



Beyond saccadic movements for interactive perception in robotics

PhD Thesis

Angel Juan Durán Bosch

Supervisor: Prof. Angel Pascual del Pobil Ferré (Universitat Jaume I)

Doctoral Programme in Computer Science

Dissertation submitted to the Escola de Doctorat in partial fulfilment of the requirements
for the Degree of Doctor by Jaume I University

Castelló de la Plana (Spain)

July 2022



Beyond saccadic movements for interactive perception in robotics

PhD Thesis

Angel Juan Durán Bosch

Supervisor: Prof. Angel Pascual del Pobil Ferré (Universitat Jaume I)

Doctoral Programme in Computer Science

Dissertation submitted to the Escola de Doctorat in partial fulfilment of the requirements for the Degree of Doctor by Jaume I University

Castelló de la Plana (Spain)

July 2022

Author: Angel Juan Durán Bosch

Supervisor: Angel Pascual del Pobil Ferré

Funding

This dissertation is funded by Ministerio de Economía y Competitividad (DPI2015-69041-R), Generalitat Valenciana (PROMETEO/2020/034) and by Universitat Jaume I (UJI-B2018-74, PREDOC/2013/06).



Creative Commons Attribution-ShareAlike 4.0 International

*–Esta tesis está dedicada a mi
mujer Ana y a mis dos hijos Eva
e Iván porque sin ellos todo esto
no tendría ningún sentido–*

Acknowledgments

This PhD thesis would not have been possible without the help and support of exceptional people around me. My first grateful words are for my PhD director Prof. Angel P. del Pobil, who offered me the opportunity to work and collaborate in the Robotic Intelligence Laboratory at Universitat Jaume I. Over the last ten years, I have gone from barely knowing what a robot is beyond science fiction movies to designing, building and programming them.

At 47 years old, almost 48, the paths that have led me to write this thesis are littered with people who have helped me. I would especially like to thank Prof. Eris Chinellato for his patience and interest, which allowed me to work with him at the Middlesex University of London under the extraordinary circumstances of the beginning of the COVID 19 pandemic.

I can not forget my friends and workmates at Robotic Intelligence Lab and Interactive & Robotic Systems Lab at Universitat Jaume I: Xavi, Juan Carlos, Jose Juan, Jose, Toni, David, Javi P., Diego, Jorge, Gabriel, Gustavo and Alejandro. I would especially mention Marco Antonelli, who guided my first steps in this research path. I also have to thank my friend Majd Kassawat, who taught me about the hardware of robotics. Furthermore, we lived together one of the most exciting experiences when participating in the Amazon Robotic Challenge in 2017 in Nagoya, Japan. Finally, I would like to thank Javier Fernández and our morning chats expressly; without them, I probably would not have been encouraged to finish this work. Thanks to his faith in Matlab and his help, I implemented the simulations of this thesis. My best wishes to all of you in the future.

There are no words to express my gratitude to my parents Angel & Amparo, my brother Vicente and his wife Elham, my parents in law Paco & Virgilia and my wife Ana. Thank you to all for your continuous support, words of encouragement, patience and understanding, for being there in each of the difficult moments that I have had since I started this thesis.

During these ten years of research, I have seen my daughter Eva grow up. She was only two years old when I started working at Robinlab. In addition, I witnessed the birth of my son Iván in 2013. Although we have suffered many adversities thanks to their love, I have completed this thesis. I hope that one day this effort can serve as an example of what hard work can achieve.

Resumen

1 Introducción

Desde el origen de la humanidad, la exploración del entorno que le rodea ha sido una de sus características que la han convertido en una especie diferente. Desde los primeros homínidos que se pasearon por las sabanas africanas hasta Neil Armstrong que puso su pie en la luna en 1969, la humanidad ha usado su habilidad para explorar y superar los límites que le impone el entorno. Estos logros han sido principalmente debidos a su capacidad de percibir, imaginar y crear con el fin de adaptarse a los diferentes entornos donde otras especies son incapaces de hacerlo. Esta característica podría definirse de muchas maneras, una de las más comunes es denominarla inteligencia.

El concepto de inteligencia es tan amplio como el número de campos del conocimiento donde se usa y por lo tanto afirmar que la inteligencia es sólo la habilidad que poseen los seres vivos de adaptarse al entorno, probablemente no es del todo preciso.

En robótica, la exploración y la inteligencia van de la mano, si el robot no es capaz de percibir de manera correcta el entorno será difícil que puede adaptarse a los cambios que se producen en él y por lo tanto presentará un comportamiento a todas luces poco inteligente.

Hoy en día existe un plausible incremento de la necesidad de sistemas robóticos inteligentes que mejoren las condiciones de vida de la sociedad en un amplio rango de campos, desde los sistemas de transportes autónomos a los que ejecutan tareas médicas.

Esta tesis intenta entender las lecciones que nos proporciona la Naturaleza, aplicada al campo de la robótica y en especial a los movimientos sacádicos de los ojos, como una herramienta que nos permite la exploración activa del mundo a través de la visión. Este trabajo deja a un lado, la mera implementación de un comportamiento en un sistema robótico específico, para estudiar la manera de generalizarlo para cualquier robot similar, dándole una nueva aplicación práctica que establece semillas para nuevos

campos de estudio y contribuyendo a la Industria 4.0 para mejorar la sociedad.

2. Motivación

2.1 Bioinspiración

Uno de los principales factores que contribuyen al progreso científico, es sin duda la observación de los fenómenos naturales. Por ejemplo, Nicolás Copérnico fue capaz de inferir de la observación de los movimientos planetarios que la tierra no era el centro del Universo, o Alexander Fleming observó cómo el crecimiento de un hongo (*Penicillium notatum*) eliminaba las bacterias presentes en un cultivo. Es razonable, por lo tanto asumir, que para poder crear sistemas inteligentes, se podría partir de la observación de las entidades consideradas como tales.

Desde esta perspectiva el estudio y observación de los organismos vivos puede aportar conocimiento que puede ser aplicado a otros campos. En este sentido, la bioinspiración se puede entender como la aplicación de los modelos biológicos a otros campos diferentes al de la biología. Ahora bien, este concepto no es nada novedoso, ya en tiempo de Leonardo Da Vinci en pleno Renacimiento, el modelado del vuelo de los pájaros le inspiró para crear máquinas voladoras.

En general, los modelos biológicos que se emplean suelen referirse a materiales, morfologías o funcionalidades, sin embargo, cuando se consideran factores extrínsecos como el entorno, la adaptabilidad de los seres vivos se convierte en un factor relevante. Los complejos sistemas biológicos actuales con sus capacidades, son el resultado de miles de millones de años de evolución donde la Naturaleza ha realizado un trabajo de mejora que puede servir como base para el desarrollo de sistemas robóticos con sus morfologías particulares que intentan desempeñar las mismas funciones.

El sistema visuo-oculomotor es probablemente el principal mecanismo con el que los seres humanos percibimos y exploramos el mundo. El considerarlo como objeto de estudio tiene ciertas ventajas, entre otras que es uno de los sistemas más estudiados en el campo de la neurobiología y medicina. En consecuencia existen numerosos modelos biológicos que tratan de explicar su funcionamiento. Por otro lado, el sistema visuo-oculomotor contiene todos los elementos que están presentes en el ciclo perceptual, presentado por Uexküll (1926): percepción del estímulo, procesamiento de la información y generación de una respuesta, que se ajustan perfectamente a los mecanismos de operación de un sistema robótico.

El replicar la morfología o funcionalidad del sistema visuo-oculomotor no necesariamente es suficiente y por lo tanto se debe considerar la adaptabilidad con el

objetivo de generar verdaderos sistemas inteligentes. Es en este punto, la bioinspiración juega de nuevo un importante rol proporcionando un modelo para crear redes neuronales artificiales que son las herramientas matemáticas para lograr el aprendizaje y la implementación de comportamientos adaptativos en los sistemas robóticos.

Los modelos biológicos pueden aplicarse a los sistemas robóticos para la exploración visual para intentar que sean funcionales, adaptativamente y morfológicamente equivalentes. En robótica, a diferencia del mundo de la biología, es habitual trabajar con sistemas únicos o en el mejor de los casos con robots del mismo fabricante. Este hecho complica la generalización y comparación de resultados. En biología, por el contrario, los individuos comparten los rasgos morfológicos, funcionales y adaptativos con otros individuos. Los individuos biológicos comparten esta información de tal manera que les permite replicar comportamiento comunes, a pesar que tengan diferentes características. Este es un modelo que puede aplicarse también en el campo de la robótica.

En definitiva, este trabajo se basa en el estudio de ciertos modelos biológicos como son las redes neuronales, el sistema visuo-oculomotor, ciclo de percepción, relación entre el genotipo, fenotipo y entorno, el proceso de fijación visual, estudiando como trasladarlos a un sistema robótico para incrementar su percepción interactiva.

2.2 Industria 4.0

La introducción de robots colaborativos en la industria han producido un cambio de paradigma. Ahora ya no es necesario separar las zonas de trabajo de personas y robots. Por el contrario se promueve la colaboración entre ambos para mejorar los procesos industriales. Este cambio en la forma de proceder contribuye de manera decisiva a la implantación de Industria 4.0. Este concepto fue introducido por el gobierno alemán en 2011 y engloba el conjunto de estrategias relacionadas con la implementación de alta tecnología en la industria. Uno de los cuatro principios fundamentales que identifica a la Industria 4.0 es la asistencia in-situ. Los sistemas automáticos deberían asistir a los humanos en los procesos de toma y ejecución de decisiones o ayudarlos con tareas difíciles o peligrosas. En este contexto el robot no es ya una máquina que trabaja de manera independiente sino que es un sistema ciberfísico que trabaja en red dotado con interoperatividad (Khaitan and McCalley, 2015) y por lo tanto debe poseer la capacidad de conectarse y comunicarse con otros dispositivos a través de Internet (Hermann et al., 2016). Así por ejemplo, Wan et al. (2016) y Kehoe et al. (2015) introducen el concepto de *cloud robotics* que trata de obtener un mejor rendimiento de sistemas robóticos individuales a través del conocimiento compartido entre múltiples sistemas robóticos.

Proyectos como *RoboEarth* (Waibel et al., 2011), recopilan almacenan y comparten datos independientemente de las plataformas robóticas utilizadas. Estructurar los datos resultantes del aprendizaje de un robot dado en un entorno para mejorar el comportamiento de robots similares en este mismo entorno, contribuye a compartir el conocimiento que requiere la Industria 4.0.

Una vez más, la Naturaleza ya ha resuelto la forma de compartir información entre los individuos de una especie. Así todo organismo vivo comparte información con las formas vivas que lo precedieron a través de los genes. Una de las motivaciones subyacentes en este trabajo es el uso de este mecanismo como fuente de inspiración para aplicarlo a un conjunto de robots interconectados de tal forma que puedan compartir conocimiento e incluso mejorar su adaptación a diferentes entornos.

2.3 Impacto en la sociedad

En los últimos 10 años la robótica ha experimentado un impulso debido al gran número de avances que han permitido mejorar sus capacidades e incrementar el interés por la misma.

- La mejora en los sensores, especialmente en el campo de la visión, con la inclusión de los sensores RGBD que ha permitido a los robots interactuar y percibir su entorno de una manera más precisa al incluir la tercera dimensión.
- La aparición de los robots colaborativos ha permitido la incorporación de la robótica en procesos de cooperación y asistencia a los operadores humanos. Pero este tipo de sistemas también ha permitido en la robótica de servicios originar un conjunto innumerable de aplicaciones que tienen efectos en nuestra vida cotidiana.
- Desde el punto de vista de la educación, la aparición de las mini computadoras como la Raspberry Pi ha facilitado la expansión de la robótica debido a las posibilidades que ofrece a la creación de sistemas robóticos de bajo coste permitiendo y promocionando el aprendizaje de la programación y de los principios básicos de la robótica.
- Finalmente, el desarrollo del Deep Learning ha tenido un impacto en todas las áreas relacionadas con el aprendizaje automático. Aunque la aplicación genérica de estas técnicas a la robótica conlleva una serie de inconvenientes, sobre todo a la hora de generalizar la soluciones, para la resolución de problemas particulares están funcionando de manera muy eficiente.

Todos estos avances han llevado a la robótica a estar cada vez más presente en nuestra vida, desde el robot aspirador, al brazo robótico que nos sirve una cerveza en la barra de un bar. Sin olvidar que hay lugares en el mundo donde los coches autónomos ya son una realidad.

Cualquiera que sea su funcionalidad, los robots necesitan obtener información del entorno con el objetivo de ejecutar sus tareas eficientemente. Este trabajo contribuye a mejorar este aspecto, usando modelos biológicos como fundamento para el desarrollo de algoritmos que permiten a los sistemas robóticos mejorar sus capacidades de percepción y adaptación a diversos entornos y por lo tanto convertirlos, en cierta manera en más inteligentes. De esta forma se incrementa su confiabilidad, contribuyendo a una mayor integración de estos sistemas en la sociedad para resolver tareas más complejas o peligrosas.

3. Objetivos

Esta tesis se centra en extender las capacidades de exploración y adaptabilidad de un sistema robótico autónomo con objeto de reproducir los movimientos sacádicos propios de los primates para una percepción interactiva. Para conseguir este objetivo se deben cumplir los siguientes hitos:

- A partir del estudio del modelo biológico del sistema visuo-oculomotor, proponer un modelo aplicable a la robótica que permita reproducir los movimientos sacádicos de una manera adaptativa a través de la exploración y el aprendizaje autónomo.
- Investigar cómo mejorar la adaptabilidad de los sistemas robóticos que reproducen los movimientos sacádicos en base a la información generada por otros sistemas que previamente ya han aprendido a operar de manera adecuada en un entorno determinado.
- Usar la capacidad de reproducir los movimientos sacádicos para mejorar ciertos aspectos de la percepción interactiva de un sistema robótico.

4. Metodología

A partir de de una serie de modelos biológicos que permiten a los seres vivos resolver un conjunto de problemas que tienen interés desde el punto de vista de la robótica, se elaboran diversos modelos matemáticos. Estos modelos son probados primero en simulación de una forma genérica y posteriormente comprobados de una forma más específica en un sistema robótico real:

- En primer lugar, se busca una solución desde el punto de vista biológico a un problema de relevancia en el campo de la robótica. Normalmente hay un modelo biológico que lo describe. En este trabajo los modelos biológicos considerados son:
 - Movimientos sacádicos de los ojos. Los movimientos sacádicos usados por los primates para explorar el mundo son un fuente de inspiración para implementar sistemas de visión activa en un sistema robótico.
 - La relación entre genotipo, fenotipo y entorno en los sistemas biológicos. Cuando un nuevo ser vivo nace, en su ADN, codifica toda la información que posteriormente, condicionada por el entorno, definirá su comportamiento y capacidades. Cuando se construye un sistema robótico, todos sus controladores y propiedades se ajustan para obtener un cierto comportamiento desde cero, sin considerar ningún conocimiento previo. Esto implica que no se reutilizan las experiencias que otros sistemas robóticos similares que realizan la mismas tareas han desarrollado previamente. Los patrones con los que la naturaleza transmite la información entre sucesivas generaciones pueden ayudar a mejorar esta circunstancia.
 - La ejecución de mecanismos y procesos complejos generados en los seres vivos, tal como el procesos de fijación visual, incrementa la información disponible en el cerebro para procesar el entorno que les rodea. Por lo tanto, la transferencia de estos mecanismos a un sistema robótico debería mejorar sus capacidades perceptuales.
- Los sistemas biológicos con sus miles de millones de años de evolución son tan complejos, que replicarlos puede convertirse en una tarea inalcanzable. Afortunadamente, gracias a la abstracción, es posible simplificar las partes esenciales de estos sistemas y modelarlos matemáticamente. Para este fin, es necesario desarrollar una serie de hipótesis que permitan la creación de modelos matemáticos simplificados.
- Como paso previo a la implementación de estos modelos, deben ser comprobados en simulación. En algunos casos, la imposibilidad de disponer de suficientes robots reales para testear estos modelos genera que la simulación sea el único sistema hábil para hacerlo. En la simulación, las hipótesis realizadas para el desarrollo de los modelos simplificados son validadas.
- Si las hipótesis desarrolladas en el diseño de los modelos bio-inspirados se validan en simulación, se implementan en un sistema robótico real, verificando que exhibe los mismos comportamientos que se habían obtenido en simulación.

Los resultados numéricos obtenidos —tanto en las simulaciones como en los sistemas robóticos reales usados en este trabajo— se comparan, o bien con otros métodos que permiten su estimación utilizando la literatura existente, o con la simulación de ciertos parámetros que nos permiten evaluar su rendimiento.

En cada modelo propuesto, se ha utilizado la misma metodología para su desarrollo e implementación, cambiando la forma de comparar los resultados o los sistemas robóticos utilizados. En algún caso específico ha sido necesario el diseño y desarrollo de un sistema robótico para poder verificar los resultados.

5. Contenido

Esta tesis se estructura en varios capítulos. Así, los capítulos 2 y 3 describen respectivamente el modelo biológico para la generación de sacadas y los modelos matemáticos para tratar el problema de la adaptación. El resto de capítulos describen el trabajo realizado para desarrollar e implementar los modelos bioinspirados como posibles soluciones a varios problemas en el campo de la robótica. En cada capítulo, los objetivos que se establecen y los resultados que se obtienen contribuyen a la solución de dichos problemas.

El primer apéndice propone una aplicación derivada de los modelos desarrollados. Usando los conceptos de percepción interactiva y técnicas de aprendizaje profundo se ofrece una posible solución al problema de la estimación de la profundidad con una sola cámara. El segundo apéndice describe la arquitectura software que hemos desarrollado específicamente para la implementación de todo este trabajo en varios sistemas robóticos.

En el capítulo 2 se presenta la base biológica de los movimientos sacádicos de los ojos. Estos movimientos son el núcleo sobre el que gira todo el trabajo desarrollado y que claramente tiene una fuerte bioinspiración. Hay modelos en biología que describen cómo los seres humanos generamos movimientos sacádicos para explorar de forma activa el mundo. Este capítulo, de manera resumida, describe cómo nuestros cerebros generan los comandos para producir los movimientos sacádicos a partir de la percepción de un estímulo. Estos movimientos son la base para los modelos que se describirán en capítulos posteriores. De igual manera, en este capítulo se detalla brevemente la morfología del sistema visuo-oculomotor que permitirá, en capítulos posteriores, ofrecer diferentes alternativas morfológicas aplicables a un sistema robótico que desee replicar este comportamiento sacádico.

El entorno, entendido como todo aquello exógeno al sistema objeto de estudio, suele ser el gran olvidado de la robótica, sin embargo, es un factor que se ha tenido

presente a lo largo de todo este trabajo. La idea de adaptabilidad llevada a un sistema robótico —entendida como la capacidad de los seres vivos de adecuarse e interactuar con el entorno— es uno de los pilares para la generación de sistemas más inteligentes. Existen herramientas matemáticas que permiten, en cierto modo, representar el modelo interno de los sistemas robóticos para adaptarse a los cambios que se producen en el entorno. Se da la circunstancia que estos modelos matemáticos son también bioinspirados, específicamente en las redes neuronales que conforman los sistemas nerviosos de los seres vivos. El concepto de aprendizaje adaptativo hace referencia a la capacidad de el sistema de aprender de los cambios en el entorno, con el objetivo de mejorar su rendimiento. En el capítulo 3 se realiza una pequeña introducción a los modelos de redes neuronales artificiales que permiten en cierta manera la adaptabilidad, así como los fundamentos matemáticos requeridos para su implementación.

En el capítulo 4 se describen las principales morfologías de cabezas robóticas que pueden imitar los movimientos generados por el sistema visuo-oculomotor descrito en el capítulo 2. Seleccionamos la morfología tipo Helmholtz con objeto de construir el sistema robótico para la ejecución de movimientos sacádicos, detallando los parámetros morfológicos que la describen. A continuación, se estudia cómo la integración de las dos señales visuales es de utilidad para generar movimientos sacádicos más precisos a partir de la interacción perceptual. Finalmente, se proponen dos arquitecturas de controladores para la ejecución de estos movimientos. Estas son comparadas con otros tipos de modelos desarrollados en robótica para la ejecución de estos movimientos exploratorios.

Una vez tenemos varios métodos descritos en el capítulo 4 para la ejecución de movimientos sacádicos basados en técnicas de aprendizaje adaptativo, en el capítulo 5 proponemos reutilizar, en cierta manera, la información de cómo un sistema ha aprendido a adaptarse a un entorno específico para mejorar el aprendizaje de otros sistemas similares. Esta relación entre las propiedades específicas de un sistema que constituyen parcialmente su morfología y los parámetros que le permiten adaptarse al entorno, nos ha llevado a proponer un modelo artificial basado en el triángulo genotipo, fenotipo y entorno descrito en los organismos biológicos. El objetivo final ha sido crear un modelo de tal forma que la morfología de un sistema robótico y su modelo interno pueden transmitirse de un sistema robótico a otro de la misma manera que se produce en un sistema biológico.

Finalmente, los movimientos sacádicos descritos en el capítulo 2 y modelados en el capítulo 4 para la exploración visual, tienen implicaciones en otros comportamientos biológicos como es el caso del proceso de fijación visual. En el capítulo 6 se explora el alcance e importancia de los movimientos sacádicos en este proceso de fijación para conseguir la información de profundidad en un entorno. Por otro lado, en este capítulo

se describe brevemente el sistema robótico desarrollado específicamente para testear los algoritmos propuestos.

6. Conclusiones

Como resultado de la aplicación de la metodología presentada en el capítulo 1, se han descrito brevemente diversos modelos biológicos que nos han permitido elaborar propuestas de modelos matemáticos para la ejecución de los movimientos sacádicos como eje central de este trabajo. A partir de la implementación de dichos movimientos se han desarrollado y explorando diversas aplicaciones en el campo de la robótica. En el capítulo 2 se ha descrito el sustrato biológico, tanto fisiológico como neurológico, que permite la generación de movimientos sacádicos en el ser humano.

Una cualidad innegable de los seres vivos es su capacidad de adaptación al entorno. Normalmente, en robótica, el entorno es algo siempre considerado como un inconveniente para nuestros algoritmos. En el caso de los seres vivos, el entorno es un elemento más de su evolución. Por ello, en este trabajo hemos tenido presente en todo momento el entorno. Hemos optado por emplear redes neuronales adaptativas, que aprenden a medida que se dispone de información, para emplear las herramientas adecuadas que nos permitan simular la adaptación de un modelo artificial a un entorno determinado. Estas herramientas matemáticas, así como sus fundamentos matemáticos, se describen en el capítulo 3. En particular, el uso de redes neuronales de una sola capa oculta, que emplean el aprendizaje supervisado, son la herramienta básica elegida como núcleo de los modelos desarrollados en el capítulo 4 para implementar los movimientos sacádicos en un sistema robótico. En el capítulo 3 se proponen dos algoritmos para adaptar estas redes neuronales, como son los basados en el filtro de Kalman o en regresiones incrementales de procesos gaussianos dispersos.

Después de un análisis de las dos configuraciones más comunes (Flick y Helmholtz) para reproducir los movimientos de los ojos humanos en un sistema robótico, seleccionamos la configuración de Helmholtz que permite una aproximación suficiente a los movimientos del ojo y tiene la ventaja sobre la configuración de Fick que requiere de un motor menos y por lo tanto un grado de control menos. Una vez se ha establecido la configuración del sistema robótico, procedemos a su caracterización paramétrica. Este proceso de parametrización pretende ir más allá de un sistema robótico específico, describiendo la morfología de estos sistemas de una manera genérica. Así, por ejemplo, la configuración de Helmholtz de un sistema robótico capaz de generar movimientos sacádicos está formada por dos cadenas cinemáticas que comienzan en la base del sistema y finalizan en cada una de las cámaras que realizan la función de visualizar el entorno. La longitud de los *links* entre cada *joint* de estas cadenas es un parámetro que

puede variar de un sistema robótico a otro y definen la morfología final del sistema. En robótica, la parte cinemática se suele diferenciar del resto para realizar la descripción paramétrica de un sistema robótico. En este caso, hemos incluido el sistema de visión como un elemento más en la definición morfológica del sistema. De esta manera, el sistema robótico se ve como un todo, donde su morfología no está definida sólo por su perceptible forma externa, sino también por todos los parámetros diferenciales de otros sistemas similares.

Una vez definida la morfología del sistema robótico capaz de mimetizar los movimientos sacádicos, estudiamos cómo implementar el comportamiento sacádico. Primero, se observa que la codificación binocular del estímulo mejora la precisión en la generación de las sacadas de un sistema robótico. Se proponen dos arquitecturas para generar las acciones de control necesarias para focalizar un estímulo en un sistema binocular por medio del movimiento sacádico: *Feedback error learning* y *Recurrent architecture*. Los resultados obtenidos en la simulación y el robot real arrojan ratios de error por debajo del 1% para ambos controladores teniendo presente la resolución de las cámaras utilizadas. También se ha observado que los datos experimentales del conjunto de test se ajustan bastante bien a una distribución de probabilidad de valor extremo.

El rendimiento obtenido con estas dos arquitecturas las convierte en claras candidatas para la implementación del comportamiento sacádico en nuestro sistema robótico. Mientras que la *Recurrent architecture* presenta mejores resultados de precisión en los movimientos, la velocidad de ejecución de la arquitectura *Feedback error learning* la convierte en favorita para muchos de los experimentos y algoritmos que son descritos a lo largo de este trabajo.

De esta manera, se define un sistema robótico con cierta morfología (configuración de Helmholtz) y capaz de ejecutar el comportamiento sacádico basándose en la arquitectura *Feedback error learning* o *Recurrent architecture*. Sin embargo, el entorno modifica y condiciona su comportamiento: por lo tanto, se introduce su definición y parametrización. De esta manera, el entorno es considerado como la región del espacio delimitada por 6 planos que definen un paralelepípedo donde un conjunto de estímulos puntuales se distribuyen aleatoriamente y con los que el sistema robótico interactúa. Los parámetros que lo definen son sus dimensiones volumétricas y su centro geométrico.

De este modo, siempre que aparezcan los estímulos adecuados, el sistema robótico puede desarrollar el comportamiento sacádico en el entorno definido. Sin embargo, para conseguirlo es necesario entrenar el sistema desde cero. Cuando un ser vivo nace, hay una serie de comportamientos preprogramados que permiten al

organismo adaptarse al nuevo entorno en lugar de aprender. Estos comportamientos preprogramados están grabados por miles de millones de años de evolución en el código genético de los seres vivos. En cambio, en la robótica los sistemas suelen tener pocas evoluciones en el mejor de los casos, siendo entidades únicas.

En el capítulo 5, sugerimos cómo se puede aprovechar la relación entre el modelo interno, capaz de generar un determinado comportamiento, y la morfología de un conjunto de sistemas robóticos para acelerar la adaptación de nuevos sistemas robóticos a un entorno determinado en lugar de partir de cero. Para lograr este objetivo, ha sido necesario utilizar diversas herramientas matemáticas para aprender la relación entre el modelo interno y la morfología.

Para disponer de una población de sistemas robóticos lo suficientemente grande como para aprender esta relación, se han entrenado desde cero unas 45.000 cabezas robóticas con el mismo número de parámetros morfológicos pero con valores aleatorios dentro de ciertos rangos. De las tres arquitecturas de redes neuronales probadas, la red neuronal paralela es la que mejores resultados ha dado a la hora de predecir el modelo interno de un sistema conociendo los parámetros morfológicos que lo definen. Una aplicación práctica que se desprende de los resultados obtenidos es que con un conocimiento parcial de la morfología es posible mejorar el aprendizaje de un nuevo sistema.

Utilizando de nuevo la herramienta de la bioinspiración, se ha formulado un genotipo artificial que caracteriza en cierta manera a cada individuo de una *especie* de sistemas robóticos. Tiene una parte específica del sistema robótico y otra común a la especie a la que pertenece. Se define a partir del conocimiento obtenido de la relación entre la morfología y el modelo interno de todos los individuos de la especie. Una vez inicializado, este genotipo puede evolucionar para adaptarse a otros entornos. Este desarrollo tiene implicaciones directas en la Industria 4.0: los nuevos robots pueden utilizar el conocimiento adquirido por otros robots de la misma especie relativo a cómo interactuar en un entorno determinado para desarrollar sus capacidades. Todo ello sin olvidar que la adaptación al entorno local donde tendrán que operar puede modificar su comportamiento a través del proceso de adaptación.

Para observar si este modelo es análogo al propuesto por la genética analítica para relacionar fenotipo, entorno y genotipo, se utiliza el concepto de norma de reacción trasladado a un conjunto de sistemas robóticos que presentan un comportamiento sacádico y cuyo genotipo artificial se ha obtenido previamente. Los resultados muestran que existe una variación fenotípica análoga a la que producen los organismos vivos en un entorno biológico.

En la última parte de este trabajo (capítulo 6), se modeló el proceso de fijación

en el que los movimientos sacádicos juegan un papel importante pero biológicamente poco claro en el ser humano. A partir de una serie de hipótesis sobre los movimientos sacádicos generados, definidos en el capítulo 4, se diseñó un algoritmo para la estimación de la imagen de profundidad alrededor del punto de fijación.

La idea desarrollada para proponer este algoritmo se basa principalmente en que los movimientos microsacádicos y de la cabeza generan pequeños desplazamientos de la retina que combinados con la señal de variación de la posición del ojo o de la cámara permiten obtener más información del entorno. Cada píxel de la imagen se convierte en un problema de minimización independiente: entre el desplazamiento óptico real y el predicho por el desplazamiento retiniano calculado a partir de la variación de posición de la cámara y considerando que el objeto que ilumina ese píxel está a una determinada distancia. Mediante la minimización de la diferencia entre los flujos ópticos predichos por el desplazamiento físico y los flujos obtenidos por la diferencia de imágenes, se puede ir variando la estimación de la distancia de cada píxel.

Como en el proceso de fijación se combinan los movimientos sacádicos y los movimientos del cuello, se ha desarrollado una simulación para evaluar su rendimiento en relación con dos referencias: la imagen de profundidad estimada en la simulación y un conjunto de marcadores (marcadores Aruco) colocados estratégicamente en el campo de visión para determinar la distancia de estas regiones a la cámara. Estos últimos son decisivos para evaluar el algoritmo en el robot real.

Las pruebas de simulación han demostrado que el algoritmo funciona de manera correcta, aunque es bastante sensible al error en la estimación de la posición de la cámara. También se ha observado que las micro-sacadas, inherentes al proceso de fijación, pueden tener relevancia para dar cierta estabilidad a la estimación de la profundidad. Las pruebas realizadas con objetos semitransparentes en simulación dieron buenos resultados que posteriormente se confirmaron en el robot real.

Para reforzar el comportamiento bioinspirado del algoritmo se estudia cómo funciona frente a la ilusión de Ouchi, produciendo una diferenciación de zonas de profundidad similar a la obtenida en el caso biológico.

Para realizar los experimentos en un robot real, necesitamos de un sistema que permitiera la ejecución de los movimientos sacádicos definidos en capítulo 4. Sin embargo, también es necesario un cuello con 6 DOF. Por este motivo, se diseña y construye un prototipo de sistema robótico basado en la conjunción de una plataforma Stewart rotatoria y un generador de sistemas sacádicos basado en la configuración de Helmholtz.

Los experimentos realizados con este sistema robótico confirman el rendimiento del algoritmo y sus limitaciones. Su principal limitación radica en la necesidad de

disponer de una adecuada precisión en la estimación del desplazamiento de la cámara. Por otro lado, uno de sus mayores puntos fuertes es la capacidad de detección de la profundidad en objetos transparentes, para los que las cámaras RGBD no funcionan de manera correcta.

En resumen, se han desarrollado tres modelos bioinspirados basados en tres procesos biológicos centrados en la generación de movimientos sacádicos para su implementación en sistemas robóticos, permitiendo:

- Aumentar su capacidad de visión y exploración activa a través de la percepción interactiva.
- Emplear el conocimiento de cómo otros sistemas robóticos han aprendido a adaptarse a un entorno determinado para mejorar y acelerar el aprendizaje de nuevos sistemas robóticos
- Aumentar la capacidad de percepción de un sistema robótico mediante la combinación de movimientos inspirados en el proceso de fijación visual.

7. Contribuciones

El trabajo descrito en esta tesis contribuye al desarrollo de varios modelos útiles y aplicables en el campo de la robótica y específicamente en la visión activa:

1. Proponemos dos arquitecturas bioinspiradas para la generación de movimientos sacádicos para la exploración activa del entorno basada en la percepción interactiva. Además, parametrizamos un sistema robótico capaz de generar estos movimientos, trascendiendo la mera descripción de las cadenas cinemáticas que lo componen, y considerando los elementos que constituyen el sistema de visión como parte del conjunto que define la morfología del robot.
2. Desarrollamos el concepto de genotipo artificial para describir un conjunto de sistemas robóticos con características morfológicas comunes y que interactúan con un mismo entorno. Este genotipo se inicia a partir de la metodología desarrollada para determinar la relación entre el modelo interno y la morfología de un sistema robótico. Podemos obtener esta relación utilizando redes neuronales de una sola capa oculta trabajando en paralelo.
3. Basándonos en la hipótesis de que los movimientos microsacádicos pueden desempeñar un papel específico en el proceso de fijación en los seres humanos, desarrollamos un algoritmo para estimar una imagen de profundidad utilizando

una sola cámara minimizando la diferencia entre el desplazamiento óptico de la imagen debido a los movimientos de fijación y el desplazamiento óptico esperado a partir del cambio de posición de la cámara.

4. Diseñamos y construimos un sistema robótico que permite generar movimientos sacádicos y movimientos coordinados del cuello. Este sistema se basa en una plataforma Stewart rotatoria combinada con un sistema Helmholtz. Hemos desarrollado los controladores de software para el control de este sistema.
5. Los modelos desarrollados basados en los movimientos microsacádicos y la percepción interactiva, nos han permitido generar y mejorar redes neuronales de Deep Learning para la estimación de la imagen de profundidad con una sola cámara (Apéndice A).
6. Para la implementación de todos los algoritmos propuestos en los sistemas robóticos empleados, desarrollamos una arquitectura de software (Apéndice B) que nos permite la programación modular y la ejecución de procesos en paralelo.

8. Trabajo futuro

El trabajo expuesto en esta tesis puede ser el inicio de futuras líneas de investigación y propone preguntas abiertas que mejorarían y ampliarían el trabajo realizado.

1. Una de las líneas de investigación puede establecerse a partir del concepto de genotipo artificial introducido en este trabajo. Estudios recientes en el campo de la biología corroboran que la plasticidad del desarrollo juega un papel fundamental en la diversificación y especialización de los organismos. Estamos convencidos de que esta área emergente de investigación en biología puede contribuir a un nuevo paradigma en la robótica evolutiva. La idea fundamental que subyace es la de superar las limitaciones actuales de la robótica evolutiva en cuanto a la decodificación genotipo-fenotipo, que en la inmensa mayoría de los casos se plantea como una correspondencia uno a uno, es decir, que los genes determinan de forma exclusiva los fenotipos de tal manera que la influencia del entorno se limita a su utilización como simple banco de pruebas para evaluar la aptitud del fenotipo. Esto contrasta con los conocimientos actuales de la biología evolutiva, según los cuales los genotipos de los organismos muestran una capacidad para expresar una serie de fenotipos diferentes en respuesta a las distintas condiciones ambientales. Esto se conoce como plasticidad fenotípica o de desarrollo y puede visualizarse mediante el uso de normas de reacción, que representan valores de un rasgo fenotípico específico para un conjunto de entornos.

A partir del modelo de genotipo artificial propuesto, hemos podido estimar una norma de reacción para una *especie* de sistemas robóticos. Los resultados obtenidos simulan el proceso en función de un genotipo para un sistema robótico que se desarrolla en diferentes entornos que conducen a la expresión de diferentes fenotipos, mostrando un comportamiento similar al de los organismos vivos en términos de su norma de reacción.

Un posible punto de partida, apoyado por varios estudios recientes en biología evolutiva, es la incorporación de mecanismos de plasticidad en el desarrollo de robots mediante un enfoque evolutivo, lo que podría ser beneficioso para obtener sistemas robóticos autónomos con mayor capacidad de adaptación al entorno. Dado que la selección natural no sólo selecciona entre genotipos, sino también entre fenotipos, el fenotipo y la variación entre fenotipos, puede jugar un papel importante en la evolución artificial de los robots, de forma que el entorno no sólo puede servir para seleccionar entre las variaciones producidas genéticamente, sino también para crear variación fenotípica y seleccionar entre esa variación. Para ello, nuestro modelo de genotipo artificial puede proporcionar este punto de partida.

La idea es que, en lugar de buscar un diseño de robot (genotipo) que puntúe más alto en una determinada función de aptitud bajo unas condiciones ambientales concretas, el diseño ganador será el que muestre la mayor plasticidad en su fenotipo y, por tanto, será más adaptable a las circunstancias ambientales cambiantes. Este resultado tendrá importantes implicaciones para las aplicaciones prácticas de los sistemas robóticos autónomos, ya que esta plasticidad o capacidad de adaptación permitirá al sistema modificar rápidamente su comportamiento ante nuevas circunstancias de su entorno para las que no fue explícitamente diseñado o, en el caso de la plasticidad morfológica, rediseñarlo de forma mucho más eficiente en función de su norma de reacción, lo que nos ayudará a predecir los rasgos óptimos que debe tener el nuevo fenotipo.

2. Otras posibles líneas de investigación surgen de las aportaciones realizadas en el capítulo 6:

- Perfeccionamiento del algoritmo de estimación de profundidad propuesto para hacer frente a los problemas que plantea el ruido en la estimación de la variación de la posición de la cámara. Se podría extender su uso a situaciones en las que el sistema robótico experimenta oscilaciones incontroladas en torno a un punto inicial, por ejemplo un dron que planea, donde podemos determinar su desplazamiento relativo respecto a una posición inicial y utilizar este desplazamiento para generar la imagen de profundidad.

- La percepción de la profundidad en los seres humanos proviene de la integración de varias señales. Los experimentos con el prototipo ad-hoc desarrollado para replicar los movimientos de fijación nos han permitido estimar la imagen de profundidad de forma independiente para cada cámara. Pero esta señal, podría integrarse con la estimación de la profundidad utilizando la disparidad generada entre la imagen izquierda y la derecha. Una posible línea de investigación sería la integración de estas tres señales para obtener una imagen de profundidad más precisa del entorno que rodea al sistema robótico.
- El uso de la aproximación geométrica para modelar el movimiento de fijación ha permitido diseñar e integrar una serie de modelos de *deep learning* para determinar la imagen de profundidad a partir de la variación de la imagen producida por un desplazamiento de la cámara (Apéndice A). El perfeccionamiento de estos modelos de aprendizaje profundo a partir de la integración del sistema binocular podría ser una posible línea de investigación.
- El resultado obtenido con la ilusión de Ouchi puede iniciar una línea de investigación para probar el sistema robótico construido con otros tipos de ilusiones ópticas, incluso parametrizarlos y confirmar que su comportamiento es similar al biológico, lo que permitiría validar el modelo propuesto y aplicar las hipótesis establecidas como punto de partida para posibles nuevos modelos biológicos.

Abstract

Billions of years of evolution have generated highly complex systems that allow them to adapt to their environments and generate a series of behaviours. Starting from this premise, it seems reasonable to assume that for building adaptive and somewhat intelligent robotic systems, it should be essential to observe biological systems exhibiting such properties. Then, the bio-inspiration concept could be a helpful tool. This thesis is based on the study of the biological model for the generation of saccadic eye movements in humans, which is one of the fundamental pillars for exploring the world around us, and an essential part of the concept of active vision. The adaptation process that guides the learning of this model in a robotic system is based on interactive perception.

The morphological characterisation of a robotic system for the execution of this type of movement, along with the development of several internal model proposals allowing the replication of this biological behaviour in a robotic system are some of the main axes of this work. The morphology selected is based on the Helmholtz setup model. This configuration is a binocular system that is a simplified replica of the primate visuo-oculomotor system. Therefore, we perform a preliminary analysis regarding the necessity of encoding the stimulus using binocular or monocular vision. Our result suggest that using the cues from both cameras notably increases the precision of the saccadic movements.

The internal model enabling the generation of behaviour from the information coming from the environment through the sensors is described by two different architectures: Feedback error learning and Recurrent architecture. These architectures are based on the principles of interactive perception. Several experiments are conducted to evaluate the characteristics of both architectures, concluding that although the Recurrent architecture offers better performance concerning the accuracy, the time needed to adapt it is a limitation that the feedback error learning architecture does not present. In either case, they are both more precise than suggested in some literature systems and have the quality of being adaptive. In this way, the evolution of the system through interactive perception turns the environment into a fundamental actor in the process, ultimately defining the behaviour of the robot.

The estimation of the internal model associated with a given morphology emerges from the interaction with the environment. The adaptive supervised learning techniques presented in this work have an important limitation once the internal model has been learned: any change in morphology or environment involves a learning process.

In this thesis, we propose a model to obtain the correlation between morphology and internal model parameters so that a new internal model can be predicted when morphological parameters are modified. Furthermore, we suggest different neural network architectures to address this dimensionally severe regression problem. Using the studied robotic system for generating saccade eye movements, we evaluate the performance of each approach. The best results are achieved for an architecture with parallel neural networks. Our results can be instrumental in state-of-the-art trends such as self-reconfigurable robots, reproducible research, cyber-physical robotic systems, or cloud robotics. Furthermore, internal models would be available as shared knowledge so that robots with different morphologies can readily exhibit a particular behaviour in a given environment.

In nature, the transmission of information between generations through genes solves the problem of starting from scratch. Thus, when a living being is born, it does not need to generate its entire internal model to display its behaviours. Instead, it starts from a model encoded by its genes, subsequently modified and adapted by the various environments in which the living being is developing. Based on this premise, the procedure and the machine learning tools used to predict the internal model from morphology allow us to establish the concept of artificial genotype, proposing what an individual and a species are from the point of view of robotics. Finally, we suggest this model exhibits a behaviour that is similar to that of living organisms regarding the concept of reaction norm.

To conclude, an algorithm based on microsaccades and head movements during visual fixation is presented. It combines the images generated by these micro-movements with the ego-motion signal in order to compute the depth map. Depth estimation is a challenge for robots and living organisms in their adaptation to evolving environment. We propose a model that abstracts head microdisplacements and microsaccadic movements. A depth map of the initial image can be obtained using the stream of images produced in the visual fixation process as a disturbance in the initial image of the fixation point. The algorithm is tested in a robot eye-in-hand simulation, and, in light of the results obtained, it can satisfactorily estimate the depth map. Also, they corroborate the fact that microsaccades are instrumental in stabilising this estimation. In order to implement this algorithm in a real robotic system, we designed and built a visuo-oculomotor system with a Helmholtz distribution and mounted it on a rotating Stewart platform enabling us to perform the neck functions. By replicating the fixation

movements, we obtained the depth image in both cameras based on the algorithm.

In summary, throughout this thesis, we present various algorithms and methodologies with a solid biological inspiration improving the perception and adaptation capabilities of robotic systems in general, being the environment the force that allows the systems to improve their performance.

Keywords: *saccadic eye movements, monocular depth estimation, bioinspiration, genotype, phenotype, norm of reaction, morphology, internal model, adaptive neural networks, visual learning, microsaccades, sensorimotor development.*

Contents

1	Introduction	1
1.1	Motivation	2
1.1.1	Bio-inspiration	2
1.1.2	Industry 4.0	4
1.1.3	Impact on society	4
1.2	Aims and scope	5
1.3	Methodology	6
1.4	Outline	9
2	Saccadic eye movements	11
2.1	Introduction	11
2.2	Objectives	14
2.3	Eye muscles: Visual system effectors	15
2.4	Retina and visual cortex: Visual sense	17
2.5	Saccades: Visual system behaviour	19
2.6	Conclusion	22
2.7	Publications supporting this chapter	22
3	Adaptive learning in neural networks	23
3.1	Introduction	23
3.1.1	Definition of artificial neural networks	25
3.1.2	Parallel and distributed computation in biology	26
3.1.3	Feedforward neural networks	27
3.1.4	Adaptation and learning process	29
3.2	Objectives	30
3.3	Supervised linear learning	31
3.3.1	Linear model	32
3.3.2	Learning linear model parameters	32
3.3.2.1	Optimization approach	32
3.3.2.2	Maximum likelihood estimation for linear model	33

3.3.2.3	Bayesian learning approach	34
3.3.3	Linear prediction	35
3.3.3.1	Output predictions	35
3.3.3.2	Probabilistic prediction	36
3.4	Supervised non-linear learning using basis functions	37
3.4.1	Basis functions	37
3.4.2	Radial basis function neural network	38
3.4.3	Trigonometric basis functions neural network	39
3.5	Supervised and adaptive learning algorithms	40
3.5.1	Kalman filter for neural network adaptive training	41
3.5.2	Incremental sparse Gaussian process regression for neural network adaptive training	44
3.6	Conclusions	46
3.7	Publications supporting this chapter	46
4	Robotic systems for visual exploration by means of saccadic movements	47
4.1	Introduction	47
4.2	Objectives	52
4.3	Morphology of a robot system for executing saccadic movements	53
4.4	Saccadic behaviour	55
4.4.1	Monocular vs. binocular encoding	56
4.4.1.1	Introduction	56
4.4.1.2	Simplified robot head model	57
4.4.1.3	Radial basis function to compute the visuo-oculomotor transformations	59
4.4.1.4	Training and testing datasets	59
4.4.1.5	Comparison monocular vs. Binocular encoding	60
4.4.1.6	Conclusion	64
4.4.2	Recurrent architecture vs. Feedback error learning	65
4.4.2.1	Feedback error learning	66
4.4.2.2	Recurrent architecture	66
4.4.2.3	Inverse linear controller	68
4.4.2.4	Adaptive controller	70
4.4.2.5	Experimental setup to estimate FEL and RA controllers	71
4.4.2.6	Simulation results	76
4.4.2.7	Robots results	81
4.4.2.8	Conclusion	85
4.5	Environment	86
4.5.1	Environment definition	87

4.5.2	Parameters defining the environment	88
4.6	Conclusion	89
4.7	Publications supporting this chapter	89
5	Predicting the internal model of a robotic system from its morphology	91
5.1	Introduction	91
5.2	Objectives	94
5.3	Morphology and Internal Model	95
5.4	Extracting knowledge from multiple robotic systems	99
5.4.1	Morphology of the robotic systems to predict the internal model	100
5.4.2	Defining an environment to adapt robotic systems	101
5.4.3	One behaviour for multiple robotic systems	104
5.5	Generating one robotic head with saccadic behaviour	105
5.6	Generating multiple robotic heads with saccadic behaviour	108
5.7	Machine learning problem	110
5.7.1	Solving the regression problem for the fixed controller	110
5.7.2	Solving the regression problem for adaptive controller	112
5.7.2.1	Single layer feedforward neural network option	113
5.7.2.2	Deep neural network option	114
5.7.2.3	Parallel feedforward neural network	118
5.7.2.4	Comparing the obtained results based on sequential test and MSE	119
5.7.2.5	Comparing the obtained results based on model selection information criteria	126
5.7.2.6	Conclusion of comparison	130
5.7.3	Family and individuals of the robotic systems	130
5.8	Applications	131
5.8.1	Learning improvement for new robot morphologies	131
5.8.2	Predicting the internal model with partial knowledge about morphology	132
5.9	Bio-inspired model of artificial genotype and norm of reaction in a robotic system	135
5.9.1	Introduction	135
5.9.2	Genotype model	138
5.9.3	Applying the GEP model to the robotic systems	143
5.9.4	Experiments to evaluate GEP model	146
5.9.5	Conclusion	150
5.10	Conclusion	150
5.11	Publications supporting this chapter	152

6	Saccadic behaviour for depth estimation	153
6.1	Introduction	153
6.2	Objectives	157
6.3	Model	158
6.3.1	Model hypothesis	158
6.3.2	Mathematical model	158
6.3.3	Depth estimation algorithm	166
6.3.4	Algorithm parameters	167
6.4	Experiments in simulation	168
6.4.1	Simulation setup	168
6.4.2	Experimental procedure	169
6.4.3	Evaluation methods	169
6.4.4	Experimental tests	172
6.4.4.1	Influence of the choice of parameters	173
6.4.4.2	Additive error influence	177
6.4.4.3	Aruco marker comparison	183
6.4.4.4	Environment with ordinary objects	183
6.4.4.5	Influence of microsaccade movements in the depth algorithm execution	191
6.4.4.6	Ouchi Illusion	193
6.5	Experiments in a real robot	195
6.5.1	Introduction	195
6.5.2	Oculomotor system	196
6.5.2.1	Mechanical and optical design	196
6.5.2.2	Oculomotor system control for saccade generation	198
6.5.3	Neck system	202
6.5.3.1	Introduction	202
6.5.3.2	Stewart Platform	204
6.5.4	Built oculomotor-neck system	204
6.5.5	Depth estimation test with the real robot	206
6.5.5.1	Environment setup	206
6.5.5.2	Parameter adjustments and fixation process	207
6.5.5.3	Results	208
6.6	Conclusions	214
6.7	Publications supporting this chapter	215
7	Conclusions and Future work	216
7.1	Conclusions	216
7.2	Contributions	220

7.3	Future work	221
A	Deep learning application to monocular depth estimation in warehouse automation	224
A.1	Introduction	224
A.2	Methodology	227
A.2.1	DepthS neural network	228
A.2.2	DepthC neural network	230
A.2.3	DepthCSx neural network	230
A.3	Experimental setup	232
A.4	Experimental tests	233
A.5	Experimental results	234
A.6	Conclusions	236
A.7	Publications supporting this appendix	238
B	Software architecture to implement the developed algorithms	239
B.1	Introduction	239
B.2	Software architecture description	242
B.2.1	Module commands	243
B.2.2	Module events	244
B.2.3	Module parameters	245
B.3	Software architecture implementation	246
B.3.1	Service base module	246
B.3.2	Action primitive module	248
B.3.3	Module combination: generating the architecture	250
B.4	Conclusions	253
B.5	Publications supporting this appendix	254
C	Mathematical fundamentals	256
C.1	Introduction	256
C.2	Mathematical notation of scalars, vectors and matrices	256
C.3	Probability	257
C.4	Probability Rules	258
C.4.1	Conditional Probability	258
C.4.2	Total Probability Rule	258
C.4.3	Bayes' Theorem	258
C.4.4	Chain Rule	259
C.5	Inference and Bayesian learning	259
C.6	Random variable and probability distribution models	260

C.7 Likelihood function	261
C.8 Maximum likelihood estimation	261
C.9 Bayesian learning to estimate model parameters	262
C.10 Covariance	263
C.11 Gaussian distribution	264
C.11.1 Definition	264
C.11.2 Sampling from a Gaussian distribution	268
C.11.3 Conditional Gaussian distribution	268
C.11.4 Marginal and conditional Gaussians	269
C.12 Introduction to the kernel concept	270
D Specifications of the components of the oculomotor and neck robotic system	272
Index of figures	276
Acronyms	283
Publications supporting this thesis	285
Bibliography	288

Chapter 1

Introduction

Do, or do not. There is no try.

**Yoda. Star Wars: Episode V
The Empire Strikes Back (1981)**

Indeed, since the origins of humanity, exploring the environment around has been one of its characteristics. From the first hominids in Africa who left the savannah to Neil Armstrong, who set foot on the moon in 1969, human beings have used their ability to explore and go beyond the established limits of the environment. These achievements have been mainly the result of the ability of human beings to perceive, imagine, and create with the aim of adapting to environments where other species are unable to do so. This ability could be defined in many ways, one of the most common is to call it intelligence..

The concept of intelligence is as broad as the fields where it is applied. Therefore, to say intelligence is just an ability of living beings to adapt to their environment is probably inaccurate. However, adaptive capacity has a considerable bearing on intelligence behaviour.

In robotics, exploration and intelligence go hand in hand. If the robot cannot perceive the environment correctly, it will be difficult for it to adapt to the changes

occurring there, and it will therefore appear unintelligent.

Interactive perception blurs the barrier between perceiving the environment and acting on it, allowing the robotic system to adapt through interaction with the environment. Interactive perception is an evolution of the concept of active perception, which [Bajcsy et al. \(2018\)](#) defines as:

“An agent is an active perceiver if it knows why it wishes to sense, and then chooses what to perceive, and determines how, when and where to achieve that perception.”

The concept of interactive perception embraces the concept of active perception because the agent not only explores the environment but also monitors the executed motor actions.

These days there is an apparent increase in the need for intelligent robotic systems that improve society’s living conditions in a wide range of fields, from autonomous transport systems to robotic systems for medical tasks.

This thesis attempts to understand the lessons of Nature applied to robotics, specifically to the saccadic movements underlying active exploration and vision through interactive perception. This work leaves aside the exclusive development of controllers to reproduce the saccadic behaviour in a particular robotic system, studying how to generalise it to any similar robot and giving it practical applications, suggesting new fields of study contributing to Industry 4.0 and improving society.

1.1 Motivation

1.1.1 Bio-inspiration

The study of natural phenomena is one of the main components of scientific advances, e.g. Nicolaus Copernicus was able to deduce from his observations of planetary movements that the Earth was not the centre of the Universe. In turn, Alexander Fleming observed how the growth of a fungus (*Penicillium notatum*) in cultivation eliminated the bacteria present. Therefore, it is reasonable to assume that to create intelligent systems, one must observe entities considered as such. From this perspective, the observation and study of living organisms can provide knowledge that can apply to other fields. Thus, Bio-inspiration is understood as the application of biological models in fields beyond biology. This way of modelling is not really a new

concept, as back in the Renaissance, Leonardo Da Vinci studied the anatomy of birds in an attempt to create flying machines.

In general, the biological models considered refer to materials, morphology or functionality. However, when contemplating extrinsic factors such as the environment, the adaptability of living beings is relevant. Furthermore, the present complex biological systems with their capabilities result from millions of years of evolution. Therefore, Nature has done an improvement work that may serve to develop robotic systems with a particular morphology intended to perform the same functions.

The visuo-oculomotor system is probably the primary mechanism for humans to explore and perceive the world. Considering it has several advantages, among others being one of the most studied systems in neurobiology and medicine, numerous biological models try to explain its functioning. In addition, the visuo-oculomotor system contains all the elements of the perceptual cycle presented by [Uexküll \(1926\)](#): perception of the stimulus, processing of the information and generation of a response, fitting the basic mechanisms of operation of robotic systems.

On the other hand, it is not enough to mimic the morphology or functionality of the visuo-oculomotor system, and it is necessary to consider adaptability to create real intelligent systems. At this point, bio-inspiration also plays an essential role in providing the model inspiring the artificial neural networks that are the mathematical basis for learning and implementing adaptive behaviours and interactive perception in robotic systems.

Biological models can be applied to a robotic system for visual exploration to attempt to be functional, adaptive and morphologically equivalent. In robotics, unlike in the biological world, it is usual to work with individual systems or at best with robots from the same manufacturer. This approach complicates the generalisation and comparison of results. In biology, on the other hand, individuals share morphological, functional and adaptive traits with other individuals (species). Furthermore, biological individuals share this information in a fashion that allows them to replicate common behaviours, albeit with different traits. Such a course of action can also be applied to the field of robotics.

Therefore, this work is based on certain biological models, such as: neural networks, visuo-oculomotor system, perception cycle, gene-phenotype-environment relationship, visual fixation process, and it studies its translation into robotic systems to improve their capabilities through interactive perception.

1.1.2 Industry 4.0

The introduction of collaborative robots in the industry has represented a paradigm shift. It is now no longer necessary to separate the work areas of people and robots, fostering collaboration between both to perform tasks. This change makes a decisive contribution to Industry 4.0. The German government introduced this concept in 2011 to name a set of strategies related to implementing high technology in the industry.

One of the four principles that identify Industry 4.0 is on-site support. Therefore, automated systems should assist humans in decision-making processes or help with complex or risky tasks. In this context, a robot is no longer regarded as a standalone machine, but rather as a networked *Cyber Physical System* (Khaitan and McCalley, 2015) endowed with interoperability, i.e., the ability to connect and communicate with other devices via the Internet (Hermann et al., 2016).

Wan et al. (2016) and Kehoe et al. (2015) introduce the concept of *cloud robotics* so that, rather than attempting to increase the performance and functionality of isolated robotic systems, knowledge is reused through the shared memory of multiple robots.

Projects such as *RoboEarth* (Waibel et al., 2011) collect, store and share data independently of the robotic platform used. Structuring all the data resulting from the learning of a given robot in one environment improves the behaviour of similar robots in the same environment, and contributes to the knowledge sharing that is needed within Industry 4.0.

Once again, this aspect is brilliantly solved by Nature. Every living organism shares information with the living forms that preceded it through its genes. One of the motivations behind this work is to use this model as a source of inspiration to apply it to interconnected robots allowing them to share knowledge or even improve their adaptation to different environments.

1.1.3 Impact on society

Over the last ten years, robotics has been boosted by several advances that have significantly improved its capabilities:

- Improvements in sensors, especially in the field of vision, have enabled robots to perceive their environment more accurately. For example, expanding RGB sensors to RGBD has become a standard. This improvement has let robots apprehend their environment in three dimensions.

- The introduction of collaborative robots, not only in industry but also for service robotics applications, has generated an innumerable set of applications contributing to turn robotics into an element of people's daily lives.
- At the educational level, the appearance of mini-computers such as the Raspberry Pi has facilitated the popularisation of robotics by enabling the creation of low-cost robotic systems, allowing and promoting the learning of programming and the basic principles of robotics.
- Finally, the development of Deep Learning has impacted all areas related to machine learning. Although it remains to be seen whether these techniques will be applicable to robotics in a generic way, it has made it possible to solve some particular problems efficiently.

All these advances have led robotics to be more and more present in our lives, from the vacuum cleaner hoovering the house, to the robotic arm that serves us a beer in a bar; without forgetting that there are some places in the world where autonomous cars are already a reality.

Whatever their functionality, robots need to obtain information from the environment to perform their tasks efficiently. This work contributes to improving this aspect by using biological models to develop algorithms that allow robots to better perceive and adapt to their environment and make them more intelligent. This improvement increases their reliability and further integrates robotic systems into society to solve more complicated tasks.

1.2 Aims and scope

This thesis focuses on extending the exploration and adaptability capabilities of an autonomous robotic system in order to reproduce primate-like saccadic movements for interactive perception.

In order to achieve this aim, several milestones need to be accomplished:

- From the study of the biological model of the human visuo-oculomotor system, propose a model applicable to robotics that allows us to generate saccadic movements adaptively through exploration and autonomous learning.
- Improve the adaptability of robotic systems that generate saccadic movements by using the information generated by other systems that have already learned to operate appropriately in a certain environment.

- Exploit the ability to reproduce saccadic movements to improve certain aspects of interactive perception in robotic systems; particularly, capturing the third dimension.

1.3 Methodology

We elaborate mathematical models in order to reach the proposed aims, based on a series of biological models that allow living beings to solve a series of problems relevant to robotics. They are first examined in simulation in a general fashion and then experimented in a particular way in a real robotic system (figure 1.1):

- First, a problem of relevance to the field of robotics is selected, and a biological solution to the problem is sought. Usually, a biological model describes it. In this work, the following biological models are considered:
 - Saccadic eyes movements. The saccadic movements that humans use to explore the world are a source of inspiration for implementing active vision in a robotic system.
 - The relationship between genotype, phenotype and environment in living beings. When a new living being comes to life, its DNA strands encode all the information that later, conditioned by the environment, will give rise to its behaviour and capabilities. On the other hand, when a robot system is created, all its controllers and properties are adjusted to generate a particular behaviour from scratch without prior knowledge. Hence, there is no reuse of the experiences that other robots performing the same task have previously acquired. Nature's pattern of transmission of information between successive generations can help improve this issue.
 - The execution of complex mechanisms and processes produced in living beings, such as the visual fixation process, increases the information available to the brain to process the surrounding environment. Therefore, transferring these mechanisms to a robotic system should improve its perceptual capabilities.
- Biological systems with billions of years of evolution are so complex that replicating their models becomes an unachievable task. Fortunately, thanks to abstraction, it is possible to simplify the essential parts of these systems and model them mathematically. For this purpose, it is necessary to establish a series of hypotheses allowing these simplified mathematical models to develop.

- Before these models can be implemented in a real robotic system, they are tested in simulation. In some cases, the unavailability of sufficient real robotic systems renders simulation the unique means of testing them. In the simulation, the assumptions for the development of models are tested to exhibit similar behaviours to biological models.
- If the hypotheses developed in the design of the bio-inspired models are verified in simulation, the implementation in a real robotic system takes place, verifying that the robot exhibits the behaviours obtained in the simulation.

The numerical results obtained —either in the simulations or in the real robotic systems used in this work— are compared either with other methods that allow their estimation using the existing literature, or with the simulation of certain parameters that allow us to evaluate their performance.

The same methodology has been applied to develop each proposed model, changing how the results are compared or the robotic systems used. In some specific cases, a robotic system has been designed and implemented specifically to verify the results.

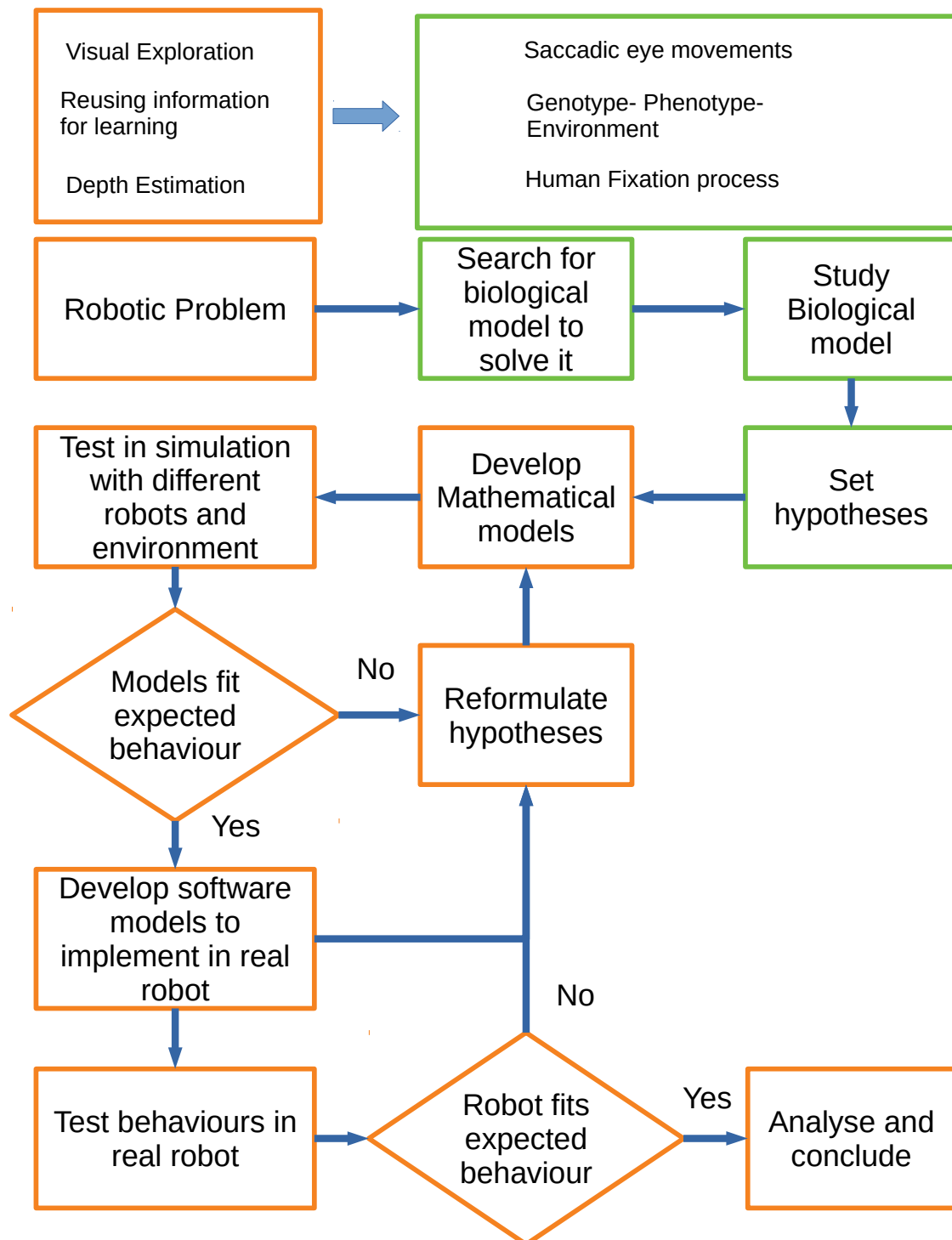


Figure 1.1: Outline of the methodology followed in this research work.

1.4 Outline

We structured this thesis in several chapters. Chapters 2 and 3 describe the biological model for saccadic generation and the mathematical models to deal with the adaptation problem respectively. The remaining chapters describe the work performed to develop and adapt various biological models, suggesting solutions to various problems in the field of robotics. In each chapter we propose some partial objectives and obtain results to solve the posed problems. The first appendix introduces an application derived from the developed models using deep learning techniques to solve the problem of depth estimation. The second appendix describes the software architecture we specifically developed to implement all this work in the various robotic systems.

- **Chapter 2. Saccadic eye movements.** The biology of saccades is presented here, since they are the core around which all our work revolves, providing it with a strong bio-inspiration. There exist models describing how human beings generate saccadic movements to explore the world actively. This chapter briefly attempts to detail how our brains generate saccadic movements as a basis for the models that will be described in subsequent chapters. Similarly, introducing the morphology of the oculomotor system here will allow us to discuss later on the different ways in which robotic modelling of the oculomotor system has been attempted.
- **Chapter 3. Adaptive learning in neural networks.** The environment is a factor that has been present throughout this work. The ability of living beings to adapt to the environment is one of the most desired qualities in robotics. A set of mathematical tools allows, in a certain way, to represent the internal model of robotic systems in an adaptive fashion, so that they can respond to the changes occurring in the environment. These mathematical models are also bio-inspired, specifically in the neural networks conforming to the nervous systems of living beings. The concept of adaptive learning refers to the ability of the system to learn from changes in the environment in order to improve its performance. This chapter introduces these artificial neural network models, that allow for some adaptability, along with the mathematics required to implement them.
- **Chapter 4. Robotic systems for visual exploration by means of saccadic movements.** This chapter describes the main types of morphology for robotic heads that allow them to imitate the movements generated by the human oculomotor system described in chapter 2. A Helmholtz setup morphology is selected for implementing the robotic system to execute saccades, and the morphological parameters describing it are detailed. Afterwards, we study how integrating the

two visual signals is useful for generating more accurate saccadic movements. Finally, two controller architectures for the execution of these movements are proposed. These are evaluated and compared with other types of models developed in robotics to execute these exploratory movements.

- **Chapter 5. Predicting the internal model of a robotic system from its morphology.** Once we have several methods described in the previous chapter for the execution of saccadic movements based on adaptive learning methods, this chapter proposes to reuse, in a certain way, the information of how a robotic system has learned to adapt in a specific environment to improve the learning of other similar robotic systems. This relationship between the specific properties of a robotic system —partially constituting its morphology— and the parameters allowing it to adapt itself to the environment, led us to propose an artificial model based on the triangle genotype, phenotype, environment, as described for biological organisms. Thus, the morphology of a robotic system and its internal model can be *transmitted* from one robot to another in the same way it occurs in a biological system.
- **Chapter 6. Saccadic behaviour for depth estimation.** The saccadic movements described in chapter 2, and modelled in chapter 4 for visual exploration, have implications for other biological behaviours such as the visual fixation process. We exploit the involvement of saccades in this fixation process to acquire depth information from the environment. This chapter also briefly describes the robotic system we have specifically developed to test the proposed algorithms.

Chapter 2

Saccadic eye movements

Exploration is wired into our brains. If we can see the horizon, we want to know what's beyond.

Aldrin and Abraham (2009)

2.1 Introduction

There is no doubt that the visual stream plays an important role in the way humans explore the world around them. Nature has endowed us with a system that generates a signal from electromagnetic radiation that —after been processed in a convenient way —produces a response in the organisms for their interaction with the surroundings.

As (Uexküll, 1926) suggests, there is a relationship between the perceived signal (stimulus), the actions carried out by the organisms and a part of the world called the environment. The meaning of a stimulus for a particular organism can be inferred from the behaviour it exhibits as a response.

These elements can be represented in a diagram (figure 2.1) where arrows symbolize the flux of information between them. The senses capture data from the environment; however, that data can be altered depending on the inner conditions of the organism. Actions are executed by the organism through its effectors after information

processing in the central nervous system—or more directly in the case of reflexes. These actions somehow modify the environment, with possibly a change in the signal perceived by the sensors as a result. These actions need not necessarily result in an actual physical change: any action that modifies the perceived signal coming from the environment—e.g. by moving the sensory organs—can also be considered as a change in the environment from the point of view of the organism.

The concept of interactive perception (Bohg et al., 2017) turns the cycle proposed in figure 2.1 into an interactive process so that, to some extent, the environment guides the cognitive process.

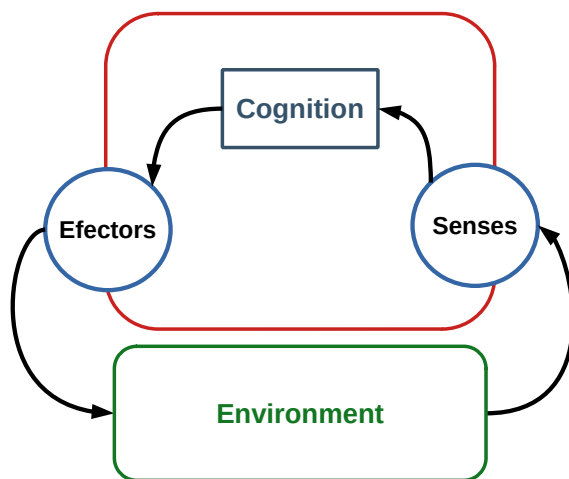


Figure 2.1: Perception and action cycle

Visual perception in humans fits this cycle perfectly. In addition, the physiological and functional model of the oculomotor system has been described almost in its entirety (Kandel et al., 2000). This knowledge about the functioning of the primate oculomotor system, and its adaptability and precision capabilities, has led to the translation of biological models into robotic systems. This adaptation of models has been carried out to solve complex problems in the field of robotics. This chapter aims to describe the biological visual system that allows the execution of this cycle for humans. In order to obtain models that could be transferred to robot-

ics, not only the physiological and anatomical descriptions that can be found in manuals for medical professionals are necessary, but also those proposed by neuroscience (Squire et al., 2012) (Rolls and Deco, 2001), experimental psychology (Wixted et al., 2018), and computer science (Mallot et al., 2000).

The oculomotor system is a part of the primates' central nervous system that is in charge of eye movements. It is composed of pathways that connect various regions of the cerebellum, brain stem, and eye nucleus, using multisynaptic joints.

A number of systems in robotics have taken inspiration for their design and control in both the visual oculomotor system, and the proposed cycle of perception and action (Murray et al., 1992; Sharkey et al., 1993; Truong et al., 2000).

Oculomotor control is also an important research area in computational neuros-

science since the primate oculomotor system can generate a rich set of behaviours in spite of been relatively simple (Shibata et al., 2001). Furthermore, applying the knowledge acquired in this area to the field of humanoid robotics allows the validation of models developed by neuroscience, providing practical and straightforward solutions to highly complex problems (Rucci et al., 2007).

If the visuo-oculomotor system is isolated from the rest of a complex human being, its behaviour can be characterized by how it acts in the presence of a stimulus, and conditioned by the environment and by the body itself.

The visual oculomotor system typically generates a series of eye movements in response to different stimuli, or as a functional part of cognitive processes. As it is well known, the fovea is the area of the retina where the receptor concentration is highest and, consequently, it yields the highest resolution. Therefore, depending on whether the behaviour of the eye is to maintain the projection of the stimulus within the fovea or to bring it onto the fovea, the eye movements can be classified into two large groups:

1. *Movements that bring a stimulus of interest onto the fovea:*

- **Saccadic movements.** These are swift eye movements that immediately change the direction of gaze. They bring the image of interest to the fovea in the shortest possible time. They are conjugated movements, i.e. in which both eyes participate in a coordinated manner. This kind of movement can reach speeds of 20 – 600 degrees per second. The saccades have two properties that make them singular compared with the other eye movements: they can be voluntary and suppressed.
- **Quick phase of Nystagmus.** It resembles a saccade. Its purpose is to reset the oculomotor system during a long rotation. In a way, it prevents the eye from being blocked in an extreme position.
- **Vergence movements.** They are in charge of bringing the image of interest to both foveas, mediating stereoscopic vision. They are slower than saccades..

2. *Movements that maintain the images of interest on the fovea:*

- **Vestibular movements.** They maintain a fixed image on the retina during short, rapid rotations of the head.
- **Optokinetic movements.** They maintain a stable image when objects move in the periphery while the head is stationary (Cohen, 2011). The combination of vestibular and optokinetic movements allows us to see and move

at the same time. In practice, although they depend on different neuronal systems, both movements are so closely related to each other, that it could be considered as a vestibular-optokinetic system.

- **Smooth pursuit movement.** It maintains a mobile object on the fovea. It is a tracking movement and can reach 20 – 30 degrees per second. When the speed of the object is too high or changes rapidly, saccadic movements intervene.
- **Fixational eye movements.** When the axis of vision of the two eyes are directed to one point, and this point of interest is in the centre of both foveas, several movements are produced to maintain this situation. The most important are: the microsaccades, the eye's tremor and drift.

Each movement has a particular function and its neural substrate. Among all these types of eye movements, this work focuses on saccadic movements. Saccades allow us to explore the world and focus our interest on a particular voluntary or involuntary stimulus. Exploring the world involves paying attention to some things and disregarding others, the saccades being the basis of such exploration. Understanding the mechanisms governing the generation of these movements allows us to design artificial systems imitating these behaviours.

This chapter is developed following the idea expressed in figure 2.1. First, the eyes —considered as the effectors of the visual system— are briefly described in section 2.3. Second, the concept of sensor applied to the human visual system cannot be circumscribed to the retina only because, as described in section 2.4, the image captured by the retina is projected to different brain areas which process it and generate the signals that trigger the resulting cognitive processes. Finally, the generation of a saccade involves many brain areas that do not act sequentially but instead in a parallel fashion; they are briefly described in section 2.5.

2.2 Objectives

This work is developed in the context of bio-inspiration; specifically, a class of eye movements involving the exploration, perception and action on the environment, as is the case of saccadic movements. For this reason, it is appropriate to briefly describe the biological processes involved in the execution of these movements. They have already been modelled from the point of view of medicine, neuroscience and cognitive psychology.

2.3 Eye muscles: Visual system effectors

The effectors of the visual-oculomotor human system are the eye muscles. They can be grouped into three pairs for each eye. Unlike the rest of the body, the eye muscles are not segmented and, therefore, they are composed of whole fibres. This property facilitates the understanding and analysis of eye movements.

In a simplified manner, they can execute eye rotation and translation movements as shown in the diagram in figure 2.2. The eye muscles are controlled by three of the

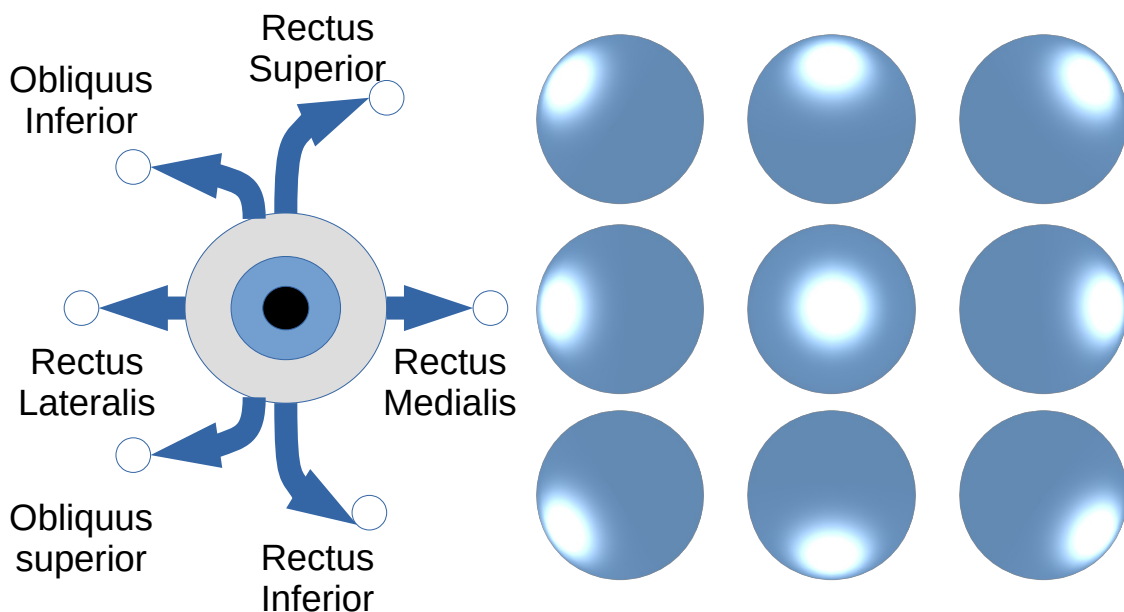


Figure 2.2: Schema of eye movements

twelve cranial nerves. The signals travelling through these nerves come from various areas of the brainstem. The discharge rate in neurons of that final common path is proportional to the angular deviation of the eye. The movement is executed by a combination of contraction and relaxation of each pair of muscles; therefore, it is far from pure translation and rotation. Although the human eye is inside a spherical cavity called a Tenon capsule, the centre of the eye's rotation can be displaced by 1 to 2 mm with each movement. Therefore, the centre of rotation of the eye sphere is usually between 13-15 mm behind the cornea. However, Dutch ophthalmologist Franciscus Cornelius Donders in 1846 established the so-called Donders' law. He stated that the eye always assumes the same position for any head position and any direction of gaze, resulting in the fixation point being focused on the fovea.

In addition, the mathematician Johann Benedict Listing described how the eye moves, the so-called Listing's law. The roll angle of the eye is always the same as if it

had turned the shortest possible way from all other positions into the current position. This fact suggests that the eye should execute compensatory movements on the roll angle. These are not a consequence of the geometry or muscles of the eye; on the contrary, it has a neural background. These two laws applied to a biological system, such as the human eyes, are important to implement eye movements in a robotic system since they define a set of design limitations.

It is noticeable that the eyes can turn in the same direction or in opposite directions. In the first case we speak of version movements. On the other hand, when the direction of the eyes is opposite, these movements are called vergence. The German physiologist Ewald Hering in the late nineteenth century, postulated his law in reference to the direction in which the eyes move, stating the conjugacy of saccadic eye movement in stereoptic animals —i.e. both eyes must move symmetrically by equal amounts— regardless of physiological and anatomical differences between the two eyes. He thought this property was innate, contradicting contemporaries like Helmholtz, who believed this coordination was learned. Today it seems that both of them were right: our eyes are yoked together at birth, but the yoke can be modified depending on the environment.

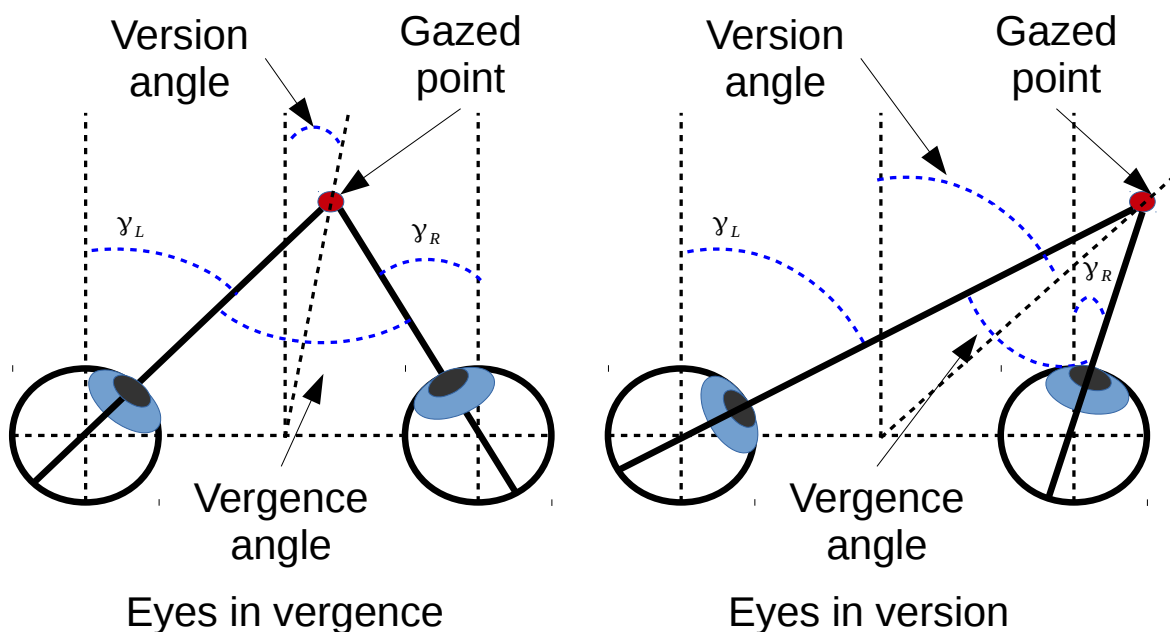


Figure 2.3: Schema of the eye vergence and version movements

These movements can be represented on a plane, giving them an angular interpretation, as it can be seen in figure 2.3 where γ_L and γ_R are the angles between the direction of each eye and the perpendicular line to the segment joining them. The version angle is the mean of both angles and corresponds geometrically to the angle

shown in figure 2.3.

$$\text{version} = \frac{\gamma_R + \gamma_L}{2} \quad (2.1)$$

On the other hand, vergence is defined as the difference between both angles:

$$\text{vergence} = \gamma_L - \gamma_R \quad (2.2)$$

From the point of view of engineering, the motor system of the eyes has a series of properties that make difficult their artificial replication with current materials and technologies:

- Eyeballs have low inertia.
- Muscle drive is reliable.
- The entire motor system has very low friction and therefore low energy dissipation.
- The oculomotor system has a high power/size ratio.
- It is a system that can reach high accelerations.
- The control mechanisms are far from being linear.

2.4 Retina and visual cortex: Visual sense

Unlike the previous case, where the eye muscles are directly assimilated to the effectors, qualifying the eyeball and the retina as the sensory system is a simplification, which is not entirely accurate. Human beings do not form the images in the retina but in particular areas of the brain. So these areas can also be considered as part of the visual sensory system.

The eye in primates is the gateway to the signal from the environment triggering the stimuli, and different parts of the visual system process them. Therefore, the eye is an adaptive optical mechanism that can be modelled. In 1911, Gullstrand presented his scheme of the human eye in "Einführung in die Methoden der Dioptrik des Auges des Menschen". Gullstrand won the Nobel Prize in 1911 for this work. In this way, given a point source of light and using the geometric interpretation of the proposed ocular model (Vojniković and Tamajo, 2013), it is possible to determine the position of the projection of that point on the retina.

In primates, the vision system perceives the observed scene through 2D projections on the left and right retinas. The retinal ganglion cells are connected to the lateral

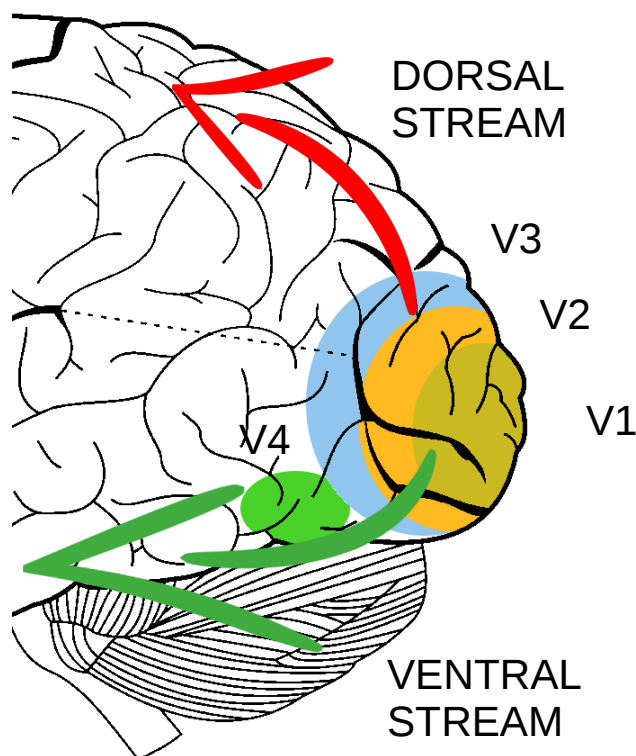


Figure 2.4: Areas of the brain involved in the sensory part of the visuo-oculomotor system

geniculate nucleus in the thalamus, which projects directly to the primary visual cortex (V1).

This area is characterized by simple and complex cells. Both types of cells have small receptive fields and are sensitive to monocular and binocular visual stimuli, such as oriented edges, moving bars or gratings, and binocular disparities (Ohzawa et al., 1990; Prince et al., 2002).

From a practical point of view, V1 can be seen as the substrate that encodes the basic binocular information provided by the retinas into a feature-based space. Downstream from V1, visual processing splits into two parallel streams.

The ventral stream performs object recognition and courses through visual cortical areas V1, V2, V4 and IT. Each of these areas is sensitive to specific features that get increasingly complex and invariant against affine transformations. While the ventral stream detects target objects, the dorsal stream estimates their spatial location and their size. The dorsal stream is also in charge of planning eye movements such as vergence and saccades, computing the pertinent sensorimotor transformations.

Although the processing stream is divided in two, both signals are somehow integrated into specific brain areas to generate motor movements. These two cues are produced by the excitation of the retinal cells that cause the activation of the visual cortex.

In addition, these two streams are used by other brain areas to generate cognitive and non-cognitive responses (Chinellato and del Pobil, 2016).

2.5 Saccades: Visual system behaviour

Saccades are fast, ballistic movements that are used to bring a visual stimulus to the centre of the field of view (named fovea in primates) and therefore, eye version and vergence are both changed (Enright, 1984). The movement can be as fast as 900 deg/s and the control of its execution is not altered by visual perception. Due to the internal model of the visual-oculomotor system, the movement of the eye is accurate (Chen-Harris et al., 2008). This internal model shows certain plasticity, as demonstrated by its ability to adapt to small changes in the oculomotor system (Lappe, 2008). While, in general, the target can be provided with both auditory and visual stimuli, herein the visual ones is only referred to in this work. From a behavioural point of view, once a target is detected, the oculomotor system triggers an eye movement to bring it to the centre of the visual field. The target of a saccade emerges from the interconnectivity between several cortical areas, such as the superior colliculus (Munoz and Wurtz, 1995), the basal ganglia, the posterior parietal cortex, the frontal eye field (FEF), the cerebellum and the brainstem.

Multiple models (Girard and Berthoz, 2005) have been developed to explain the flow of information and the pathways through the different brain areas. From a modelling point of view, sequencing the different events and processes occurring in the brain areas involved is the most straightforward fashion to interpret the experimental results. However, the neural processes responsible for target selection and execution of the saccade occur concurrently in an interconnected network throughout the brain from front to back and top to bottom (Chalupa and Werner, 2004) (Chapter 92). In figure 2.5, a schema with the central relationships between different areas of the brain involved in the saccade generation is presented.

There is an area playing a central role in saccadic triggering: the superior colliculus (SC). It is one of the brain regions that has experienced the most significant changes throughout evolution. This structure performs an initial visual analysis and generates response commands in the simplest animals such as fish or amphibians. In these species, the motor commands generated by the SC not only involve eye movements, which are a fraction of the total number of actions performed by the SC. With the increase in size of the cerebral cortex as a result of evolution in primates, some of the functions of the SC have been extended to new cortical areas. In monkeys and humans, the SC is relatively small compared to other zones but is still a significant structure. It

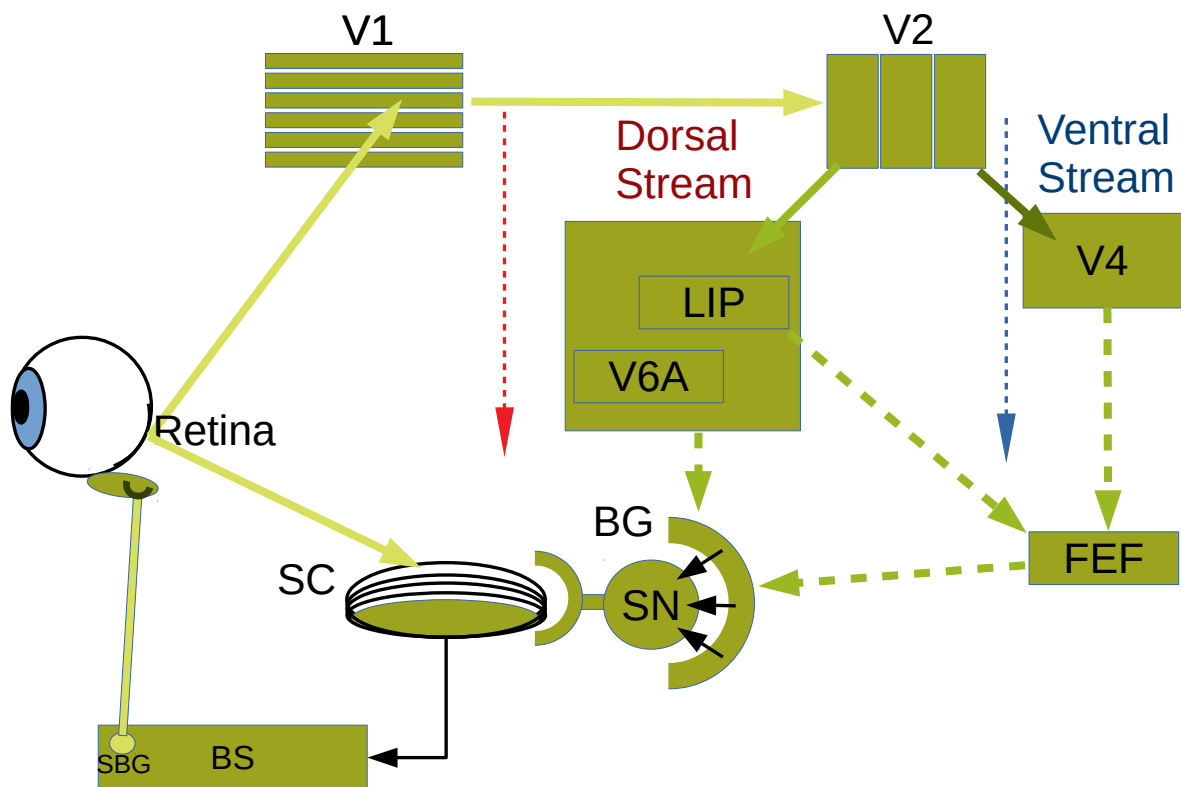


Figure 2.5: Schema of the main brain areas involved in saccadic behaviour

has been suggested that this area is responsible for involuntary saccades in primates (Bell and Munoz, 2008).

Signals from the retina and the visual cortex are received by the superior colliculus (SC). This region has a layered structure. A retinotopic map is located in the upper layers of neurons in this region. Unlike those in the V1 region, these cells are sensitive to small stimuli but are invariant to other characteristics such as shape and colour. The activation in this retinotopic area of the SC arises where there are potential targets to execute a saccade. An exciting feature of the layered structure of the SC is that there is little interaction between them, but each one has a remarkable relationship with other areas of the brain. The entry signals come from the retina directly to the most superficial layer of the SC through neurons called W neurons. On the other hand, from the fifth layer of neurons of the visual cortex V1, some neuronal structures project the visual signal onto the intermediate layers of the SC.

The deep layers of neurons in the SC express sensitivity to sensory stimuli from several modalities (e.g., vision, audition, somatosensation), and another overlapping group of neurons discharges a vigorous premotor burst during the orienting movement (Gandhi and Katnani, 2011). This signal is processed by the cerebellum and the saccadic burst generators (SBG) in the brainstem (BS). The cerebellum has an adaptive

modulation function since it is where the internal models of the motor apparatus are stored (Wolpert et al., 1998).

A saccade is generated by computing the direction and size of the saccadic vector needed to null the retinal error between the present and intended eye position (Moschovakis, 1996; Poletti et al., 2020). This procedure is the basic coding principle in the SC (Gandhi and Katnani, 2011). The execution of a saccade by the SC is produced with a slight error when the saccade vector is short. As the size of the saccade vector increases, it is necessary to perform a second correction saccade.

During maintained fixation, very small saccades (microsaccades) are generated with variable frequency and amplitude. Even though microsaccades and saccades exhibit similar motor characteristics and share a common neural substrate (Ko et al., 2010), there has been a long controversy over the visual functions of these movements. Recent studies show that these microsaccades are precisely directed and play a fundamental role in enhancing visual acuity (Intoy and Rucci, 2020)

Attending to studies about the generation of microsaccades indicate to be similar to the saccades at the level of Superior Colliculus and downstream of it, in pre-motor and motor brainstem circuits (Hafed, 2011; Thier et al., 2015). A microsaccade involves significant suppressive influences on neural activity in the Superior Colliculus. This is complementary to the fact that these eye movements, along with larger saccades and movements of the head and body, contribute to retinal image motion and, therefore, can help prevent fading when no other sources of image motion exist (Hafed and Krauzlis, 2010).

The cue that induces the generation of saccadic movement produced in the SC is conditioned by activity in other brain areas. These areas process simultaneously the signal received in the retina following the circuits described in section 2.4.

Signals from both the ventral and dorsal streams converge in the frontal field eye area FEF. The group of neurons in this region contains a retinotopically organized map of visual, visuo-movement and movement cells (Schall, 1991). While the visual cells respond to the onset of visual stimuli, the movement cells respond to the onset of a saccade and thereby encode the expected landing position of the eyes after a saccade. The FEF is bidirectionally connected to area V4 (ventral stream)(Hamker, 2005b; Szczepanski and Saalmann, 2013) and to the lateral intraparietal area (LIP) in the dorsal stream. From these areas, it receives information regarding visual features of the target (V4) and its spatial location (parietal regions as V6A (Buneo et al., 2002)). These interactions are not limited to eye movements but to reentrant processing in general, e.g. to the deployment of visual attention (Hamker, 2005a). In this way, the

visual system maintains a consistent object representation in all cortical areas (frontal eye field, ventral and dorsal streams).

The dorsal stream is also in charge of computing the sensorimotor transformations required to control eye movements. These transformations are likely to be performed through the gain field effect of neurons in the posterior parietal cortex (PPC). In particular, neurons in the V6A area have been found to contextually encode different representations of the target position, allowing for easy reference frame transformations. In addition, neurons in V6A with retinotopically organized fields are modulated by gaze direction to encode spatial positions (Marzocchi et al., 2008; Bosco et al., 2010).

The processed cues from both the dorsal and ventral streams generate signals that reach several neural structures located in the area of Basal Ganglia BG. This region, in turn, controls a group of neurons in the area known as Substantia Nigra SN which is one of the primary circuits generating an inhibitory signal in the SC. In this way, when the visuo-oculomotor system is used to explore the environment, hundreds of stimuli appear on the retina. However, thanks to these inhibitory circuits, only one of these stimuli reaches the lower layers of the SC and generates a saccadic movement.

2.6 Conclusion

Beyond the sea of acronyms, this chapter attempts to describe which parts of the human brain and body are involved in performing a saccadic movement that, as indicated, allows us to direct our visual attention. In summary, for perceiving the world around us, not only the eyes are used, but there are many areas of the brain involved (V1, V2, V4, LIP, IT,...). After the description of these different cerebral regions, we can conclude that visual signal processing is done in a parallel and redundant fashion; in fact, this is a general principle in biological computation. As it can be seen throughout this work, bioinspiration is the basis of our goal: developing complex behaviours in a robot for the exploration of the surrounding environment.

2.7 Publications supporting this chapter

- Antonelli, M., Gibaldi, A., Beuth, F., Duran, A.J., Canessa, A., Chessa, M., Solari, F., del Pobil, A.P., Hamker, F., Chinellato, E., Sabatini, S.P., 2014, "A hierarchical system for a distributed representation of the peripersonal space of a humanoid robot", *IEEE Transactions on Autonomous Mental Development*, Vol. 6, No. 4, pp. 259-273. DOI: 10.1109/TAMD.2014.2332875.

Chapter 3

Adaptive learning in neural networks

Learning denotes changes in the system that are adaptive in the sense that they enable the system to do the same task or tasks drawn from a population of similar tasks more effectively the next time.

Herbert A. Simon (1983)

3.1 Introduction

In a real scenario, where a robot agent has to develop its activity, the interchange of data with the environment constantly varies over time. Machine learning techniques play a central role in this landscape ([Esposito et al., 2004](#)). In a way, the robotic system should predict and model its surroundings to adapt to these changes. In this chapter, a set of machine learning tools are presented to deal with the adaptation problem.

Classical classification of machine learning techniques regarding the type of problem to be solved is shown in figure [3.1](#). However, the issue is not so much the problem that these tools want to solve, but how the problem is solved.

One of the main objectives of this work is to reproduce the saccade movement of the eyes in primates in a robot head based on detecting a visual stimulus and generating the ocular movement to centre the stimulus on a retinotopic image. This movement

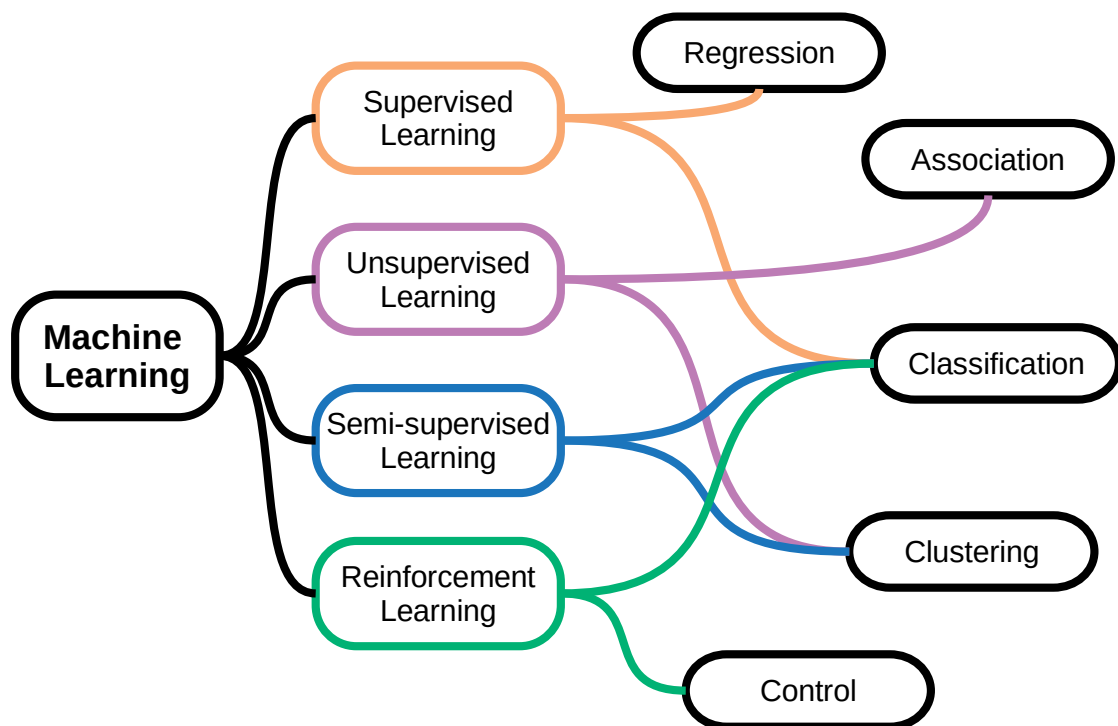


Figure 3.1: Schema of the machine learning types depending on the kind of problem to be solved.

involves a transformation from visual coordinate space to motor coordinate space. Therefore, learning this visuo-oculomotor transformation can be treated as a function approximation problem. However, in order to strengthen the biological plausibility, the saccade control is learned through the interaction with the surrounding space (Chinellato et al., 2011). Consequently, among all the techniques proposed in the classification in figure 3.1, this work is focused on supervised learning for the resolution of regression problems. In this category, there are many different techniques, among other: decision trees, support vector regression, linear and bridge regression, ensemble methods, gaussian process and neural networks.

As a consequence of the aforementioned biological inspiration, the choice of artificial neural networks seems a natural consequence. The primary foundation of these particular machine learning tools was the modelling and abstraction of their biological counterparts. However, the model has currently evolved beyond biological behaviour to focus more on its practical application.

3.1.1 Definition of artificial neural networks

When one wants to give a precise and simple definition of what a neural network consists of, it is simply not possible to find it because the definitions given in the literature are made within a given context. Here are some examples:

- *“An artificial neural network is an information-processing system that has certain performance characteristics in common with biological neural networks.”*(Fausett, 1994)
- *“Artificial neural networks are parallel computational models, comprising densely interconnected adaptive processing units.”*(Hassoun, 1995)
- *“Neural networks are mathematical models developed in an attempt to emulate human neural systems.”* (Chen and Dong, 1998)
- *“A neural network is an interconnected assembly of simple processing elements, units or nodes, whose functionality is loosely based on the animal neuron. The processing ability of the network is stored in the interunit connection strengths, or weights, obtained by the process of adaptation to or learning from a set of training patterns.”* (Gurney, 1997)
- *“A neural network is a massively parallel distributed processor made up of simple processing units that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects: Knowledge is acquired by the network from its environment through a learning process; Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.”* (Haykin et al., 2009)
- *“An artificial neuron is a model of a biological neuron. Each artificial neuron receives signals from the environment, or other artificial neurons, gathers these signals, and when fired, transmits a signal to all connected artificial neurons. An artificial neural network is a layered network of artificial neurons.”* (Eberhart and Shi, 2011)

The issue shared by almost all of the above definitions is biological inspiration. However, after careful comparison, the relevance of other concepts can be observed: parallel and distributed computation, adaptation and learning process.

3.1.2 Parallel and distributed computation in biology

The computational load in the brains of living beings is distributed among the units that form their nervous system called neurons. Therefore, a processing unit of an artificial neural network models its biological counterpart. Neuronal computing takes place in the cell body of the neuron, where the nucleus is located. The diffusion of ions across the cell membrane is the motor causing the transport of charge along the neuron. The neuron membrane has ion channels allowing the charge interchange between the cell and its environment. In addition, neurotransmitters regulate this flux. Thus, the charge movements produce a difference of electric potential between the inside and outside of the neuron. When this potential difference exceeds a certain threshold, at this point, a phenomenon called depolarization occurs, and an electrical signal is generated (action potential).

Neuronal activity flows from one neuron to another in terms of electrical impulses that travel along the neuronal axon through voltage-dependent processes of ion exchange along the axon and diffusion of neurotransmitter molecules across the cell membrane within the synaptic space between neurons. This space is called the synaptic cleft. Signal transmission in that space depends on the neurotransmitters that the presynaptic neuron has released when the action potential has been triggered and the pre-existing chemicals in the synaptic cleft. Modelling these electrochemical processes is highly complex when a low level of abstraction and multiple neurons are considered. A simple example of a biological neural network is shown in figure 3.2.

To define an abstract representation of a biological neural network from a computational point of view, each neuron is considered as a node in a directed graph. These nodes have particular properties in this schema that depend on their function within the network.

The input nodes symbolize the presynaptic neurons. The signals prior to these neurons triggered by their depolarization are not considered, what matters is the numerical value representing the action potential (x_1, x_2, x_3 in figure 3.2). A single number or weight embodies how the synapse process modified the signal from the presynaptic neurons ($w_{1,1}, w_{2,1}, w_{3,1}$ in figure 3.2). Next, the integration of all these modified signals in the dendrites determines the activation of a neuron. When it exceeds a certain threshold, it is depolarized, transmitting the signal forward in the network. A non-linear function of the integration of the input signals (ϕ_1 in figure 3.2) represents the activation of the neuron. In a directed graph, this kind of processing unit is symbolized by an activation node. The signal generated in the processing units is transmitted forward in the network and modified by the synapses of the subsequent neurons ($W_{1,1}, W_{1,2}$ in

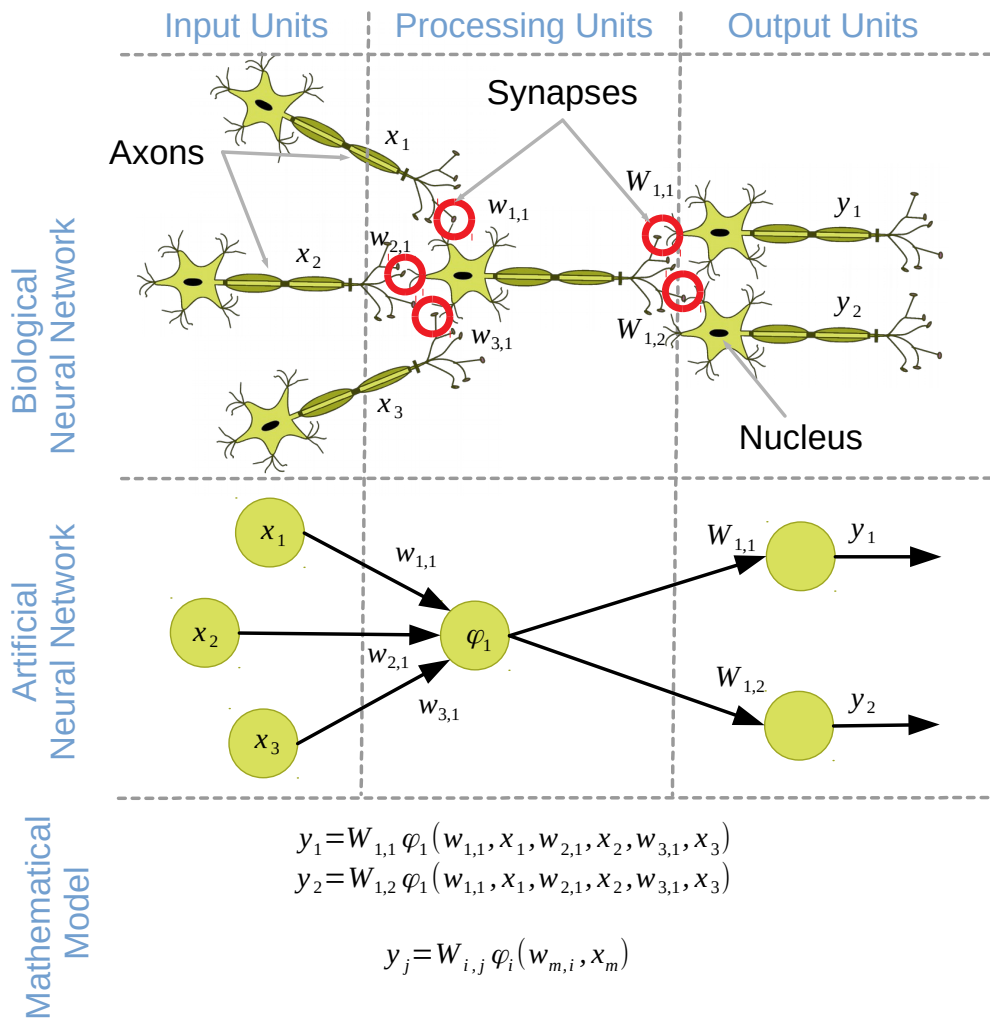


Figure 3.2: Biological concepts of synapse and information transmission are abstracted from a biological neural network, to generate the model of an artificial neural network and a mathematical model that describes it.

figure 3.2)).

This highly abstract model is probably very distant from the actual biological schema because it merely attempts to imitate its functional behaviour. Even so, it has demonstrated its ability to solve countless problems which otherwise would not have been possible to address.

3.1.3 Feedforward neural networks

In biology, two key elements of nerve tissue contribute to signal processing and the generation of responses to a dynamic environment: the neural connections and the synapses established between them. These elements can also be found in the artificial

neural network model, which also defines their behaviour. One factor is the network topology, and quite another is the algorithm used to adapt its weights.

The concept of a layer of neurons is often used to represent the topology of a neural network. Each of these layers groups a set of neurons with the same processing function, e.g. the input layer gathers the neurons which generate a signal but do not receive signals from any other neuron. Similarly, the output layer consists of the neurons that do not transmit signals to other neurons, but only receive them. The hidden layers are formed by a set of neurons that share the same activation function. The term “hidden” refers to the fact that this layer is not directly connected to either the input or the output of the neural network. Instead, these neurons behave as processing units that modify their input signal by generating an output signal.

The simplest network is composed of an input layer and an output layer. The connections go from the inputs to the outputs but not vice versa. This kind of neural network is called feedforward single-layer. The left-hand side of figure 3.3 shows the topology of a single-layer neural network.

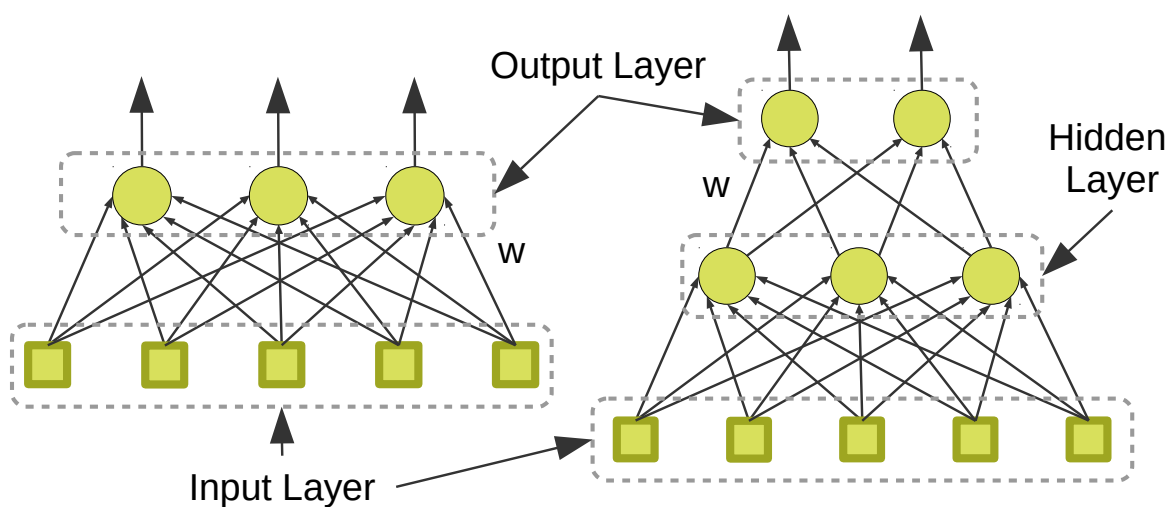


Figure 3.3: Different schematic representations of a single-layer network.

A neural network is considered as multilayer feedforward (MFN) when one or more hidden layers are present. However, this definition can lead to confusion with some graphic representations of a neural network. For example, in the network on the right of figure 3.3. In this case, the defined hidden layer has a special connection to the input layer. It can be seen that the synaptic weights between the input layer and the hidden layer are not indicated. The hidden layer acts as a predefined spatial transformation of the inputs taking into account the value of all inputs. The arrows between the input neurons and the hidden layer units indicate that these transformations are dependent

on all input units. This layer has no actual activation functions. The synaptic weights are set between the hidden layer and the units of the output layer. This scheme is still a single-layer neural network.

This work will mainly focus on this second type of network (right-hand of figure 3.3), although MFN will also be used on some occasions.

3.1.4 Adaptation and learning process

In living beings with a central nervous system, the role of the brain is to process information from the environment and generate behaviour in response. However, the environment is always dynamic, and therefore the brain must change how it processes the received information to adapt to this dynamism. Biology and neuroscience suggest that the reason for this ability to adapt is the plasticity of the synaptic processes occurring in the brain. At least two mechanisms are available that enable that adaptation: (i) Rebuilding the neural network connections by destroying and generating synapses. (ii) Strengthening and weakening previous synaptic connections and altering the information fluxes that circulate through the neural networks in the brain.

In artificial neural networks, the topology refers to the combination of nodes and connections that are established. Therefore, changing it by adding or removing nodes and connections depending on the inputs and outputs of the network is a form of abstraction for the implementation of neural plasticity. This problem has been addressed by pruning and growing algorithms from an initial model of the network. Nowadays, pruning techniques are becoming more popular because of their application to problems addressed by deep learning (Molchanov et al., 2019; Anwar et al., 2017; Zhu and Gupta, 2018). Initially, an adapted topology for a specific problem is formulated, and then through pruning techniques, the model size is reduced to optimise the learning process. This reduction is generally based on eliminating nodes with lower synaptic weights or which contribute less to the network performance. On the other hand, growing methods are less widespread and are often based on combinations with heuristics, genetic algorithms or reinforcement learning to add new nodes to the network.

The most widely used approach to plasticity in artificial neural networks is not based on changing the topology but on varying the synaptic weights according to the expected inputs and outputs. In the proposed abstraction, the synaptic weights are represented by a value. The critical question is how these values should be modified so that the network works as expected and adapts to the data coming from the environment.

Two approaches can be considered to manage this problem: the classical batch

approach and incremental techniques. The former option is currently the most widely used due to the rise of deep learning techniques. However, it entails the need to adapt the whole model, considering the current sample and previous ones. Moreover, this method can cause inefficiency in the adaptation process when the information gained with the new sample is irrelevant in regard to the information stored in the proposed model.

The model obtained through the learning process should show two main characteristics apparently in conflict: on the one hand, stability in order to keep significant knowledge and, on the other hand, plasticity to update the model when new relevant information is available (Pérez-Sánchez et al., 2018). The ideal scenario is that the environment guides the learning of the system by sequentially providing relevant data so that the most recent signals are more important than the oldest ones (Kubat et al., 2004).

This chapter is centred on describing several approaches to this incremental learning. Depending on the field of knowledge considered, the name that defines this concept may be different: on-line learning, adaptation, incremental or sequential models are terms used to describe this issue. In this way, weight learning in neural networks will be focused on estimating the parameters of a mathematical model that describes the behaviour of the network.

Our aim is not to present a treatise on mathematics, but rather, to enumerate a series of concepts that allow the introduction of the mathematical nomenclature that will be used throughout this work. In addition, appendix C briefly summarises the mathematical foundations and concepts that are prerequisites for a better understanding of the techniques developed in this chapter.

3.2 Objectives

One of the proposed objectives is to learn the visuo-oculomotor transformations for the execution of saccadic movements. After millions of years of evolution, living beings can provide clues on how to deal with this problem so that a system can adapt to changes in its environment in an efficient way. Based on biological systems, one of the tools available within Machine Learning are artificial neural networks. Due to the nature of the problem to be addressed, supervised learning algorithms are presented, where the *teacher* is the environment that generates the signals to adapt the synaptic weights incrementally. In this chapter, a series of neural network topologies (section 3.3 and section 3.4) and some adaptive algorithms (section 3.5) are presented for further development of the objectives of this work.

3.3 Supervised linear learning

Linear models can approximate many processes in real environments. Even when the real model is complex, local linear approximations can be made. One of the main advantages of linearizing problems is that they can be solved analytically. The most straightforward neural network that can be considered is formed by a single layer corresponding to the output neurons. In figure 3.4, a scheme with the nomenclature and topology of this kind of network is shown. As discussed in the introduction (section 3.1), this contribution focuses on the use of supervised learning. Two different states can be distinguished in this type of learning: training state and prediction state.

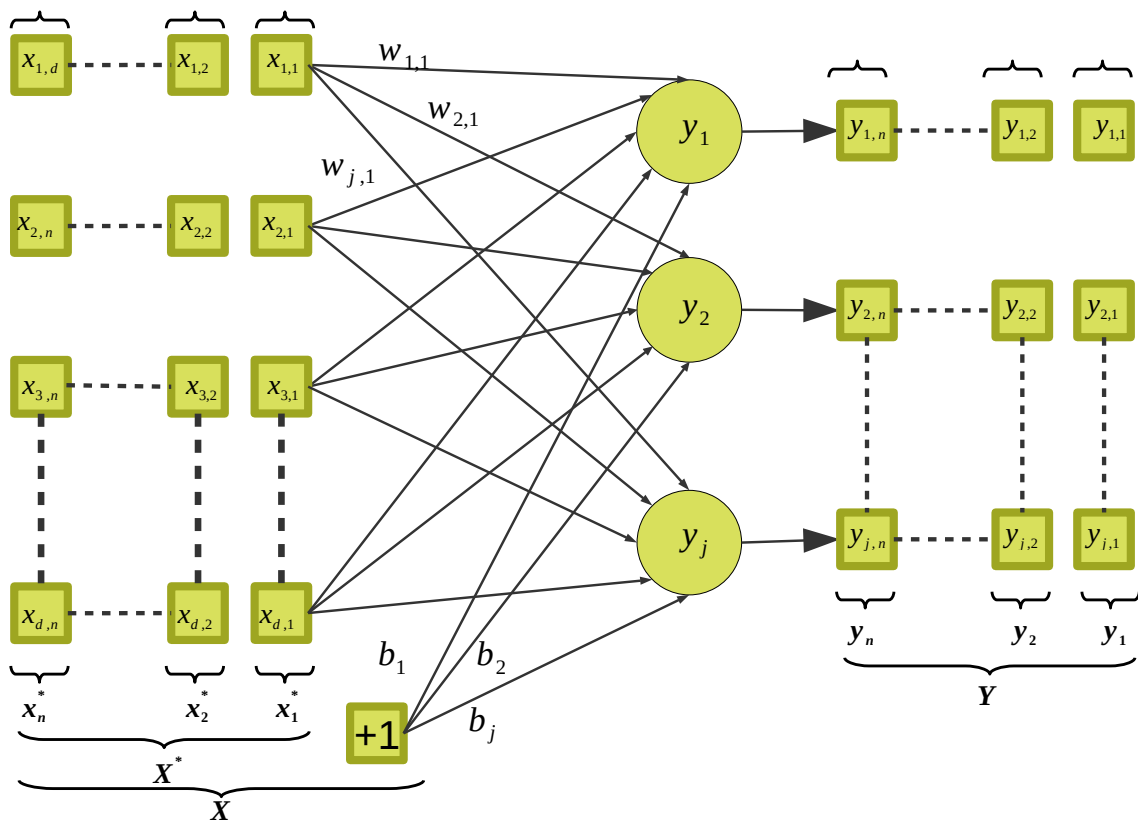


Figure 3.4: Linear single layer neural network schema.

In the former, the synaptic network weights are adapted from the difference between the expected and actual output corresponding to a known input. Given a new input that was not previously used for the training process and thanks to the model parameters estimated during this phase, it is possible to estimate a value for the output in the prediction state.

3.3.1 Linear model

Consequently, looking at figure 3.4, and given a training set with n instances of input-output pairs $\{\mathbf{X}_{1:n}, \mathbf{Y}_{1:n}\}$ where each input $\mathbf{x}_i^* \in \mathbb{R}^d$ is a vector with d attributes ($\mathbf{x}_i^* = \{x_{1,i}, x_{2,i}, \dots, x_{d,i}\}^T$) and $\mathbf{y}_i \in \mathbb{R}^j$ is a vector with the network outputs associated with each input ($\mathbf{y}_i = \{y_{1,i}, y_{2,i}, \dots, y_{j,i}\}^T$), it is possible to define a linear model (equation (3.1)) between the inputs and outputs.

$$\mathbf{y}_i = \mathbf{W}\mathbf{x}_i^* + \mathbf{b} \quad (3.1)$$

A trick can be done to avoid considering the term \mathbf{b} . If the number of attributes of the input vector is extended to one more attribute with value of just 1 so that $\mathbf{x}_i = \{1, x_{1,i}, x_{2,i}, \dots, x_{d,i}\}^T$, the equation (3.1) is simplified to equation (3.2)

$$\mathbf{y}_i = \mathbf{W}\mathbf{x}_i \quad (3.2)$$

Where:

$$\mathbf{W} = \begin{pmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,d+1} \\ w_{2,1} & w_{2,2} & \dots & w_{2,d+1} \\ \vdots & \vdots & \ddots & \vdots \\ w_{j,1} & w_{j,2} & \dots & w_{j,d+1} \end{pmatrix} \quad (3.3)$$

Moreover, they represent the weights of the network. After the training process, if n tends to infinity, the set of obtained weights should fit the model perfectly as long as the modelled process was linear. Because n is finite, the resulting weights are at best an approximation ($\hat{\mathbf{W}}$) to the real model. Therefore, the value of $\hat{\mathbf{y}}_i$ obtained with the trained weights ($\hat{\mathbf{W}}$) according to equation (3.2) is an estimation of the true \mathbf{y}_i . Based on this difference, an error parameter can be defined:

$$\boldsymbol{\epsilon}_i = \mathbf{y}_i - \hat{\mathbf{y}}_i = \mathbf{y}_i - \hat{\mathbf{W}}\mathbf{x}_i \quad (3.4)$$

3.3.2 Learning linear model parameters

Depending on the field in science in which the linear model is considered, the way to obtain the parameters that define it can be approached from different points of view. Two possible methods that converge to the same solution are presented below.

3.3.2.1 Optimization approach

Equation (3.4) is the error for input i , if the mean square error (MSE) for the whole dataset is considered, the resulting expression can be written as:

$$\mathbf{E} = \frac{1}{n} \sum_{i=1}^n \left(\mathbf{y}_i - \hat{\mathbf{W}}\mathbf{x}_i \right)^T \left(\mathbf{y}_i - \hat{\mathbf{W}}\mathbf{x}_i \right) \quad (3.5)$$

Using the procedures to compute the minimum of a function it is possible to obtain the values of the weights that minimizes this error:

$$\hat{W} = YX^T (XX^T)^{-1} \quad (3.6)$$

This equation can present certain problems when the inverse of the matrix is calculated, for example in the case that the resulting system of equations is ill-conditioned. One way to avoid this situation, is to add a small penalty or regularization term to ensure a good conditioning of XX^T :

$$\hat{W} = YX^T (XX^T + \delta^2 I)^{-1} \quad (3.7)$$

where I is a diagonal matrix of size d and δ^2 is a scale factor to graduate the penalty term. This technique is called ridge regression or Tikonov regularization and is the solution for a quadratic cost equation like:

$$E = \frac{1}{n} \sum_{i=1}^n (\mathbf{y}_i - \hat{W}\mathbf{x}_i)^T (\mathbf{y}_i - \hat{W}\mathbf{x}_i) + \delta^2 \|\hat{W}\|_f^2 \quad (3.8)$$

The term $\|\hat{W}\|_f^2$ is the squared Frobenius norm of the matrix of weights.

3.3.2.2 Maximum likelihood estimation for linear model

From the point of view of probability, it is possible to assume that each of the outputs of the neural network proposed in figure 3.4 corresponds to a random variable \mathbf{y}_i that has a probability density distribution that follows a multivariate Gaussian model. The mean of that distribution should be $W\mathbf{X}$ and we assume that every point has the same covariance Σ . As stated in appendix C.11, it can be written as: $Y \sim \mathcal{N}(W\mathbf{X}, \Sigma)$. Following equation (C.23), the likelihood function can be written as:

$$\mathcal{L}(W, \Sigma) = p(Y|X, W, \Sigma) = |2\pi\Sigma|^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{Y}-W\mathbf{X})^T \Sigma^{-1}(\mathbf{Y}-W\mathbf{X})} \quad (3.9)$$

According to the procedure for calculating the MLE (appendix C.8), the support function is defined as the logarithm of equation (3.9):

$$l(W, \Sigma) = -\frac{1}{2} \log(|2\pi\Sigma|) - \frac{1}{2}(\mathbf{Y} - W\mathbf{X})^T \Sigma^{-1}(\mathbf{Y} - W\mathbf{X}) \quad (3.10)$$

By calculating the derivative with respect to the weights of the linear model and equalling zero, the maximum likelihood estimator of the weights is obtained:

$$\frac{\partial l(W, \Sigma)}{\partial W} = 0 \rightarrow \hat{W}_{MLE} = YX^T (XX^T)^{-1} \quad (3.11)$$

The result is analogous to that obtained by approaching the problem from the point of view of optimization (equation (3.6)). However, this method has an advantage in terms of calculating not only the maximum likelihood estimator for the weights, but also, for the covariance.

$$\frac{\partial l(\mathbf{W}, \Sigma)}{\partial \Sigma} = 0 \rightarrow \Sigma_{\text{MLE}} = \frac{1}{n} \sum_{i=1}^n (\mathbf{y}_i - \mathbf{W}\mathbf{x}_i)^T (\mathbf{y}_i - \mathbf{W}\mathbf{x}_i) \quad (3.12)$$

The usefulness of Σ_{MLE} lies in indicating the uncertainty of the obtained values. Given a training set $D = \{\mathbf{X}, \mathbf{Y}\}$, and once the maximum likelihood estimators of the weights ($\hat{\mathbf{W}}_{\text{MLE}}$) and covariance (Σ_{MLE}) have been obtained, it is possible to predict, for a given new unknown value \mathbf{X}_* , the probability distribution of the neural network output: $p(\hat{\mathbf{Y}}|\mathbf{X}_*, D) = \mathcal{N}(\mathbf{Y}|\mathbf{W}\mathbf{X}_*, \Sigma)$.

3.3.2.3 Bayesian learning approach

In this case, the procedure described in appendix C.9 is followed to determine the parameters of the model. To do this, a dataset $D = \{\mathbf{X}, \mathbf{Y}\}$ is the starting point.

1. Then, define the likelihood function that the neural network outputs in figure 3.4 are assumed to have. That is, a probability model is proposed to obtain \mathbf{Y} from $\mathbf{W}\mathbf{X}$ and also to consider that there is any uncertainty in this estimate represented by a covariance Σ . Under these conditions, it can be assumed that the output of the neural network, given parameters, input values and uncertainty, could follow a Gaussian distribution:

$$\mathcal{L}(\mathbf{Y}|\mathbf{W}\mathbf{X}, \Sigma) = \mathcal{N}(\mathbf{Y}|\mathbf{W}\mathbf{X}, \Sigma) \quad (3.13)$$

Considering the network outputs as independent variables, $\mathcal{N}(\mathbf{Y}|\mathbf{W}, \mathbf{X}, \Sigma)$ can be written as the joint probability of multiple normal distributions for each obtained \mathbf{y}_i :

$$\begin{aligned} \mathcal{N}(\mathbf{Y}|\mathbf{W}, \mathbf{X}, \Sigma) &= \prod_{i=1}^n \frac{1}{(2\pi\sigma^2)^{\frac{1}{2}}} e^{-\frac{1}{2\sigma^2}(\mathbf{y}_i - \mathbf{W}\mathbf{x}_i)^2} \\ &= \frac{1}{(2\pi\sigma^2)^{\frac{n}{2}}} e^{-\frac{1}{2\sigma^2} \sum_{i=1}^n (\mathbf{y}_i - \mathbf{W}\mathbf{x}_i)^2} \end{aligned} \quad (3.14)$$

2. To assign a priori probability to the parameters. When they are estimated in the training process, they must follow a probability model. Assuming that the exact parameters that would describe the model are the weights \mathbf{W}_0 , it can be stated that the estimation of \mathbf{W} would be associated with an uncertainty \mathbf{V}_0 . Therefore,

a proper probability density distribution of the estimation of the parameters can be defined:

$$p(\mathbf{W}) = \mathcal{N}(\mathbf{W}|\mathbf{W}_0, \mathbf{V}_0) \quad (3.15)$$

The weights obtained in the training process follow a distribution that can be considered as a Gaussian over the exact weights, with a particular covariance.

3. Applying Bayes' theorem for the estimation of the *a posteriori* probability:

$$P(\mathbf{W}|\mathbf{X}, \mathbf{Y}, \Sigma) \propto \mathcal{N}(\mathbf{Y}|\mathbf{W}\mathbf{X}, \Sigma)\mathcal{N}(\mathbf{W}|\mathbf{W}_0, \mathbf{V}_0) \quad (3.16)$$

To develop this equation, the result obtained in *Marginal and conditional Gaussians* (appendix C.11.4) can be considered. Thus, the *a posteriori* probability is a Gaussian $\mathcal{N}(\mathbf{W}|\mathbf{W}_n, \mathbf{V}_n)$ where:

$$\mathbf{W}_n = (\mathbf{V}_0^{-1}\mathbf{W}_0 + \beta\mathbf{Y}\mathbf{X}^T) \mathbf{V}_n^T \quad (3.17)$$

$$\mathbf{V}_n^{-1} = \mathbf{V}_0^{-1} + \beta\mathbf{X}^T\mathbf{X} \quad (3.18)$$

In the particular case that the prior probability of \mathbf{W} is a zero-mean isotropic Gaussian governed by a single precision parameter α so that: $p(\mathbf{W}) = p(\mathbf{W}|\alpha) = \mathcal{N}(\mathbf{W}|0, \alpha^{-1}\mathcal{I})$. The corresponding posterior distribution over \mathbf{W} is simplified to:

$$\mathbf{W}_n = \beta\mathbf{Y}\mathbf{X}^T\mathbf{V}_n^T \quad (3.19)$$

$$\mathbf{V}_n^{-1} = \alpha\mathcal{I} + \beta\mathbf{X}^T\mathbf{X} \quad (3.20)$$

By combining these expressions and grouping the parameters, the next equation is obtained:

$$\mathbf{W}_n = \mathbf{Y}\mathbf{X}^T (\alpha\beta^{-1}\mathcal{I} + \mathbf{X}\mathbf{X}^T)^{-1} \quad (3.21)$$

If equation (3.21) and equation (3.7) are compared, it can be seen that the result obtained is analogous to the approach taken for the ridge regression. However, an estimation of the uncertainty defined by equation (3.20) is now available.

3.3.3 Linear prediction

3.3.3.1 Output predictions

In general, known the trained weights, the output values of a network, such as the one shown in figure 3.4, can be calculated using the expression:

$$\hat{y}_{k,i} = \sum_{m=0}^d \hat{w}_{k,m} x_{m,i} \quad (3.22)$$

where:

- $\hat{y}_{k,i}$ is the predicted output of neuron k for the input \mathbf{x}_i ;
- $\hat{w}_{k,m}$ are the trained weights corresponding to all the i^{nth} inputs related to the neuron k .
- $x_{m,i}$ represents the m^{nth} feature of the i^{nth} input vector; it could be assumed that $x_{0,i} = 1$ and therefore, the value of $\hat{w}_{k,0}$ could be considered as the bias or offset.

In particular, for a new input \mathbf{x}_* not previously used to estimate \hat{W} , the value generated at the output of the neural network according to equation (3.1) should be:

$$\mathbf{y}_* = \hat{W}\mathbf{x}_* \quad (3.23)$$

3.3.3.2 Probabilistic prediction

Using equation (3.21) to estimate the weights of the neural network by Bayesian learning approach, equation (3.23) can be rewritten as:

$$\mathbf{y}_* = \mathbf{W}_n \mathbf{x}_* = \left[\mathbf{Y} \mathbf{X}^T (\alpha \beta^{-1} \mathcal{I} + \mathbf{X} \mathbf{X}^T)^{-1} \right] \mathbf{x}_* \quad (3.24)$$

We assume that the parameters α and β are fixed and known in advance, applying the predictive posterior distribution definition (equation (C.16)):

$$f(\mathbf{y}_* | \mathbf{x}_*, D) = \int f(\mathbf{y}_* | \mathbf{x}_*, \mathbf{W}, \alpha, \beta) f(\mathbf{W} | D) d\mathbf{W} \quad (3.25)$$

Using the Gaussian notation defined in section 3.3.2.3, that expression is transformed:

$$\begin{aligned} f(\mathbf{y}_* | \mathbf{x}_*, D) &= \int \mathcal{N}(\mathbf{y}_* | \mathbf{W} \mathbf{x}_*, \Sigma) \mathcal{N}(\mathbf{W} | \mathbf{W}_n, \mathbf{V}_n) d\mathbf{W} \\ &= \mathcal{N}(\mathbf{y}_* | \mathbf{W} \mathbf{x}_*, \Sigma + \mathbf{x}_*^T \mathbf{V}_n \mathbf{x}_*) \end{aligned} \quad (3.26)$$

As seen with this approach, the whole space of the weights that have been evolving during the training is considered.

The probability distribution followed by the network output and calculated using only the MLE approach (equation (3.11) and equation (3.12)) can be defined as:

$$f(\mathbf{y}_* | \mathbf{x}_*, D) = \mathcal{N}(\mathbf{y}_* | \mathbf{W}_{MLE} \mathbf{x}_*, \Sigma_{MLE}) \quad (3.27)$$

While the Bayesian learning approach considers the whole parameter space, the estimation using MLE only grants a value for \mathbf{W}_{MLE} that is the one that maximises the

probability. Furthermore, while the covariance in the case of the Bayesian approach varies depending on the input of the network during the training and prediction process, in the case of the MLE it is constant.

3.4 Supervised non-linear learning using basis functions

3.4.1 Basis functions

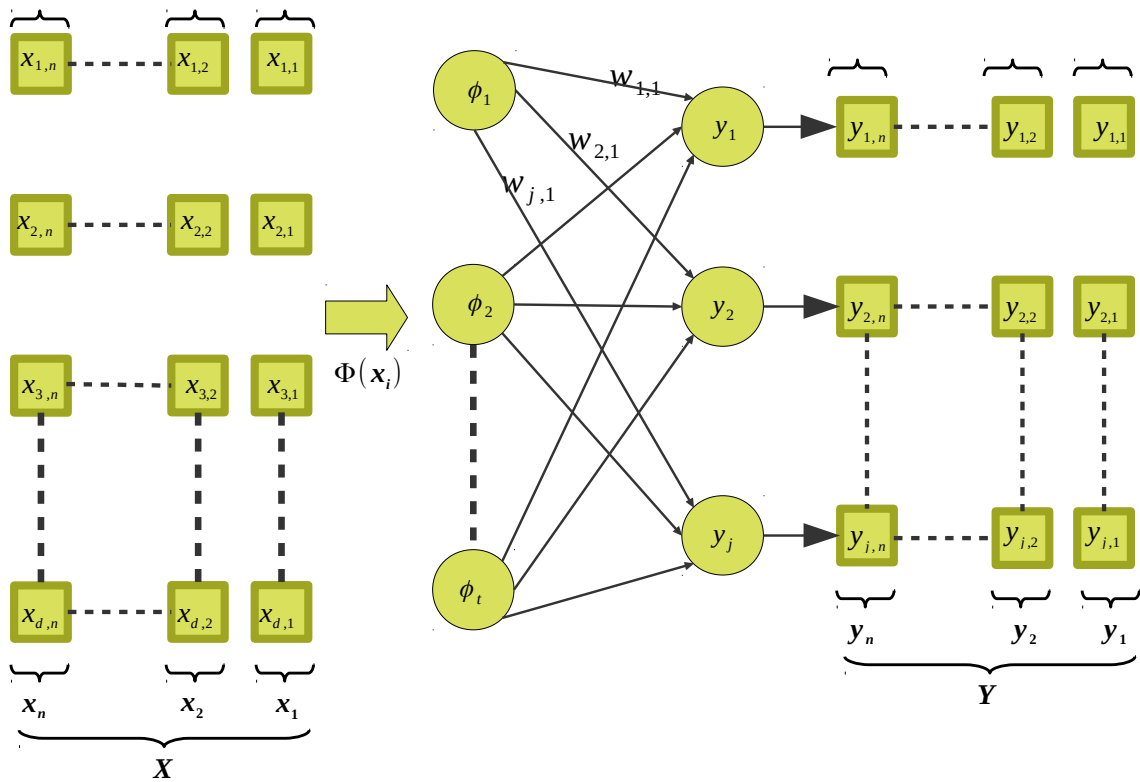


Figure 3.5: Schema of linear single layer neural network with generic basis function.

Basis functions are a set of standard mathematical functions:

$\Phi(\mathbf{x}_i) = \{\phi_1(\mathbf{x}_i), \phi_2(\mathbf{x}_i), \dots, \phi_t(\mathbf{x}_i)\}$ that are linearly independent and they can be combined to estimate any function $\mathbf{y}_i(\mathbf{x}_i, \mathbf{W})$:

$$\mathbf{y}_i = \mathbf{W}\Phi(\mathbf{x}_i) \tag{3.28}$$

Where \mathbf{W} are the parameters of the model. In the particular case that $\Phi(\mathbf{x}_i) = \mathbf{x}_i$ and $\phi_1(\mathbf{x}_i) = 1$ the above expression is transformed into a simple linear model as shown

Table 3.1: Some of the most used standard basis functions

Type of basis	$\phi_t(\mathbf{x}_i)$
Polynomial functions	\mathbf{x}_i^t
Gaussians or radial functions	$\exp\left(-\frac{(\mathbf{x}_i - \mu_t)^2}{2\sigma^2}\right)$
Sigmoidal functions	$\frac{1}{1+e^{-a}}\left(\frac{\mathbf{x}_i - \mu_t}{\sigma}\right)$
Fourier functions	$\phi_{2t-1} = \sin(2\pi t \mathbf{x}_i); \phi_{2t} = \cos(2\pi t \mathbf{x}_i)$

in equation (3.2). Sometimes it is useful to employ the transposition of equation (3.28):

$$\mathbf{y}^T = \Phi(\mathbf{x}_i)^T \mathbf{W}^T \rightarrow \boldsymbol{\gamma} = \Phi(\mathbf{x}_i)^T \boldsymbol{\mathcal{W}} \quad (3.29)$$

where $\boldsymbol{\gamma}$ is \mathbf{y}^T and $\boldsymbol{\mathcal{W}} = \mathbf{W}^T$.

A single-layer neural network schema described by this model is shown in figure 3.5.

The objective of the basis functions is to make a change of space in such a way that the non-linear relationship between \mathbf{y}_i and \mathbf{x}_i is transformed into linear when the basis function is applied to \mathbf{x}_i . In this way, all the available techniques for estimating the parameters that define this new linear relationship can be used.

The choice of the type of basis function depends on the type of model to be addressed. There are standard and widely used sets of these functions (table 3.1). All these basis functions depend on specific parameters that must be set before starting the learning process. In some cases, approximation or prior unsupervised learning of these parameters can be done from an initial dataset.

The estimation of the weights of a neural network like the one shown in figure 3.5, with a hidden layer in which the activation function is one of these basis functions, can be done by any of the approaches seen in the previous section. Thus, for example, using Bayesian learning, equation (3.21) can be rewritten using basis functions as:

$$\mathbf{W}_n = \mathbf{Y} \Phi(\mathbf{X})^T (\alpha \beta^{-1} \mathcal{I} + \Phi(\mathbf{X}) \Phi(\mathbf{X})^T)^{-1} \quad (3.30)$$

3.4.2 Radial basis function neural network

A radial basis function neural network (**RBFNN**) has the topology of a full connected single-layer network. Their particularity is the presence of a hidden layer that transforms the inputs using a radial basis function (**RBF**). Radial basis function networks can

potentially approximate any function with the desired precision (Park and Sandberg, 1991). They were actually introduced to interpolate functions (Powell, 1987). It was precisely this concept of interpolation that related radial basis functions to neural networks (Broomhead and Lowe, 1988).

Given an input $\mathbf{x} \in \mathbb{R}^n$, a RBF is a radially symmetric function, i.e. it is a real-valued function whose value depends only on the distance from some other point $\mathbf{c} \in \mathbb{R}^n$, called a center, $\phi(\mathbf{x}) = f(\|\mathbf{x} - \mathbf{c}\|)$. Typically, the Gaussian is used as a RBF:

$$\phi(\mathbf{x}) = e^{\left(-\frac{(\mathbf{x}-\mu)^2}{2\sigma^2}\right)} \quad (3.31)$$

Where μ and σ are the centre and radius of the RBF. For a particular neuron i of the hidden layer, the centre μ represents the centre of a cluster that is used as a basis for comparison. The standard deviation σ for this cluster defines the range of the RBF. The choice of σ and μ for each neuron determines a partition of the input space. As it is the case for other parameters, this choice conditions the network performance and its learning ability. The values of the centres can be chosen randomly (Broomhead and Lowe, 1988). However, unsupervised methods —such as k-means— can be used to estimate the RBF centres (Moody and Darken, 1989).

Therefore, the model that describes a RBF neural network with RBFs as activation functions in its hidden layer can be expressed, with the nomenclature of figure 3.5, as:

$$y_{k,i} = \sum_{l=1}^t w_{k,l} \phi_l(x_i) \quad (3.32)$$

where, in this case, the activation function ϕ_l is defined as a Gaussian:

$$\phi_l(\mathbf{x}_i) = \exp\left(-\sum_{m=1}^d \frac{(x_{m,i} - \mu_{m,l})^2}{2\sigma_l^2}\right) \quad (3.33)$$

3.4.3 Trigonometric basis functions neural network

From the theory of Fourier series, any continuous function $f(x)$ can be decomposed as follows:

$$f(x) = c + \sum_{l=1}^{\infty} a_l \cos(lx) + \sum_{l=1}^{\infty} b_l \sin(lx) \quad (3.34)$$

If we accept an error in this approximation, and only use a finite number of terms, the trigonometric functions defined in the above expression can act as basis functions. Thus, the higher the maximum value of l , the better the obtained accuracy. Moreover, the use of these trigonometric functions avoids to some degree the requirement for normalising the inputs.

A particular model based on the application of this kind of trigonometric basis functions is defined by [Lázaro-Gredilla et al. \(2010\)](#).

$$f(\mathbf{x}_i) = \sum_{l=1}^t a_l \cos(2\pi\Omega_l^T \mathbf{x}_i) + b_l \sin(2\pi\Omega_l^T \mathbf{x}_i) \quad (3.35)$$

Where Ω_l represents a vector of spectral frequencies that is shared by each pair of trigonometric functions. The values of these frequencies are previously predefined, whereas the amplitudes a_l and b_l are considered as independent parameters that follow a particular Gaussian distribution:

$$a_l \sim \mathcal{N}\left(0, \frac{\sigma_0^2}{t}\right); b_l \sim \mathcal{N}\left(0, \frac{\sigma_0^2}{t}\right) \quad (3.36)$$

where σ_0^2 is the variance of the prior distribution that it is independent of the inputs. In this way, it is possible to define a set of pairs of trigonometric functions that compose the basis functions of the neural network:

$$\phi(\mathbf{x}_i) = [\cos(2\pi\Omega_1^T \mathbf{x}_i) \sin(2\pi\Omega_1^T \mathbf{x}_i), \dots, \cos(2\pi\Omega_t^T \mathbf{x}_i) \sin(2\pi\Omega_t^T \mathbf{x}_i)] \quad (3.37)$$

3.5 Supervised and adaptive learning algorithms

The use of basis functions as indicated in section 3.4 implies that a neural network can approximate any function. Its main advantage is that once the basis transformation has been carried out, the problem of estimating the model parameters has an analytical solution, so long as a sufficient and consistent number of data is provided. Nevertheless, as stated in section 3.1.4, our aim is to adapt the learning to the changes in the environment; for this reason, the direct use of the analytical solution for the estimation of the weights of the neural network is not practical because it requires gathering information beforehand about the environment to estimate its model. This is, however, not always possible since the information coming from the environment is a stream that must be processed as it is received. Moreover, it modifies and conditions what is learned about the environment.

Two different approaches to solve this problem are presented below. They are based on the scheme in figure 3.5, considering a generic type of basis function.

3.5.1 Kalman filter for neural network adaptive training

In the context of linear dynamical systems, the Kalman filter is a fundamental tool to resolve recursively linear optimal filtering problems in both stationary and non-stationary environments (Haykin, 2001). The Kalman filter incrementally updates the network state, represented by the weights, each time the training point is available. Therefore, the expressions that describe the transfer between the states (weights) and the observation of the effects of this transfer can be defined, given a neural network whose model is described by equation (3.28).

The state space model of the neural network can be expressed by this equation:

$$\mathbf{W}_i = \Gamma \mathbf{W}_{i-1} + \rho \quad (3.38)$$

And the observation of the network state is given by:

$$\mathbf{y}_i = \mathbf{W}_i \phi(\mathbf{x}_i) + \epsilon \quad (3.39)$$

Where:

- Γ is the transition model matrix between the network states.
- ρ is a white noise associated with the confidence in the transition between the states, so $\rho \sim \mathcal{N}(0, \mathbf{Q})$. Where \mathbf{Q} is the variance matrix of this Gaussian noise.
- ϵ is the white noise regarding the measure of the state: $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$.
- \mathbf{W}_i represents the neural weights at iteration i .
- $\phi(\mathbf{x}_i)$ is a basis function transform over the input in the current iteration \mathbf{x}_i .
- \mathbf{y}_i is the output of the neural network at current i iteration.

The goal of training is to update the weights given the past and current observations. At any time $i > 1$ the sequence of observations $\mathcal{D}_{i-1} = \{\mathbf{y}_{i-1}, \mathbf{y}_{i-2}, \dots, \mathbf{y}_1\}$ can be defined. From this point, it is possible to consider a conditional prior probability distribution $f(\mathbf{W}_i | \mathcal{D}_{i-1})$. This term depends on the previous observations and the current state. Applying equation (C.16) to this conditional probability:

$$f(\mathbf{W}_i | \mathcal{D}_{i-1}) = \int f(\mathbf{W}_i | \mathbf{W}_{i-1} \mathcal{D}_{i-1}) f(\mathbf{W}_{i-1} | \mathcal{D}_{i-1}) d\mathbf{W}_{i-1} \quad (3.40)$$

using the Chapman-Kolmogorov expansion this equation is transformed to:

$$f(\mathbf{W}_i | \mathcal{D}_{i-1}) = \int f(\mathbf{W}_i | \mathbf{W}_{i-1}) f(\mathbf{W}_{i-1} | \mathcal{D}_{i-1}) d\mathbf{W}_{i-1} \quad (3.41)$$

$f(\mathbf{W}_i|\mathbf{W}_{i-1})$ is defined by equation (3.38) as a Gaussian distribution with mean $\Gamma\mathbf{W}_{i-1}$ and variance \mathbf{Q} , i.e. $f(\mathbf{W}_i|\mathbf{W}_{i-1}) = \mathcal{N}(\Gamma\mathbf{W}_{i-1}, \mathbf{Q})$. Due to recursion, as it will be seen below, the term $f(\mathbf{W}_{i-1}|\mathcal{D}_{i-1})$ would correspond to the prior probability calculated in the previous step of the iteration. For now, we assume that this probability is a Gaussian distribution with a mean $\boldsymbol{\mu}_{i-1}$ and $\boldsymbol{\Sigma}_{i-1}$ covariance: $f(\mathbf{W}_{i-1}|\mathcal{D}_{i-1}) \sim \mathcal{N}(\boldsymbol{\mu}_{i-1}, \boldsymbol{\Sigma}_{i-1})$. The above integral does not need to be calculated numerically if certain properties of Gaussians are used. Considering the joint distribution ($f(\mathbf{W}_{i-1}, |\mathcal{D}_{i-1}), f(\mathbf{W}_i|(\mathbf{W}_{i-1}, |\mathcal{D}_{i-1}))$) and using equation (C.37):

$$\begin{bmatrix} f(\mathbf{W}_{i-1}, |\mathcal{D}_{i-1}) \\ f(\mathbf{W}_i|(\mathbf{W}_{i-1}, |\mathcal{D}_{i-1})) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_{i-1} \\ \Gamma\boldsymbol{\mu}_{i-1} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{i-1} & \boldsymbol{\Sigma}_{i-1}\Gamma^T \\ \Gamma\boldsymbol{\Sigma}_{i-1} & \Gamma\boldsymbol{\Sigma}_{i-1}\Gamma^T + \mathbf{Q} \end{bmatrix}\right) \quad (3.42)$$

Therefore, the marginal distribution of $f(\mathbf{W}_i|\mathcal{D}_{i-1})$ is:

$$f(\mathbf{W}_i|\mathcal{D}_{i-1}) \sim \mathcal{N}(\Gamma\boldsymbol{\mu}_{i-1}, \mathbf{P}_i = \Gamma\boldsymbol{\Sigma}_{i-1}\Gamma^T + \mathbf{Q}). \quad (3.43)$$

On the other hand, the network model defines the probability of obtaining its output \mathbf{y} from the weights of the network: $f(\mathbf{y}_i|\mathbf{W}_i, \mathcal{D}_{i-1}) = f(\mathbf{y}_i|\mathbf{W}_i) \sim \mathcal{N}(\mathbf{W}_i\phi(\mathbf{x}_i), \sigma_n^2)$. Using equation (C.37) again, the joint distribution of \mathbf{y}_i and \mathbf{W}_i conditioned by previous steps can be written as:

$$\begin{bmatrix} f(\mathbf{W}_i, |\mathcal{D}_{i-1}) \\ f(\mathbf{y}_i|(\mathbf{W}_i, |\mathcal{D}_{i-1})) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \Gamma\boldsymbol{\mu}_{i-1} \\ \Gamma\boldsymbol{\mu}_{i-1}\phi(\mathbf{x}_i) \end{bmatrix}, \begin{bmatrix} \mathbf{P}_i & \mathbf{P}_i\phi(\mathbf{x}_i) \\ \phi(\mathbf{x}_i)^T\mathbf{P}_i & \phi(\mathbf{x}_i)^T\mathbf{P}_i\phi(\mathbf{x}_i) + \sigma_n^2 \end{bmatrix}\right) \quad (3.44)$$

Renaming the variance of $f(\mathbf{y}_i|(\mathbf{W}_i, |\mathcal{D}_{i-1}))$ as:

$$\mathbf{S}_i = \phi(\mathbf{x}_i)^T\mathbf{P}_i\phi(\mathbf{x}_i) + \sigma_n^2 \quad (3.45)$$

And using equation (C.31) and equation (C.32) from appendix C.11.3 to calculate the joint distribution of two Gaussian that are conditioned: $f(\mathbf{W}_i|y_i\mathcal{D}_{i-1}) \sim \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$

$$\begin{aligned} \boldsymbol{\mu}_i &= \Gamma\boldsymbol{\mu}_{i-1} + \mathbf{P}_i\phi(\mathbf{x}_i)\mathbf{S}_i^{-1}(\mathbf{y}_i - \Gamma\boldsymbol{\mu}_{i-1}\phi(\mathbf{x}_i)) \\ \boldsymbol{\Sigma}_i &= \mathbf{P}_i - \mathbf{P}_i\phi(\mathbf{x}_i)\mathbf{S}_i^{-1}\phi(\mathbf{x}_i)^T\mathbf{P}_i \end{aligned} \quad (3.46)$$

The term $\mathbf{P}_i\phi(\mathbf{x}_i)\mathbf{S}_i^{-1}$ represents the linear coefficient of recursion:

$$\mathbf{K}_i = \frac{\boldsymbol{\mu}_i - \Gamma\boldsymbol{\mu}_{i-1}}{(\mathbf{y}_i - \Gamma\boldsymbol{\mu}_{i-1}\phi(\mathbf{x}_i))} = \mathbf{P}_i\phi(\mathbf{x}_i)\mathbf{S}_i^{-1} \quad (3.47)$$

Algorithm 1 Adaptive algorithm to train a neural network based on Kalman filter**Require:** $W_0, P_0, Q, \sigma_n^2, \Gamma$; $i \leftarrow 1$;**while** $i < n$ **do**

$$W_i \leftarrow \Gamma W_{i-1};$$

$$P_i \leftarrow \Gamma P_{i-1} \Gamma^T + Q;$$

$$S_i \leftarrow \phi(\mathbf{x}_i)^T P_i \phi(\mathbf{x}_i) + \sigma_n^2;$$

$$K_i \leftarrow P_i \phi(\mathbf{x}_i) S_i^{-1}$$

$$W_i \leftarrow W_i + K_i (\mathbf{y}_i - W_i \phi(\mathbf{x}_i));$$

$$P_i \leftarrow P_i (\mathbf{I} - K_i \phi(\mathbf{x}_i))^T;$$

 $i \leftarrow i + 1$ **end while**

Below, algorithm 1 shows how these equations can be expressed and sequenced in algorithmic form.

Although the deduction of the recursive least squares algorithm (RLS) can be made independently of the Kalman filter, an exciting result is that for $\Gamma = I$ and $Q = 0$, i.e. no variability is assumed to the regression parameters, the Kalman filter algorithm is equivalent to the recursive least squares algorithm (Teixeira and Rodrigues, 1997) (see algorithm 2).

Algorithm 2 Adaptive algorithm to train a neural network based on Recursive Least Squares**Require:** W_0, P_0 ; $i \leftarrow 1$;**while** $i < n$ **do**

$$S_i \leftarrow 1 + \phi(\mathbf{x}_i)^T P_{i-1} \phi(\mathbf{x}_i);$$

$$K_i \leftarrow P_{i-1} \phi(\mathbf{x}_i) S_i^{-1}$$

$$W_i \leftarrow W_{i-1} + K_i (\mathbf{y}_i - W_{i-1} \phi(\mathbf{x}_i));$$

$$P_i \leftarrow P_{i-1} - K_i \phi(\mathbf{x}_i)^T P_{i-1};$$

 $i \leftarrow i + 1$ **end while**

3.5.2 Incremental sparse Gaussian process regression for neural network adaptive training

This method was proposed by [Rasmussen \(2003\)](#), and it is based on the linear model expressed by equation¹:

$$\boldsymbol{\gamma} = \boldsymbol{\phi}(\mathbf{x})^T \mathbf{W} + \epsilon \quad (3.48)$$

The additive noise ϵ is an identically distributed Gaussian distribution with zero mean and variance σ_n^2 : $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$ —as in section 3.5.1. Therefore, the prior likelihood $f(\boldsymbol{\gamma} | \mathbf{W}, \boldsymbol{\phi}(\mathbf{x})^T)$ can be described by a Gaussian distribution with mean $\boldsymbol{\phi}(\mathbf{x})^T \mathbf{W}$ and variance σ_n^2 : $f(\boldsymbol{\gamma} | \mathbf{W}, \boldsymbol{\phi}(\mathbf{x})) \sim \mathcal{N}(\boldsymbol{\phi}(\mathbf{x})^T \mathbf{W}, \sigma_n^2)$.

The prior distribution over the weights is considered a Gaussian with zero mean and variance Σ_p : $\mathbf{W} \sim \mathcal{N}(0, \Sigma_p)$. Under these conditions, it is possible to use Bayes' theorem and the posterior probability $f(\mathbf{W} | \boldsymbol{\gamma}, \mathbf{x})$ can be estimated. Using the Gaussian properties developed in appendix C.11.4, applying equation (C.34) and equation (C.35) to the proposed neural network model:

$$\Sigma_{\mathbf{W} | \boldsymbol{\gamma}} = \Sigma_p^{-1} + \sigma_n^{-2} \boldsymbol{\phi}(\mathbf{x}) \boldsymbol{\phi}(\mathbf{x})^T \quad (3.49)$$

$$\boldsymbol{\mu}_{\mathbf{W} | \boldsymbol{\gamma}} = \sigma_n^{-2} \Sigma_{\mathbf{W} | \boldsymbol{\gamma}}^{-1} \boldsymbol{\phi}(\mathbf{x}) \boldsymbol{\gamma} \quad (3.50)$$

If the transformation $\mathbf{A} = \boldsymbol{\phi}(\mathbf{x}) \boldsymbol{\phi}(\mathbf{x})^T + \sigma_n^2 \Sigma_p^{-1}$ is considered:

$$f(\mathbf{W} | \boldsymbol{\gamma}, \mathbf{x}) = \mathcal{N}(\mathbf{A}^{-1} \boldsymbol{\phi}(\mathbf{x}) \boldsymbol{\gamma}, \sigma_n^2 \mathbf{A}^{-1}) = \mathcal{N}(\hat{\mathbf{W}}, \sigma_n^2 \mathbf{A}^{-1}) \quad (3.51)$$

where $\hat{\mathbf{W}} = \mathbf{A}^{-1} \mathbf{b}$ and $\mathbf{b} = \boldsymbol{\phi}(\mathbf{x}) \boldsymbol{\gamma}$.

Once a set of samples has been used to estimate the network weights and their variance using the above equations, the predictive posterior distribution for a new test sample $(\boldsymbol{\phi}(\mathbf{x}_*), \boldsymbol{\gamma}_*)$ is:

$$f(\boldsymbol{\gamma}_* | \boldsymbol{\phi}(\mathbf{x}_*), \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\gamma}) = \mathcal{N}(\boldsymbol{\phi}(\mathbf{x}_*)^T \mathbf{A}^{-1} \boldsymbol{\phi}(\mathbf{x}) \boldsymbol{\gamma}, \sigma_n^2 (1 + \boldsymbol{\phi}(\mathbf{x}_*) \mathbf{A}^{-1} \boldsymbol{\phi}(\mathbf{x}_*)^T)) \quad (3.52)$$

Let $(\mathbf{x}_i, \boldsymbol{\gamma}_i)$ be a new sample in i iteration, matrix \mathbf{A} can be formulated recursively (([Gijsberts and Metta, 2013](#))) as:

$$\mathbf{A}_i = \mathbf{A}_{i-1} + \boldsymbol{\phi}(\mathbf{x}_i) \boldsymbol{\phi}(\mathbf{x}_i)^T \quad (3.53)$$

¹This model is expressed in this way based on equation (3.29) that defines the neural network model in figure 3.5

Furthermore, if the term $\phi(\mathbf{x}_i)\gamma_i$ is named \mathbf{b}_i , a recursion can be defined as:

$$\mathbf{b}_i = \mathbf{b}_{i-1} + \phi(\mathbf{x}_i)\gamma_i \quad (3.54)$$

The initial configuration for recursion is defined by $\mathbf{A}_0 = \sigma_n^2 \Sigma_p^{-1}$ and $\mathbf{b}_0 = 0$. From equation (3.51), it is possible to write:

$$\mathbf{A}_i \hat{\underline{\mathbf{w}}}_i = \mathbf{b}_i \quad (3.55)$$

The network weights in each iteration can be obtained from this system of linear equations, but it is necessary to compute the inverse of matrix \mathbf{A}_i . Since this is a covariance matrix, anyway, it fulfils the properties mentioned in appendix C.11.2, and it can be transformed using Cholesky decomposition. Let \mathbf{R}_i be an upper triangular Cholesky factor resulting from \mathbf{A}_i decomposition: $\mathbf{A}_i = \mathbf{R}_i^T \mathbf{R}_i$, equation (3.53) can be transformed using Cholesky decomposition as:

$$\mathbf{R}_i^T \mathbf{R}_i = \mathbf{R}_{i-1}^T \mathbf{R}_{i-1} + \phi(\mathbf{x}_i)\phi(\mathbf{x}_i)^T \quad (3.56)$$

An efficient way to estimate \mathbf{R}_i is to use Cholesky rank-1 update, as proposed by (Gijssberts and Metta, 2013). In any case, the network weights can be computed from matrix \mathbf{R}_i :

$$\hat{\mathbf{W}}_i = \mathbf{R}_i^{-1} (\mathbf{R}_i^T)^{-1} \mathbf{b}_i \quad (3.57)$$

Algorithm 3 Adaptive algorithm to train a neural network based on iterative Gaussian Process and Cholesky rank-1 update

Require: $\mathbf{R}_0, \mathbf{W}_0, \mathbf{b}_0$;

$i \leftarrow 1$;

while $i < n$ **do**

$\gamma_i \leftarrow \phi(\mathbf{x}_i)^T \hat{\mathbf{W}}_{i-1}$;

$\mathbf{b}_i \leftarrow \mathbf{b}_{i-1} + \phi(\mathbf{x}_i)\gamma_i$;

$\mathbf{R}_i \leftarrow \text{CholeskyUpdate}(\mathbf{R}_i, \phi(\mathbf{x}_i))$;

$\hat{\mathbf{W}}_i \leftarrow \mathbf{R}_i^{-1} (\mathbf{R}_i^T)^{-1} \mathbf{b}_i$;

$i \leftarrow i + 1$

end while

By ordering all these equations and considering the initial values, it is possible to develop an adaptive algorithm 3.

3.6 Conclusions

The biochemical and signal transport processes in biological neurons are far more complex than the abstractions resulting from the models described in this chapter. However, they replicate several fundamental aspects of the biological reality, that are instrumental in creating artificial models. This chapter introduced some mathematical tools that allow a simple artificial neural network to adapt its model as new information becomes available. This ability is particularly relevant in a robot attempting to operate in a changing environment. In this way, adaptive artificial neural networks become the basis for defining the internal models of robotic systems and for processing information from the environment, adapting their response accordingly. Since most of the algorithms proposed in this work are based on this concept of adaptability, we have introduced it here from a mathematical point of view.

3.7 Publications supporting this chapter

- **Antonelli, M., Duran, A.J., del Pobil, A.P., 2013**, "Application of the Visuo-Oculomotor Transformation to Ballistic and Visual-Guided Eye Movements", in *Proc. International Joint Conference on Neural Networks (IJCNN 2013)*, Dallas, Texas, USA, pp. 813-820. ISBN: 978-1-4673-6129-3/13.

Chapter 4

Robotic systems for visual exploration by means of saccadic movements

We have to accept that we are just machines. That's certainly what modern molecular biology says about us.

Rodney Brooks

4.1 Introduction

Due mainly to its sensory information-cost ratio, one of the standard sensors that a robot can be endowed with is a camera in any of its variations. Although some cameras can cover spherical fields of vision of almost 360 degrees, most of them have a field of vision of a few tens of degrees. Therefore, if they cannot move, it is not possible to perceive other areas of the environment that surround them. As described in section 2.1, living beings have developed mechanisms to perform visual exploration of their environment based on eye movements. One of the most important is saccadic movement (section 2.5). The desired goal is to design a robotic system that performs exploratory movements to acquire information from the environment to respond and adapt to change, using saccadic movement as a source of inspiration.

Developing a robotic mechanism that imitates the human oculomotor system is a very complex task due to the mechanical and physical properties of the biological system (section 2.3). Of necessity, reducing the problem is called for. The first approach is to consider that the displacement of the eyes is a pure rotation with no translation. In this way, the six degrees of freedom are simplified to three. Moreover, if Donders' law is considered, the eye exploits only two of these degrees of freedom. Under these conditions, the displacement of the two eyes can be expressed as three rotations per eye: the first rotation around the horizontal axis that is common to both, and the second rotation around the vertical axis. Finally, the third rotation would be around the line-of-sight/optic axis. This model is called Helmholtz coordinates and it has served as basis for the development of a number of robotic systems. In this case, the cameras tilt up and down about a common elevation platform and verge independently about axes perpendicular to the elevation plane (figure 4.1).

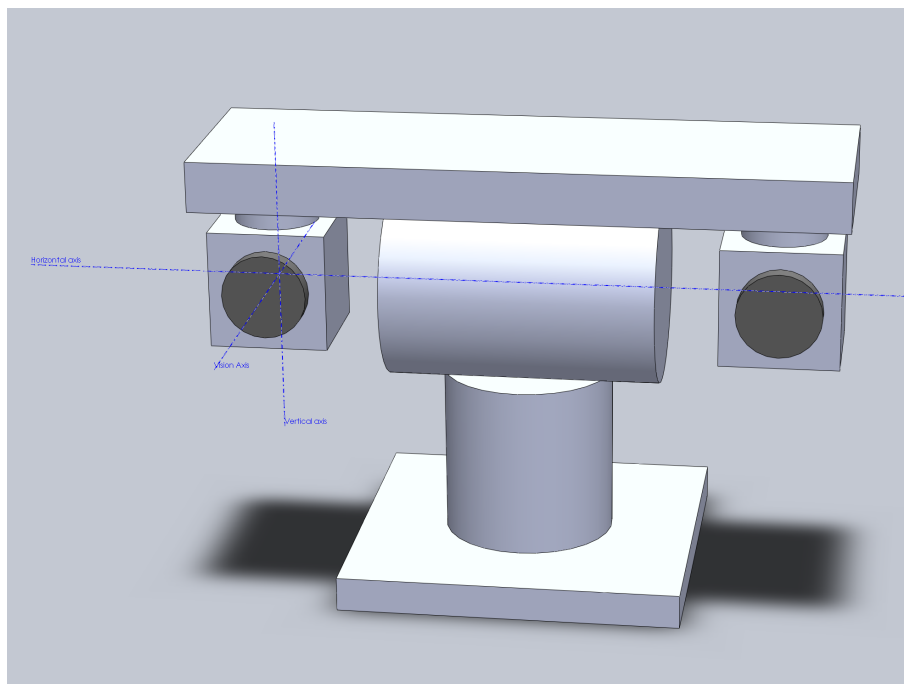


Figure 4.1: Conceptual design of a robot oculomotor system based on Helmholtz configuration

There is an alternative model to the one based on Helmholtz's angles. This model is called Fick and is based on two grouped systems with four degrees of freedom in which the horizontal axis of rotation is common to both but can be actuated independently (figure 4.2). From the point of view of vision algorithms in robotics, there is no clear advantage of one model with respect to the other (Murray et al., 1992); for this reason, the configuration of choice is usually the one with the lowest cost and requiring only three actuators, i.e. the model based on Helmholtz configuration.

A robotic system based on the design in figure 4.1 needs a series of components

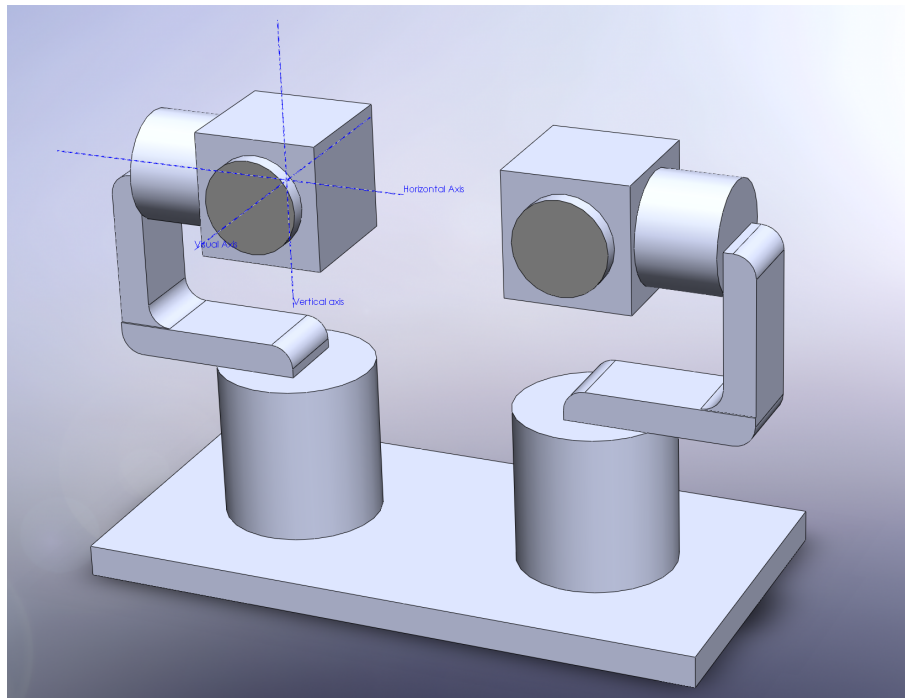


Figure 4.2: Conceptual design of a robot oculomotor system based on Fick configuration

such as motors, cameras and sockets. They are organised to replicate the Helmholtz configuration. That is, all the components are assembled according to the three axes as defined in this configuration. In biology, the study of the structure conditioned by the size and shape of animals, plants, and microorganisms and of the relationships of their constituent parts is referred to as morphology (Vilee, 2019). Translating this concept to the world of robotics, we can speak of the morphology of a robot when referring to the structure of the elements that constitute the proposed conceptual configurations, along with their specific design parameters that define the relationships between the components.

In engineering, the rotations defined in the previous models must be implemented by some design that incorporates an actuator, the simplest case being a motor. The sensing part that would imitate the operation of the eyes would be composed of two cameras that must be integrated into the design.

The use of a camera to replicate the behaviour of the eye, when the pinhole model is also used to operate it, is not entirely correct since the model of the eye proposed by Gullstrand (section 2.4) considers the crystalline to be a real lens and therefore with two nodal points. Besides, the retina has a particular curvature and CCD or CMOS sensors in cameras are usually flat matrices. In any case, our objective is not to precisely replicate the eye morphology, but rather to understand how the biological design works in order to develop a robotic system that performs similar functions.

Both commercial humanoid robots and robotic platforms built from off-the-shelf components often incorporate a vision system. Not all platforms count with specific mechatronics to imitate a visual configuration such as that of the human oculomotor system, as table 4.1 shows. The more complex humanoid robots focus on the control of bipedal and manipulation parts. On the other hand, vision mechanisms are elementary systems based on stereo or RGB-D cameras in most cases, and their movement is included in the body kinematics of the robot.

Table 4.1: Visual systems integrated in high-end robotic platforms.

Robot Platform	Assembler or manufacturer	Comment
Asimo	Honda R&D Co.Ltd	Stereoscopic cameras fixed to the head. Its movements are achieved through the rotation of the joints that compose the robot body (Sakagami et al., 2002).
Armar III	University of Karlsruhe	This robot has distributed cameras with a Helmholtz configuration mounted on the head (Asfour et al., 2006).
iCub	Italian Institute of Technology and University of Genova	This robot is specialised in bioinspired cognitive studies and features a head where two cameras with a Helmholtz configuration are placed (Metta et al., 2008).
Baxter	Rethink Robotics	Although this robot platform has three cameras, only one is mounted on the head which only has two degrees of freedom. (Cremer et al., 2016)
HRP-4	Institute of Advanced Industrial Science and Technology (AIST)	Humanoid robot focused on replicating human appearance. Its visual system is based on two fixed cameras which can be moved with the two degrees of freedom of the head (Kaneko et al., 2011)
Rollin' Justin	DLR - German Aerospace Center	This robot has two stereo cameras placed on a 2-DOF head (Fuchs et al., 2009).
Continued on next page		

Table 4.1 – continued from previous page

Robot Platform	Assembler or manufacturer	Comment
Talos	PAL-Robotics	This robot has a head with two degrees of freedom and can mount either an RGB-D camera or a stereoscopic camera. (Stasse et al., 2017)
Nao	Aldebaran-Robotics	This small humanoid robot has a 2-DOF head; the two cameras are fixed but, in contrast to other robots, they are not placed in the apparent eyes and have different orientations. (Shamsuddin et al., 2011)
Tombatossals	Jaume I University	It is a multipurpose humanoid torso for research in autonomous grasping and manipulation. The head is composed of a TO-40 pan-tilt-vergence system with two RGB cameras and one RGB-D sensor. The configuration of the cameras is based on the Helmholtz model. (Felip et al., 2015)

In any case, the three implemented systems (Armar III, iCub and Tombatossals) that have independent control over the degrees of freedom of each camera are configured according to the Helmholtz model.

In table 4.1, the term head is mentioned several times, but in most cases, the function of the head is just to incorporate the vision sensors. Since often the only purpose of the head is to hold the visual system, we will refer to the "robotic head" as a synonym of the visuo-oculomotor system.

In addition to the above set of robots, a number of more specific systems have been prepared to imitate and study eye movements and their application to the field of robotics (Rucci et al., 1999, 2000, 2007).

Following the cycle of perception and action in living beings (figure 2.1), the concept of morphology and configuration of the visuo-oculomotor system relates the effectors (motors) and the sensors (cameras). If the models developed in robotics pursue to imitate this cycle, it is necessary to consider two more components: the environment and cognition. The concept of environment has a relative character and fuzzy limits. The definition of environment is usually made according to what is not the system. However, the environment of living beings played an essential role in

developing the theory of evolution. The adaptation to the environment induced changes and selected individuals causing their evolution. In robotics, there is a more robo-centric view of the environment and especially when it is unstructured. The environment is everything that is not a robot and interacts with it. However, the environment can be seen as an obstacle because it prevents or complicates the robot operating function. The environment is constantly changing and indeterminate; it is challenging to consider all possible states when a deterministic approach is followed.

Cognition in living beings is the mental process of modelling and understanding the environment through interaction, experience, and perception. A third observer perceives this modelling as a behaviour of the analyzed system. Saccadic behaviour is the result of processes (section 2.5) that allow us to receive information from the environment in the form of stimuli and triggering a series of neurophysical processes, provoking an active response from the oculomotor system. In robotics, this problem involves developing a controller to process the visual input and generate the motor command.

The concepts of morphology, behaviour and environment are elaborated in this chapter. Our purpose is to design a robotic system that is able to execute saccadic movements to actively explore the world.

The first part of this chapter (section 4.3) deals with the description of a characteristic morphology of a robot head and how to parameterize it generically, abstracting from particular configurations. The next section 4.4 deals with implementing the behaviour of the robotic system. Two adaptive control architectures are proposed, and both are evaluated in simulation and then with a real system. A brief overview of the concept of environment is given in the last part of the chapter (section 4.5) to complete the three above-mentioned aspects. This part is dealt with in more detail in further chapters.

4.2 Objectives

One of the objectives of this work is to study saccadic behaviour and implement it in a robotic system. Therefore, this chapter aims to describe a robotic system in a generic way by using the parameters corresponding to a Helmholtz configuration, and allowing the implementation of saccadic behaviour within a specific environment. Furthermore, the different components must also be defined within a generic approach, so that the conclusions derived from its study can be extended beyond a particular case.

4.3 Morphology of a robot system for executing saccadic movements

The proposed system is based on a Helmholtz configuration. We use two kinematic chains (left and right) to describe the geometry of the robotic head where the end effectors are the cameras forming the vision system. These chains have the particularity that they share a joint, which corresponds to the angle of rotation of the neck.

This system presents morphological parameters corresponding to actuators and sensors since it comprises two cameras and three controlled DOF. Furthermore, eight additional prismatic joints are defined in the model so that changes in the head morphology can be introduced, resulting in a total of eleven DOF (figure 4.3).

The Denavit-Hartenberg model for the left-hand side of the head is specified in table 4.2; the right-hand side has an identical description, and both share the first joint ρ_t .

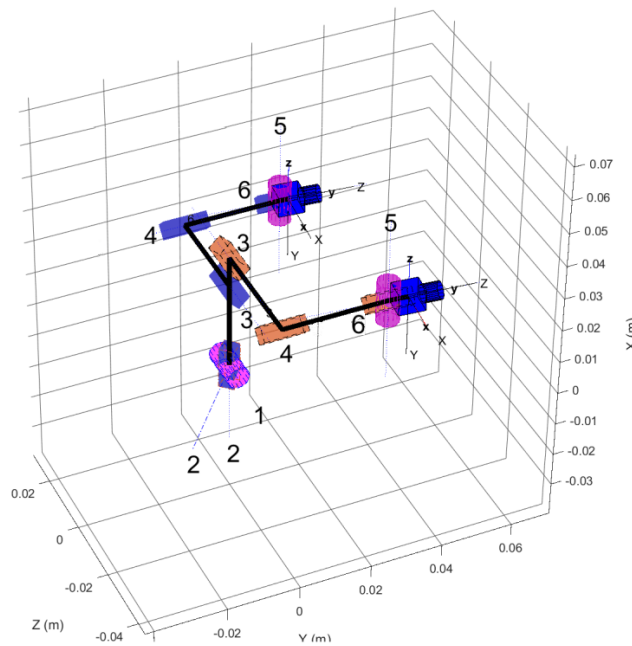


Figure 4.3: Example of the head model. In this schema, there are two overlapping kinematic chains. Purple cylinders represent the three revolute joints. Joint number one is shared by both chains. Parallelepipeds represent prismatic joints that modify the head morphology; the two colors blue and orange correspond to the two sides of the head.

The prismatic joints $\{q_2, q_3, q_4, q_6\}_{left}$ and $\{q_2, q_3, q_4, q_6\}_{right}$ define the geometry of the head and do not modify their values while the head is executing a task.

The value of joint q_2 would represent the height of the neck with respect to the tilt

Table 4.2: Denavit-Hartenberg model of the left side of the head. ρ_p, ρ_t are the revolutes joints for the pan and common tilt motors. The model for the right-hand side is the same, and it shares the ρ_t joint

joint	$\rho(\text{rad})$	r(m)	a(m)	$\alpha(\text{rad})$	Offset	Type
q_1	ρ_t	0	0	$\pi/2$	$\pi/2$	R
q_2	0	0	0	$-\pi/2$	0	P
q_3	$\pi/2$	0.055	0	$\pi/2$	0	P
q_4	$\pi/2$	0.055	0	$\pi/2$	0	P
q_5	ρ_p	0	0	$\pi/2$	π	R
q_6	0	0.01	0	$\pi/2$	0	P

angle of the neck. As defined, the system q_2 can be different for the right or left chain. Although this configuration does not seem to have an equivalent in the biological world, from a system engineering design point of view it might be possible.

The joint q_3 sets for both the right and left chain the value of the distance of each camera to the geometric centre of the system defined by the neck axis. As in the case of q_2 this distance can be different for each kinematic chain and thus, the symmetry of the system is regulated.

The joints q_4 and q_6 apparently generate a displacement in the same direction, i.e. perpendicular to the plane containing the joints q_2 and q_3 . The joint q_4 represents the displacement of the cameras in relation to the plane containing q_2 and q_3 . In this case, as in the previous ones (q_2 and q_3), it is possible to establish a differential configuration for each kinematic chain.

The joint q_6 is defined in the kinematic chains after q_5 , so even if the displacement occurs in the same direction as q_4 , the effective value of this displacement is affected by the value of q_5 . The value of q_6 for each chain represents the distances from the planes of projection of the cameras to their axis of rotation. This is equivalent to the nodal distance.

The values of these eight parameters ($\Gamma^{(a)} \in \mathbb{R}^8$) describe the robot morphology in terms of head actuators. It is assumed that the movement of the joints are precise enough so that noise does not need to be added to the model.

The system is completed with two camera sensors, and the pinhole model is taken to simulate their behavior. Each camera can be described by using at least four parameters: focal length (f), pixel size (s) (supposed squared), width (w) and height (h) of the images. Therefore, the parameters $\{f, s, w, h\}_{left}$ and $\{f, s, w, h\}_{right}$ are regarded as the morphological sensor parameters ($\Gamma^{(s)} \in \mathbb{R}^8$). Thus, there are a total

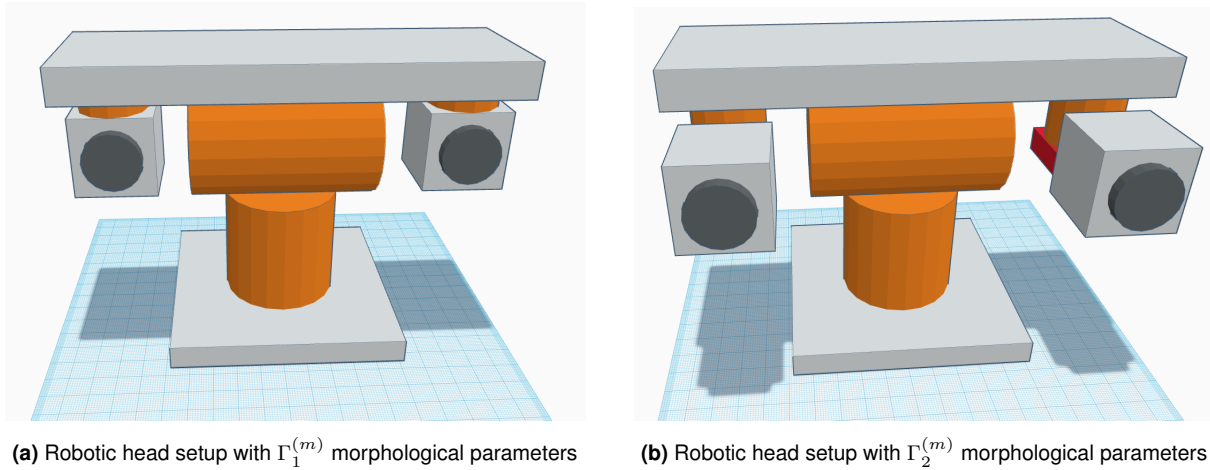


Figure 4.4: Two examples of how the morphologies of the robot heads change by varying their morphological parameters.

of 16 morphological parameters for each robotic head configuration ($\Gamma^{(m)} \in \mathbb{R}^{16}$). By varying these 16 $\Gamma^{(m)}$ parameters infinite vision system setups based on the Helmholtz configuration can be obtained. Many of them would not make any physical sense. Variation intervals can be defined for each parameter to avoid this problem, so that the resulting systems, although still infinite, have a greater physical sense. For example, in figure 4.4, two different configurations can be seen. Each of them has a set of parameters that define it morphologically. The configuration in figure 4.4a is equivalent to that in figure 4.1. The parameters $\{q_2, q_3, q_4, q_6\}_{left}$ and $\{q_2, q_3, q_4, q_6\}_{right}$ for this case are $\{0, -a, 0, 0\}_{left}$ and $\{0, a, 0, 0\}_{right}$; where a is the distance between the geometric centres of the cameras divided by 2. In contrast, figure 4.4b shows how the cameras are now slightly moved forward. The morphological parameters for this head are $\{0, -a, b, 0\}_{left}$ and $\{0, a, b, 0\}_{right}$; where b is the magnitude of the forward displacement of the cameras.

4.4 Saccadic behaviour

Performing an accurate saccade requires converting the visual position of the target into a shift of the eye position and to generate an eye movement to get the desired eye position. Beyond optimally solving the control problem that this transformation of signal into control action entails, we assume that the revolute joints that have been introduced in section 4.3, have a control system precise enough to reach a particular position reliably and accurately. This work is focused on the transformation that links the visual position of the stimulus with a target position of the eye. The retinotopic position of the stimulus —i.e. its position on the image captured by the camera— has to be

converted into a shift of the eye (camera) position. These saccadic gaze shifts are planned in 3D space. Therefore, from the robot point of view, generating a saccade requires solving an inverse control problem. In this saccadic behaviour, the visual position of the stimulus in relation to a target position of the eyes is an open loop with respect to vision. Learning this transformation requires learning the inverse kinematic model of the robot head. Several strategies have been proposed for learning this model. An early proposed strategy is the direct inverse modelling (Kuperstein, 1988). In its original formulation, direct inverse modelling performs random movements and then learns the inverse association between the motor command and its perceptual outcome. This technique was employed for learning saccade control in several works together with some ad hoc strategies to reduce the exploration process (Schenck and Möller, 2006; Chao et al., 2010; Antonelli et al., 2013b). This approach has two shortcomings: it does not consider redundant systems, and it is not goal-directed. In the proposed morphology (section 4.3), there is no redundancy, so this factor would not be a problem for the use of this method. However, in view of empirical evidence that infants perform goal-directed action right from the outset of motor learning (Rohde et al., 2019), the visuomotor transformations should be adapted as they occur in order to have a greater biological plausibility. If a saccade of a particular direction and size consistently fails to reach the intended target, the transformation parameters in the brain internal model are adjusted; this phenomenon is known as saccadic adaptation (Lappe, 2009).

Feedback error learning (FEL) (Kawato, 1990) is an attempt to overcome these limitations. This controller has been used for both saccadic control (Bruske et al., 1997) and smooth pursuit (Shibata et al., 2001) in robotics.

Furthermore, there is a computer model of the cerebellum that has been tested in various simulations (Porrill and Dean, 2007) and in a robotic system (Lenz et al., 2008) that allows to learn the visuo-oculomotor transformations. The version of this controller presented by Porrill et al. (2004) is based on a recurrent architecture (RA).

4.4.1 Monocular vs. binocular encoding

4.4.1.1 Introduction

In the morphological definition of the visuo-oculomotor system in section 4.3, we can consider the two kinematic chains as independent since they only have one shared joint. Therefore, if the position of the stimulus in one of the cameras is considered as input, then —using the appropriate methodology— it is possible to determine the position increment that should be applied to the joints of this chain to bring the stimulus

to the centre of the image. This procedure represents a monocular approach, and several works have elaborated on it (Marjanovic et al., 1996; Chao et al., 2010; McBride et al., 2010).

In contrast, if the stimulus information from the two cameras is used to estimate the motor commands to produce the appropriate displacements in the system for a saccadic behaviour, then we would be following a binocular approach (Hoffmann et al., 2005; Schenck and Möller, 2006; Forssén, 2007; Nori et al., 2007; Rucci et al., 2007).

In principle, we assume that the binocular option is more precise than the monocular one because it implicitly encodes the distance to the stimulus. In addition, it is necessary to somehow consider depth information since in our morphology the camera nodal point may not lie in the centre of the camera rotation axis. In the following sections, this problem will be addressed: the formulation of this hypothesis will be tested to calculate the visuomotor transformations for the generation of saccadic behaviours.

4.4.1.2 Simplified robot head model

The proposed model for the robotic head system presented in section 4.3 starts from the kinematic description of the system using two kinematic chains that share the same tilt joint. Simplifying all the possible combinations possible to be generated with the proposed model, we assume that the two cameras lie on the same plane parallel to the axis of rotation of the tilt joint. This configuration matches the one shown in figure 4.4a. Taking this plane for a given tilt angle, the problem becomes two-dimensional (figure 4.5) and therefore, the transformations defined in equation (2.1) and equation (2.2) apply (compare figure 4.5 with its biological counterpart in figure 2.3). Three classes of magnitudes are represented in figure 4.5:

1. The variation in the normalized visual shift, represented in figure 4.5 by u_r and u_l . The normalizing factor is the width of the image in pixels.
2. The angular position of both the right (θ_r) and left (θ_l) camera in relation to their axes of rotation perpendicular to the plane.
3. γ_l and γ_r represent the desired angular positions of the left and right cameras when the robot is gazing at the target.

Therefore, the starting points are: an angular position represented by $\theta = [\theta_r, \theta_l]^T$, and a projection of the stimulus on the images $\mathbf{u} = [u_r, u_l]^T$. The goal is to come up with the values $\gamma = [\gamma_r, \gamma_l]$ that would bring the gaze point at the target point, performing the

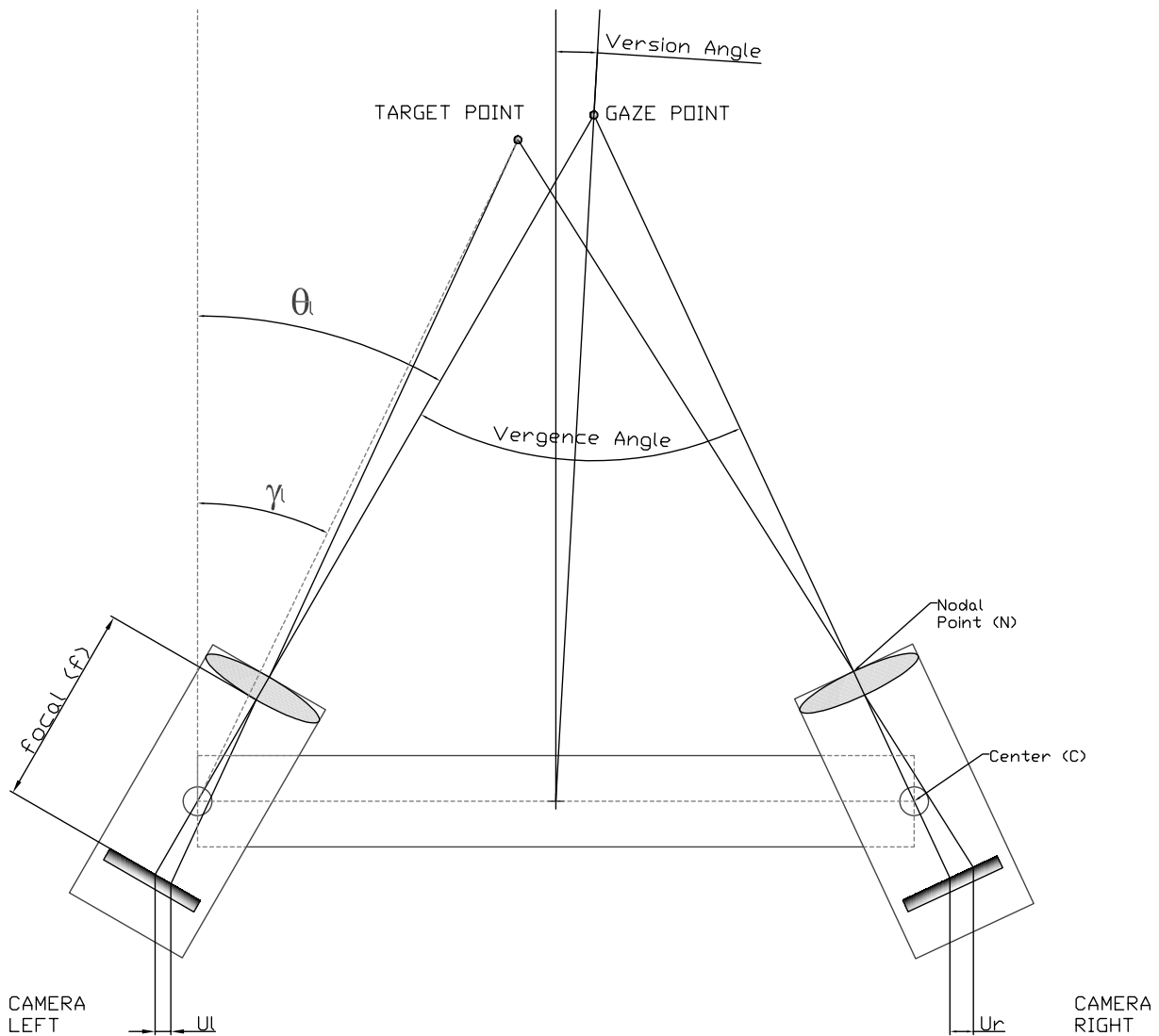


Figure 4.5: 2D model of the robot head.

desired saccade. In this way it is possible to define the monocular visuo-oculomotor transformation as:

$$\gamma_{\{l|r\}} = T_{v \rightarrow o}^M(u_{\{l|r\}}, \theta_{\{l|r\}}) \quad (4.1)$$

In contrast, the binocular approach can be formulated as follows:

$$\gamma = T_{v \rightarrow o}^B(\mathbf{u}, \boldsymbol{\theta}) \quad (4.2)$$

Table 4.3: RBFNN setups to learn the visuo-oculomotor transformations

Transf.	Inputs	Outputs	Units in hidden layer
$T_{v \rightarrow o}^M(u_{\{l r\}}, \theta_{\{l r\}})$	$u_{\{l r\}}, \theta_{\{l r\}}$	$\gamma_{\{l r\}}$	24(6x4)
$T_{v \rightarrow o}^B(\mathbf{u}, \boldsymbol{\theta})$	$u_l, u_r, \theta_l, \theta_r$	γ_l, γ_r	576(6x6x4x4)

4.4.1.3 Radial basis function to compute the visuo-oculomotor transformations

The models defined by equation (4.1) and equation (4.2) represent a visuo-oculomotor transformation that must be learned by the system and adapted to each interaction with the environment to have some biological plausibility. As seen in section 3.4.2, RBFNN can potentially approximate any function with sufficient precision. Moreover, in section 3.5 several algorithms have been proposed for adaptive learning of these networks. A RBFNN for each approach is implemented. The input signal is encoded with a fixed number of the basis functions. Accordingly, six units are required to code the visual inputs, and four units are assigned to four proprioceptive cues. Considering this configuration, the definition of each RBFNN used to learn each transformation is shown in table 4.3. The centres determining each unit of the hidden layer are regularly distributed within the normalised space of the input variables. The variance defining the radii of these radial functions is three times the value of the distance between two consecutive centres. The projection in the plane of the input variables of the radial functions in the case of the monocular transformation can be seen in figure 4.6.

The algorithm used to adapt the weights of these networks is the recursive least-squares (RLS)(algorithm 2, section 3.5.1). The state matrix P is initialized to $10^3 I$ where I is the identity matrix and the weights are initialized to zero.

4.4.1.4 Training and testing datasets

A dataset must be generated to train the proposed RBFNNs. This dataset comprises a series of virtual points in the 3D space in front of the robotic system. Generating this dataset using random points that ensure that the stimulus appears in both cameras simultaneously at the current position of the cameras implies that many attempts must be made to get a single point. For this reason, to avoid this problem, the reverse situation is proposed. The points are generated by giving random values of version and distance from the midpoint between the cameras. This way, it is ensured that any of these points can appear in both images simultaneously if the cameras are placed at the appropriate angles. The dataset is built using 30 values for distance and 13 for version.

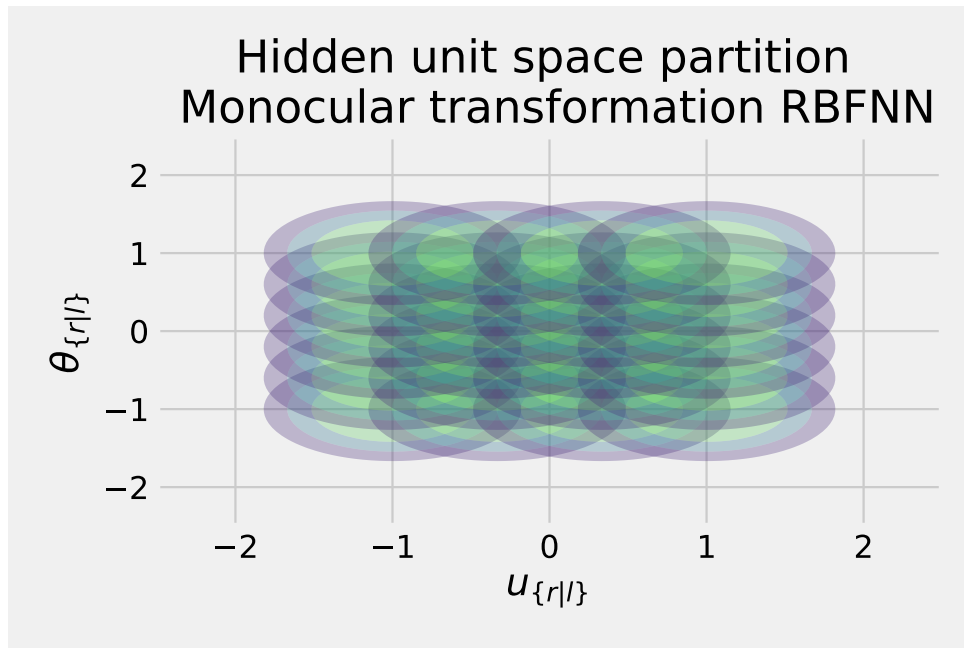


Figure 4.6: Radial basis function space partition in monocular transformation neural network

The graphical representation of this dataset is shown in figure 4.7.

At the same time, a smaller space of 200 random points is created with the version and distance values (red circles). These points represent the gaze points; hence they are the starting position of the head. From these points, saccades are executed to all the points appearing in the images of the two cameras. As the generation of this dataset is proposed, it is impossible to transfer it to a real robotic system because it is challenging to produce stimuli with this distribution and know its exact position. However, in order to check if the monocular or binocular option is better at the simulation level, it is enough.

Once trained, the neural networks are tested with another dataset generated similarly to the training one. In this case, 20969 input-output pairs have been generated.

4.4.1.5 Comparison monocular vs. Binocular encoding

As indicated in the previous section, the neural networks (table 4.3) are adaptively trained using RLS algorithm. The corresponding training curves can be seen in figure 4.8. The behaviour of the network for the monocular case is better in the early iterations; however, it reaches around 2000 iterations that stabilise and even suffer an overtraining effect. In contrast, the network designed for the binocular case requires more effort to achieve the results obtained in the monocular model. However, once the monocular networks are stabilised, the binocular network performance improves and remains so

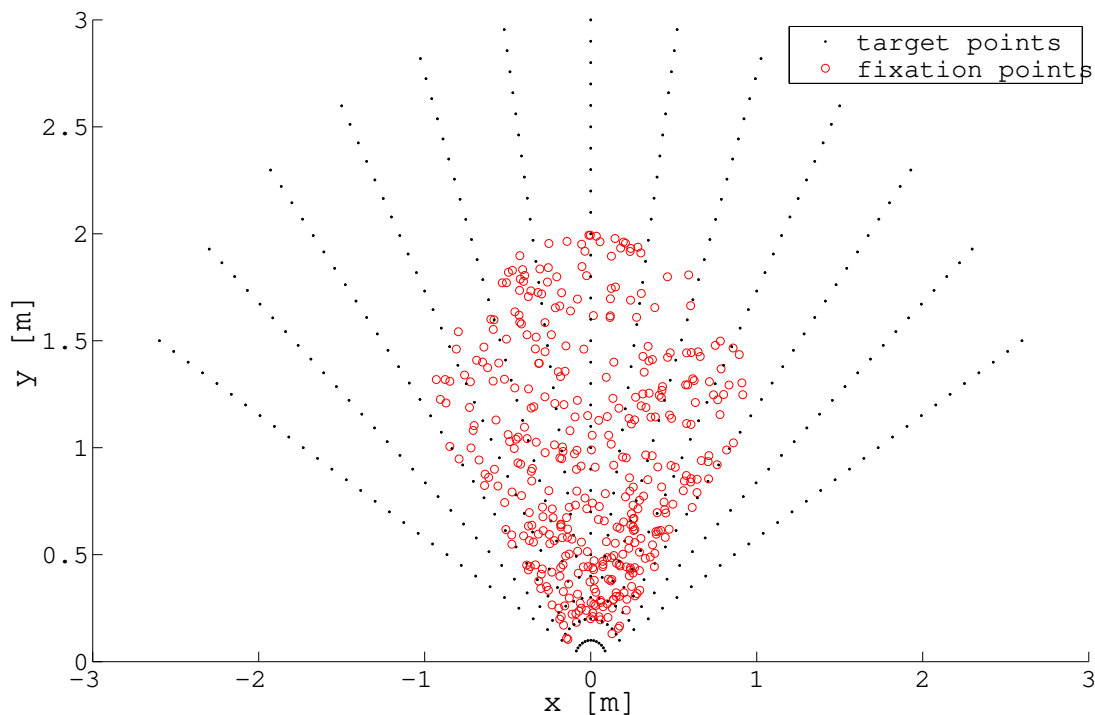


Figure 4.7: Generated dataset to train the RBFNNs.

for the successive 3000 iterations. This behaviour is consistent with the design and complexity of both networks. However, the monocular network is more straightforward, and its training behaviour shows that there is a point where it is not possible to learn more, but rather the opposite. On the other hand, the neural network that models the binocular transformation is more complex, and therefore it takes longer to adapt its weights to obtain similar results; however, it continues improving these once the monocular limit has been reached. The trained networks are used to estimate the visuo-oculomotor transformation of the test dataset to evaluate these two models. This transformation is then executed, and the stimulus's new visual position in the cameras' images is obtained. The euclidean distance to the centre of the image reflects the visual error of the transformation. From the tests performed, a histogram like the one shown in figure 4.9 is built, giving an idea of which is the probability density distribution that these results follow. The results obtained suggest the difference between using the monocular and binocular models.

From the qualitative point of view, it can be seen in figure 4.9 that both distributions are pretty symmetric and have a maximum central value. However, there is a clear difference in the dispersion of the results. Whereas there is a great dispersion in the monocular case, the values are concentrated around zero in the binocular model.



Figure 4.8: Learning neural networks curves for the proposed neural networks comparing the binocular and monocular training behaviour.

Initially, this distribution was adjusted to a Gaussian model so that it could give an approximation of the most probable value of visual error made by each model. Later, the fit to a Gaussian distribution was inaccurate because the data are more concentrated around the central value than in a Gaussian distribution. Finally, after performing the successive tests with different distributions, it is concluded that the data resulting from the simulation with the test dataset fit better to a Lévy alpha-stable probability distribution (Mandelbrot, 1960).

The mean value is obtained in -0.025 pixels with a standard deviation of 1.687 pixels for binocular case truncating this distribution in the range $[-10,10]$ pixels. On the contrary, the maximum expected value for the monocular model is -0.096 pixels with a standard deviation of 10.028 pixels. The enormous dispersion of results in the monocular experimental distribution suggests that it does not adequately model the problem.

These results are confirmed when the estimation of the Euclidean distance between the Cartesian points used to generate the test dataset and the points projected

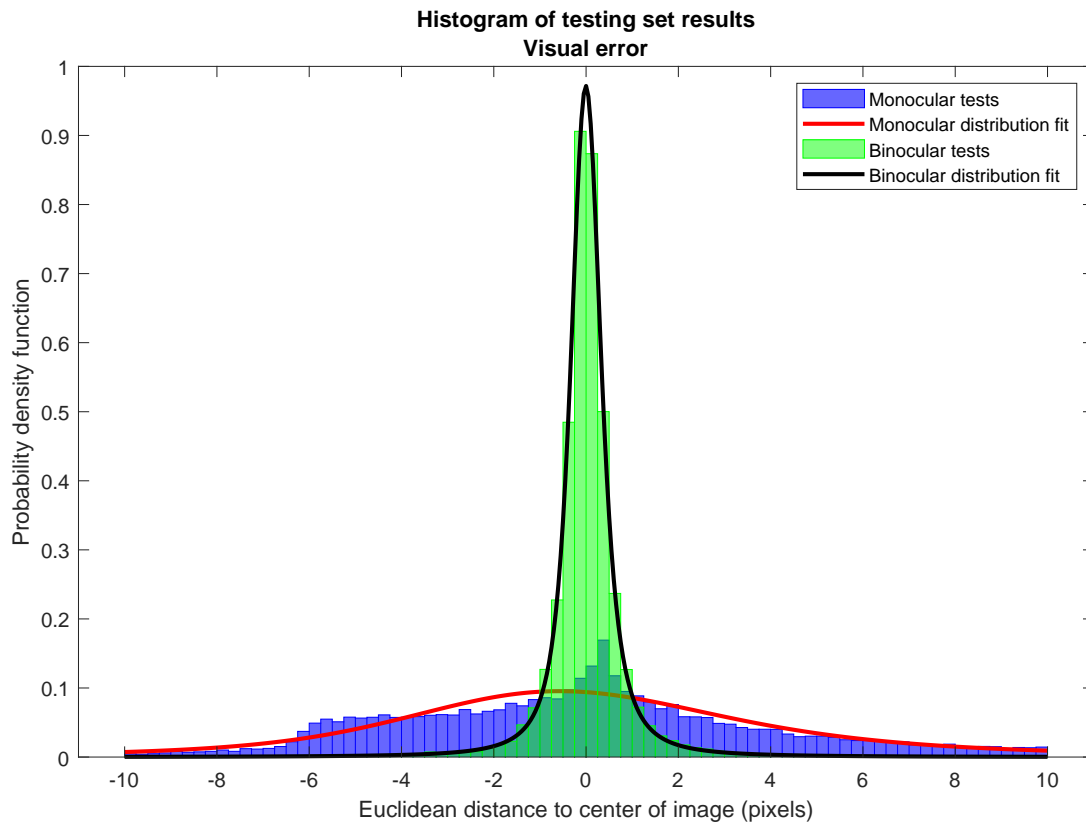


Figure 4.9: Results comparison obtained for the test dataset using as a reference measure the distance to the center of the image (maintaining the sign). The lines represent the probability density distribution that best fits the experimental data in the working range. In this case, it is a Lévy alpha-stable distribution

from the camera positions after executing the transformation using the proposed neural networks are considered (figure 4.10). In this case, the probability density distribution that best fits the data obtained from the test set evaluation is a Birnbaum-Saunders distribution (Birnbaum and Saunders, 1969). After comparing the experimental data with the model that generates this distribution using the Kolmogorov-Smirnov test (De Leeuw, 2009), the equivalence is concluded with a significance level of 5%. Considering these distributions, the monocular model has an average error value of 46.0 mm with a standard deviation of 66.9 mm. This error is reduced to 8.5 mm with a standard deviation of 10.9 mm in the binocular case.

The worst performance of the monocular approach concerning the binocular one is mainly due to the lack of depth cues. These are necessary because the centre of rotation of the camera does not lie on the optical centre of the camera. Using the test dataset and grouping the points by distance, the visual error is computed as a function

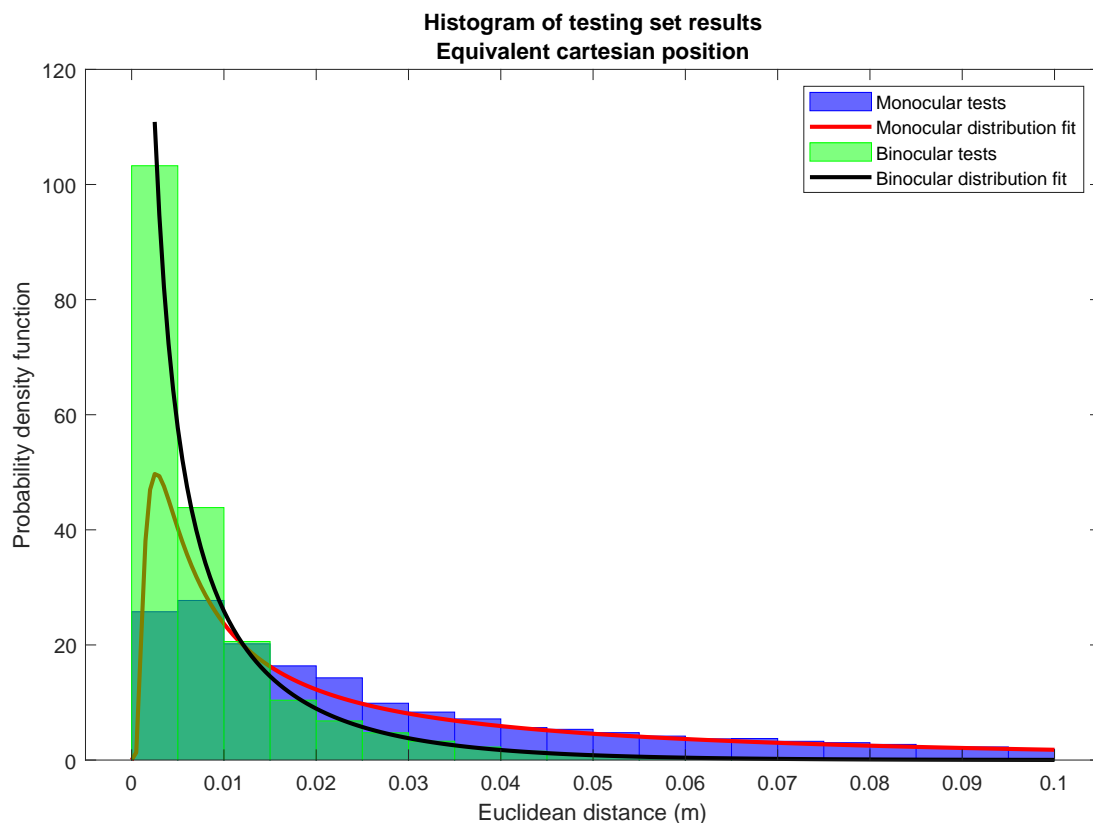


Figure 4.10: Results comparison obtained for the test dataset using as a reference measure the Euclidean distance between the projected point using the final camera angle position and the corresponding simulation point created in test dataset. The lines represent the probability density distribution that best fits the experimental data in the working range. In this case, it is a Birnbaum–Saunders distribution

of this parameter for each approach. figure 4.11 shows the results obtained. For the binocular approach, the gazing error tends to increase with the distance of the target, and this is expected because the binocular disparity, like other visual cues, are inversely proportional to the distance. In the monocular case, the tendency to gaze error with distance is noted as well, though with a more significant variation.

4.4.1.6 Conclusion

As in the biological world, the binocular model implicitly considers the distance to the target, this information is also coded in the case of the artificial binocular model using neural networks, and therefore the saccadic behaviour cannot be implemented independently for each camera, but both should be considered together.

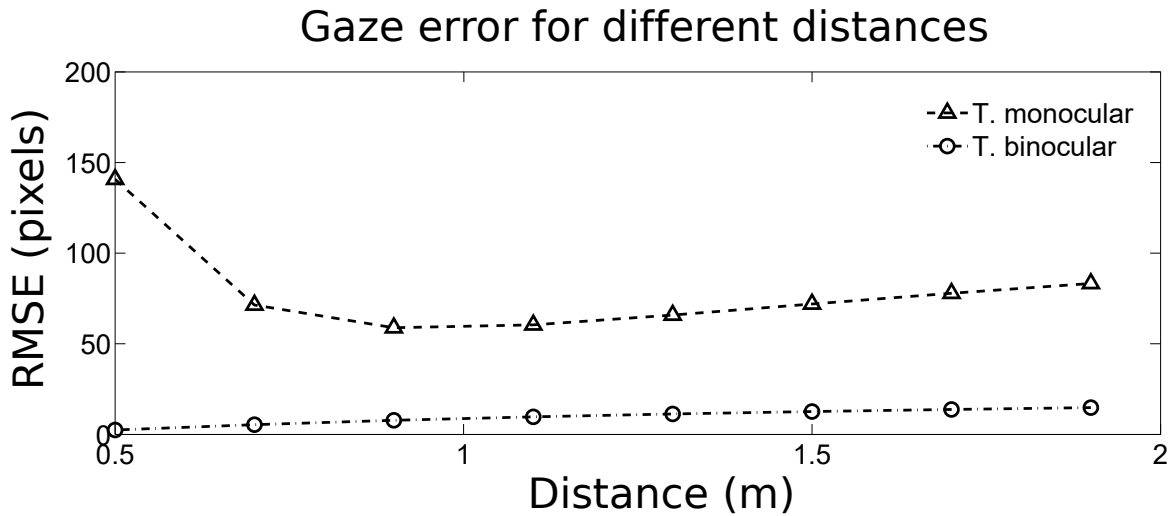


Figure 4.11: Gazing error of the monocular and binocular encoding as a function of the target distance.

4.4.2 Recurrent architecture vs. Feedback error learning

In section 4.4.1, saccadic behaviour has been implemented by learning the visuo-oculomotor transformation directly. This is possible because the calculations are done in simulation, and therefore, the environment is known completely. The experiments are designed so that the target is always in the desired visual range, and only two degrees of freedom in the motor system are considered. This section describes two kinds of saccadic controllers: Feedback error learning (FEL) and recurrent architecture (RA). The main goal is to maintain biological plausibility and to be able to implement saccadic behaviour in a Helmholtz design. Therefore, from a control engineering point of view, the objective is to convert a visual target t and the current system state e into a change of state Δe , which brings the stimulus to the centre of the visual field.

Both approaches are composed by a fixed controller B and a non-linear adaptive controller C_x . In this work, B is considered as a linear inverse model of the plant (P). Even if using B alone is enough to drive the eyes toward the target, the execution of a ballistic movement does not provide a precise saccade. Due mainly to the non-linearity of the system, the controller cannot have adequate results. The intervention of the adaptive controller (C_x) allows to compensate for this poor performance of the linear controller. The difference between the two control schemes is how the controllers B and C_x are interconnected, and the role of the adaptive controller C_x , which is an inverse model (C_f) in the FEL and a forward model (C_r) in the RA.

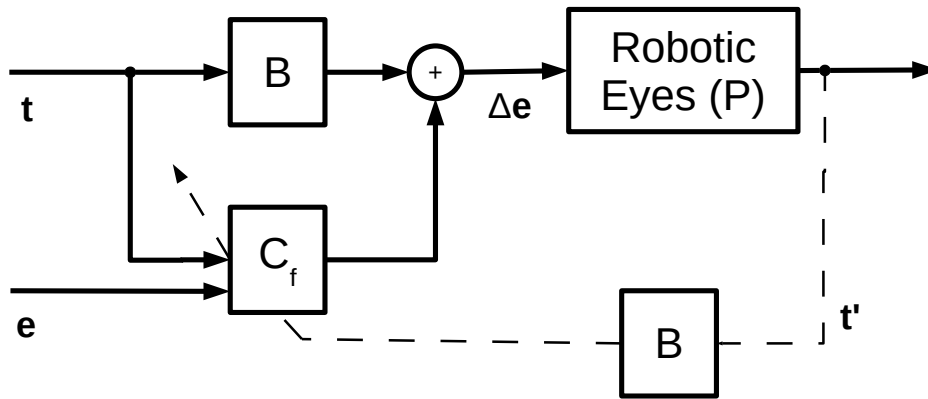


Figure 4.12: FEL schema. The visual position of the stimulus (t) and the current eye position e are converted into a motor command Δe by summing up the contribution of a fixed element B and adaptive element (C_f).

4.4.2.1 Feedback error learning

With this operating scheme, the fixed feedback controller (B) slowly drives the system toward the target and provides a learning cue to a second adaptive controller (C_f) (figure 4.12). The output of B is rectified by the adaptive controller that provides an inverse model of the plant (robotic head). The inputs of the FEL controller are the visual target (t) and the position of the current camera described by the Helmholtz angles (e). The saccade commands Δe are the output of the controller and are sent to the head actuators to move the eyes accordingly. As it can be seen in algorithm 4, it is a unique adaptation loop where the movement of the head generates the signals necessary to modify adaptive controller functions. After the movement, the new visual position of the stimulus t' is converted into a motor error to adapt the inverse controller (C_f). The adaptive filter learns to compensate for the poor response of the fixed feedback control using this approach. According to (Porrill and Dean, 2007) this behavior is similar to the expected response of the cerebellum $C_f = P^{-1} - B$.

The performance of the system is evaluated by the visual error after a saccadic movement. The visual stimulus should be centered in the image after a saccade, but the approximation $P^{-1}(t') \approx Bt'$ introduces an error in the model (see figure 4.12), even though the noise in the visual stimulus is not considered.

4.4.2.2 Recurrent architecture

Similar to FEL, in RA, an inverse linear model (B) is used. However, its function is not the same. In this case, the adaptive controller (C_r) provides a signal of correction to

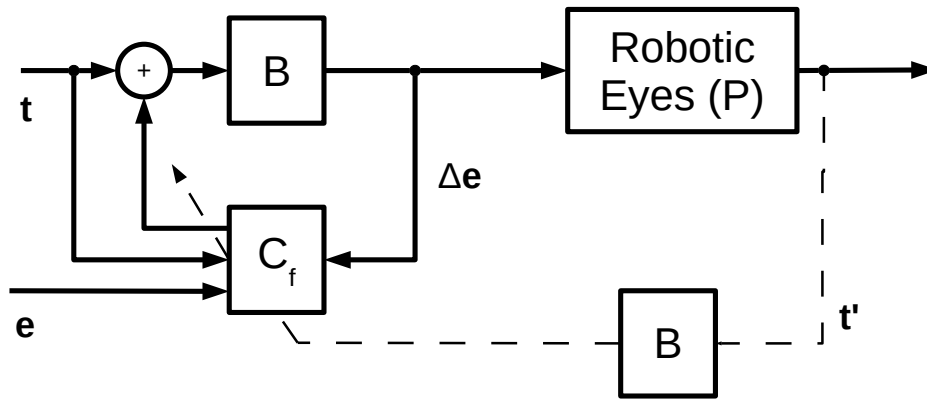


Figure 4.13: RA schema. The visual stimulus cue is processed by the fixed controller B generating a motor command Δe . The inputs of the adaptive controller C_r are provided by the generated motor command Δe , the visual target position (t) and the current cameras angular position (e). The output of the adaptive controller (C_f) is the correction of the input to the fixed controller B generating a recursively process.

Algorithm 4 Feedback error learning adaptation algorithm

Require: B_0, g_f, FEL ; {FEL is an adaptive neural network}

while True **do**

$e \leftarrow getHeadPosition();$
 $t \leftarrow getStimulus();$
 $C_f \leftarrow FEL.feedforward(e, t);$
 $\Delta e \leftarrow g_f B t + C_f;$
 $moveToHeadPosition(e + \Delta e);$
 $t' \leftarrow getStimulus();$
 $FEL.adaptWeights(g_f B t' + C_f);$

end while

the input of the inverse linear controller. A new motor command shift (Δe) is produced by the fixed controller fed by the adaptive correction. This signal has dual functionality, it is sent to the plant to generate the change in the system, and it is in turn used as input in the adaptive controller in conjunction with the visual signal (t) and the current state of the motors (e).

The response of the plant is the teaching signal to adapt C_r . As it can be seen in figure 4.13 and algorithm 5 a loop between the fixed and adaptive elements is created. This loop terminates when the motor command converges to a stable value ($C_r \approx B^{-1} - P$). One of the main advantages of using this approach is that the teaching signal is provided by the sensory error t' , without needing an additional transformation

as in the case of FEL.

Algorithm 5 Recurrent architecture adaptation algorithm

Require: $B_0, g_f, err, n_{max}, RA$; {RA is an adaptive neural network}

while True **do**

$e \leftarrow getHeadPosition()$;

$t \leftarrow getStimulus()$;

$C_r(0) \leftarrow 0$;

$\Delta e \leftarrow g_f B t$;

$i \leftarrow 1$;

while $i < n_{max}$ AND $C_r(i) - C_r(i - 1) < err$ **do**

$C_r(i) \leftarrow RA.feedforward(e, t, \Delta e)$;

$\Delta e \leftarrow g_f B(t + C_r(i))$;

$i \leftarrow i + 1$

end while

$moveToHeadPosition(\Delta e)$;

$t' \leftarrow getStimulus()$;

$RA.adaptWeights(t + C_r(i))$;

end while

4.4.2.3 Inverse linear controller

The previous sections described the controllers that will be considered to execute the saccadic behaviour in a robotic head. Both FEL and RA base their operation on two more basic types of controllers that interact with each other to generate the desired behaviour.

The inverse linear controller is shared to FEL and RA and depends on the type of coding chosen to describe the state of the system. As seen in section 4.4.1.5, the binocular encoding is suggested to be better adapted to saccadic behaviour.

Among the possible alternatives to deal with binocular visual information, the cyclopean image representation is based on combining the information of the position of the stimulus of both images in three coordinates. Two of them (c_x, c_y) are the average values of the stimulus position in each image, and the third is the disparity (d). Thus, it is possible to obtain the coordinates $[c_x, c_y, d]^T$ by the following linear transformation from

the stimulus coordinates in the left (u_l, v_l) and right (u_r, v_r) images (see equation (4.3)).

$$\begin{bmatrix} c_y \\ c_x \\ d \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_l \\ u_r \\ v_l \\ v_r \end{bmatrix} \quad (4.3)$$

In addition, the angular position of the system can be described for the binocular system using the definitions of vergence (θ_{vg}) and version (θ_{vs}) in section 2.3. Thus, given the pan (θ_l, θ_r) and tilt (θ_t) angles of the system and using equations (2.1) and (2.2) the following linear transformations can be defined:

$$\begin{bmatrix} \theta_t \\ \theta_{vs} \\ \theta_{vg} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.5 & 0.5 \\ 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} \theta_t \\ \theta_l \\ \theta_r \end{bmatrix} \quad (4.4)$$

$$\begin{bmatrix} \theta_t \\ \theta_l \\ \theta_r \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0.5 \\ 0 & 1 & -0.5 \end{bmatrix} \begin{bmatrix} \theta_t \\ \theta_{vs} \\ \theta_{vg} \end{bmatrix} \quad (4.5)$$

The inverse linear controller according to the diagrams in figures 4.12 and 4.13 receives the visual signal as an input and generates an actuation command for the plant. Then, using the nomenclature described by the above transformations, the inverse linear model (\mathbf{B}) is the proportionality factor between the visual information and the increment in the angular position of the system expressed by the following equation:

$$\begin{bmatrix} \Delta\theta_t \\ \Delta\theta_{vs} \\ \Delta\theta_{vg} \end{bmatrix} = \mathbf{B}_{3 \times 3} \begin{bmatrix} c_y \\ c_x \\ d \end{bmatrix} \quad (4.6)$$

Using this coordinate system to define the transformations ensures that \mathbf{B} is a 3x3 square matrix, allowing for easy handling. As it will be seen afterwards, there is the option of directly correlating the visual information without any previous transformation with the motor action, being in this case \mathbf{B} a non-square matrix of 3x4 shape.

$$\begin{bmatrix} \Delta\theta_t \\ \Delta\theta_l \\ \Delta\theta_r \end{bmatrix} = \mathbf{B}_{3 \times 4} \begin{bmatrix} u_l \\ u_r \\ v_l \\ v_r \end{bmatrix} \quad (4.7)$$

Choosing as $\mathbf{t} = [c_y, c_x, d]^T$ and $\Delta\mathbf{e} = [\Delta\theta_t, \Delta\theta_{vs}, \Delta\theta_{vg}]^T$, the value of \mathbf{B} can be previously calculated by the following procedure.

1. Place a stimulus in the center of the image of each camera.
2. Generate a motor babbling (Saegusa et al., 2008).
3. Record the angular increment produced in the motors (Δe) and the apparent gain in the position of the stimulus in both images and perform the described transformations (Δt).
4. Repeat this process n times.

After n iterations a set of n pairs $\{t, \Delta e\}$ is defined so that it is possible to build the arrays: $V_{m \times n} = \{t_1, t_2, \dots, t_n\}$ and $\Delta \Theta_{3 \times n} = \{\Delta e_1, \Delta e_2, \dots, \Delta e_n\}$. Depending on the visual stimulus encoding m can be 3 or 4.

Under these conditions, the variation of the visual target position ($\Delta t = t$) in the images can be correlated with the motor command (Δe) generated by the motor babbling. To do this, and considering a linear relationship exists between both terms, it is possible to use a linear regression (section 3.3.2) and directly determine the value of B (equation (3.6)).

$$B = \Delta \Theta V^T (V V^T)^{-1} \quad (4.8)$$

4.4.2.4 Adaptive controller

The number of outputs and inputs of each adaptive controller, according to the schemes proposed in figures 4.12 and 4.13, is different. However, the method of implementing them is the same. If the adaptive controller were linear, it would probably not be necessary because the fixed controller would be sufficient to model the problem. Therefore the tool used to learn this controller from the dynamic interaction with the environment is based on adaptive networks. (section 3.4). A feedforward single-layer neural network is used to implement these controllers. The activation functions shaping the hidden layer of this network are trigonometric basis functions (section 3.4.3).

The number of inputs and outputs of each controller is different, as can be seen in the diagrams of figures 4.12 and 4.13. Therefore, even if the type of network is the same, the structure defined by the inputs and outputs and the units in the layer is not. In the case of FEL, C_f has a six-dimensional input composed of the visual stimulus (t) and the current eye position (e). It is the case as long as the cyclopean representation of the visual stimulus is applied (equation (4.3)). If the stimulus coordinates are directly used in each image, the number of inputs in the network would be 7. This high dimensionality precisely defines the type of structure that the hidden layer of the network should have.

Table 4.4: Tombatossals head parameters setup according to morphology definition in section 4.3

Left				Right			
q_2	q_3	q_4	q_6	q_2	q_3	q_4	q_6
0	13.5 cm	0	-4.8 cm	0	-13.5 cm	0	-4.8 cm
f	s	w	h	f	s	w	h
5 mm	4.65 μm	1024 px	768 px	5 mm	4.65 μm	1024 px	768 px

Using an RBFNN, with a distribution of neurons in the hidden layer similar to that used in the comparison between monocular and binocular coding (table 4.3), the number of neurons in the hidden layer would rise to $7^7 = 823543$ in the worst case and $3^7 = 729$ at best. This high number of neurons is one of the main reasons radial base functions are not used and are replaced by trigonometric base functions. It is enhanced when the adaptive controller for the case of RA is considered. C_r has a nine-dimensional input composed of the stimulus cue (t), the current eyes position (e) and the upcoming eyes movement (Δe).

In the case of both FEL and RA, the number of neural network outputs is three. Therefore, apart from the number of hidden layer neurons and the algorithm for adapting the weights, the architecture of the neural networks of the adaptive controllers of both proposals are defined.

4.4.2.5 Experimental setup to estimate FEL and RA controllers

A set of experiments are performed to extract the advantages and disadvantages of using the two proposed controllers to implement saccadic behaviour in a robotic head. One of the available robotic systems in our laboratory called *Tombatossals* is used to accomplish these tests. *Tombatossals* is a humanoid torso endowed with a mechatronic figure 4.15a head ¹ and two multi-joint arms Mitsubishi PA10 (Bompos et al., 2007).

Two cameras ² with a resolution of 1024x768 pixels are mounted. These cameras allow the capture of colour images at a rate of 30 Hz. Comparing figures 4.4a and 4.15b it is clear that the configuration of this robotic head follows a Helmholtz design; therefore, its morphology can be defined according to the parameters provided in section 4.3. The values of these parameters can be seen in table 4.4.

¹The robotic head model is Robosoft TO40

² The camera model is Imaging Source DFK31AF03-Z2

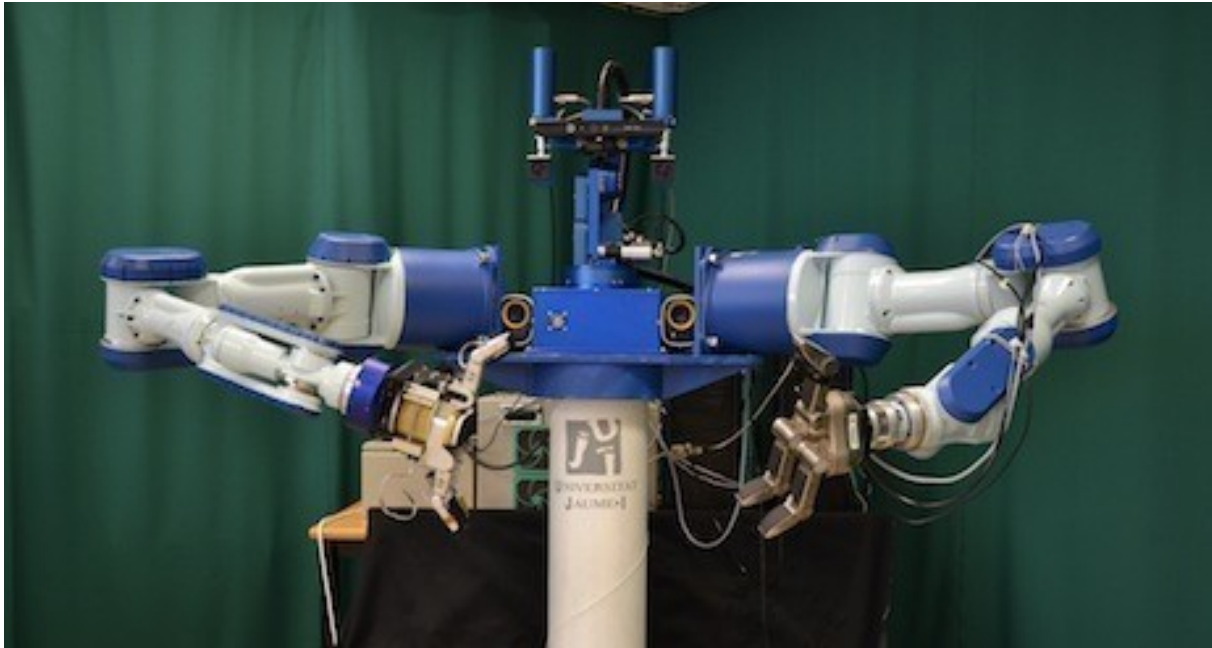


Figure 4.14: Tombatossals Robot.

In the head of Tombatossals, Microsoft Kinect® is installed as a supplementary vision system. The cameras can be actively operated employing a joint that rotates the whole system called pan, which will not be used. All other active joints are equivalent to the Helmholtz configuration arrangement: a common tilt (θ_t) and two independent pan motors (θ_l) and (θ_r). The rotation detail of these joints can be seen in figure 4.15b.

The baseline between the cameras is about 27 cm. The centre of rotation of the camera does not coincide with the projection plane. Therefore, when the head is moved, there is a rotation-translation of the projection plane of the camera. Due to this translational component, the visuo-oculomotor transformation depends on the focal length of the camera, and it is present virtually in every robotic system and also in the human eyes (Chinellato et al., 2012b). This circumstance is reflected in the morphology of the head through the parameter q_6 (table 4.3), and the estimation of its value is performed in an experimental approach.

The proposed controllers FEL and RA are compared using the 3D simulation of Tombatossals' robotic head. Nevertheless, first, a dataset is generated following the procedure described in section 4.4.1.4. In this case, 8205 points are considered. The projection of the spatial distribution of these points can be seen in figure 4.16. In table 4.5, the maximum and minimum values of the variables used to adapt these controllers are shown.

Secondly, the value of the inverse linear controller (B) has been determined for both architectures following the procedure described in section 4.4.2.3. This value

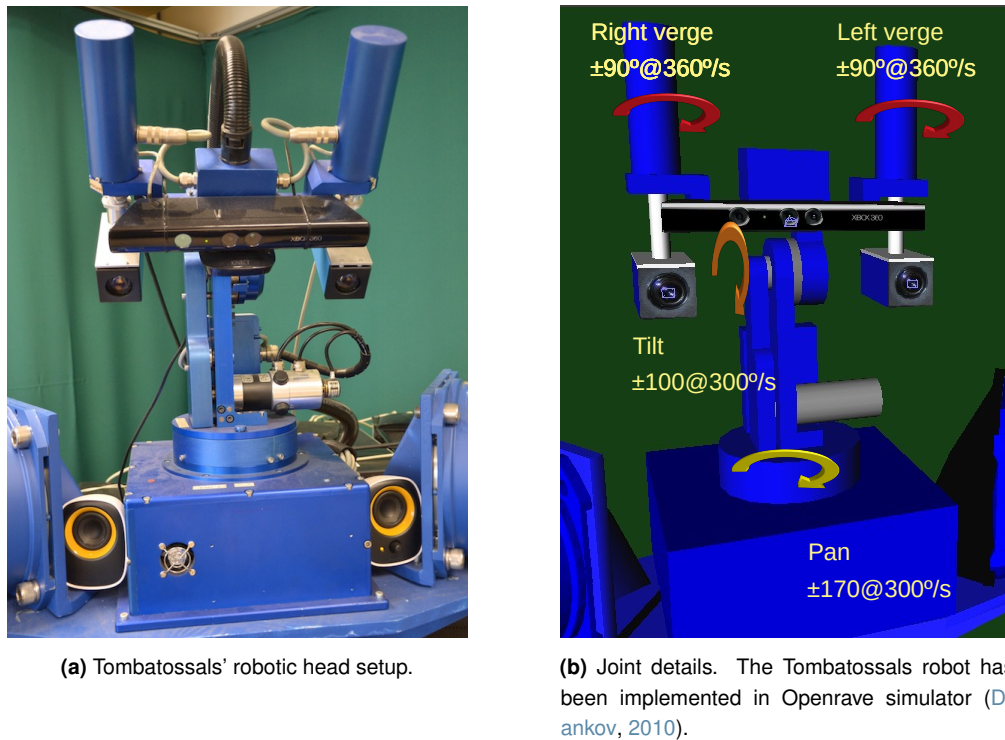


Figure 4.15: Tombatossals' visual robotic system details.

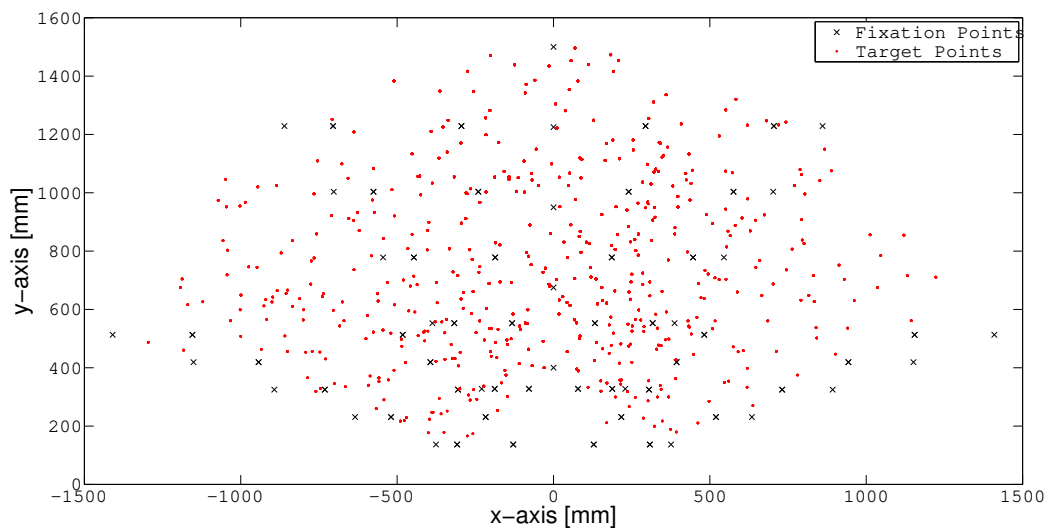


Figure 4.16: Distribution of the initial fixation points (black crosses) and target points (black dots) used for creating the dataset.

is multiplied by a factor g_f , which takes different values in a range of $[0.5, 1.4]$ and helps to know the influence of the inverse linear controller on the performance of both architectures. In this particular case, B is expressed in cyclopean coordinates and has a value of:

$$B = \begin{bmatrix} 0.4887 & -0.0002 & 0.0009 \\ 0.0012 & 0.4292 & -0.0007 \\ 0.0016 & 0.0005 & 0.4294 \end{bmatrix}$$

Table 4.5: range and distribution of the input data provided by the dataset.

Input	Min. Value	Max. value	Mean	Std. Dev.
c_y [pixels]	-382.6	372.6	-1.4	180.91
c_x [pixels]	-503.4	507.7	7.3	240.8
d [pixels]	-560.4	625.1	-17.1	204.9
θ_t [degrees]	-70.25	70.25	-0.33	23.91
θ_{vs} [degrees]	-70.21	70.22	-0.50	24.62
θ_{vg} [degrees]	3.50	38.19	17.39	8.87

In these experiments, the neural networks are trained by adopting an online approach that can be replicated in the robot. Specifically, the incremental sparse spectrum Gaussian process regression (ISSGPR) described in section 3.5.2 is used. For each sample of the training, the eye displacement (Δe) is calculated by multiplying the visual position of the stimulus (t) by the matrix gain (B). The adaptive controller C_x corrects this achieved eye shift. This correction is direct in the case of FEL; however, in the case of RA, there is an adaptation loop that uses the corrected estimation of the displacement of the cameras together with the initial visual stimulus and position to adapt the controller iteratively. It is considered a stopping criterion either the visual error achieved by less than one pixel or when 30 iterations are reached.

The hidden layer of the neural networks composing the adaptive controllers is formed by 500 random features (section 3.4.3). The weights are adapted using the ISSGPR algorithm (section 3.5.2).

K-Fold cross-validation with K=5 is used to train and test both architectures. In particular, the training set consists of 6564 (4/5 of 8205) and the test set has 1641 (1/5 of 8205) elements. These elements are chosen randomly in each pass of the cross-validation, although it should be noted that none of the elements of the test set has been previously used to generate the training set.

Although the controllers have been trained and tested in cyclopean coordinates to express the results and analyse them in a more intuitive way, the transformations expressed in section 4.4.2.3 has been used to convert the different errors in the Euclidean visual distance of the target projection to the center of the image after the saccade.

$$MRVE = \frac{\sqrt{u_r^2 + v_r^2} + \sqrt{u_l^2 + v_l^2}}{2} \quad (4.9)$$

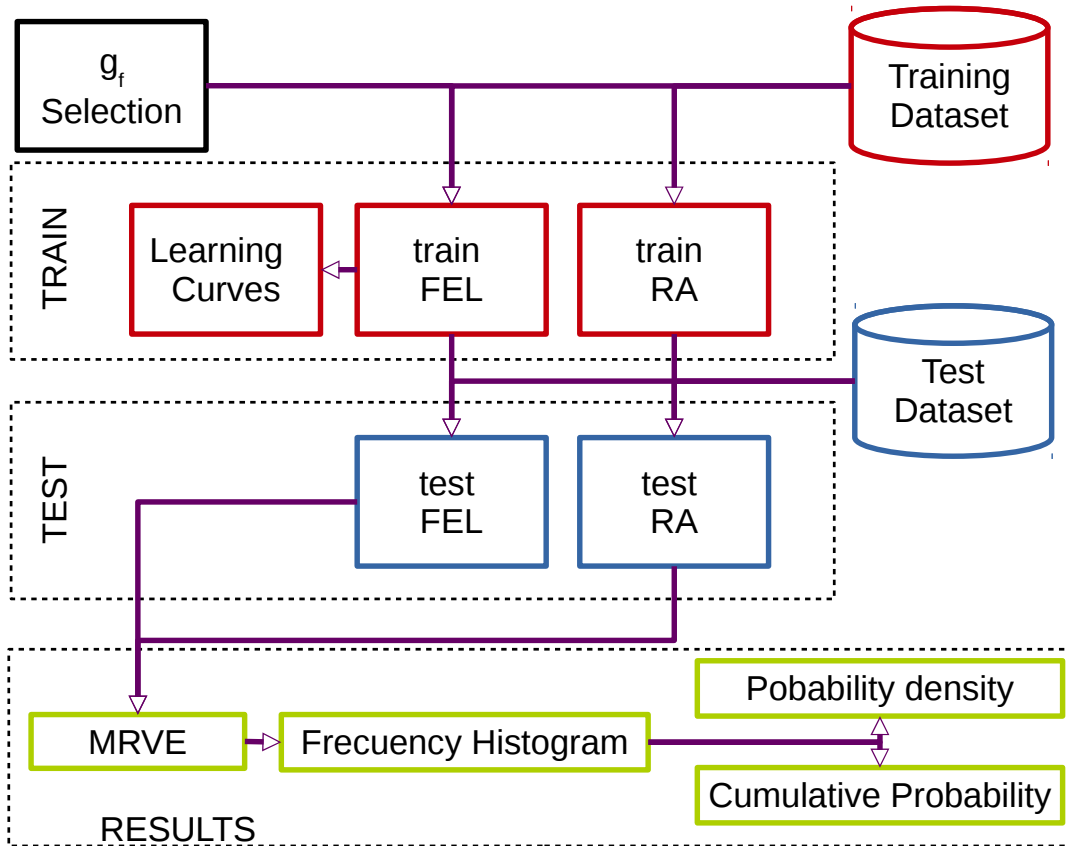


Figure 4.17: Diagram of the process followed to generate the data used to compare the performance of the FEL and RA controllers.

This magnitude is used to show the progress of the training process and also to obtain the distribution of the saccade prediction error in the test dataset.

As it can be seen in the scheme of figure 4.17, the procedure for generating a result that serves to compare both controllers is based on the selection of a gain factor that is used with the training set to generate two trained controllers. The evolution of this training is represented by the learning curves, where the evolution of the accumulated mean visual error obtained to adapt the controller in each iteration is evaluated. Thus, given the MRVE in iteration i calculated from the output of the neural network that constitutes the adaptive part of the controller, the training curve for each iteration i is obtained with the following expression:

$$l_c(i) = MRVE_{i-1} + \frac{MRVE_i - MRVE_{i-1}}{i} \quad (4.10)$$

Once the controllers have been trained, their ability is evaluated using the dataset test. For each stimulus and initial position of the head, a saccade is performed. The obtained MRVE after the saccade movement is checked. This error should ideally be

zero. From these MRVE values, it is possible to construct a frequency histogram. These frequencies are used to obtain the approximate probability density function and the cumulative probability.

4.4.2.6 Simulation results

A summary of the results obtained in the training process using the FEL controller can be seen in figure 4.18a. The value of g_f influences the learning behaviour. When it is closer to 1.0, the training process evolves to lower error values. If only the learning curve were considered, the best value of g_f would be 1.0. Nevertheless, when the trained system is checked with a test dataset, the probability density distribution of the visual error obtained (figure 4.18b) suggests that the best value of g_f is 1.2 instead of 1.0.

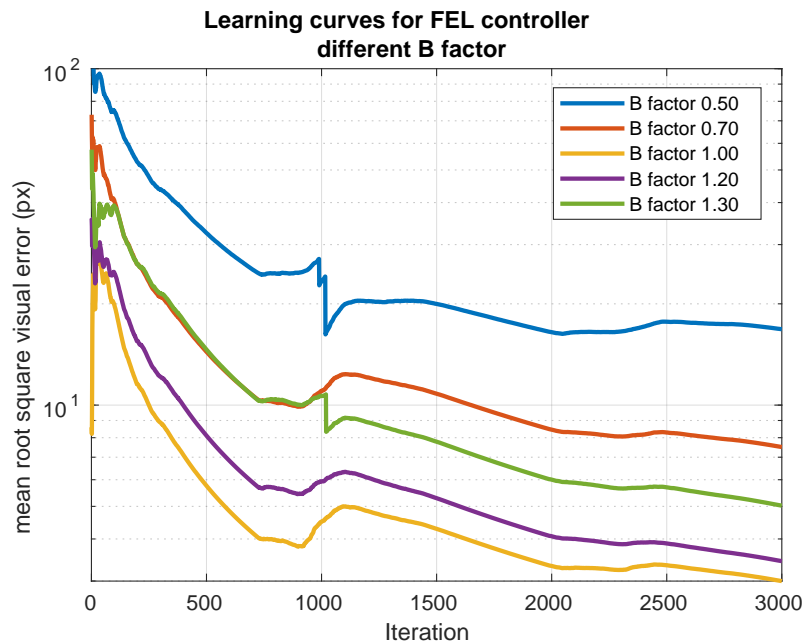
On the contrary, in the case of RA, the behaviour of the controller trained with the test dataset (figure 4.19b) confirms the trend shown in the training curves (figure 4.19a) where the best results are obtained for a value of g_f equal to 1. Thus, from the observation of the training curves in the RA controller, it can be deduced that the factor g_f affects strongly this architecture.

The first intuitive approach to compare these two controllers is to compute the average visual error for each test dataset and to study how this error varies with the value of the factor g_f . These results can be seen in figure 4.20a. The mean visual error suggests that the choice of the factor g_f affects the FEL controller significantly; in contrast, the RA controller has a lower sensitivity to this value. However, this difference is not so marked when the dispersion of results is considered.

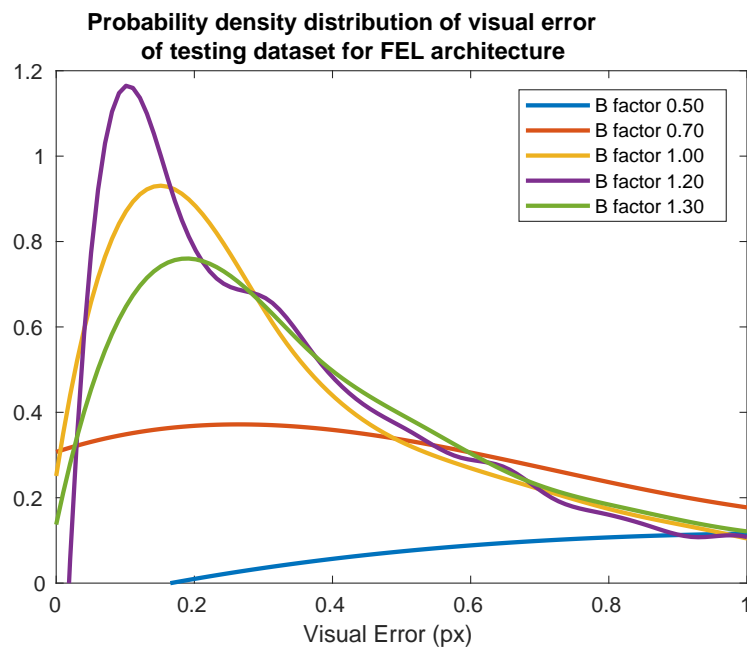
Using FEL controller, the best performance was achieved with gain factor set to 1.2. In this condition, the mean error was (1.07 ± 1.57) pixels. In the RA, the best performance was reached with a gain factor set to 1.0. In this context, the mean error was (0.47 ± 0.72) pixels.

Although these values seem to suggest that RA performs better than FEL, the probability density distribution of the error obtained for each method needs to be assessed in detail to establish a valid conclusion. Figure 4.20b reveals that the probability density function obtained for FEL and RA is at best far from a Gaussian distribution (figure C.1). Therefore, considering the mean and standard deviation only does not seem to be the most appropriate option for comparing both distributions.

According to figure 4.20b, the most probable values of the visual error obtained are 0.08 pixels in the case of RA and 0.10 pixels in the case of FEL. Therefore, the

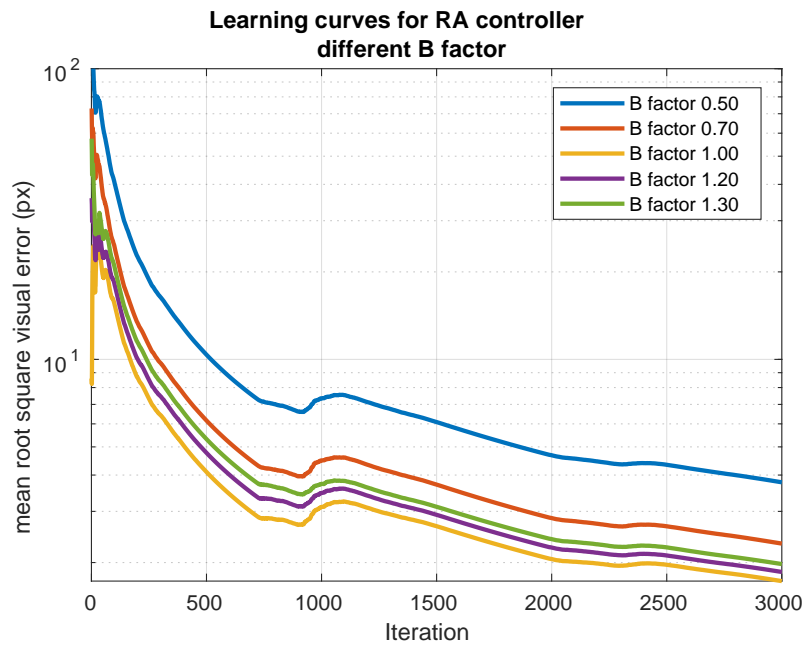


(a) Training curves for different values of the B factor (g_f) using FEL architecture. Each curve is the mean of the five training curves for each K-Fold iteration. In order to simplify the visualization, only the first 3000 iterations have been considered, although the trend has hardly changed up to 6500.

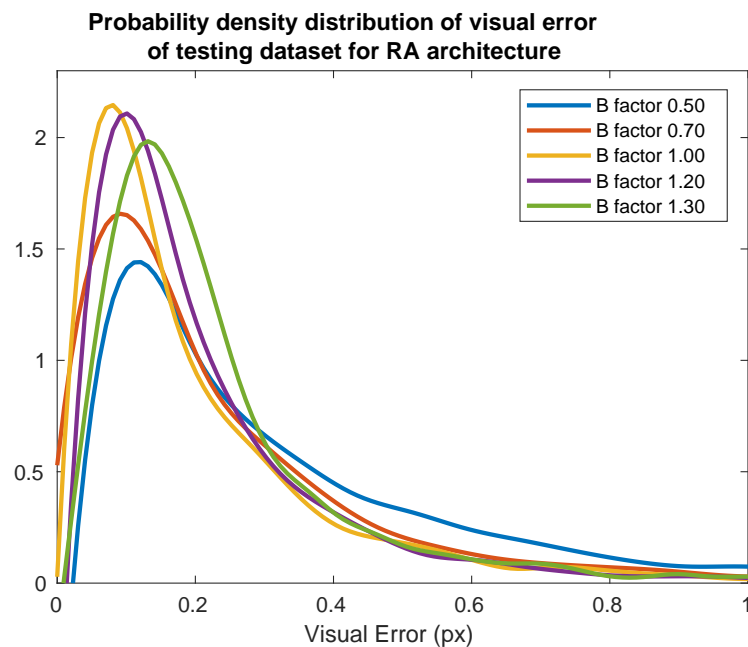


(b) Probability density distribution of the visual error generated from the testing dataset using the trained FEL controller. The visual error is limited to 1 pixel on the x-axis for visualization purposes. The area under each curve considering the full range of x is 1

Figure 4.18: Training and testing curves fro FEL controller

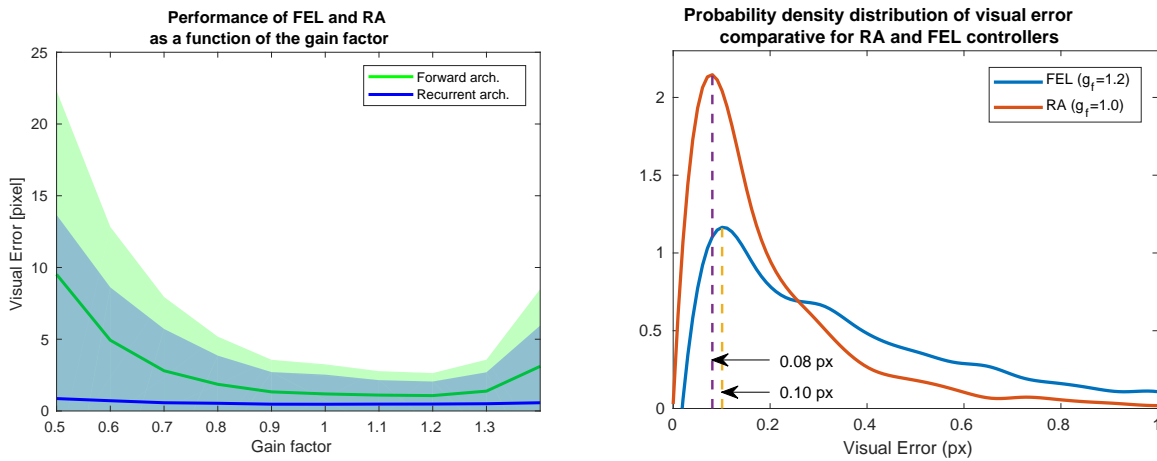


(a) Training curves for different values of the B factor (g_f) using RA architecture. Each curve is the mean of the five training curves for each K-Fold iteration. In order to simplify the visualization, only the first 3000 iterations have been considered, although the trend has hardly changed up to 6500.



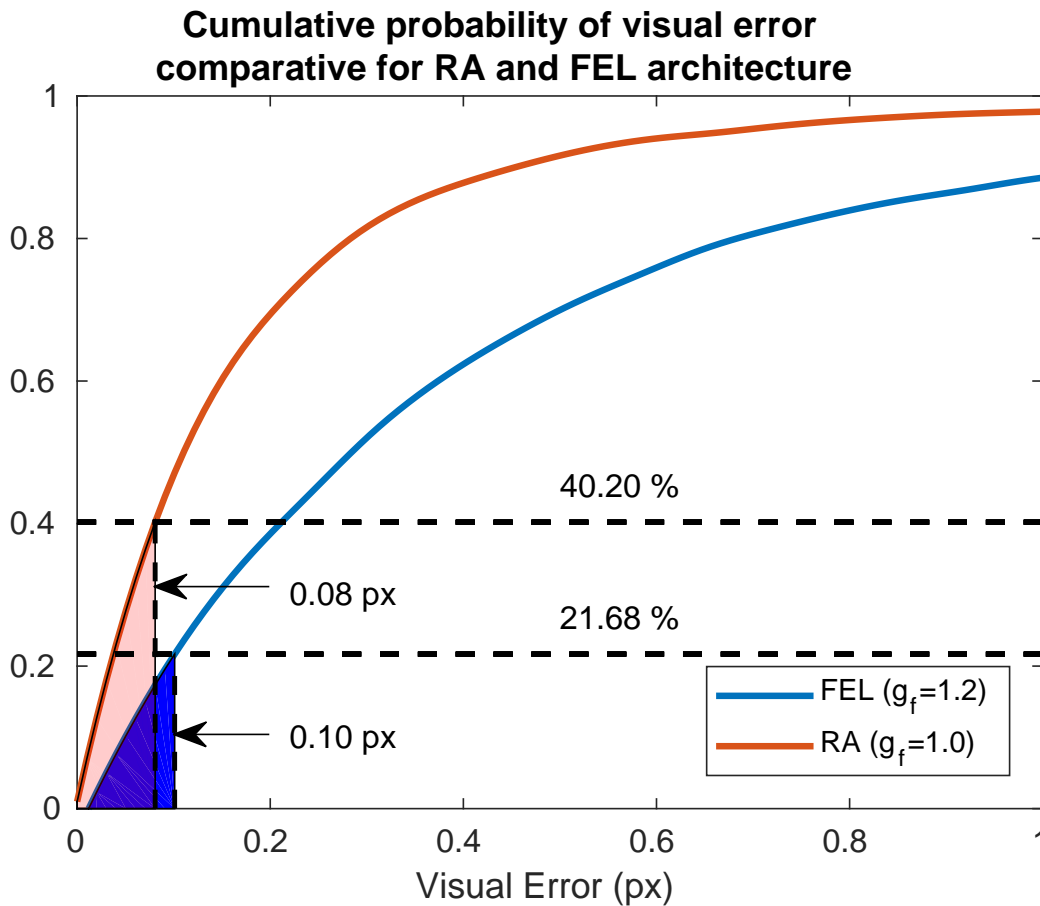
(b) Probability density distribution of the visual error generated from the testing dataset using the trained RA controller. The visual error is limited to 1 pixel on the x-axis for visualization purposes. The area under each curve considering the full range of x is 1

Figure 4.19: Training and testing curves from RA controller



(a) Mean visual error of FEL and RA depending on the gain factor applied to alter the inverse linear controller. The areas represent the standard deviation of the visual error. Both areas have as their lower limit the zero of the y-axis.

(b) Probability density distribution of the visual error generated from the testing dataset for the two better results of FEL and RA controllers according to g_f variation



(c) Comparison of the cumulative probability of the visual error distribution obtained from using the FEL and RA controllers with the test dataset. The maximum values of the probability density distribution are marked for each controller and the corresponding accumulated probability at these points

Figure 4.20: Comparative between FEL and RA controllers

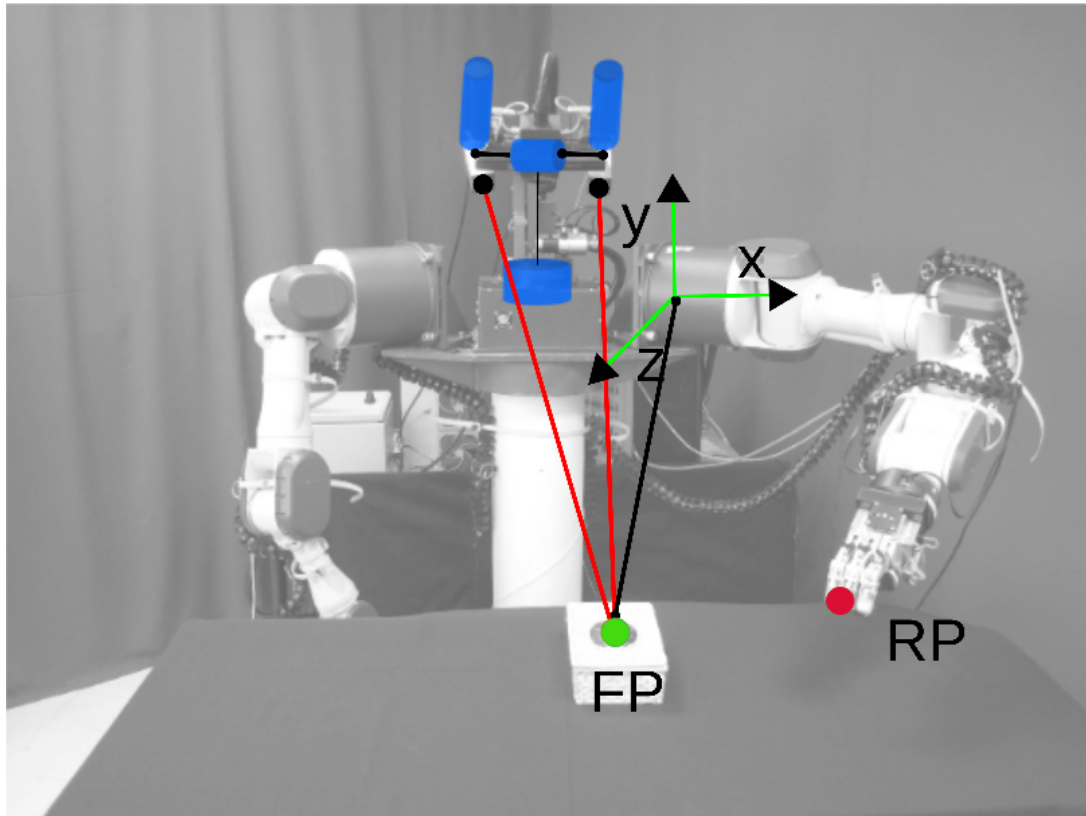


Figure 4.21: The UJI humanoid torso: Tombatossals. Blue cylinders represent the four joints of the head. The configuration of the head creates an implicit representation of the fixation target (FP) that can be made explicit through triangulation (red lines). Reaching a target (RP) requires converting the gaze direction into a 3D point of the Cartesian space centred in the arm's shoulder.

value of the arithmetic mean of the visual errors overestimates the most likely error. Furthermore, this result —considering the maximum of the probability density curve of each system— does not suggest a clear difference in the goodness of RA versus FEL.

A representation of the cumulative probability of the two cases (figure 4.20c) is necessary to appreciate a difference. As it can be seen in figure 4.20c, the FEL controller approximately 20% of visual error obtained in the estimation is below the maximum probability, whereas this value rises to 40% in the case of RA. In addition, this curve allows to compare the probability to obtain a visual error below 1 pixel, indicating, in the FEL controller, it is approximately 88%, and it is almost 99% in RA. These results suggest that at least in simulation, the RA controller performs better than the FEL controller. However, both give very precise approximations bearing in mind the image size, which is 1024 x 720 pixels.

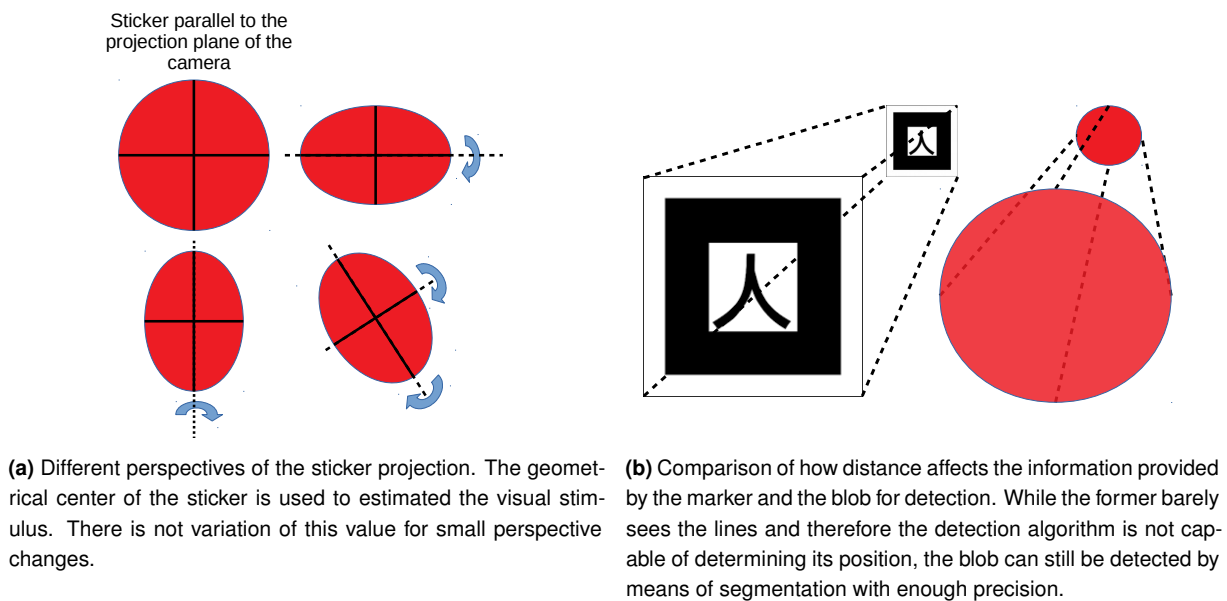


Figure 4.22: Different types of real stimulus

4.4.2.7 Robots results

It is necessary to solve several previous problems not considered in the simulation to replicate experiments in the real robot. In the simulation, the environment and the stimuli generating the saccades are created simply from virtual points in the surrounding space of the robot. The position and location of these points are as precise as required. In contrast, in the real robot, the points of attention are physical and therefore subject to external factors that condition their precision. Fortunately, thanks to the robot's arms, it has accurate access to the surrounding environment (figure 4.21). Therefore, a red circular sticker with a diameter of 2 cm is placed at a known point in the kinematic chain of one of the arms to generate the stimuli and then trigger the saccades' execution. This reference point is the fingertip of one of the hands (red dot in figure 4.21). In this way, it is possible to solve the inverse kinematics to move this point along a Cartesian space in front of the robot. During this displacement, the pose of the end effector (robot's finger) where the sticker is fixed, is kept in such a way that the plane of the sticker is parallel to the plane of the cameras planes as much as possible to minimise the change of perspective.

A blob detector is used to perceive the sticker. It provides the projection area of the sticker within each image. The input of the proposed controllers is two coordinates for each image of the stimulus. The geometrical centre of the projected circle is used to select these. The error introduced by this approach should not be significant as long as there is not a considerable variation in perspective (figure 4.22a). First, an attempt was

made to use a marker belonging to the Artoolkit³ library, which offers more precision. However, due to the requirements defined by the robot's finger dimensions, the marker size could not exceed 2 cm.

Nevertheless, with this size, because of the decrease in the resolution of the lines with distance, the marker's detection capacity was considerably reduced (figure 4.22b). On the other hand, although with more significant noise, detection could be maintained with the blob. The distances considered are defined by the range of the robotic arm with the appropriate arrangement to display the marker in front of the cameras. In this case, the maximum distance is 1.16 m. The Cartesian positions of the arm are distributed on a grid of 0.4x0.4x0.3m.

For both control architectures, the adaptive networks are trained using the same procedure applied in simulation. The stimulus (red sticker in the robot fingertip) is placed in 125 positions of the defined Cartesian space. These points are the seed to start in 26 different gaze directions. From each position of the head, the robot performed a saccade toward the target and then used the post-saccadic visual error to train the network. The total number of points acquired was 3250. After training, the controllers are tested with 1000 randomly generated points within the defined Cartesian space. The same value of B has been used as in the simulation for both the training and the test procedure. In addition, the value of $g_f = 1.0$ and $g_f = 0.5$ are tested to confirm whether the gain factor g_f affects the result in the same way as it does in simulation. Finally, the post-saccadic visual error is used as it had been previously done in simulation (equation (4.9)) to evaluate the outcomes, thus enabling a comparison of both results.

In figure 4.23, the visual error histograms for the test dataset for each controller and gain factor can be seen. In addition, the data obtained by grouping the visual error in the histograms have been adjusted using a known probability density distribution. This function is generalized extreme value distribution (GVA) (De Haan and Ferreira, 2007). This distribution can be used because there is a set of identical and distributed random variables such as the visual error that when the number of iterations tends to infinity, the value of visual error should tend to a minimum, in this case, zero. As can be seen in figures 4.23a to 4.23d the fitting is suitable enough to represent the pattern of visual error behaviour in this particular case. Using these curves to represent the results allows determining the value where the visual error presents a higher density of probability for each of the cases. As can be seen in (figure 4.24a), the better results are obtained for RA controller with gain factor 1.0. As has been done in simulation, it is possible to build cumulative probability curves for each one of these distributions (figure 4.24b). This type of function can also explain the distribution of the results beyond

³<http://www.hitl.washington.edu/artoolkit/>

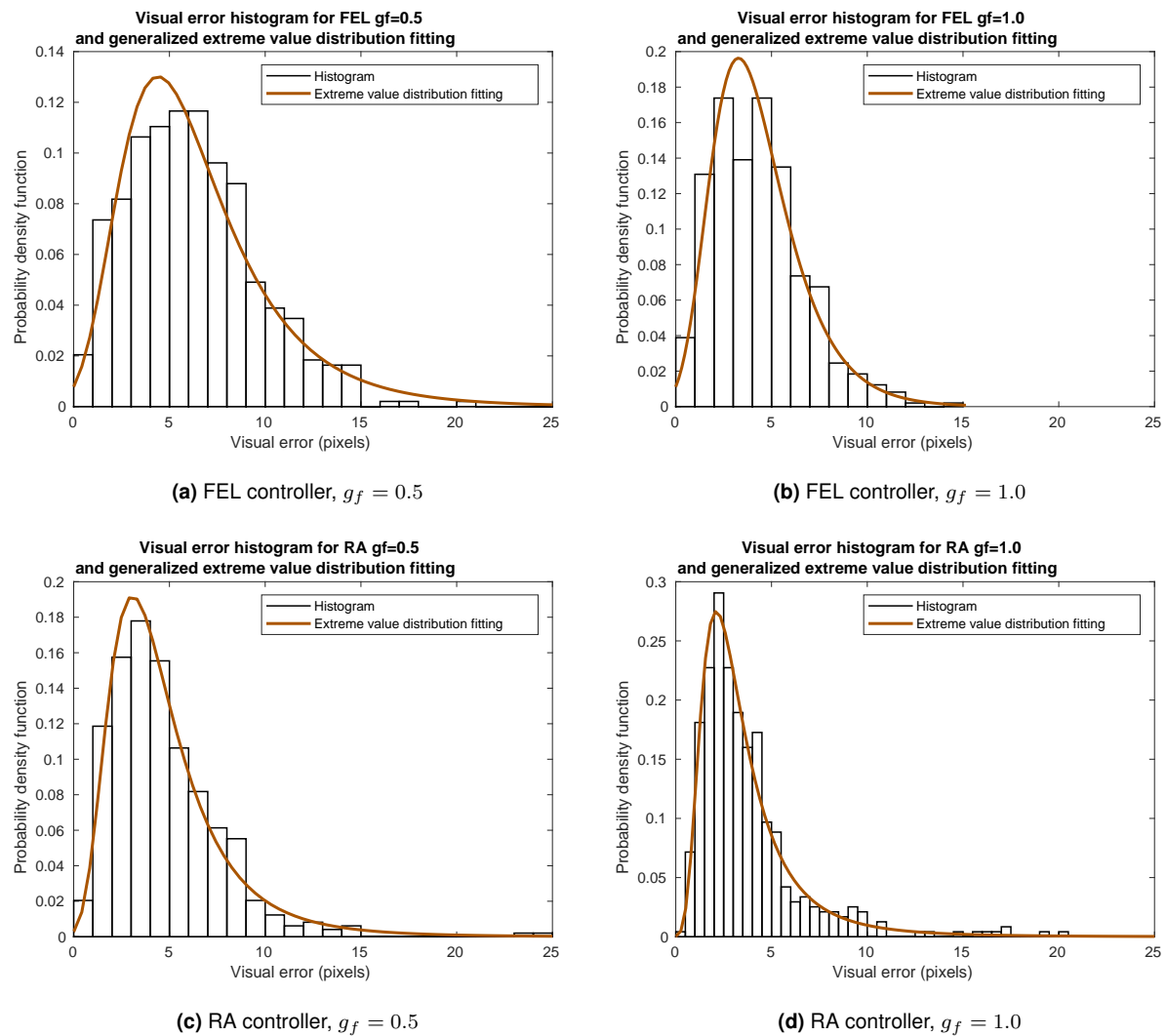


Figure 4.23: Histograms generated from the visual error obtained by using the controllers learned in the robot applied to the testing space. The curves represent the probability density distribution fitted to the results obtained using the generalized extreme value distribution as model.

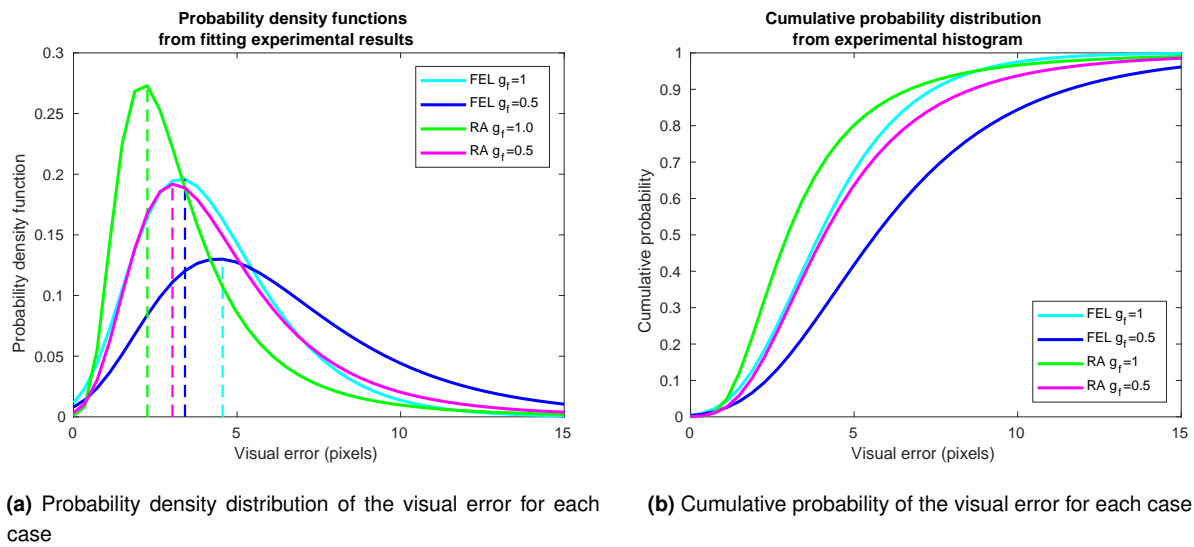


Figure 4.24: Probability density distribution and cumulative probability curves of the visual error estimated with the robot results

Table 4.6: Summary of the results in the robot extracted from the analysis of the fitting curves obtained

Controller	$MRVE_{max}$	$\mathcal{P}(MRVE < MRVE_{max})$	$\mathcal{P}(MRVE < x) < 0.5$
FEL $g_f = 1.0$	3.408	0.399	4.175
FEL $g_f = 0.5$	4.558	0.361	5.708
RA $g_f = 1.0$	2.258	0.322	3.024
RA $g_f = 0.5$	3.024	0.305	4.175

the maximum probability density value.

In table 4.6, some of the information gathered from these curves is summarised. The value of the visual error in which the probability density is maximum is shown for each controller in the first column. As can be seen, the results for both **FEL** and **RA** are in accordance with the trend of those obtained in simulation. Although the numerical result is ostensibly more significant in the real robot, the sorting by performance is the same. Thus, according to this value, **FEL** with half the gain value has worse results since the gain value has more influence than in the case of the **RA** controller. On the contrary, the **RA** controller with a gain value of 1 has better results. This arrangement can be seen graphically in figure 4.24a.

In either case, both **FEL** and **RA** with gain 1 have a similar behaviour, being the accumulated probability for a lower error of 7 pixels around 0.8.

Table 4.7: Performance of the binocular visual-oculomotor transformation in the literature. This information has been gathered from the corresponding reference

Approach	Resolution	Error (pixels)	% Err.
FEL	1024x768	4.32 ± 2.34	0.48%
RA	1024x768	3.70 ± 2.75	0.56%
(Forssén, 2007)	640x480	5.50	1.15%
(Bruske et al., 1997)	512x512	2.50	0.49%
(Schenck and Möller, 2004)	1x1	0.06	6.00%

In the literature, several studies are available where this visual error is estimated for different controllers. However, the distribution of visual error is usually not considered, and an average value is used to estimate its performance. Therefore, in order to be able to compare these cases, the average value is calculated for both controllers. The comparative results with other approaches can be seen in table 4.7.

This comparison is merely indicative since the distribution of error obtained in the different approaches is not available. In addition, other factors must be considered, such as the distance between the cameras that, in the particular case of Tombatossals, usually double the standard value for this type of head, which are not taken into account. Nevertheless, as each robotic system has been tested with different cameras, in the absence of more data, it is possible to calculate (last column of table 4.7) a ratio between the average visual error obtained in the different tests and the value of the resolution of the smaller dimension of the image. This ratio allows us to compare the results obtained by abstracting them from the specificity of the cameras.

4.4.2.8 Conclusion

Two adaptive controllers have been proposed and tested in the preceding sections to generate saccadic behaviour in a robotic binocular system. The results obtained in the simulation and the real robot achieve rates of less than 1% of visual error for both controllers, bearing in mind the cameras' resolution. Furthermore, it has been found that the experimental data visual error in the test set approximates a generalised extreme value distribution. Although this point does not seem to have been taken into account, the comparative results with other methods of solving this problem that exists in the literature suggest that RA gives better experimental results in the robot. The improvement of RA concerning FEL is achieved by the estimation loop that uses RA to boost the results of the adaptive controller. This loop makes this architecture very stable about the fixed controller estimation as it can be seen in figure 4.20a but at the

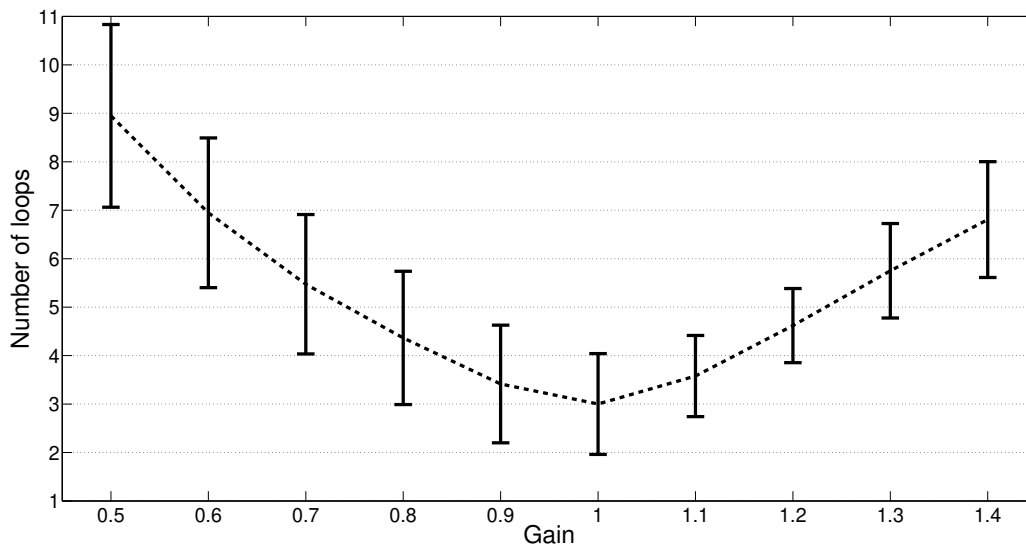


Figure 4.25: Number of loops (mean and standard deviation) in the RA as a function of the gain

cost of increasing the number of iterations of this loop as it is shown in figure 4.25. The controller learns this factor is less important because the number of iterations tends to be 1 when the controller is trained. However, the training time varies depending on the value and the precision of the gain estimation. However, this is not the case with the FEL controller that is more sensitive to the value of the gain because just one pass is given in each iteration loop, allowing a constant time calculation.

In any case, the results in simulation and the robot suggest that both proposed controllers can generate the desired saccadic behaviour. Furthermore, as indicated in (Porrill et al., 2004), the design of these controllers is bio-inspired by the behaviour of the cerebellum, which is essential for triggering eye movement during saccades (section 2.5).

4.5 Environment

In section 2.1, when the cycle of perception and action is described, a reference is made to the environment. Intuitively, from the point of view described there, the environment was an entity outside the system considered a source of information and a receiver of actions modifying its state. In a system with a predefined model of the environment, the way it is adapted is decided when the model is created. The changes that may occur in this environment and the established model has not foreseen may lead to inappropriate responses or actions towards the environment. The proposed saccadic system is intended to adapt to environmental conditions and modify its internal

model to suit these circumstances. The environment, therefore, plays an important role, and it is necessary to define and characterize it in some way outside the robotic system in question.

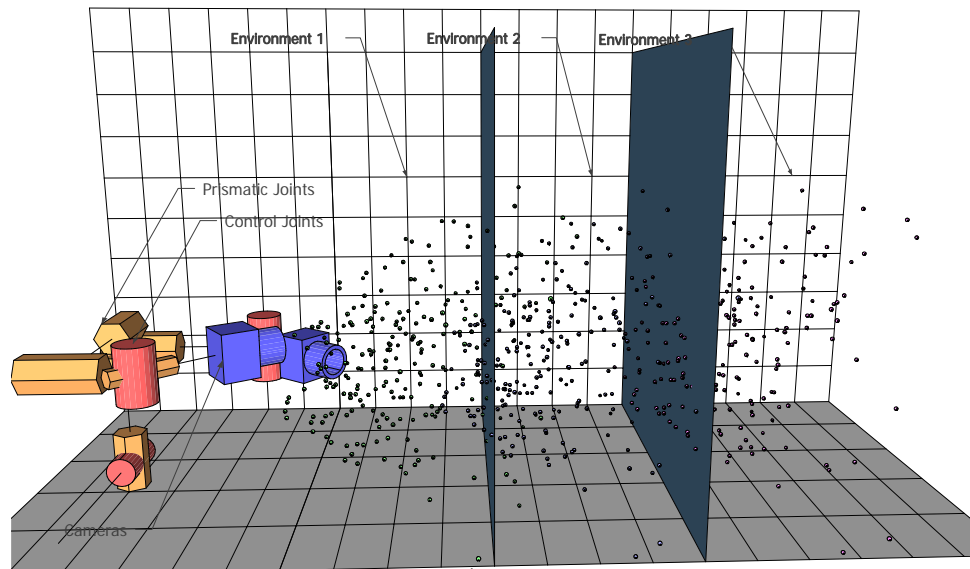


Figure 4.26: Example of how the environment can be established in various forms.

4.5.1 Environment definition

A first attempt to define the environment would be everything that is not the system. This definition, however, is too general. For example, if the points in the plane proposed in figure 4.16 are considered, everything that is not the system would be the infinite points that are in all directions where the robot can perform a saccade. However, the space for the positions of the visual stimuli where a saccade can be triggered is limited.

The limits defining the environment are its distinguishing characteristics. For example, in figure 4.26, a set of virtual stimuli can generate multiple environments that present different properties affecting in different ways the ability of the robotic system to adapt.

In conclusion, the environment has its entity and therefore, the parameters that define it must be established in order to determine how changing these factors affects the robotic system that is adapting to it.

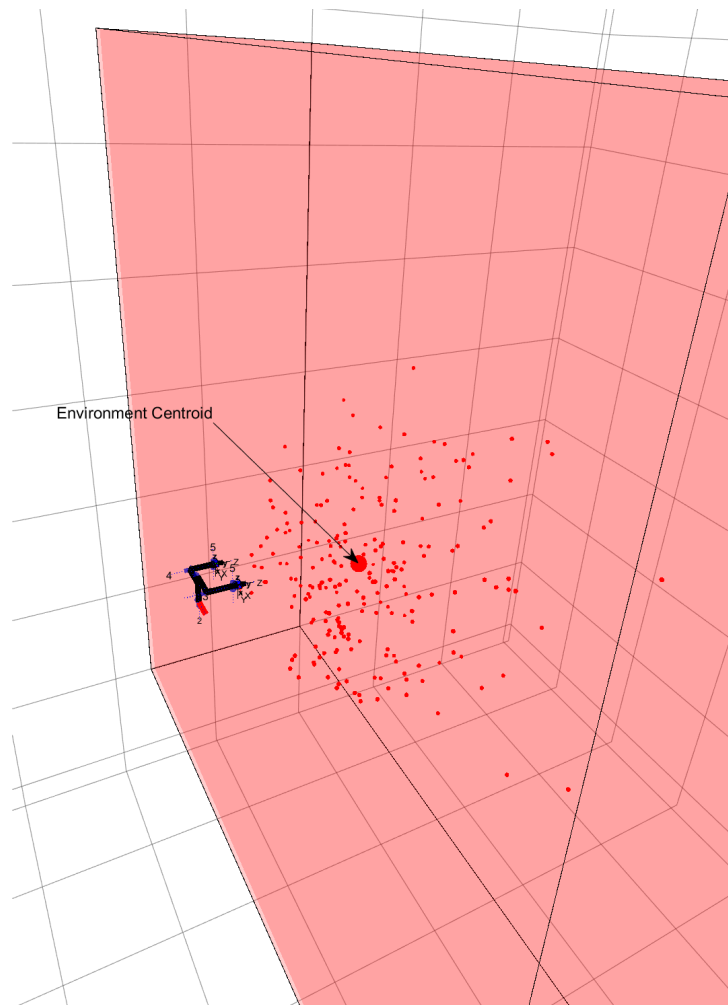


Figure 4.27: Parallelepiped region defined as environment.

4.5.2 Parameters defining the environment

In this case, the environment is defined by a three-dimensional region, where the visuo-oculomotor system can capture stimuli enabling it to perform saccadic movements. This portion of the space can be delimited. The boundaries establish the area where a stimulus can be captured. This volume can be defined in multiple ways, from an irregular volume to a set of parallel planes defining a parallelepiped (figure 4.27).

For simplicity, this is how the environment is defined for the proposed robotic system. A set of parallel planes delimit the region in front of the robotic head in which the saccadic behaviour is to be implemented. In addition, a centroid is defined that corresponds to the average value of the stimuli that can be generated within this volume (figure 4.27).

4.6 Conclusion

Transferring the perceptual model of living beings to a robotic system involves describing three component factors constituting it:

The morphology of the robotic system in which a widely used configuration typology (Helmholtz) has been started and parameterized in a systematic way to be able to generalize the concepts that are developed for a particular robotic system to a broader set of systems that can follow this configuration.

The second aspect is the implementation of the behaviour. Throughout this chapter, several ways of implementing and characterizing the saccadic behaviour from the robotic point of view have been presented, performing tests in a particular robot suggesting the adequacy of the results obtained in simulation.

Finally, the environment factor in an adaptive system is highlighted in the corresponding section of this chapter, describing what should be understood by the environment and how the parameters describing it are defined.

4.7 Publications supporting this chapter

- **Antonelli, M., Duran, A.J., Chinellato, E., del Pobil, A.P., 2015**, "Learning the Visual-Oculomotor Transformation: Effects on Saccade Control and Space Representation", *Robotics and Autonomous Systems*, Vol. 71, pp. 13-22. DOI: 10.1016/j.robot.2014.11.018.
- **del Pobil, A.P., Duran, A.J., Antonelli, M., Felip, J. Morales, A., Prats, M., Chinellato, E., 2013**, "Integration of Visuomotor Learning, Cognitive Grasping and Sensor-Based Physical Interaction in the UJI Humanoid Torso", in *Designing Intelligent Robots: Reintegrating AI*, edited by B. Boots et al., AAAI Press, Palo Alto, California, pp. 6-11. ISBN: 978-1-57735-601-1
- **Antonelli, M., Duran, A. J., Chinellato, E., & Del Pobil, A. P., 2013**, "Speeding-up the Learning of Saccade Control" in *Biomimetic and Biohybrid Systems*, edited by N.F. Lepora et al., *Lecture Notes in Artificial Intelligence* Vol. 8064, Springer-Verlag, Berlin, pp. 12-23. ISBN: 978-3-642-39801-8. DOI 10.1007/978-3-642-39802-5_2.
- **Antonelli, M., Duran, A.J., del Pobil, A.P., 2013**, "Application of the Visuo-Oculomotor Transformation to Ballistic and Visual-Guided Eye Movements", in

Proc. International Joint Conference on Neural Networks (IJCNN 2013), Dallas, Texas, USA, pp. 813-820. ISBN: 978-1-4673-6129-3/13.

- **Antonelli, M., Duran, A.J., Chinellato, E., del Pobil, A.P., 2015**, "Adaptive saccade controller inspired by the primates' cerebellum", in *Proc. IEEE International Conference on Robotics and Automation (ICRA 2015)*, Seattle, USA, pp. 5048-5053. DOI: 10.1109/ICRA.2015.7139901
- **Felip, J., Duran, A.J., Antonelli, A., Morales, A., del Pobil, A.P., 2015**, "Tombato: A humanoid torso for autonomous sensor-based tasks", in *Proc. 15th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2015)*, Seoul, Korea. pp. 475-481, DOI: 10.1109/HUMANOIDS.2015.7363592

Chapter 5

Predicting the internal model of a robotic system from its morphology

Work of each individual contributes to a totality and so becomes undying part of a totality. That totality is human life. Past and present and to come forms a tapestry that has been in existence now for many tens and thousands of years. And has been growing more elaborate, and on the whole more beautiful.

Isaac Asimov, Robots and Empire

5.1 Introduction

The perceptual loop exposed in the section [2.1](#) relates three elements that can be transferred to the field of robotics. First, from the engineering point of view, cognition concerns the internal model generated by the robot to develop tasks within a particular environment. Second, the sensors and effectors pertain to the morphology. Finally, the environment is the glue that cements the whole set. Its morphology and internal model characterize a robot system interacting with a particular environment. The morphology could be considered as a representation of the physical properties of the robotic system. Most of these properties can be measured. In turn, the internal model represents the interaction between the robot system and the environment. Different research areas

within Robotics have been established that differ in how the relations among these three elements are handled.

- ([Bongard et al., 2006](#)) point out that the interaction with the environment can determine the morphology of the robot.
- ([Vaughan and Zuluaga, 2006](#)) suggest that the simulation of the environment and incomplete self-knowledge model the robot behaviour and, therefore, its internal model
- Based on the maxim that knowledge comes from experience, the estimation of the internal model can be obtained from the robot's interaction with the surrounding environment. These model learning approaches typically consider the relationships between states and actions exclusively, and the information about the states and actions of the past, present and even the expected future is needed to model the robot behaviour. The process of learning is a regression problem where the training samples are obtained from the state and controls of the plant along with time ([Sigaud et al., 2011](#)). Internal-model-based control theory is well established, but internal models are typically expressed as mathematical models of the plant, normally employing a set of differential equations ([Isidori et al., 2003](#)).

Classical robotics relies on a previous generation of models, for example, performing a series of reality simplifications that make it possible to generate behaviour in a controlled environment. However, a cognitive and autonomous robot must be able to generate its models from the information streams and the interaction with the environment ([Nguyen-Tuong and Peters, 2011](#)). Two approaches can be considered to implement this model learning using the environment as guidance operationally:

- From an engineering point of view, by using online parametric identification ([Siciliano and et al., 2008](#)).
- From a machine learning point of view, by using the tools described in chapter 3. In chapter 4, we decided to use machine learning as a working tool, mainly supervised learning. In this way, it is not necessary to make a-priori assumptions about the structure of the model and we can include all relevant phenomena in a general function built out of experimental data.

In any case, according to ([Pfeifer et al., 2007](#)), a relationship always exists between the internal model and the morphology. Moreover, they are inseparable because both affect how information is processed in the robotic system. In the previous chapter, an

internal model (behaviour) has been implemented without considering how the morphology of the robotic system affects it. The proposed approach was based on the adaptation of the robotic system to the environment perspective. The morphological parameters defining the robotic head were preset, and from this point and through an adaptation process, the internal model defining the behaviour was modelled. As in the natural world, where the morphological characteristics of a living being condition its adaptation to the environment, it is appropriate to introduce a method for relating this part of the congenital cycle to the other two (internal model and environment) in order to have a more general view of the behaviour under study.

The reference is going to be changed to achieve this objective. In the previous chapter, morphology was considered constant and we studied how the internal model adapted to the environment. Therefore, a robotic system and variations of the environment served as a guide to learn the relationship between both. On the contrary, now the environment remains constant, and there are multiple robotic systems with different morphologies interacting with the same environment to determine their internal model. In this way, it is feasible to study how these robotic systems' morphology and the internal model are related.

Since this relationship is unknown, we propose an approach based on neural networks to learn it. These methods are instrumental in solving problems when a complete formulation is not known, or a mathematical representation is not explicitly available (Hudson and Cohen, 2000).

The ability to predict the internal model of a robot from its morphology has a great interest in the current state of the art in robotics research and applications. So-called *morphofunctional* machines can change their functionality by modifying their morphology and some modular self-reconfigurable robots can *morph* (Pfeifer et al., 2007). The rationale is that much of the functionality of a robot is due to its particular morphology, and by altering it depending on the task, its performance, adaptivity and versatility, will substantially improve.

In the last decade, there has been a growing concern in the community regarding how to progress in robotics research has been hampered due to differences in hardware that make it difficult to compare alternative approaches, apply benchmarks, or replicate results (Bonsignorio and del Pobil, 2015). The problem is that robots with a similar design still have differences in their morphological parameters. A possible solution is to make knowledge transferable to different *embodiments* or morphologies. For instance, Felip et al. (Felip et al., 2013) proposed a methodology based on abstract state machines that are automatically translated to embodiment-specific models. Getting the internal model directly from the morphology will contribute to progress in this direction.

Arguably, this research can have a high potential for impact in the 4th industrial revolution, the so-called Industry 4.0. In this context, a robot is no longer regarded as a standalone machine, but rather as a networked *Cyber Physical System* (Khaitan and McCalley, 2015; del Pobil, 2018) endowed with interoperability, i.e., the ability to connect and communicate with other devices via the Internet (Hermann et al., 2016). In this sense, the rationale of so-called *Cloud Robotics* (Wan et al., 2016)(Kehoe et al., 2015) is that instead of trying to increase the performance and functionality of isolated robot systems, knowledge is reused through the shared memory of multiple robots. Endeavours such as *RoboEarth* (Waibel et al., 2011) collect, store, and share data independent of specific robot hardware. However, internal models should be available online so that different robots can exhibit the same behaviour in a given environment for this to be fully operational. This chapter aims to verify how the internal model built by the robot from its adaptation process to the environment is related to its morphology for saccade execution behaviour. Therefore, this chapter is structured with an initial part where a model and a computation methodology to elucidate this relationship is presented.

Next, several methods for implementing the model defining the relationship between the internal model and the morphology are compared. In order to evaluate them properly, it is necessary to generate a dataset based on the typology of the robotic system described in the previous chapter, exhibiting a particular saccadic behaviour. Finally, after evaluating the proposed tools for learning the suggested model, the best performing method is used to describe several applications in robotics.

In the last part of the chapter, it is argued that understanding the relationship between morphology and internal modelling involves exploiting the way previous systems have learned to adapt to a given environment. This procedure has an unmistakable resemblance to the concept of gene and phenotype in the genetics of living organisms. When a living being is born, it does not start from an absolute lack of knowledge; on the contrary, the programming of its genes allows it to adjust its behaviour to adapt to the different environments in which it develops. The last section presents an artificial genotype model related to the relationship between morphology and internal parameters of a robotic family.

5.2 Objectives

Discovering the relationship between the internal model and the morphology of a robotic system able to execute a saccade movement is the aim of this chapter- To do so, we generalise the particular result of the previous chapter to any robotic head by

means of a parameter range established by the morphology. Correlating the morphology and the internal model facilitates this generalisation. Several techniques for learning are tested to discover the searched relationship. Their performance is compared and evaluated. Finally, various applications of the presented model are presented.

5.3 The relationship between morphological and internal model parameters

The starting point for systematically describing a mathematical model to achieve the proposed objectives is the cognitive loop shown in figure 2.1. For this purpose, an analogous loop for a robotic system is presented in figure 5.1.

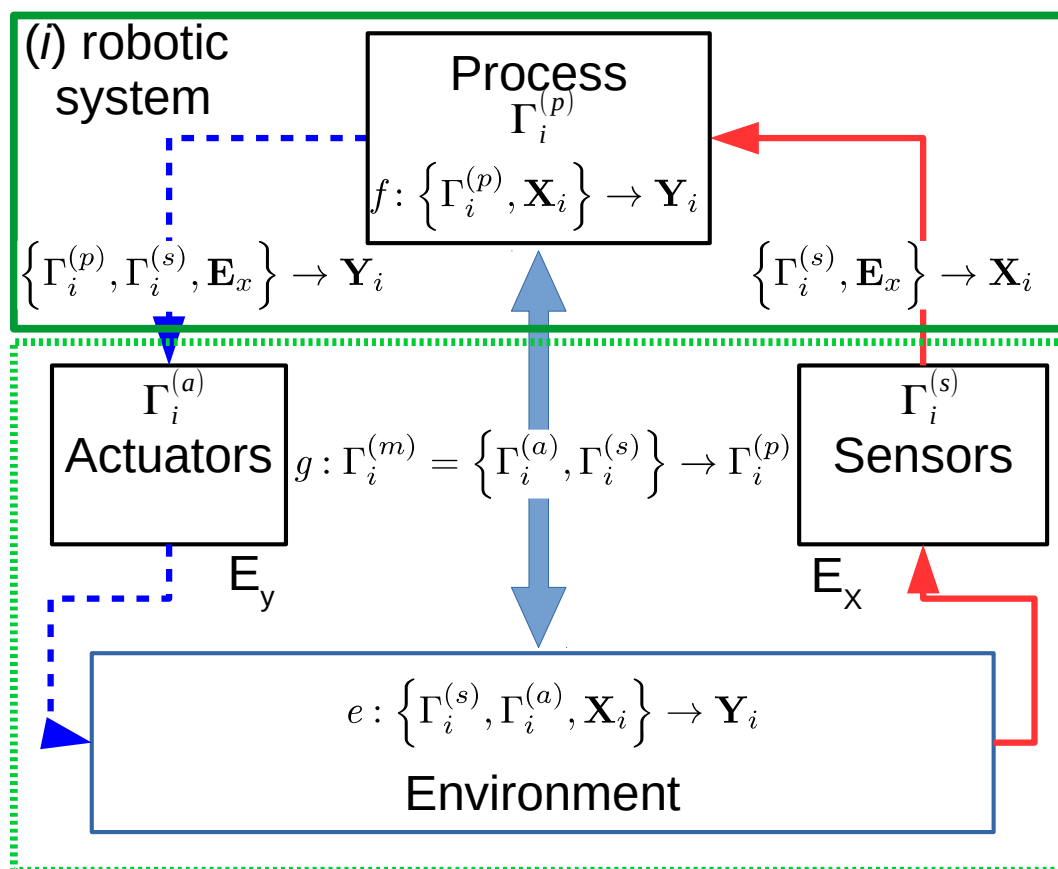


Figure 5.1: Schema showing the information fluxes and transformations among the different system components and the environment.

A robotic system (i) immersed in a particular and stable environment is considered. Each information flow exchanged by the robot with the environment is analysed. Nevertheless, this discussion is made from two different points of view:

- **From the robot point of view.** This is a classical approach where the sensors and actuators of the robot are part of it, and the whole system is somehow separated from the surrounding environment (solid green line in figure 5.1). The flow of information through the various elements of the system can be decomposed into the following stages:
 - The sensors of the robotic system transform the data flux (\mathbf{E}_x) from the environment and generate an input (\mathbf{X}_i) for the robot internal model. The flux \mathbf{E}_x only depends on the environment state, however the \mathbf{X}_i input depends on the robot morphology (generally on the sensor parameters ($\Gamma_i^{(s)}$) and \mathbf{E}_x).
 - The internal model processes the inputs and generates the outputs (\mathbf{Y}_i) to the robot actuators. The input flux \mathbf{X}_i is processed by the internal model (f) which takes into account the internal model parameters $\Gamma_i^{(p)} = \{\gamma_{i,1}^{(p)}, \gamma_{i,2}^{(p)}, \dots, \gamma_{i,k}^{(p)}\}$, where k is the number of internal parameters for (i) system. The function f embodies (figure 5.1) everything there is to know about the relationship \mathbf{X}_i and \mathbf{Y}_i through the internal model parameters.
 - The actuators of the robot modify the state of the environment. This fact can be represented by an output data flux (\mathbf{E}_y) in some way. This flux is a function of the parameters of the robot actuators ($\Gamma_i^{(a)}$).
 - The set $\Gamma_i = \{\Gamma_i^{(s)} \cup \Gamma_i^{(p)} \cup \Gamma_i^{(a)}\}$ represents the parameters of the robotic system, both morphological and those of the internal model. The set $\Gamma_i^{(m)} = \{\Gamma_i^{(s)} \cup \Gamma_i^{(a)}\} = \{\gamma_{i,1}^{(m)}, \gamma_{i,2}^{(m)}, \dots, \gamma_{i,h}^{(m)}\}$ contains the morphological parameters of the system, and h is the total number of sensor and actuator parameters.
- **From the environment point of view:** Now, focusing on the environment and defining new limits, the set formed by the environment, sensors and actuators are considered excluding the internal model as a new system (dotted green line in figure 5.1). The information streams can be decomposed into the elements of the considered system, as in the previous case:
 - The input flux of the considered system is the output flux of the internal model (\mathbf{Y}_i).
 - As in the previous case, the actuators modify or transform the information generating changes in the environment (\mathbf{E}_y).
 - In the environment, there is an inherent relationship (e) to the model that governs it, regulating the changes that can occur in it, generating signals that are captured by the sensors of the robotic system. This can be formulated as $\mathbf{E}_y = e(\mathbf{E}_x)$ or the inverse function $\mathbf{E}_y = e^{(-1)}(\mathbf{E}_x)$.

- The output stream of the environment is the input flow to the sensors that modify it, generating the output flux of the new system under consideration \mathbf{X}_i .

From the environment point of view the relationship (e) between \mathbf{Y}_i and \mathbf{X}_i depends only on the morphological parameters of the system ($\Gamma_i^{(m)}$) and \mathbf{X}_i . Whereas from the system point of view, the relationship between \mathbf{Y}_i and \mathbf{X}_i is only a function of the internal model parameters ($f(\Gamma_i^{(p)}, \mathbf{X}_i)$).

However, it is necessary to define which of the two systems is the adaptive one. The nature of the environment yields the model as defined by the functional relationship (e). In a stable environment like the one considered in this function, the inputs and outputs should vary, but not the type of function. On the contrary, the internal model defined by the functional relation (f) receives some input signals and generates an output when exposed to this environment. If the information reaching the environment generated by the internal model produces the expected changes, in this case, it is possible to consider that the system is well adapted to the environment.

For a suitable interaction with the environment, the internal model (f) must be an approximation of the real model of the environment (e). The correlation between the morphological and internal model parameters is given by the following functional relationships (figure 5.1):

$$\Gamma_i^{(p)} = g(\Gamma_i^{(m)}) \quad (5.1)$$

This statement will be true if the internal model is properly learned and the morphological parameters satisfy:

$$\left. \frac{\partial \Gamma_i^{(s)}}{\partial \mathbf{E}_x} \right|_i = \left. \frac{\partial \Gamma_i^{(a)}}{\partial \mathbf{E}_y} \right|_i = 0 \quad (5.2)$$

This equation (5.2) defines the notion of morphological parameters in our formalism, in the sense that they must accomplish the condition that they are independent of the input and output data fluxes from and to the environment for the system (i). Thus, for example, the camera's focal length is considered a morphological parameter if its value does not vary with the input information; i.e., it is not in autofocus mode.

As shown in the previous chapter, the relationship defined by function f , is estimated using the inputs (\mathbf{X}_i) and outputs (\mathbf{Y}_i) of the system, and the g transformation is embodied in the environment data flux transformations. The inputs \mathbf{X}_i and outputs \mathbf{Y}_i are represented by a sequential set of vectors $\mathbf{X}_i = \{\mathbf{x}_{i,1}, \mathbf{x}_{i,2}, \dots, \mathbf{x}_{i,u}\}$ and

$\mathbf{Y}_i = \{\mathbf{y}_{i,1}, \mathbf{y}_{i,2}, \dots, \mathbf{y}_{i,u}\}$ where u is the number of samples ($\mathcal{D}_i = \{\mathbf{X}_i, \mathbf{Y}_i\}$). We assume that each pair of components of \mathcal{D}_i are independent measures of the system.

In model learning frameworks, the optimal model structure should be obtained from training data [Nguyen-Tuong and Peters \(2011\)](#), and the model can be fitted into the following function:

$$\mathbf{Y}_i = f(\mathbf{X}_i, \Gamma_i^{(p)}) + \epsilon_i; \quad f(\mathbf{X}_i, \Gamma_i^{(p)}) = \phi(\mathbf{X}_i)^T \Gamma_i^{(p)} \quad (5.3)$$

From a probabilistic point of view, $f(\mathbf{X}_i, \Gamma_i^{(p)})$ can be considered as a conditional expectation $\mathbb{E}(\mathbf{Y}_i | \mathbf{X}_i)$, therefore ϵ_i is an expectational error term that can be expressed as $\epsilon_i \equiv Y_i - \mathbb{E}(Y_i | X_i)$.

The model is defined by the value of the internal model parameters $\Gamma_i^{(p)}$. Due to the independence of the measures in \mathcal{D}_i , the likelihood function for $\Gamma_i^{(p)}$ is:

$$\mathcal{L}_u(\Gamma_i^{(p)} | \mathcal{D}_i) = \frac{1}{u} \sum_{s=1}^u \log(\mathbb{P}(\mathcal{D}_{i,s} | \Gamma_i^{(p)})) \quad (5.4)$$

Maximizing the function $\mathcal{L}_u(\Gamma_i^{(p)} | \mathcal{D}_i)$, we obtain the maximum likelihood estimator (MLE) for $\Gamma_i^{(p)}$ as:

$$\hat{\Gamma}_i^{(p)} \in \underset{\Gamma_i^{(p)}}{\operatorname{argmax}} \mathcal{L}_u(\Gamma_i^{(p)} | \mathcal{D}_i) \quad (5.5)$$

There exist many optimisation procedures to solve this equation (5.5). However, the above definition is not complete because there is no guarantee that such a maximum exists or, when it does exist, it is unique. That is, the proposed system is not able to properly learn the environment model. Depending on the similarity between the real model of the environment and the model proposed by equation (5.3), this statement will be valid. In this work, the supposition that all proposed systems can learn the environment model is considered. As a result of equation (5.5), two properties are satisfied:

- MLE for $\Gamma_i^{(p)}$ is consistent, that is, when u is large enough, the estimator $\hat{\Gamma}_i^{(p)}$ converges into $\Gamma_i^{(p)}$:

$$\lim_{u \rightarrow \infty} \mathbb{P} \left(|\hat{\Gamma}_i^{(p)} - \Gamma_i^{(p)}| > \delta \right) \rightarrow 0 \quad (5.6)$$

From this expression can be concluded that the value of $\hat{\Gamma}_i^{(p)}$ can be considered constant when u is large enough. That is, there exists a unique set of optimum parameters $\hat{\Gamma}_i^{(p)}$, that leads to the best approximation of the true function by a certain model.

- The maximum likelihood estimator is asymptotically normal. That is, as u becomes large, $\hat{\Gamma}_i^{(p)}$ converges to a multivariate normal random variable whose variance is a diagonal matrix. The asymptotic normality of the maximum likelihood estimator is expressed as:

$$\sqrt{u} \left(\hat{\Gamma}_i^{(p)} - \Gamma_i^{(p)} \right) \xrightarrow{d} \mathcal{N} \left(0, \sigma_{ML}^2 \mathbf{I} \right) \quad (5.7)$$

where σ_{ML}^2 is called the asymptotic variance of the estimate $\hat{\Gamma}_i^{(p)}$. Asymptotic normality says that the estimator not only converges to the unknown parameter $\Gamma_i^{(p)}$, but it converges fast enough, at a rate $1/\sqrt{u}$.

5.4 Extracting knowledge from multiple robotic systems

If instead of only one system (i), v similar systems with different morphologies¹ are considered, and each one has an internal model whose properties are fitted into (5.3), the result is a set of pairs $\{\hat{\Gamma}^{(p)}, \Gamma^{(m)}\} = \{\{\hat{\Gamma}_i^{(p)}, \Gamma_i^{(m)}\} \forall i \in [1, v]\}$. From a statistical point of view, both variables are considered as random because we randomly define $\Gamma^{(m)}$ as the system morphology in a particular range of values and each $\hat{\Gamma}_i^{(p)}$ was obtained from independent trials. Therefore, a regression model such as (5.3) can be used to learn the relationship between them:

$$\hat{\Gamma}^{(p)} = g(\Gamma^{(m)}, \mathbf{W}) + \xi; \quad g(\Gamma^{(m)}, \mathbf{W}) = \Phi(\Gamma^{(m)})^T \mathbf{W} \quad (5.8)$$

As in the previous case, a maximum likelihood estimator for $\hat{\mathbf{W}}$ can be used for fitting the values of morphological parameters —obtained for each system— to $\hat{\Gamma}^{(p)}$. Once the value of $\hat{\mathbf{W}}$ is calculated, a prediction model can be used to obtain an estimation of $\hat{\Gamma}_i^{(p)}$ from the specific morphological parameters. Given a new system (j) —defined by its morphological parameters— its internal model parameters can be estimated as:

$$\hat{\Gamma}_j^{(p)*} = \Phi(\Gamma_j^{(m)})^T \hat{\mathbf{W}} \quad (5.9)$$

If equation (5.9) is compared with equation (3.28), it is possible to elucidate that the techniques described in section 3.4 can be valid for addressing the learning problem.

The problem posed by equation (5.1) is to find the function g to map the morphological parameters and the internal model parameters. For this, v robotic systems are needed. Each one of these v systems will have previously interacted with the same environment in order to obtain its set of internal model parameters $\Gamma_i^{(p)}$.

¹Same number and kind of morphological parameters but different values

The size of this set depends on the number of inputs and outputs of the robotic system and the adaptation solution used for estimating the internal model. In turn, the size of $\Gamma_i^{(m)}$ is related to the complexity of the system morphology, and the number of morphological parameters is usually substantially smaller than the number of internal model parameters $\Gamma_i^{(p)}$.

In order to contextualise this problem in the scope of this work, it is necessary to have ν robotic heads —characterised by a variable morphology within a range— that interact with the same environment and develop the same behaviour: the saccadic movement. In the previous chapter, these points have been defined for a single robotic head; now, we will vary the parameters that define the morphology of the head defined in section 4.3, allowing the system to adapt to an environment that is common to all systems, by using the learning algorithms described in section 4.4.2.

5.4.1 Morphology of the robotic systems to predict the internal model

A set of robotic heads with a Helmholtz-type design needs to be defined. For this purpose, the parameters defined in section 4.3 are used. This system presents morphological parameters that correspond both to actuators and sensors. According to section 4.3, a range for each parameter can be defined. Table 5.1 shows this range of values that each morphological parameter can take. If the limits had been defined totally randomly, many of the configurations obtained would probably have been unviable. Therefore, the range of the parameters on the right side is defined according to the range on the left side to avoid this situation as much as possible. Examples of different morphologies generated using these parameter ranges can be seen in figure 5.2. In these examples, the variations in the camera's parameters should also be considered.

For further use in learning, the values corresponding to the set of morphological parameters are transformed to maintain a homogeneous scale across all inputs. For example, as shown in section 4.3, the camera resolution is in pixels. Therefore, the image dimensions are transformed to decimetres using the pixel size value and are thus of the same scale as the rest of the morphological parameters. In turn, the pixel size values are also converted from metres to decimetres to maintain the homogeneity of the set.

Table 5.1: Morphological parameters to generate a set of robotic systems.

Prismatic joints (cm)	
Left side	Right side
$\Gamma_{i,1}^p = q_2 \in [-0.054, 0.054]$	$\Gamma_{i,2}^p = q_2 = \Gamma_{i,1}^p + [0, 0.01]$
$\Gamma_{i,3}^p = q_3 \in [0, 0.07]$	$\Gamma_{i,4}^p = q_3 = \Gamma_{i,3}^p + [0.035, 0.07]$
$\Gamma_{i,5}^p = q_4 \in [-0.02, 0.054]$	$\Gamma_{i,6}^p = q_4 = \Gamma_{i,5}^p + [0, 0.02]$
$\Gamma_{i,7}^p = q_6 \in [0, 0.01]$	$\Gamma_{i,8}^p = q_6 = \Gamma_{i,7}^p + [0, 0.01]$
Cameras parameters:	$f(\text{px}); s(\text{m/px}); w(\text{px}); h(\text{px})$
Left camera	Right camera
$\Gamma_{i,9}^p = f_l \in [340, 1920]$	$\Gamma_{i,10}^p = f_r = \Gamma_{i,9}^p + [0, 200]$
$\Gamma_{i,11}^p = s_l \in [3 \cdot 10^{-6}, 7 \cdot 10^{-6}]$	$\Gamma_{i,12}^p = s_r \in [3 \cdot 10^{-6}, 7 \cdot 10^{-6}]$
$\Gamma_{i,13}^p = h_l \in [340, 1920]$	$\Gamma_{i,14}^p = h_r = \Gamma_{i,13}^p + [0, 200]$
$\Gamma_{i,15}^p = w_l \in [340, 1920]$	$\Gamma_{i,16}^p = w_r = \Gamma_{i,15}^p + [0, 200]$
if $\Gamma_{i,13}^p > \Gamma_{i,15}^p$, swap ($\Gamma_{i,13}^p, \Gamma_{i,15}^p$)	if $\Gamma_{i,14}^p > \Gamma_{i,16}^p$, swap ($\Gamma_{i,14}^p, \Gamma_{i,16}^p$)

5.4.2 Defining an environment to adapt robotic systems

All robotic systems described by the morphological parameters considered in the previous section must interact with a single environment. The environment had been defined in section 4.5.1, a prismatic region determined by six planes and a centroid. From a theoretical point of view, any point within this space could be a stimulus that triggers the saccade in a certain robotic system. However, from a practical point of view, it is not such an easy problem to solve.

The stimulus must appear in both cameras of the robotic system simultaneously to execute the saccadic movement. A first solution to solve this situation would be to generate a cloud of random points uniformly distributed that would act as stimuli within the region defined as the environment. Depending on the density of points per unit of volume, there would be more or less probability that the projection of some of them on the image planes of each camera would fall within the limits of the image size. In addition, the point chosen should have the projection on both images. For example, considering 5000 points distributed within the space that defines the environment, it would be necessary to calculate the projections of these 5000 points on the planes of each camera and look for the ones that are within both. All this computation effort is only to determine a unique, valid stimulus. Bearing in mind, as suggested in section 4.4.2.7 that it was necessary to execute about 400 movements, the total number of projections that should be computed could rise to 200000 to train a single robotic system.

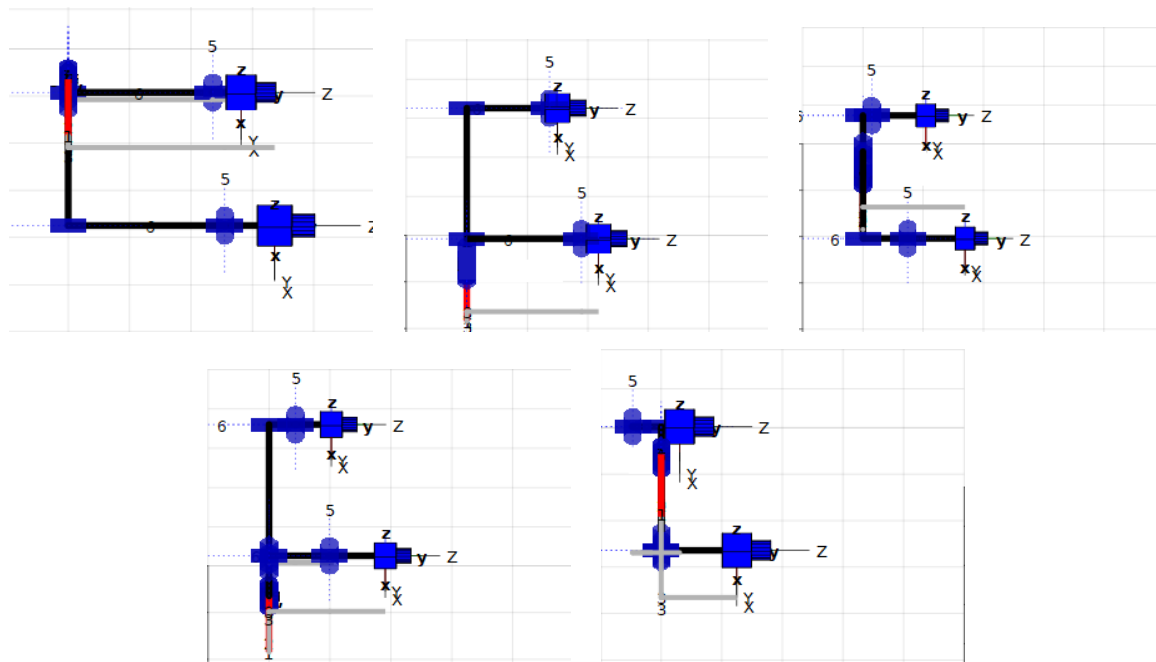


Figure 5.2: Different morphologies of robotic systems generated from the ranges defined in table 5.1

A computer graphics technique (*frustum culling*) is used to circumvent this problem and reduce the number of projections and calculations that have to be made for the process of adapting the robotic system to the environment. In figure 5.3, the parallelepiped defined as environment is represented and contains all possible stimuli that can be generated within this environment (red points). These stimuli are generated once and they are characteristic of the environment. Therefore all robotic systems interact with the same stimuli. In figure 5.3, the three-volume intersection is also represented. The volume of the environment is intersected with the frustum defined by each camera's near and distant plane so that only the points within these three volumes are possible stimuli that can trigger a saccade in the current position of the cameras. The equations of the planes defining the environment are always the same if the base of the robotic system is taken as the origin reference frame. However, the planes forming the frustum of the cameras must be recalculated when the cameras are moved.

If the cameras are moved by giving angular position values to the joint motors at random, it can be the case that the frustum of cameras never intersect, and therefore many calculations are wasted. In a biological system, the eyes act in a coordinated manner. If instead of using the angular positions of the robotic system directly, random values of vergence and version are calculated; then, from equations (2.1) and (2.2), the angular positions for each motor are easily obtained from:

$$\gamma_R = \frac{2version - vergence}{2} \quad (5.10)$$

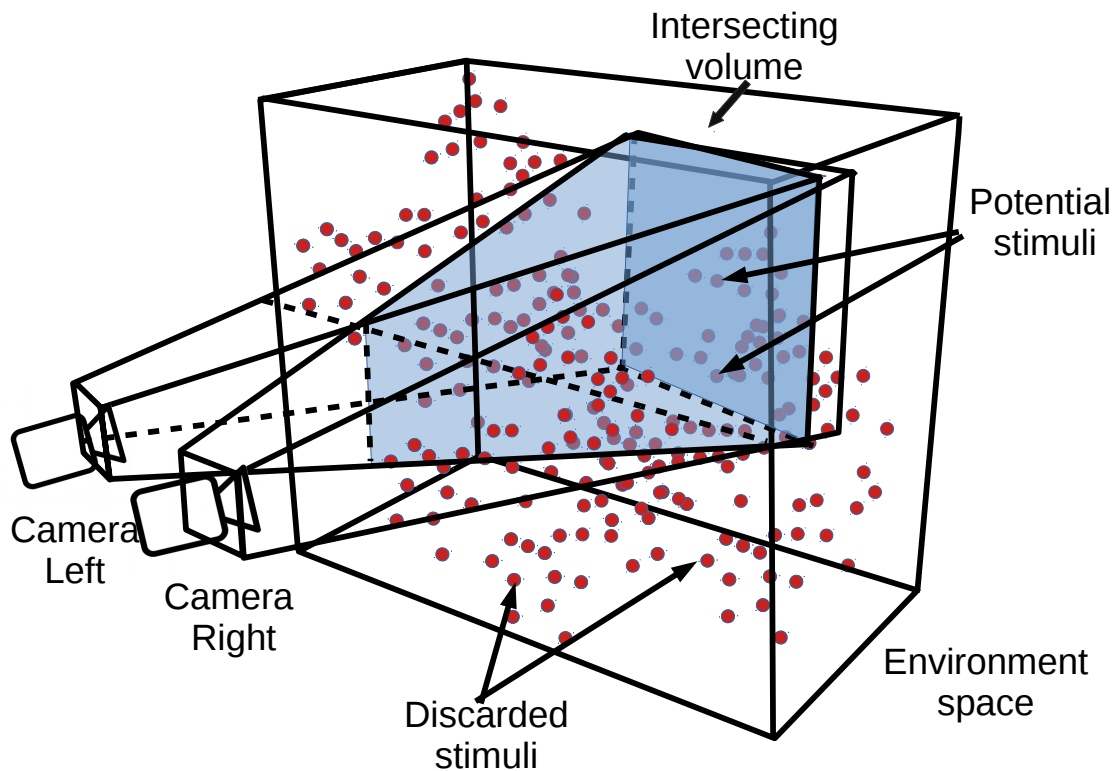


Figure 5.3: Environment and frustum of the cameras intersecting

$$\gamma_L = \text{vergence} + \gamma_R \quad (5.11)$$

This aspect does not apply to the tilt angle position since it is common to both cameras and is directly calculated randomly.

Considering this calculation method, the obtained angular positions ensure that the frustums of the cameras have an intersecting space at some point. Therefore, the same environment is maintained instead of positioning the stimuli in front of the robotic systems to be adapted, although the robotic system moves. This movement is performed in the space of vergence and version of each system.

The normals defining the planes forming the environment and the cameras' frustums point towards the centre of each volume. Thus, to determine whether a point (x,y,z) is inside or outside the region bounded by these planes, a dot product of the coefficients defining these planes and the coordinates of the point must be calculated. A result greater than zero implies that the point is defined within the volume.

The main advantage of this method is that by performing this calculation for one of the cameras, many points are discarded that no longer need to be considered for the other camera. Thus, this procedure speeds up the process of obtaining a set of candidate stimuli that are present in both images.

5.4.3 One behaviour for multiple robotic systems

In the previous chapter, two architectures were presented to implement saccadic movements: feedback error learning **FEL** and the recurrent architecture **RA**. Both yield positive results (section 4.4.2.8), with **RA** being better in the final performance, but at the cost of execution time. This time factor is definitive in the selection of the **FEL** architecture to implement the robotic systems used to achieve the objectives of this chapter.

In any case, both proposals have in common that they need a fixed controller to guide the initial movements in the adaptation process. The fixed controller (**B**) slowly drives the system toward the target and provides a learning cue to an adaptive controller (**C_f**). Depending on the chosen coordinate system, the fixed controller is a transformation matrix as indicated in equation (4.3).

Before starting the adaptation process, regardless of the chosen architecture, an *a priori* estimation of **B** is needed to convert the visual target into a movement of the eyes. The procedure to estimate the fixed controller is based on the fixation of the stimulus in the center of the camera images. Afterwards, motor babbling is generated [Saegusa et al. \(2008\)](#). Under these conditions, the variation of the visual target position ($\Delta \mathbf{t} = \mathbf{t}$) in the images can be correlated with the motor command (Δe). Therefore, if a set of b pairs $\{\mathbf{t}_i, \Delta \mathbf{e}_i\} = \{\{t_{i,j}, \Delta e_{i,j}\} \forall j \in [1, b]\}$ for a particular robotic system (i) is generated by motor babbling, **B_i** can be directly obtained from least squares regression (see equation (4.8)):

$$\mathbf{B}_i = \Delta \mathbf{e} \mathbf{t}_i^T (\mathbf{t}_i \mathbf{t}_i^T)^{-1} \quad (5.12)$$

Once the controller **FEL** is chosen to perform the process of adaptation, it is necessary to comprehend the learning as a process of transmission of information that flows from the environment to the robotic system in such a way that, using this information, the robot can discover the model that governs the environment and therefore interact with it.

From a practical point of view, learning using the proposed **FEL** controller is equivalent to adapting the weights of a neural network (section 4.4.2.4).

From equation (5.3), to learn the relation between the signals that arrive at the system and those that it generates to interact with the environment is to discover for a given architecture ($\phi(\mathbf{X}_i)^T$) the parameters of the internal model $\Gamma_i^{(p)}$.

Therefore, the weights of the neural network are an essential part of the parameters of the internal model ($\Gamma_i^{(p)}$) described in equation (5.3). Hence, the term $\phi(\mathbf{X}_i)^T$ is the

one corresponding to the neural network architecture used by the FEL controller.

These weights store the information from the interaction of the system with the environment. Both the fixed controller parameters \mathbf{B}_i and the adaptive controller parameters (the neural network weights θ_i) represent the internal model of the robotic system in this example $\hat{\Gamma}_i^{(p)} = \{\mathbf{B}_i, \theta_i\}$.

5.5 Generating one robotic head with saccadic behaviour

As it is indicated in section 5.4, to learn the relationship between the internal model and the morphology of a robotic system, it is not enough to consider one case; it is also necessary to generate ν robotic systems. In order to produce one of these systems, it is necessary to follow a scheme like the one shown in figure 5.4. As can be seen in figure 5.4, the generation process is divided into three main phases:

1. **Robotic system generation:** A robot head is created from the ranges defined for each morphological parameter described in table 5.1 ($\Gamma_i^{(m)} \in \mathbb{R}^{16}$). However, it is also necessary to initialize the two components of the internal model.
 - In the case of the internal model corresponding to the fixed controller \mathbf{B}_i , it is necessary to perform an estimate prior to the adaptation process as indicated in the previous section. In order to obtain an adequate precision in the estimation of the fixed controller, the number of movements (b) in the motor babbling process that are necessary to execute for each robotic system must be taken into account. As a result, a set of pairs $\{\mathbf{t}_j, \Delta \mathbf{e}_j\} \forall j \in [1, b]$ is obtained and using equation (5.12), \mathbf{B}_i can be estimated. To avoid unnecessary calculations for each robotic system, a previous study is performed to estimate a value of b to give satisfactory results without having to increase excessively the iterations to generate them.

A number of head setups (1000) were chosen randomly. The values in \mathbf{B}_i were estimated by varying the number of iterations (b) for the selected robot heads. Afterwards, the calculated \mathbf{B}_i is used to predict the visual stimulus position after a movement. The mean square error between the real stimulus position variation in the images and the predicted position shift can be considered as a quality measure for \mathbf{B}_i . Figure (5.5) illustrates the high stability of \mathbf{B}_i estimation after some 500 iterations. Therefore, a value of b greater than 500 ($b = 600$) is selected.

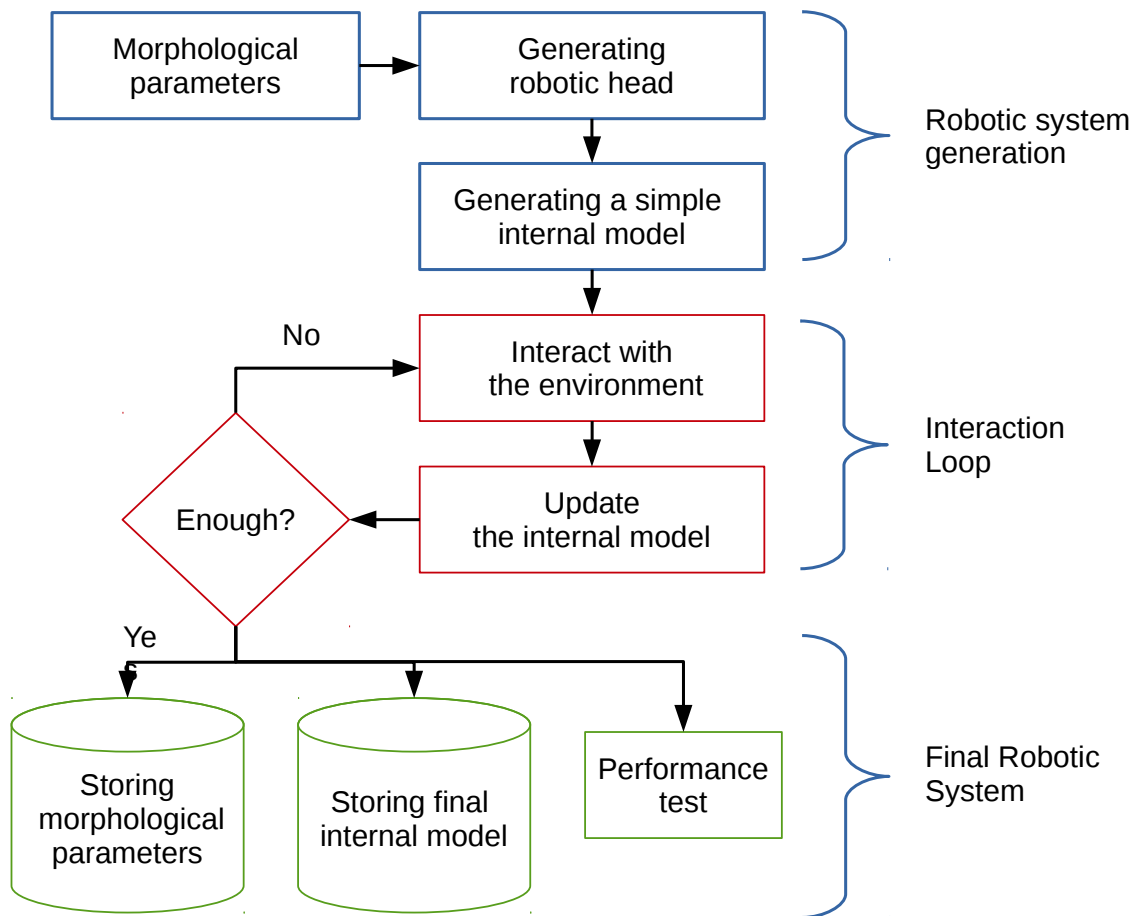


Figure 5.4: Schema describing the generation of one of the robotic heads needed to learn the relationship between the internal model and the morphology

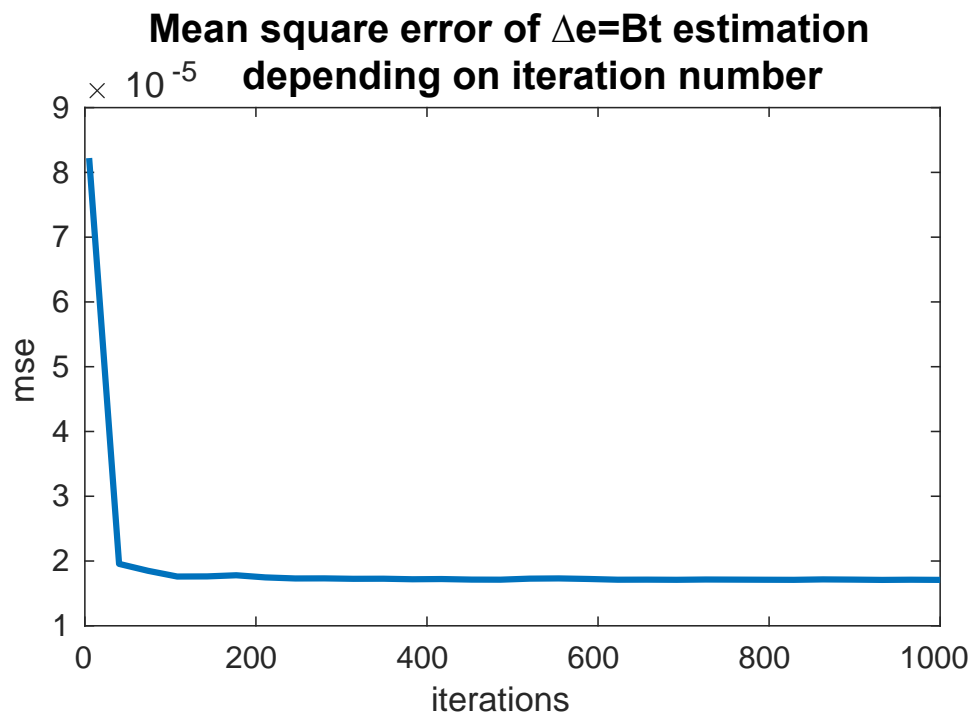


Figure 5.5: Mean square error for \mathbf{B}_i estimation with 1000 robot head setups versus the number of iterations

- In the case of the internal model corresponding to the adaptive controller, a neural network with a single hidden layer was used, keeping the same architecture for all the setups. Namely, an input layer with seven neurons: four neurons for stimulus position in the image (\mathbf{t}) and three for the head motor commands (\mathbf{e}); the output layer has three neurons according to the control command variations ($\Delta\mathbf{e}$). Gaussian activations using random sparse features (section 3.5.2) are used for the hidden layer. The randomness of these features allows us to use this set of features for all neural networks. In this way, these features are considered as a component of the neural network architecture. A unique set of random sparse features is generated (Ω_m), and they are used for the training and validation processes. Incremental sparse spectrum Gaussian process regression is used for adapting the weights of the neural network. The parameters of the training algorithm have been previously tuned (Gijsberts and Metta, 2013): variance of the model ($\sigma_n^2 = 0.1$), signal variance ($\sigma_f^2 = 1.0$) and number of projections ($D=300$), therefore $\Omega_m \in \mathbb{R}^{600}$. Each neural network used for learning the adaptive controller of each head setup is initialised using random weights (θ_i^0). As indicated in section 3.5.2, a covariance matrix (\mathbf{A}) and a vector (\mathbf{b}) are needed to train the neural network using a gaussian process. The weights of the networks are adapted using

- (A) and (b). Once the network is trained, these two elements are not used to estimate the network's output. The mapping between inputs and outputs is given solely by the trained weights θ_i , regardless of how the weights were learned.
2. **Iteration loop.** Once the robotic system has been configured, it must be exposed to the environment. This procedure is common to all the generated robotic systems, and it is defined as indicated in section 5.4.2. For adapting, the cameras are moved randomly in vergence and version mode, a random virtual object placed in both images is selected as a stimulus, the cameras of the i robot head acquire the images, and the visual position of the object is estimated. Using the current output of the neural network (\mathbf{C}_{f_i}) and the value provided by the fixed controller, previously calculated (\mathbf{B}_i), an incremental control action for the head motors is generated. The head moves according to these control commands, and the cameras acquire the new visual stimulus position on the images. The neural network is adapted using the visual error in this position. After a number of iterations, the neural network weights converge to stable values, and the visual error is stabilised. To ensure convergence is reached, and considering the time consumption of the training process, each robot head is trained for 1000 iterations. This training process has been repeated three times for each head setup, and the final θ_i values are taken as the mean of the three trials.
 3. **Final robotic system.** At this point, the resulting weights of each neural network representing an adaptive controller are considered as part of the internal model parameters, since they partially determine the system's behaviour. Each trained weight matrix is represented by $\theta_i \in \mathbb{R}^{600 \times 3}$. Together with the fixed controller parameters, they constitute the set of characteristic internal model of each robotic system $\Gamma_i^{(p)} = \{\mathbf{B}_i \in \mathbb{R}^{3 \times 4}, \theta_i \in \mathbb{R}^{600 \times 3}\}$. After the training process, the estimated \mathbf{C}_f and \mathbf{B} are used for executing saccadic movements. In particular, to estimate the error in the adaptation process, each robot head setup is tested using 500 random saccades.

5.6 Generating multiple robotic heads with saccadic behaviour

The previous section described how a robot head is generated and adapted to a particular environment to develop a saccadic behaviour. This process must now be repeated as many times as required to obtain sufficient data to learn the relationship

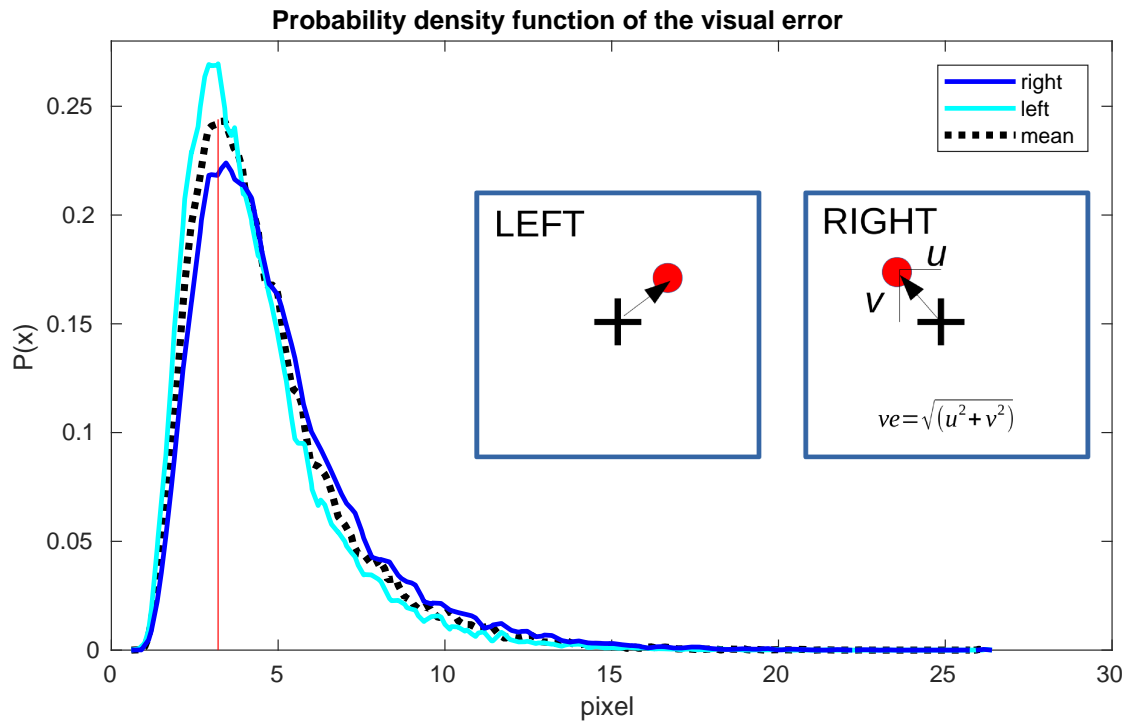


Figure 5.6: Visual error probability density functions for the 44271 trained head setups using the estimated \mathbf{B} and C_f for each one. The error for the two cameras and the average is shown. The considered measure of the final visual error is the mean visual distance in pixels between the stimulus position and the gaze point for each camera. Thus, these three curves represent the visual error after training for the proposed set of morphologies. These probability density functions fit fairly well into a log-normal distribution. The estimation of their parameters expressed in pixels are: for the mean ($\mu = 4.6, \sigma = 2.26$); left camera ($\mu = 4.3, \sigma = 2.08$); right camera ($\mu = 4.9$ and $\sigma = 2.4$).

between the internal model and the morphology. Through simulation, 44271 robotic systems are generated and adapted in the same environment.

After this intensive calculation process, the result is a dataset \mathcal{D} composed by a set of pairs $\Gamma_i^{(m)} \in \mathbb{R}^{16}, \Gamma_i^{(p)} = \{\mathbf{B}_i \in \mathbb{R}^{3 \times 4}, \theta_i \in \mathbb{R}^{600 \times 3}\} \mid i \in [1, v = 44271]$ and an average performance in the final execution of the saccade behaviour for each generated robotic system. Figure 5.6 shows the probability density function for the visual mean error based on performed tests for the 44271 robotic heads.

It should be noted that the values of the covariance matrix R_i and the term b_i in the algorithm 3 are not stored in the dataset based on the idea that both elements depend on the adaptation process and represent the uncertainty in that process. In contrast, the final weights are the terms that contain the knowledge acquired in the learning

procedure.

5.7 Machine learning problem

A machine learning tool is needed to approximate equation (5.8). This tool should be flexible, and sometimes it will have to manage a significant number of components for $\Gamma_i^{(p)}$ and $\Gamma_i^{(m)}$. An approach based on neural networks is feasible due to the scalability of the resulting architectures. Since typically the number of components in $\Gamma_i^{(p)}$ is large, the high dimension of the neural network output is a challenging problem (L'Heureux et al., 2017) and several approaches to solve it are tested.

Fortunately, the problem formulated in (5.8) can be decomposed into two parts for the proposed case study: on the one hand $\hat{\mathbf{B}} = g_1(\Gamma^m)$:

$$\mathbf{B} = g_1(\Gamma^{(m)}, \mathbf{W}_1) + \xi_1; g_1(\Gamma^{(m)}, \mathbf{W}_1) = \Phi_1(\Gamma^{(m)})^T \mathbf{W}_1 \quad (5.13)$$

and on the other hand $\hat{\theta} = g_2(\Gamma^m)$:

$$\theta = g_2(\Gamma^{(m)}, \mathbf{W}_2) + \xi_2; g_2(\Gamma^{(m)}, \mathbf{W}_2) = \Phi_2(\Gamma^{(m)})^T \mathbf{W}_2 \quad (5.14)$$

These equations represent the two regression problems that must be solved to learn the relationship between morphology and the internal model in this set of robotic systems.

5.7.1 Solving the regression problem for the fixed controller

Given a dataset \mathcal{D} generated from the interaction with an environment of many robotic systems, a machine learning approach can be used to solve the regression problem posed by the equation 5.13. The dataset (\mathcal{D}) is split into three partitions to reduce potential bias in the data as much as possible: 26500 items for training, 4500 items for validation and 13271 items for testing.

The regression problem in equation (5.13) can be solved using a basic single layer neural network as show in figure 3.5. Bearing in mind these aspects:

- The number of neurons in the hidden layer(h) is undoubtedly one of the foremost parameters to be modified in this type of neural network.
- The input layer contains 16 neurons corresponding to the 16 morphological parameters. In turn, there are 12 neurons in the output for 3x4 \mathbf{B} parameters.

- The hyperbolic tangent function was used for the hidden layer, and the output layer was linear.
- The algorithm used to train the network was *scaled conjugate gradient back-propagation* (SCG) Møller (1993).

A schema of this networks can be seen in figure 5.7. Several variations of the same neural network schema have been trained by changing the number of hidden layer units (h) in order to identify the most optimal parameter \mathcal{D} and the proposed network architecture. The training process and its posterior test have been repeated

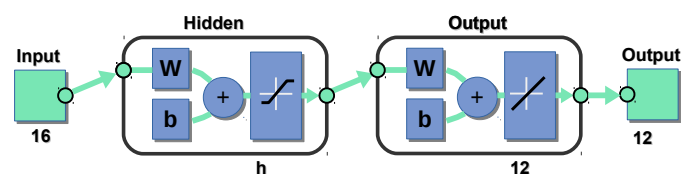


Figure 5.7: Schema of the single layer feedforward neural network proposed for learning the relation between the B matrix of the fixed controller and the morphological parameters.

three times splitting randomly each time \mathcal{D} in the proposed subsets. Figure 5.8

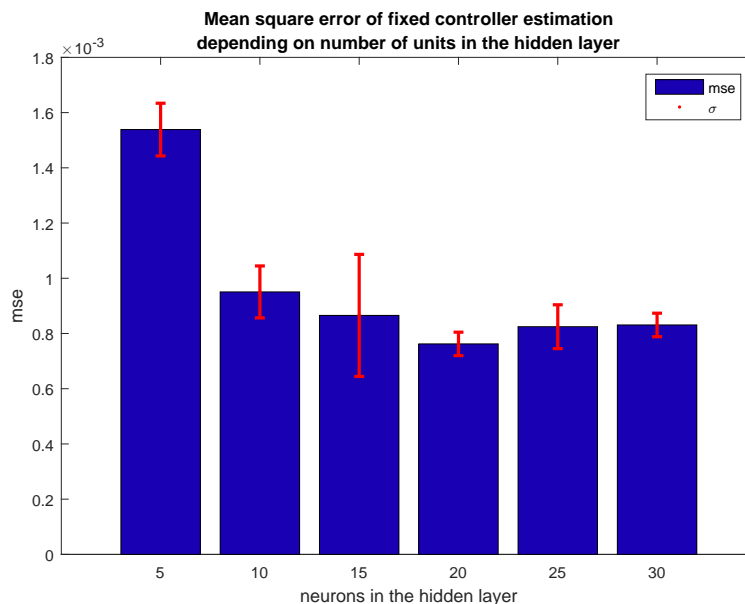


Figure 5.8: Mean square error obtained by different neural network hidden layer setups. The red line represents the standard deviation for three repetitions of the training

shows the obtained results for the mean and standard deviation of the mean square error for B estimation after training. From these results, the best performance for this

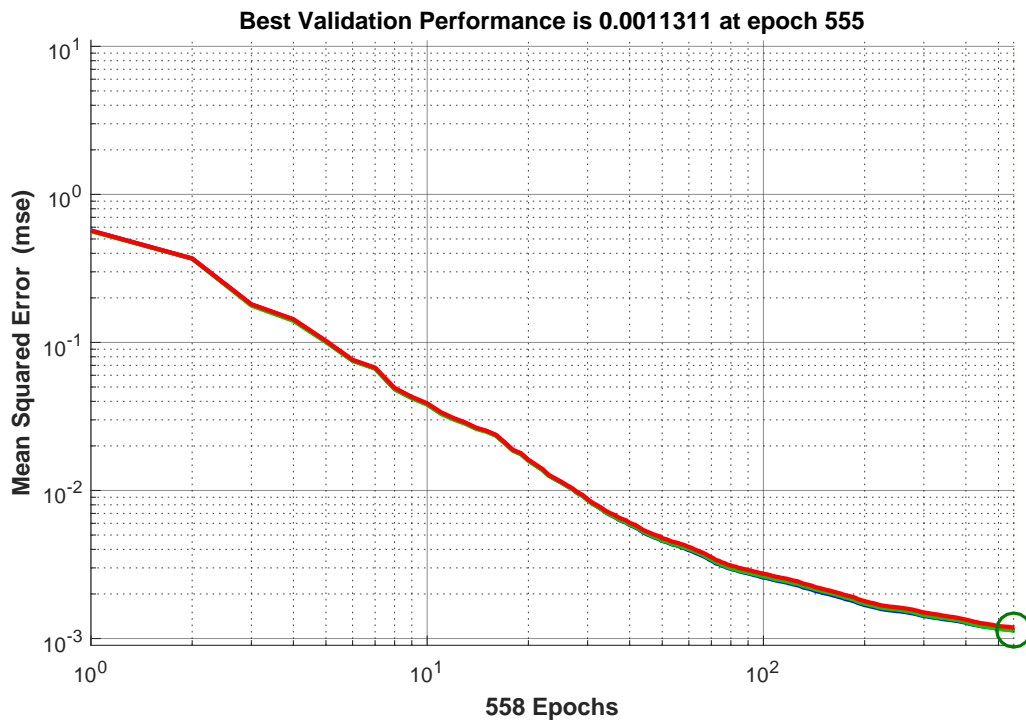


Figure 5.9: The learning curve for \mathbf{B} and $\Gamma^{(m)}$ regression

dataset is reached using 20 neurons in the hidden layer. Therefore, the best training performance for the estimation of \mathbf{B} is reached for 20 neurons in the hidden layer with a $MSE = (0.763 \pm 0.043)10^{-3}$. The training and validation curve with the defined data sets and a hidden layer with 20 units can be seen in figure 5.9

5.7.2 Solving the regression problem for adaptive controller

Solving equation (5.14) can be challenging due to the curse of dimensionality, with 16 input neurons and 1800 output neurons. When the dimensionality of the inputs is increased, the number of training samples needs to be increased exponentially for a nonparametric model regression (Kohler et al., 2009). The following sections present three neural network architectures for dealing with this problem. The purpose is to evaluate their performance as methods for learning the relationship between the internal model and the morphological parameters. The selection of the best neural network model to solve this learning problem can be treated as a model selection problem, noting that these three architectures are really different. Two strategies are applied to the three approaches.

- The first strategy is based on using a sequential test. A huge amount of morphologies are used for training in order to be in a data-rich situation. Therefore, the

best approach to model selection, according to Hastie et al. (Hastie et al., 2009) is to divide the dataset into three parts, a training set, a validation test and a test set. The first two are used for the training process, while the test set is employed to assess the generalisation error of the final chosen model. This procedure is repeated three times. The average of the mean square error (MSE) of each trial for each neural network is estimated to compare the different models.

- The second strategy is based on information criteria. Whereas the network complexity is not explicitly considered in the case of sequential tests; the information criteria addresses the model complexity through generalised degrees of freedom (GDF) (Ye, 1998). This second strategy is usually applied to compare the different approaches only when the final network architecture for each approach has been decided.

5.7.2.1 Single layer feedforward neural network option

A classic single-layer network is proposed to solve the problem of regression between the 16 parameters conforming to the morphology of the robotic system ($\Gamma^{(m)}$) and the 1800 parameters representing the internal model parameters (θ) of the trained adaptive controller. This network is similar to the one proposed for learning the fixed controller.

The elements of the network topology remain the same as in the case of the fixed controller, i.e., the activation functions of the neurons in the hidden layer are the hyperbolic tangent, and the output layer is a linear function (figure 5.10). The algorithm used for training the network is also SCG. However, in this case, the number of units of the hidden layer must be greater. Because a priori, this value is not known, tests varying the number of hidden layer neurons are performed. Three scenarios are considered: 500 units (SL_1), 1000 units (SL_2) and 2000 units (SL_3) in the hidden layer.

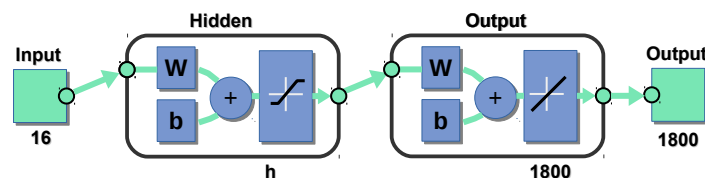


Figure 5.10: Schema of the single-layer feedforward neural network proposed for learning the relationship between the weights of the adaptive controller and the morphological parameters. The value h is the number of units in the hidden layer

As will be seen below, the aim of these experiments is not to obtain an exact value

of neurons in the hidden layer in order to achieve the best result for this type of neural network, but rather the goal is to reach a certain order of magnitude of **MSE** by means of the proposed architecture.

Taking as inputs the morphological parameters of each robotic system and, as the desired output, the value of the adaptive controller weight matrix, an off-line training of these networks was performed using for this purpose the dataset partitions proposed in section 5.7.1. The results for the test set, using the trained networks and considering the repetition of the partitioning and training processes three times, can be seen in table 5.2.

Table 5.2: Results for the test set of the three proposed single layer networks

Id	Hidden layer (h)	\overline{MSE}
SL_1	500	$(1.064 \pm 0.034)10^{-3}$
SL_2	1000	$(1.259 \pm 0.046)10^{-3}$
SL_3	2000	$(2.163 \pm 0.056)10^{-3}$

The **MSE** values obtained have no meaning per se. They are used as a reference for comparing the different methods. Afterwards, the process of adapting the robotic system to the environment will be used to assess whether the predictions performed by these trained networks are more or less adequate.

5.7.2.2 Deep neural network option

A deep neural network architecture combining two stacks of autoencoders is presented in this section to deal with the stated regression problem defined in equation (5.14).

An autoencoder (**AE**) is a simple learning circuit that aims to transform inputs into outputs with the least possible amount of distortion (Baldi, 2012). The autoencoder neural network is an unsupervised learning algorithm that tries to learn an approximation of the identity function subjected to several constraints, such as limiting the number of the hidden units. An autoencoder has two parts, a decoder and an encoder. The output of the encoder is a representation of the input, whereas the output of the decoder is the input reconstruction from the encoder representation (Rifai and Muller, 2011). The autoencoders have been tested and compared with other classical methods, such as principal component analysis (**PCA**), to reduce the data dimensionality (Holden and et al., 2006).

Depending on the number of hidden units of the autoencoder with respect to the number of neurons in the input layer, two kinds of autoencoders can be considered. In the *contractive autoencoders* (CAE), the number of neurons in the hidden layer is smaller than the number of neurons in the input layer; therefore, the autoencoder is forced to learn a short representation of the input. In turn, when the input layer has fewer neurons than the hidden layer, and a sparsity constraint is imposed, the autoencoder will still discover an interesting structure in the data (Ng, 2011; Meng et al., 2017). This kind of autoencoder is called sparse autoencoder (SAE).

A stack of autoencoders is built by chaining the hidden layer activations of one autoencoder as inputs for the next one (Bengio et al., 2007). In this way, the autoencoders generate a hierarchical stack. One of these stacked of autoencoders is composed by CAEs, and the other one is built using SAEs. Finally, both are linked with each other using two single feedforward neural networks (Figure 5.11).

The underlying idea for proposing this architecture is to take advantage of the properties of both classes of autoencoders for compensating the huge difference between the inputs and outputs. Therefore, the morphological parameters ($\Gamma^{(m)}$) are the inputs of the sparse autoencoder stack. Since these autoencoders expand the information provided by the morphological parameters, the condition ($u'_2 > u'_0$) is satisfied. In turn, the adaptive controller parameters (θ) are the inputs of the stack of contractive autoencoders. In this case, the compression of the information is intended, therefore $u_2 < u_0$ (see Fig. 5.11).

Many combinations of these network architectures have been tested in this work. One of their advantages is the possibility to train each level of the network separately. However, the obtained conclusion is that if the number of layers is significant for the contractive autoencoders, the hidden layer activations become saturated as their number of units is smaller. In turn, the activation of the hidden layers of the stack of sparse autoencoders tends to zero when the number of layers is more significant. Thus, the best configuration for the network layout is to use two contractive autoencoders and two sparse autoencoders to address this problem. The activation function of the hidden layer for all autoencoders is the logistic function, and the output function is linear.

The last activation layer of the stack of sparse autoencoders is the input of a single feedforward neural network with u'_L units in its hidden layer. The output of this neural network is the last layer of the stack of contractive autoencoders. The inverse single feedforward neural network can be trained too. These single networks have a hyperbolic tangent as an activation function. Both the autoencoders and the single neural networks use the SCG algorithm for training.

To properly define each autoencoder that form the proposed stack, it is necessary to determine the number of neurons in each layer ($\{\{u_0, u_1, u_2\}, \{u'_0, u'_1, u'_2\}, \{u_L, u'_L\}\}$) as well as the parameters that are used for their training.

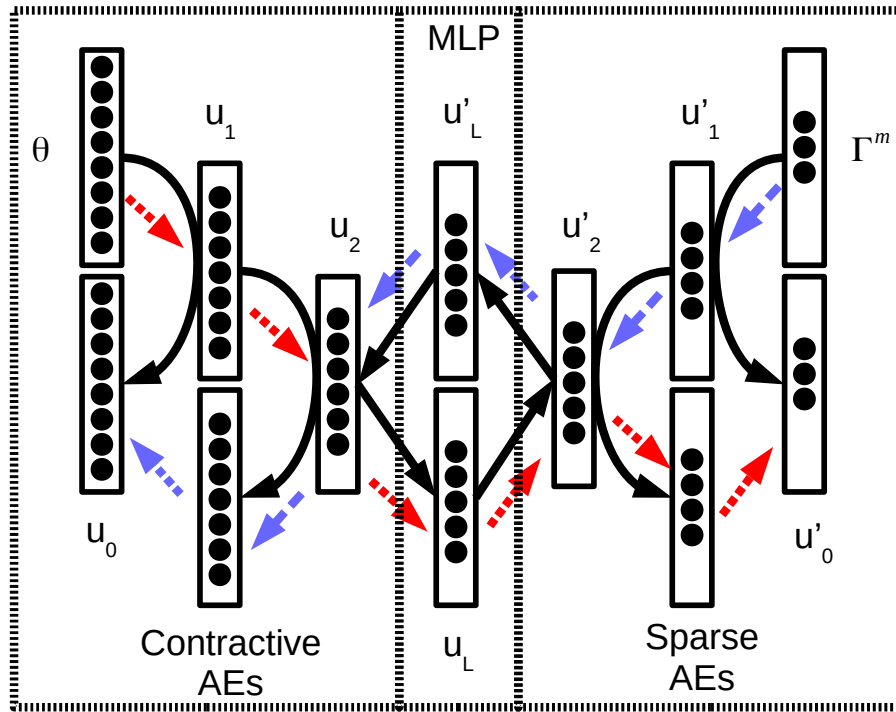


Figure 5.11: The proposed deep neural network architecture. It is composed of two stacks of autoencoders connected by two feedforward neural networks. The parameters $\{\{u_0, u_1, u_2\}, \{u'_0, u'_1, u'_2\}, \{u_L, u'_L\}\}$ are the number of neurons in each layer.

In this particular case, u_0 is the weight matrix of the trained adaptive controller which, as defined, is a 600×3 matrix, i.e. 1800 inputs. The value u'_0 is the input of the sparse autoencoder corresponding to the vector of parameters defining the morphology of the considered robotic system, in this particular case $u'_0 = 16$. The possible combinations in the number of neurons in each of the remaining layers causes the generation of different networks with different performances. A large number of network setups and training parameters are tested to develop the proper architecture for the described network. Finally, the number of layers and the regularisation parameters are fixed for defining each layer's sparse and contractive autoencoders.

In this way, three networks setups were defined ($Mirror_1$, $Mirror_2$ and $Mirror_3$). Their settings, together with the parameters used for training each auto-encoder, are summarized in tables 5.3 and 5.4.

To train and test these architectures, the same partitions are used as for single layer neural networks in previous section. The results obtained for all the proposed networks can be seen in table 5.5.

Table 5.3: Parameters used for training each autoencoder. These are fixed for the three network setups.

Type	CAE	CAE	SAE	SAE
L2. reg	10^{-6}	10^{-2}	10^{-5}	10^{-5}
Sparsity. reg	10^{-6}	10^{-4}	10^{-5}	10^{-5}
Sparsity.prop	10^{-2}	10^{-2}	10^{-1}	10^{-1}

Table 5.4: Distribution of the neurons in the different hidden layers of the autoencoder stacks and the feedforward neural networks (see Fig. 5.11). The layers u_0 and u'_0 are common to every architectures and they have 1800 and 16 neurons respectively.

Layer	Neurons					
	u_1	u_2	u_L	u_L	u'_2	u'_1
<i>Mirror</i> ₁	600	300	100	100	300	50
<i>Mirror</i> ₂	600	150	100	100	150	60
<i>Mirror</i> ₃	900	300	100	100	300	100

Table 5.5: MSE obtained to test the three proposed networks.

Id	\overline{MSE}
<i>mirror</i> ₁	$(0.527 \pm 0.021)10^{-3}$
<i>mirror</i> ₂	$(0.480 \pm 0.020)10^{-3}$
<i>mirror</i> ₃	$(0.543 \pm 0.043)10^{-3}$

5.7.2.3 Parallel feedforward neural network

Thus far, the properties of $\hat{\Gamma}^{(p)}$ described in section 5.3 have not been considered. As stated in equation (5.7), $\hat{\Gamma}^{(p)}$ tends to a multivariate normal distribution with a diagonal matrix as variance. Therefore, each component of $\hat{\Gamma}^{(p)}$ can be viewed as an independent, normally distributed variable, and this applies also to the components of $\hat{\theta} = \{\hat{\gamma}_1, \hat{\gamma}_2, \dots, \hat{\gamma}_k\}$. At this point, the problem of discovering the relationship between $\Gamma^{(m)}$ and $\Gamma^{(p)}$ is decomposed into many small problems which have easier solutions. Formally, equation (5.14) is decomposed into many simpler equations:

$$\begin{aligned}
 \hat{\gamma}_1 &= e_1(\Gamma^{(m)}, \omega_1) + \xi_1; e_1(\Gamma^{(m)}, \omega_1) = \Psi_1(\Gamma^{(m)})^T \omega_1 \\
 \hat{\gamma}_2 &= e_2(\Gamma^{(m)}, \omega_2) + \xi_2; e_2(\Gamma^{(m)}, \omega_2) = \Psi_2(\Gamma^{(m)})^T \omega_2 \\
 &\dots\dots\dots \\
 \hat{\gamma}_k &= e_k(\Gamma^{(m)}, \omega_k) + \xi_k; e_k(\Gamma^{(m)}, \omega_k) = \Psi_k(\Gamma^{(m)})^T \omega_k
 \end{aligned}
 \tag{5.15}$$

In this particular case, for each trained robotic system i , a θ_i has been obtained,

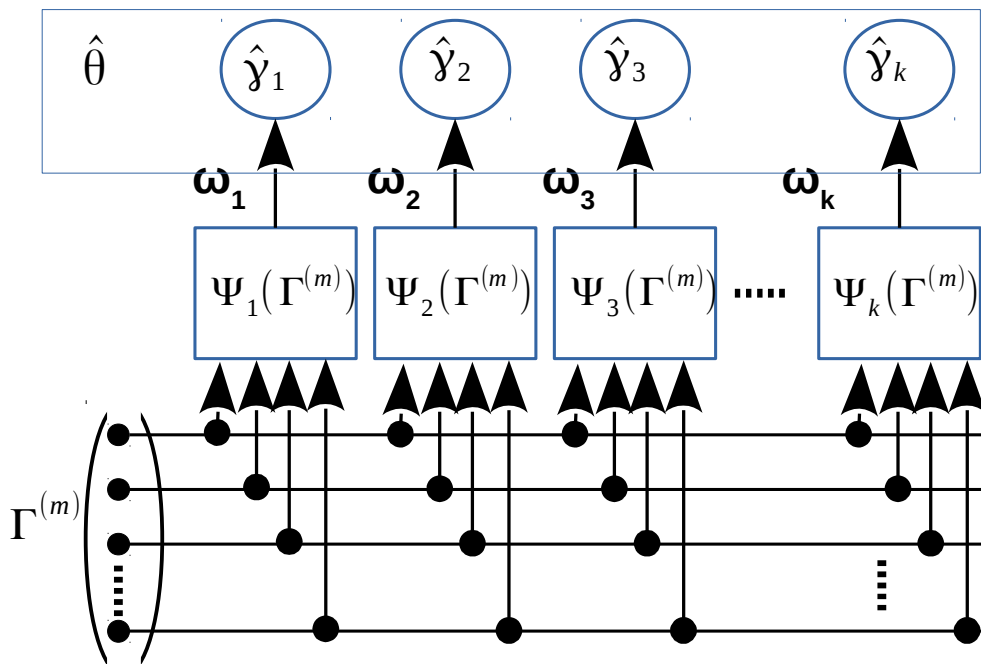


Figure 5.12: Proposed regression problem decomposition into many straightforward regressions.

therefore $\hat{\gamma}_{i,t}$ is an element of the matrix θ_i where $i \in [1, 44271], t \in [1, 1800]$. Figure 5.12 graphically shows the proposed decomposition. Each of these equations is dealt with by using a single layer feedforward neural network. Now, 1800 neural networks with

Table 5.6: Obtained results for the three proposed parallel neural networks

Id	\overline{MSE}
pnn_5	$(0.254 \pm 0.083)10^{-3}$
pnn_{10}	$(0.233 \pm 0.091)10^{-3}$
pnn_{20}	$(0.226 \pm 0.093)10^{-3}$

16 inputs and one output can be used instead of one network with 16 inputs and 1800 outputs. These neural networks must be trained in parallel since they share the same input but have different outputs. In this case, for all networks, the activation function in the hidden layer is a hyperbolic tangent, and the unique output corresponds to one weight of the adaptive controller. The input layer is shared by all networks and has 16 units. The architecture of each neural network corresponds to the one shown in figure 5.10 but with only one neuron in the output layer.

One single neural network is generated for each output, and they are trained sequentially. In this case, as in the previous ones, it is necessary to tune the best value of the number of units of the hidden layer. For this reason, three different possibilities are considered: the networks pnn_5 , pnn_{10} , pnn_{20} , have in their hidden layer 5, 10 and 20 units respectively. Using the same \mathcal{D} dataset partitioning as the previous options, each of these networks are trained sequentially for each robotic system.

After training, these parallel networks are tested using the test dataset for estimating the mean square error. This process is repeated three times. The best result is obtained for 20 units in each hidden layer. The obtained results are summarized in table 5.6.

5.7.2.4 Comparing the obtained results based on sequential test and MSE

In previous sections, three different options have been proposed to solve the regression problem posed. Each proposed alternative has advantages and disadvantages and produces results that must be compared to determine which of the three options offers the best solution.

From figure 5.13, it can be concluded that the best option for learning the relationship defined by equation (equation (5.14)) is the parallel neural networks (pnn_{20}) with 20 neurons in each hidden layer. Even though the deep network solution ($Mirror_2$) is close to these results, it does not reach the same precision. The worst behaviour is that of the single-layer feedforward neural network (SL_1).

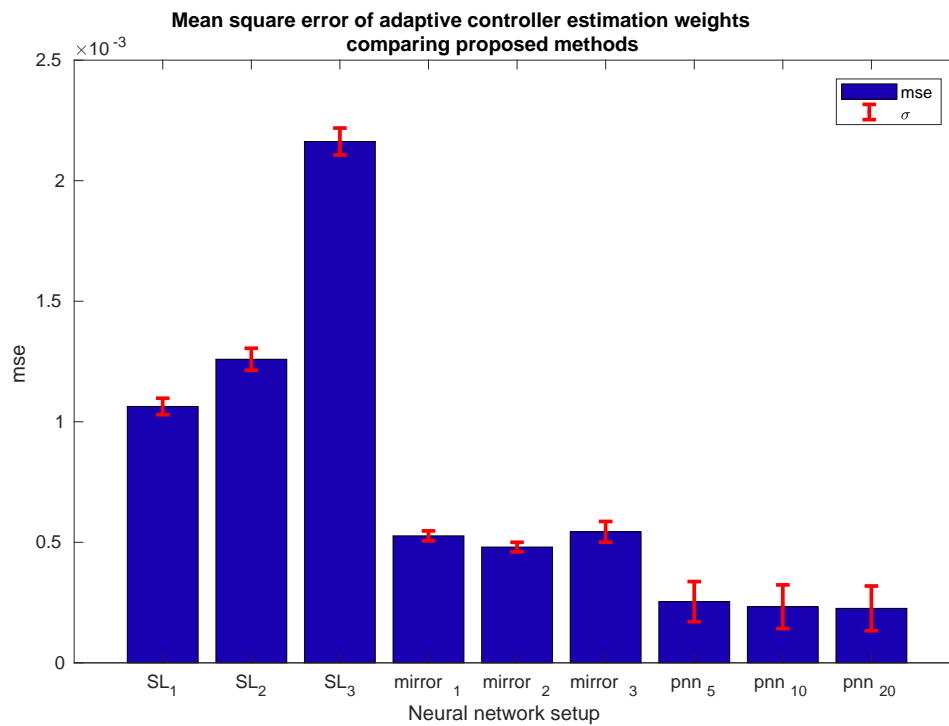


Figure 5.13: Mean square error obtained for different neural network setups. The red line represents the standard deviation over the three repetitions of the training.

Comparing the three methods and taking the best performance as a base, the MSE for *Mirror*₂ is around 2.4 times the value of *pnn*₂₀, and the MSE for *SL*₁ is about 4.7 times the value of the best performance. *SL*₁ is affected by the curse of dimensionality due to the high dimension of the output.

The problem with using **MSE** as an index to compare the three proposed network models lies in the lack of direct translation into how the robotic system is affected when its morphology is estimated by one method or another. The calculation of **MSE** simply indicates the mathematical semblance between the values predicted by the different networks and the values of the weights obtained for each adaptive controller of each robotic system from the dataset \mathcal{D} . The aim now is to check whether the proposed methods can learn the relationship between the internal model and the morphological parameters beyond the **MSE** value.

To achieve this goal, 500 head setups were selected from the test dataset. Each one of these robot heads is characterized by its morphology, defined by $\Gamma^{(m)}$. Using the proposed machine learning methods, equations (5.13) and (5.14) are modeled to predict the internal model parameters for each head setup $\{\mathbf{B}_i, \theta_i\}$.

For estimating the fixed controller of these 500 head setups, the network configuration with the best performance, as described in section 5.7.1 (20 neurons in the hidden layer), predicts $\hat{\mathbf{B}}_i$ from $\Gamma_i^{(m)}$ for each head.

In turn, the initial adaptive controller parameters are estimated for different cases:

- i Taking their initial values randomly ($\hat{\theta}_o$ randomly).
- ii Using previously trained weights (with 1000 iterations) ($\hat{\theta}_o$ trained).
- iii Employing the three neural networks described in previous sections with the best performance for initialising the weights of the adaptive controller (SL_1 , $mirror_2$ and pnn_{20}).

It should be remembered that the value of $\hat{\theta}$ represents the parameters of the internal model, but that they are materialised in the weights of the neural network that defines the adaptive controller. In addition to the weights, to run the updating process according to section 3.5.2, the covariance matrix ($\mathbf{A}_t = \mathbf{R}_t^T \mathbf{R}_t$) and the transformation of the input vectors (\mathbf{b}_t) are needed. If \mathbf{R}_t and \mathbf{b}_t are not correct, the weight vectors will diverge quickly. The point is how to initialise the algorithm from the estimation of the weights. The weights in the training phase are obtained using a gaussian process where ($\mathbf{A}_t = \mathbf{R}_t^T \mathbf{R}_t$) is a covariance matrix and \mathbf{b}_t is a vector of the neural network error transformation. The mean value of the weights in the gaussian process is estimated using equation (3.57):

$$\mathbf{W}_t = \mathbf{A}_t^{-1} \mathbf{b}_t \rightarrow \mathbf{b}_t = \mathbf{A}_t \mathbf{W}_t$$

Therefore, given \mathbf{A}_t and \mathbf{W}_t , the algorithm would start at the point it was (t), and the error would start at the same point. In this case, the value of \mathbf{R}_t was not stored when the training process was finished; however, the neural network has been trained with an upper bound of the covariance matrix, this is the initial value \mathbf{R}_0 before the training process. Therefore, that value could be used to estimate \mathbf{b}_0 as $\mathbf{b}_0 = \mathbf{R}_0^T \mathbf{R}_0 \hat{\mathbf{W}}_t$. From here, as the mean of \mathbf{W} is close to the probable value of \mathbf{W} , the gaussian process algorithm uses the new points to reduce just the variance.

From this, \mathbf{b}_0 is computed as: $\mathbf{b}_0 = \mathbf{R}_0^T \mathbf{R}_0 \hat{\theta}_{xx}$. Where \mathbf{R}_0 is the initial covariance matrix used to configure the adaptive controllers prior to the adaptation process and $\hat{\theta}_{xx}$ are the predicted values by each of the models presented.

Each of these 500 robot heads undergo a training process using the estimated $\{\hat{\mathbf{B}}_i, \hat{\theta}_{xx_i}\}$ for the different five cases. The mean visual error concerning the number

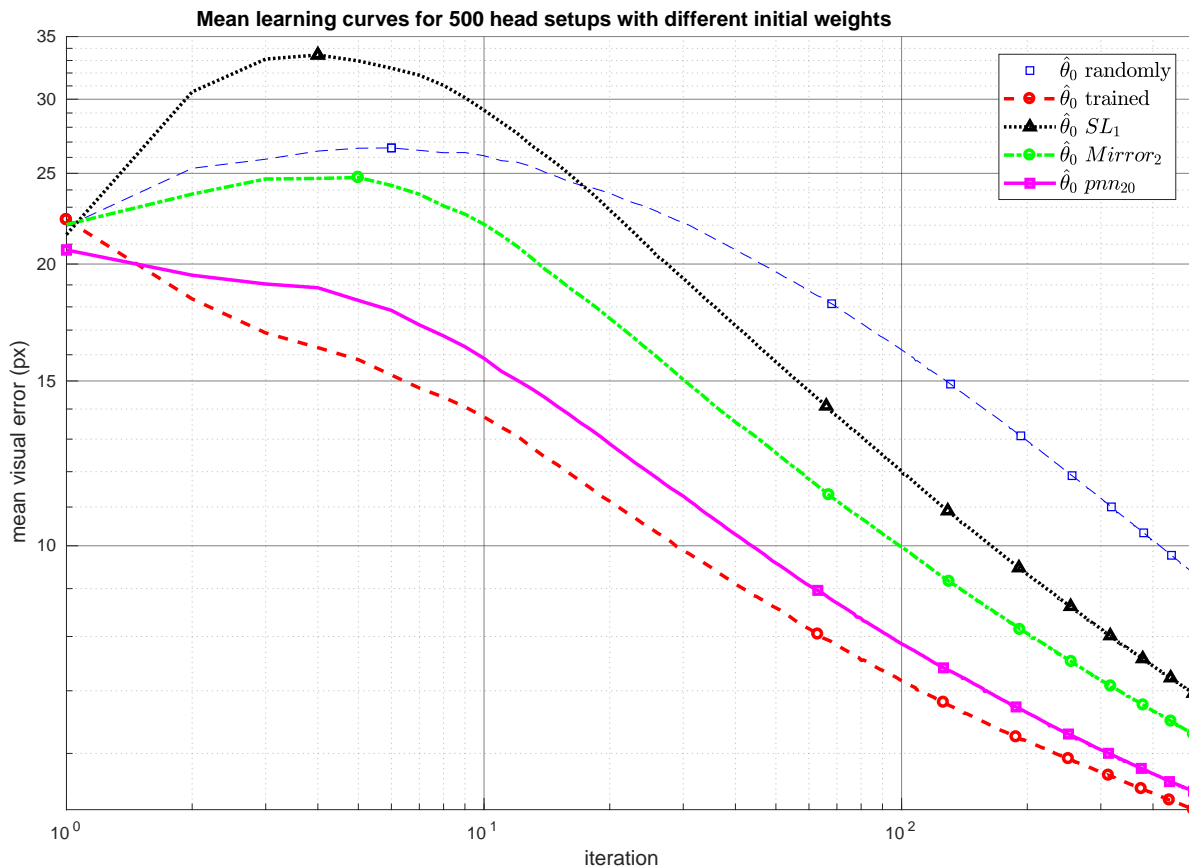


Figure 5.14: Mean of the visual error for 500 robot head setups during the training process using the estimations of the fixed controller and the initialization of the adaptive controller from the morphological parameters for different machine learning procedures

of iterations is shown in figure 5.14. The standard deviation of the visual error for the training process is depicted in figure 5.15.

These experiments evaluate to what extent the different proposed neural networks improve the performance of the training process. Using equations (5.13) and (5.14), a fixed and an adaptive controller are predicted for a particular morphology, and then the adaptation process continues. In this way, if the prediction were perfect, the adaptation process should continue at that point. In Figure 5.14 the visual error is used as a measure of the real performance of the system.

The curve ($\hat{\theta}_0$ randomly) represents the learning curve when there is no previous information about the system. Additionally, the ($\hat{\theta}_0$ trained) curve represents that the networks start the adaptation process initialising the adaptive controller with previously trained weights. In this case, the visual error is stabilised after just a few iterations. This short updating period of the visual error is due to the I-SSGPR algorithm that needs to update other parameters besides the weights (e.g. the system covariance matrix),

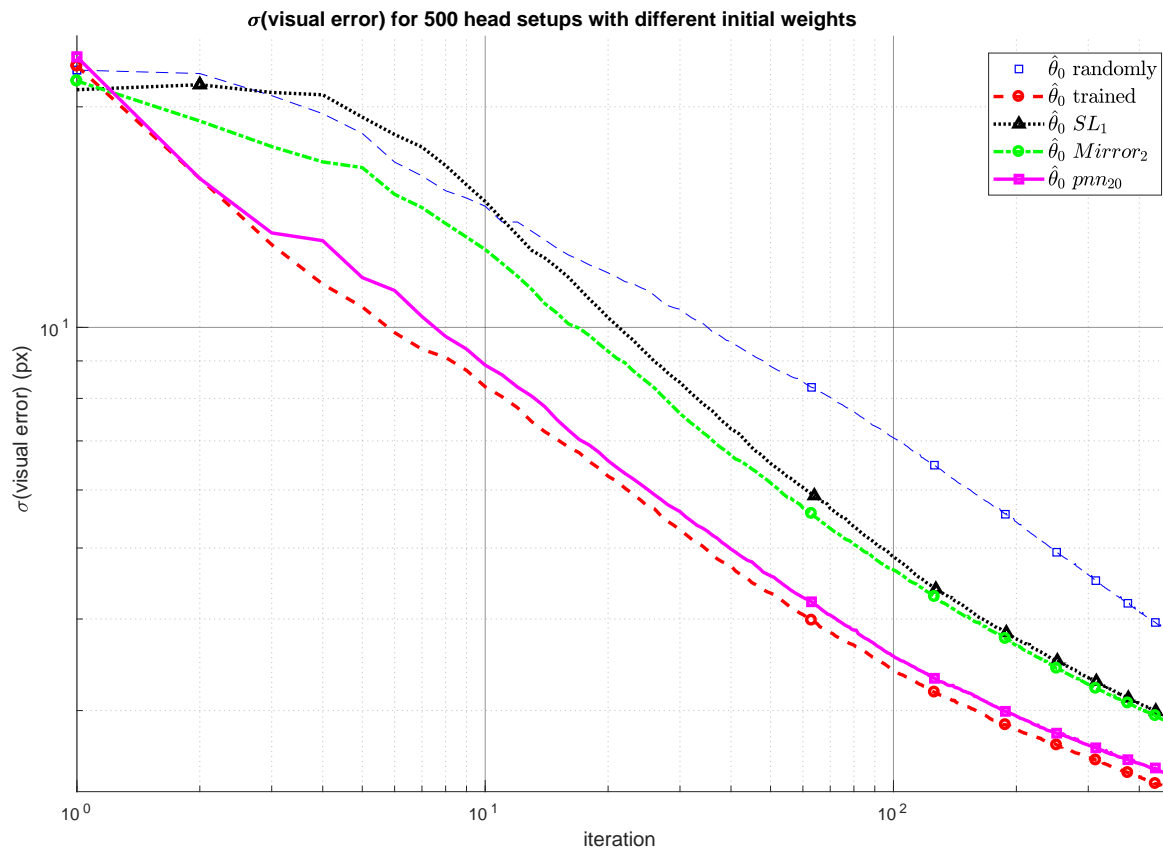


Figure 5.15: Standard deviation of the visual error for 500 robot head setups during the training process using the estimations of the fixed controller and the initialization of the adaptive controller from the morphological parameters for different machine learning procedures

which are initialised to start the training process with the same value for all setups. The curve (θ_0 *trained*) represents that previous information about the system is known, and it would virtually continue the training at the point where it was. The more similar a training curve is to (θ_0 *trained*), the better its performance is.

The learning curves for the three proposed methods lie between (θ_0 *randomly*) and (θ_0 *trained*). Their relative performances correspond to their mean square error as previously described in this section.

As expected, (θ_0 *pnn*₂₀) is the most similar to (θ_0 *trained*), whereas the curve for *SL*₁ yields the worst visual error after the untrained network case, and (θ_0 *Mirror*₂) is an intermediate case. This order is confirmed when the standard deviation is considered. In the updating process, the standard deviation decreases until a constant value. Figure 5.15 illustrates how these variations are grouped; indeed, for (θ_0 *trained*) it is very close to (θ_0 *pnn*₂₀), and similarly for (θ_0 *Mirror*₂) and (θ_0 *SL*₁). That confirms that the behavior of the training curves using the values predicted by *pnn*₂₀ is very similar to (θ_0 *trained*), that uses the trained weights.

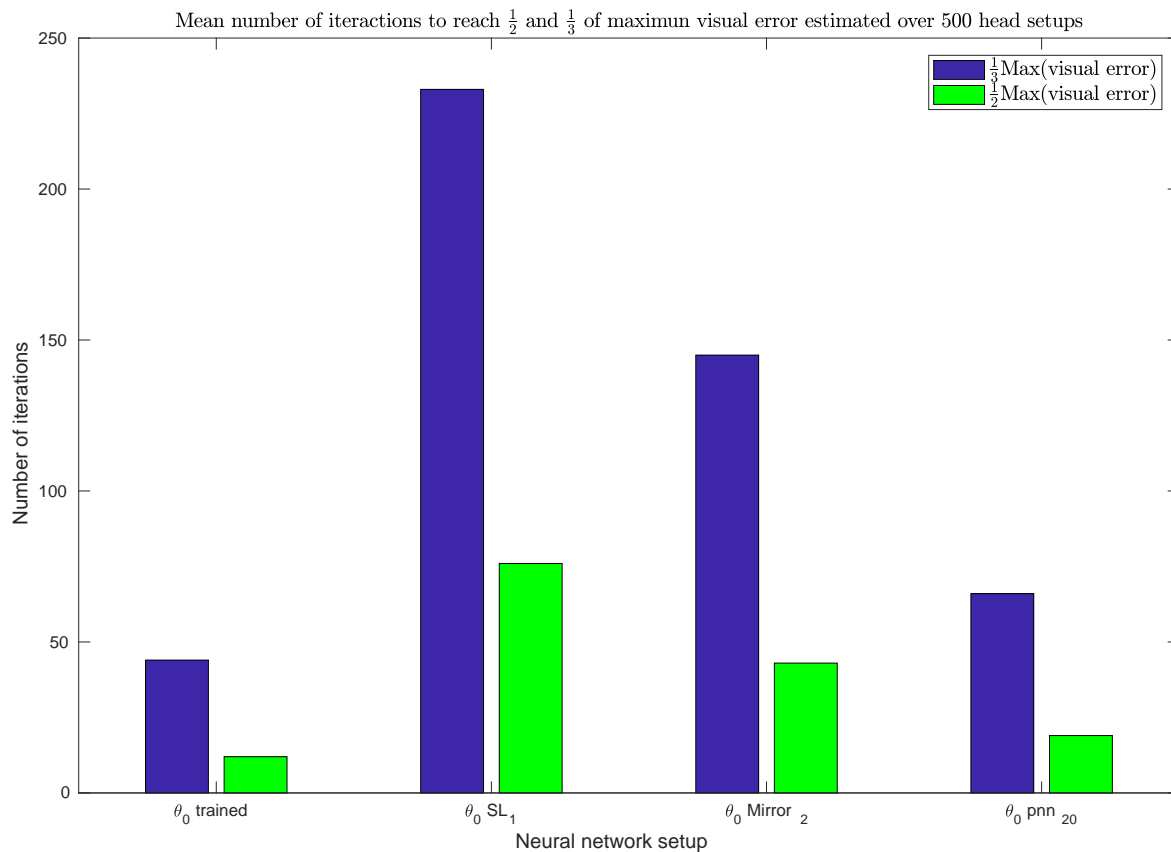


Figure 5.16: Comparison of the three models proposed with the 500 testing robotic heads considering the average variation in the number of iterations executed in the adaptation process to reach 1/3 and 1/2 of the maximum visual error

Another exciting way to confront the three architectures is to compare the speed in reaching a particular visual error of the system; considering the number of average iterations in the adaptation process, the results expected from the obtained MSE are more clearly visualised (figure 5.16).

Finally, the results suggest that the pnn_20 architecture solves better the problem posed. In this case, if the estimated weights with the morphological parameters of each system and the weights of the trained controller within the test set are compared using the MSE, and this value is represented against the visual error obtained after the training of each adaptive controller, figure 5.17 is obtained. It can be appreciated that there exists a clear correlation between them. On the other hand, if the estimation of the weights is not correct, there should be no correlation between the MSE of the weights and the visual error.

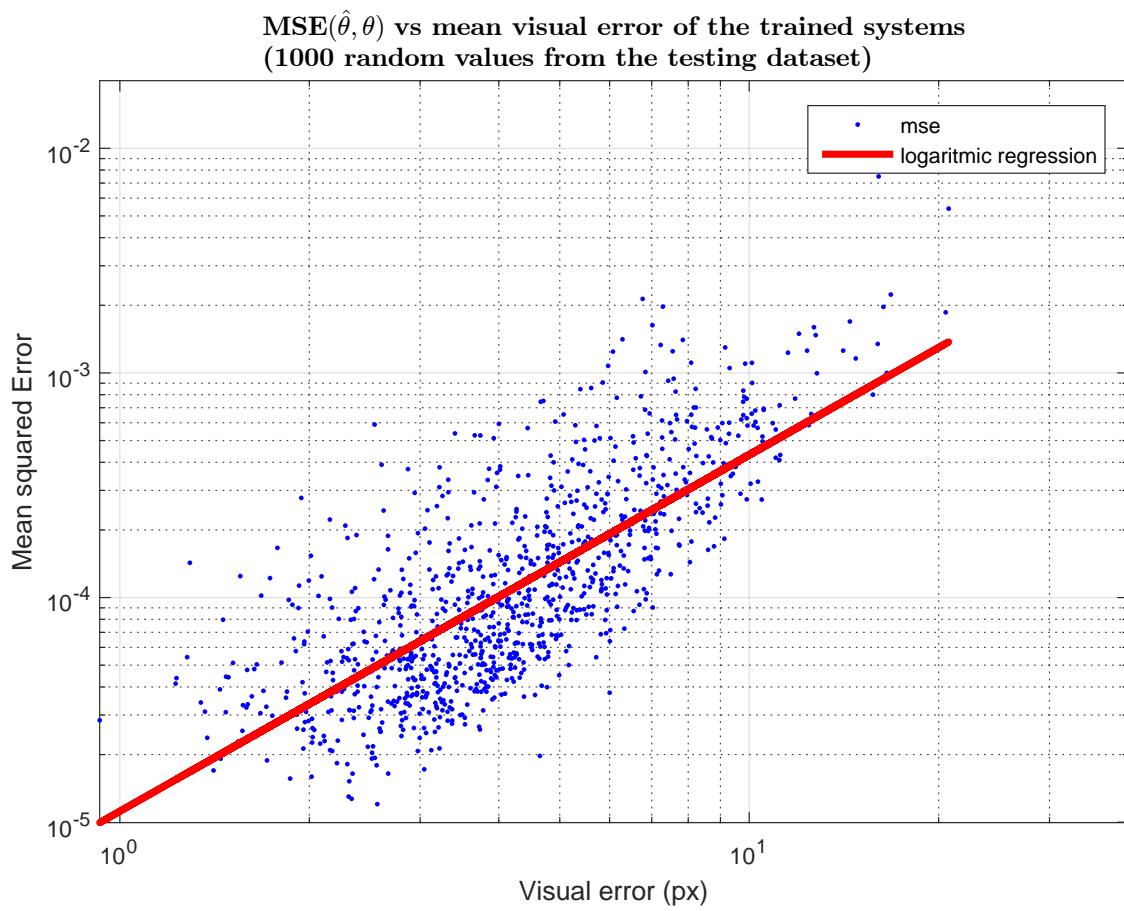


Figure 5.17: The MSE $\{\theta_0, \hat{\theta}_0\}$ vs mean visual error resulting from the θ training for pnn_{20} architecture.

5.7.2.5 Comparing the obtained results based on model selection information criteria

In the previous section, the models proposed to solve the problem of learning the internal model from the parameters defining the morphology of a robotic system have been compared from a practical point of view. Furthermore, since there is a significant amount of data available to train the different proposed models, the utilisation of a test set to evaluate them has been used to assess the performance of the proposals.

The three neural network models have completely different architectures. Although the same training algorithm has been used for all of them, selecting which model suits best the problem is not a straightforward task. Thus, the case of neural networks is a particular case of model selection theory. The weights of the neural networks are just the parameters that describe the model, and therefore given all the possible sets of parameters of the various proposed neural networks, the question is to decide which one fits best the problem posed. However, neuronal networks have a particular characteristic that must be considered when establishing statistical criteria to evaluate the goodness of fitting the model's parameters. [Anders and Korn \(1999\)](#) proposed several strategies for selecting neural network models based on statistical concepts like *hypothesis testing*, *information criteria* and *cross validation methods*. The latter has been used in section [5.7.2.4](#).

In the current section, information criteria are used to select from the three proposed models: single-layer neural network, a stack of autoencoders and parallel neural networks which provides the best solution for learning the adaptive controller weights from the morphology of each robotic system. It is reasonable to think that as the number of parameters (weights) and complexity of a given neural network grows, the effort to determine as accurately as possible the parameters most suited to the underlying model should be greater.

The properties described in section [5.3](#) related to the maximum likelihood estimator are also valid, given a model like the one used to learn the relationship between morphological parameters and the internal model of a set of robotic systems that have experienced a process of adaptation in a shared environment (equation [\(5.14\)](#)). Therefore, the maximum likelihood estimator tends asymptotically to a normal distribution with a variance of σ_{ML}^2 for a given number of training samples (u in equation [\(5.7\)](#)). Applying this to the model for the estimation of the adaptive controller's neural network weights from the morphological parameters of v robotic systems, equation [\(5.7\)](#) can be written as:

$$\sqrt{v} \left(\hat{\mathbf{W}}_2 - \mathbf{W}_2 \right) \xrightarrow{d} \mathcal{N} \left(0, \sigma_{ML}^2 \mathbf{I} \right) \quad (5.16)$$

The precision with which each parameter is determined must be higher in order to maintain the final normal distribution as the complexity of the neural network grows and, therefore, the number of parameters to be estimated (\hat{W}_2) for the same number of samples (v) increases. When a model increases complexity, it uses the training data to adapt the underlying structures, decreasing the bias but increasing the variance.

Training error is not a good estimator of test error. If the complexity of a model is increased sufficiently, it is possible to achieve zero training error; however, due to overfitting, the generalisation ability of the system is weak.

The information criteria for model selection attempts to evaluate the fairness of the fit, i.e. to estimate a measure of σ_{ML}^2 , but adds a penalty associated with the complexity of the model.

A widely used criteria is the *Akaike Information Criterion (AIC)* (Akaike, 1998) based on maximum likelihood estimator and Kullback-Leibler distance (Kullback and Leibler, 1951).

Given a set of models attempting to approximate equation (5.14) and characterised by a set of parameters $\alpha = \{SL_1, mirror_2, pnn_{20}\}$, AIC is a sum of two terms: the first one is an estimation of the error of the model and the second one is a term related to the complexity of the model. In a generic way, AIC expression can be written as (section 7.5 Hastie et al. (2009)):

$$AIC_\alpha = e\hat{r}_\alpha + 2\frac{df_\alpha}{v}\hat{\sigma}_\epsilon^2 \quad (5.17)$$

where:

- $e\hat{r}_\alpha$ is the expected error. It is an estimation of ξ_2 in equation (5.14) for a particular model α . Typically, this error can be computed as:

$$e\hat{r}_\alpha = \frac{1}{v} \sum_{j=1}^v (\theta - \hat{\theta}_\alpha)^T (\theta - \hat{\theta}_\alpha) \quad (5.18)$$

Usually, the calculation of AIC is limited to the training phase to try to predict which variation in the architecture of the used learning model produces minor errors and better adapts to the underlying model. This procedure in the training phase causes an optimistic estimation of $e\hat{r}_\alpha$ because the model adjusts its parameters to describe the available training data better.

In this case, the results of the different tests performed on the variations of each architecture used in the cross-validation methods indicate that the architecture within the same model is more suitable to the regression problem. Therefore, to estimate the error, the values of the adaptive controller weights corresponding

to each morphology in the test set and the weights estimated by the different proposed architectures are used:

$$\begin{aligned} e^{\hat{r}}_{SL_1} &= \frac{1}{v} \sum_{j=1}^v \left(\theta^{(s)} - \hat{\theta}_{SL_1}^{(s)} \right)^T \left(\theta^{(s)} - \hat{\theta}_{SL_1}^{(s)} \right) \\ e^{\hat{r}}_{mirror_2} &= \frac{1}{v} \sum_{j=1}^v \left(\theta^{(s)} - \hat{\theta}_{mirror_2}^{(s)} \right)^T \left(\theta^{(s)} - \hat{\theta}_{mirror_2}^{(s)} \right) \\ e^{\hat{r}}_{pnn_{20}} &= \frac{1}{v} \sum_{j=1}^v \left(\theta^{(s)} - \hat{\theta}_{pnn_{20}}^{(s)} \right)^T \left(\theta^{(s)} - \hat{\theta}_{pnn_{20}}^{(s)} \right) \end{aligned}$$

Where $\theta^{(s)}$ are the values of the weights of the trained adaptive controller from the test set and $\hat{\theta}_{\alpha}^{(s)}$ is the prediction made by the corresponding neural network using the morphology parameters of the test set as inputs.

- df_{α} is related to the number of parameters of the used model to solve the regression problem. When this model is linear, d_{α} is the number of parameters of the model. For example, a simple linear model such as $Y = \Theta_0 + X\Theta_1$ would have a value of $df_{\alpha} = 2$. if the model is non-linear, the concept of *generalized degrees of freedom* **GDF** is applied. Considering an additive-error model (equation (5.14)), a general definition of **GDF** can be written as:

$$df_{\alpha} = \frac{\sum_{j=1}^v Cov(\hat{\theta}_{\alpha,j}^{(s)}, \theta_j^{(s)})}{\sigma_{\alpha,\xi_2}^2} \quad (5.19)$$

Where $\sigma_{\xi_2}^2$ is the variance of the additive error of the model:

$$\sigma_{\alpha,\xi_2}^2 = Var(\hat{\theta}_{\alpha,j}^{(s)} - \theta_j^{(s)}) \quad (5.20)$$

- The value of $\hat{\sigma}_{\epsilon}^2$ is an estimate of the noise variance obtained from mean squared error (**MSE**) of a low-bias model. In this case, the value of **MSE** is the one obtained in the previous section for pnn_{20} network.

AIC is useful to compare several models. However, this value does not give information about the performance of a particular architecture to approximate the real model. Moreover, **AIC** is strongly dependent on sample size; for this reason, it is worth calculating **AIC** differences. In this way, given a set of models α , **AIC** difference for each model is:

$$\Delta_{\alpha} = AIC_{\alpha} - \min(AIC_{\alpha}) \quad (5.21)$$

Such differences estimate the relative expected Kullback-Leibler distance between the real and proposed models. The model estimated to be best has $\Delta_{\alpha} \equiv \min(\Delta_{\alpha}) \equiv 0$. Therefore Δ_{α} is useful in ranking the models and it is possible to approximate the likelihood of a model given data ([Burnham and Anderson, 2002](#)):

Parameter \ Network	pnn_{20}	$mirror_2$	SL_1
p	649,800	2,377,421	910,300
df_α	363,780	135,403	56,975
$MSE \cdot 10^3$	0.226	0.480	1.064
AIC_α	0.347	0.885	1.9218
Δ_α	0	0.538	1.575
$\mathcal{L}(\alpha s)$	1	0.764	0.455
w_α	0.451	0.344	0.205

Table 5.7: Different performance indicators for model selection are shown with the purpose of comparing the network architectures. The values of df_α have been computed according to equation (5.19). p is the total number of parameters (weights and bias) in each architecture. MSE corresponds to the obtained values in previous section. The indicators AIC_α , Δ_α , $\mathcal{L}(\alpha|s)$ and w_α are calculated based on equations (5.17) and (5.21) to (5.23) respectively.

$$\mathcal{L}(\alpha|s) \propto \exp\left(-\frac{1}{2}\Delta_\alpha\right) \quad (5.22)$$

These likelihoods give an idea of the relative strength of evidence for each model. To normalize these values, the Akaike weights are defined:

$$w_\alpha = \frac{\exp\left(-\frac{1}{2}\Delta_\alpha\right)}{\sum_\alpha \exp\left(-\frac{1}{2}\Delta_\alpha\right)} \quad (5.23)$$

These weights are values that identify the evidence that a given alpha model has the best Kullback-Leibler distance from the proposed set of models within a normalised range.

Indeed, AIC and its multiple variants give information on how the proposed models approximate the theoretical model, and in order to perform this comparison with some guarantee, it is necessary to consider that the data used for estimating the error are the same for all the models. In all cases, the structure of the error of the model is the same defined in equation (5.14).

The proposed performance indicators based on information criteria are summarized in section 5.7.2.5. These support the obtained results based on sequential test and MSE to conclude that pnn_{20} is the preferred option. Also shown in section 5.7.2.5, the number of weights and bias (parameters) for each model (p) is appreciably higher than GDF, being the lowest value for pnn_{20} and the highest value for $mirror_2$. However, the

value of **GDF** is higher for pnn_{20} than it is for $mirror_2$; this suggests a better exploitation of the network weights in the case of pnn_{20}

5.7.2.6 Conclusion of comparison

Both the results obtained in the sequential training using the **MSE** as a comparison indicator and the results based on information criteria suggest that the model based on a parallel network with 20 neurons in the hidden layer is the most appropriate architecture to learn the relationship between the morphology and internal model in the adaptive controller case.

The key to understanding this behaviour is to analyse the traits of θ , i.e. the outputs of the proposed neural network models. θ are the final weights of an online neural network that estimates the adaptive controller for the robot head (C_f). When the robot starts to adapt, it is learning the model that is defined by the environment. This knowledge is partially stored in the weights of the online neural network. Therefore their values barely change once the model has been learned. These weights correspond to the maximum likelihood estimator. When C_f is regarded as learned, equation (5.7) is accomplished, and therefore each weight component is probabilistically independent of the rest of them. This trait is a particularity that only applies to the parallel network model. For the other two network models, the weights are statistically related.

5.7.3 Family and individuals of the robotic systems

Because of the results obtained and the ability to model the relationship between the morphology and the internal model of a robotic system, it is possible to define two concepts that could systematically describe how a set of robotic systems adapt to an environment.

Family of robotic systems: The set of robotic systems exhibiting a particular behaviour in interaction with an environment and sharing the same morphological parameters defined in a range. In the case studied in the previous section, the family of robotic systems would be composed of each robotic system whose morphology has been randomly obtained from table 5.1 and trained in the same environment. Thus, all robotic heads exhibit the same behaviour and are implemented in the same way.

It seems clear from section 5.6 that in order to mathematically describe a family of robotic systems, it is necessary to know: the internal model defined by $\Gamma_i^{(p)} = \{\mathbf{B}_i \in \mathbb{R}^{3 \times 4}, \theta_i \in \mathbb{R}^{600 \times 3}\}$ and the morphology defined by $\Gamma_i^{(m)} \in \mathbb{R}^{16}$. It must also be kept in mind how all the family members have been generated from the same set of sparse

features (Ω_m) constituting the adaptive controller and, therefore, the family.

In conclusion, a family of robotic systems can be defined from equation (5.8) on the basis of these parameters:

$$\mathcal{F} = \{\Omega_m; \{\Gamma^{(p)} = \Phi(\Gamma^{(m)})^T \mathbf{W} + \xi\}\} \quad (5.24)$$

Where \mathbf{W} are the trained parameters of proposed solutions from the previous section. Therefore, the family of robotic systems is defined by parameters common to all of them together with a function correlating the morphology with the internal model capable of performing a particular behaviour in a specific environment.

Individual robotic system: A robotic system characterised by belonging to a specific family of robotic systems (\mathcal{F}). Given equation (5.24), an individual must have the same set of parameters Ω_m as the rest of the members of the family. Furthermore, its morphological parameters should range within the valid limits of the relationship of the internal model and the morphology. Therefore, an individual belonging to a family of robotic systems ($\mathcal{I} \in \mathcal{F}$) can be represented as follows:

$$\mathcal{I} = \{\Omega_m; \{\Phi(\Gamma_i^{(m)})^T \mathbf{W} + \xi\}\} \quad (5.25)$$

5.8 Applications

5.8.1 Learning improvement for new robot morphologies

When a saccadic robot system with a learned internal model changes its morphology, the internal model has to be learned or tuned again consequently. As can be deduced from section 5.3, the internal model partially codifies the robot morphology along with the environment interaction.

Frequently, when an internal model is learned, the inputs and outputs of the robot system are considered directly or indirectly, whereas the information regarding robot morphology is omitted because this is implicitly related to the system inputs/outputs. Most of this information is, however, available by physical measuring.

Robots are usually considered complex systems that must follow a control law to deal with a particular environment, and therefore learning implies knowing this function. However, if the behaviour and environment are the same for robotic systems that are different in morphology, the problem is how to use knowledge regarding how a robotic

system has been geared for the environment in order to adapt another robot better.

The experiment described in section 5.7.2.4 intends to evaluate the predictions of different proposals to estimate the internal model. In addition, the experiment shows (figure 5.14) that the initialisation of the adaptive controller from the estimation of the weights—given the morphology of the robotic system—improves the performance of the adaptation in any circumstance. That is, using equations (5.13) and (5.14), a fixed and an adaptive controller are predicted for a particular morphology, and then the adaptation process continues. Implicitly, the knowledge of how the 26500 robotic heads have adapted to the environment is being used to improve—giving a similar head—the training process.

As indicated in the introduction to this chapter, the availability of a global database of systems defined with the same set of parameters makes it possible to predict the internal model in this type of system directly. In this way, they can start adapting to the environment with previous knowledge, speeding up the learning process.

The problem is posed—using the terminology introduced in section 5.7.3—as follows: how the knowledge learned from adapting a family of robotic systems to an environment enhances an individual's adaptation to that family.

5.8.2 Predicting the internal model with partial knowledge about morphology

In section 5.4.1, most of the parameters describing the morphology of a robotic system allowing saccadic movements have been considered. Thus, given a new robotic system where the morphological parameters are known, it is possible to determine the internal model using, for example, one of the proposed solutions of section 5.7. However, not all parameters may be known for the new robotic system, e.g. the dimensions of the kinematic chain may be known, but for example, the camera's pixel size could not be available. This factor would, in principle, preclude the use of the established procedure for estimating the internal model from morphology.

However, when defining both the concept of a family of robotic systems and morphology, no fixed value has been established for the number of morphological parameters used. In the proposed practical case, all the parameters composing the simulation are known, allowing the creation of the dataset of robotic systems. However, considering a subset $\Gamma^{(m)*} \in \Gamma^{(m)}$, it is possible to define the family from equation (5.24) using this subset:

$$\mathcal{F} = \{\Omega_m; \{\Gamma^{(p)} = \Phi(\Gamma^{(m)*})^T \mathbf{W}^* + \xi^*\}\} \quad (5.26)$$

In the case of the regression function defining the family in equation (5.24), all morphological parameters are assumed to be known. In contrast, a partial knowledge of the morphological parameters is present in equation (5.26). However, the internal parameters inferred from both regressions must be the same for the family to perform in the environment.

Sixteen morphological parameters have been considered in the proposed robotic system to illustrate this with an example. There are readily determinable parameters, such as the length of a specific link of the kinematic chain. However, others, such as the camera's nodal point or the pixel size, can be more difficult to measure.

Instead of considering the complete set of parameters represented by the $\Gamma^{(m)}$, the distance from the camera nodal point and the pixel size of each camera is not considered; therefore, the subset $\Gamma^{(m)*}$ is defined with 12 parameters.

Following the procedure of learning the relationship between the morphological parameters and the internal model using a parallel network, it is possible to estimate equation (5.24). Actually, in this case, it is not necessary to generate a new dataset with the unavailable parameters; it is simply necessary to retrain the parallel neural network (pnn_{20}) but varying the number of outputs from 16 to 12. In this way, \mathbf{W}^* of equation (5.26) is obtained. The model has been trained without taking into account the complete set of morphological parameters but just a part of them, and therefore, it is understandable that the value of ξ^* is higher than ξ .

It is possible to compare the performance of the approximation that is made of the internal model of the family of robotic systems when knowledge of morphology is partial by using the same procedure as in section 5.7.2.4. Figure 5.18 summarizes results obtained in this comparison. The four curves shown represent the average of the 500 training curves obtained for 500 robotic systems that are part of the test set and therefore have not been used for training the parallel network (pnn_{20}) to predict the internal model.

The values that are used to initialise the weights of the adaptive controller (internal model) are calculated in four different ways:

1. The initial value of the adaptive controller weights is initialised with random values according to a normal distribution with zero mean. Thus, in figure 5.18, the blue curve corresponds to this case.

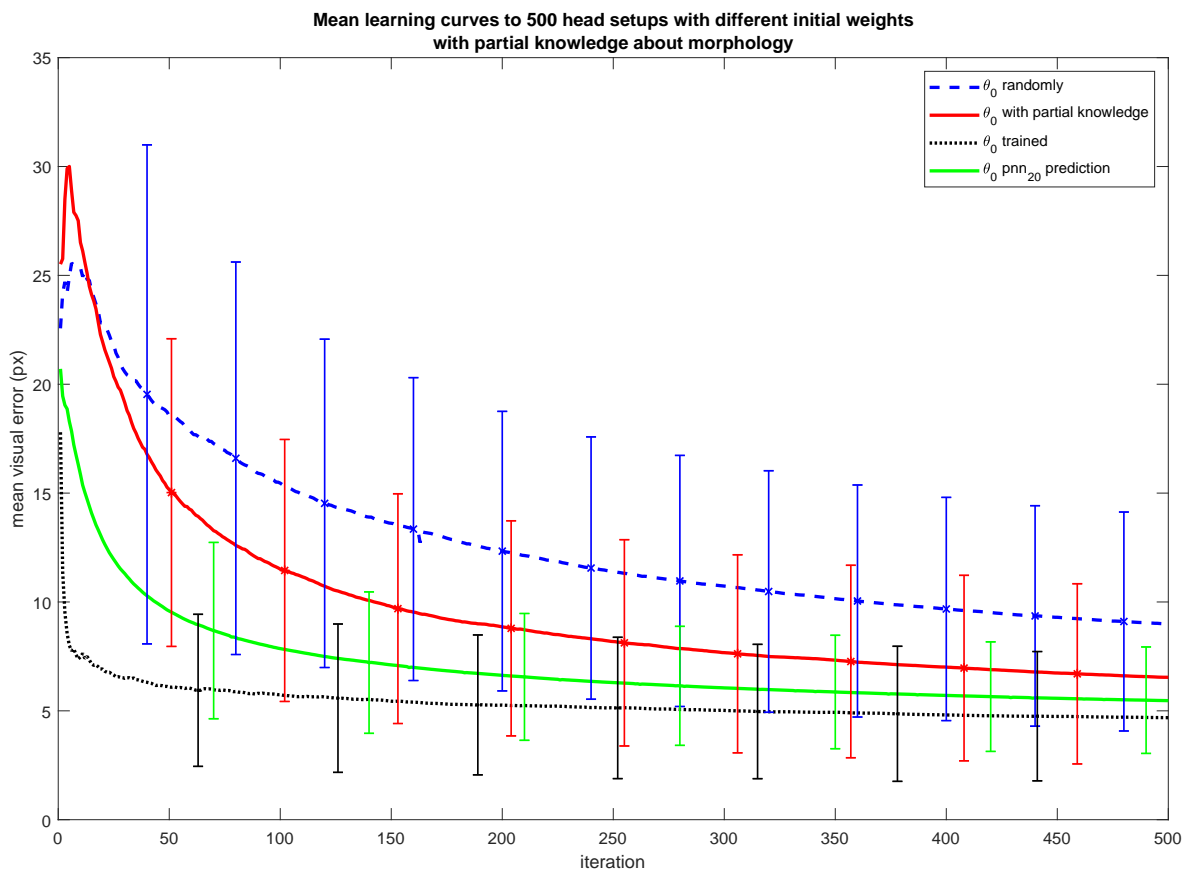


Figure 5.18: Average of the learning curves for 500 systems in the test dataset that has been initialised with the prediction made by the internal model estimation from the morphology.

2. The black curve is the one that represents the progress of the training using the weights at the last point where they were, but resetting the values of the covariance matrix. This curve represents the theoretical maximum point that could be reached in the prediction of the internal model.
3. The green line is the curve obtained from the training of the 500 robotic heads initialised with the prediction of the internal model performed knowing all the morphological parameters.
4. Finally, the red line in figure 5.18 represents the average training curve of the robotic systems under study, where their adaptive controllers have been initialised with the weights predicted from partial knowledge of morphology, i.e. considering only 12 of the 16 parameters constituting the model of the robotic systems executing the saccadic movements.

In every curve of figure 5.18, the error bars identifying the variance at different stages of the adaptation process are shown. As it can be seen, the variance decreases as the adaptation process progresses in all cases. Furthermore, comparatively the highest variance corresponds to the random initiation of the weights, and the lowest variance relates to continuing the training at the point where it stopped. The other two cases are in between these two extremes. Considering all morphological parameters makes the variance lower than partially knowing the morphology of the system.

5.9 Bio-inspired model of artificial genotype and norm of reaction in a robotic system

5.9.1 Introduction

The concepts introduced in section 5.7.3 to define a family of robotic systems and an individual belonging to a given family have strong parallelism to the concepts of species and individual belonging to a species in the natural world. This section presents a model aiming to go beyond simple confrontation. This model is applied to the set of robotic systems presenting the behaviour under study in this paper.

Nature has created a mechanism for the transmission of information that allows organisms to improve throughout evolution. This information is encoded in their genetic material. How this information is decoded in living organisms can be considered from distinct abstraction levels. The low level regards the biochemistry and molecular reactions involved. Hence, a *gene* is a section of a threadlike double-helical molecule

called deoxyribonucleic acid (Griffiths, 2005). The genes dictate the inherited properties of a species, and allelic variations cause hereditary variation within the species. The main elements of form in organisms are proteins. The main task of the living system is to convert the information contained in the DNA of genes into proteins (Griffiths, 2008).

A higher abstraction level considers how to connect the genetic information (genotype) stored in the DNA molecules with a specific characteristic of a living organism (phenotype). In the theoretical scheme proposed by evolutionary genetics, development is the function that maps the genotype onto the phenotype ($G \rightarrow P$).

It is known that the relationship genotype-phenotype is not one-to-one at the lowest levels. At higher levels of interaction, such as morphological traits, the genotype-phenotype relationship is even more complex (Alberch, 1991). Genes can not generate the structure of an organism by themselves. For a gene to influence a phenotype, it must act in concert with many other genes and the external and internal environment. Hence, the $G \rightarrow P$ map is really $G \xrightarrow{E} P$ (GEP) map. For an understanding of this concept, it is fundamental to consider the role of phenotype plasticity and the idea of reaction norm, which are introduced as the basic link relating the three variables (GEP). *Phenotype plasticity* is the property of a given genotype to produce different phenotypes in response to distinct environmental conditions. The fundamental conceptual research tool in phenotypic plasticity is the idea of *norm of reaction* (Pigliucci, 2001). Figure 5.19 shows a diagram relating these concepts.

A norm of reaction is a function that relates the environments in which a particular genotype is exposed and the phenotypes that can be produced. In practice, such a tabulation can only be made for a partial genotype, a partial phenotype, and some particular aspects of the environment (Griffiths, 2005).

Frequently, this abstraction level has been used to model evolutionary behaviours in artificial systems. The $G \rightarrow P$ map is usually the basis of bio-inspired genetic algorithms (GAs). However, such algorithms have been more concerned with imitating the evolution process results to solve searching and optimisation problems. Genetic algorithms emphasise the use of a “genotype” that is decoded and evaluated. These genotypes are often simple data structures (Whitley and Sutton, 2012). Genetic algorithms are a simple form of evolutionary algorithms EAs. These are composed of four components: a genotype, $G \rightarrow P$ mapping, a set of variation operators, and a user-defined function to be optimised called a fitness function. The EAs are often classified as “black-box optimization algorithms” (Doncieux and Mouret, 2014). Overall, this kind of algorithm proposes that, although evolution manifests itself as a succession of changes in a species’ features, it is the changes in the genetic material that form the essence of evolution (Srinivas and Patnaik, 1994). The main idea of these methods is based on

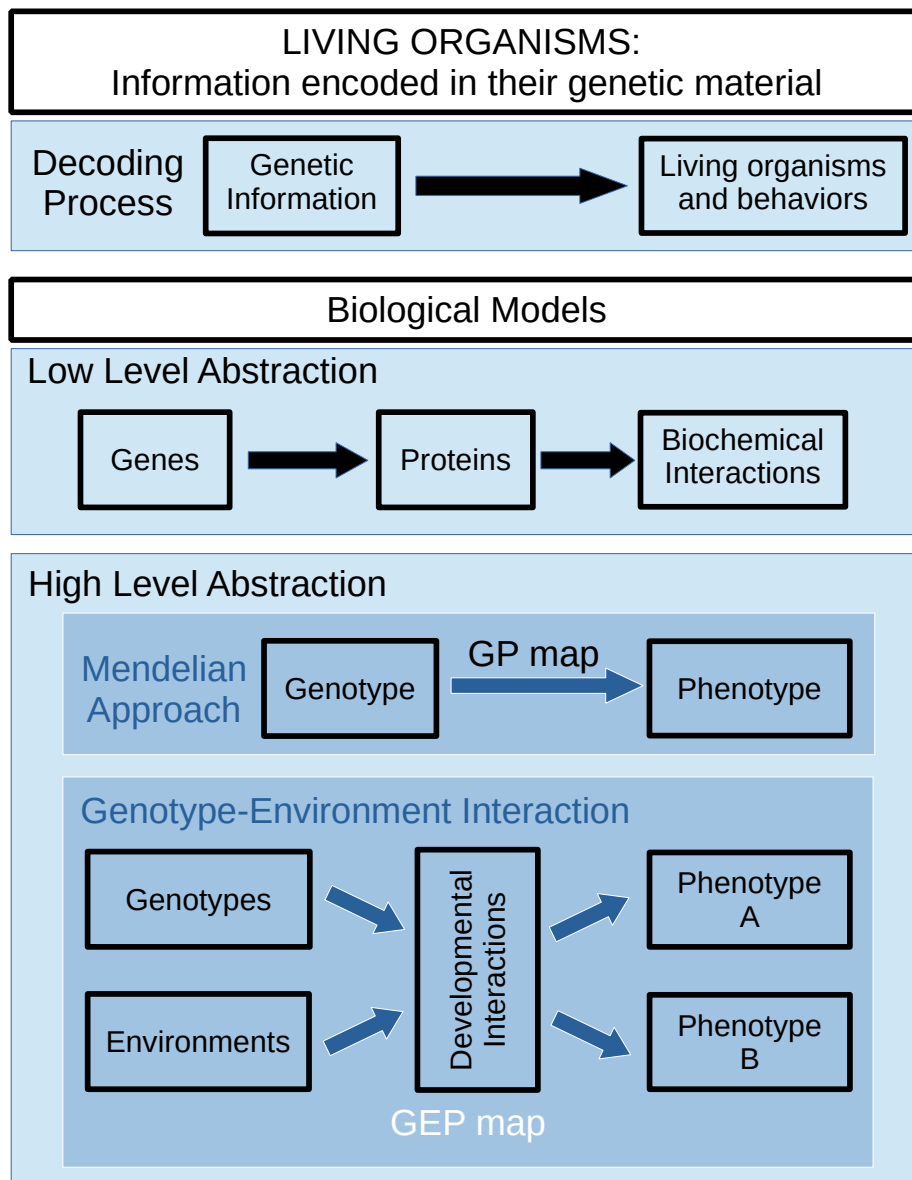


Figure 5.19: This scheme is an attempt to summarise the genetic concepts from an abstract point of view to model them. The decoding of genetic material governs the process of the creation of living beings and their future behaviour. At a low level of abstraction, this process is described and modelled by biochemical and biological processes. Moving up the level of abstraction, two types of models can be envisaged: the simplest is to obviate the intervention of the environment and consider that phenotype and genotype are a one-to-one relationship, this being a Mendelian approach, and the second possibility is to consider the intervention of the environment in the process of genetic decoding and therefore that there is a genotype-environment interaction that produces diverse phenotypes.

the “genetic blueprint” or a “genetic programme”. In other words, genes determine phenotypes. This sort of answer bypasses the process of development, which is treated as an incidental black box with no direct causal relevance to the evolutionary process (Pigliucci, 2010). From this point of view, changes in the species are produced by isolated changes in the individuals. In addition, the influence of the environment is limited to be used merely as a testbed to evaluate the phenotype fitness. Evolutionary Robotics **ER** proposes to employ **EAs** to design robots or, more often, control systems for robots.

Over millions of years of evolution, living organisms have adapted to different environments and have competed for survival, allowing them to improve their phenotypic attributes. From a conceptual standpoint, the information to generate living organisms has been transmitted in successive generations, improving and diversifying in each iteration and generating the particular attributes in each species. Nowadays, any species has the same common phenotype due to evolution because this information is transferred to the new members by inheritance. Of course, the species’ individuals have differences that are usually morphological, but the primary mechanism that accounts for these allelic differences is not a mutation in genes, as in classical **EAs**.

Transferring these concepts to the world of robotics and making a comparison with those introduced in section 5.7.3, a parallel between them can be established. The terms species and individual in biology are equivalent to the terms family and individual defined for robotics. The morphology and behaviour of a robotic system would be equivalent to the phenotypic description of a living being. Finally, in this section, an artificial genotype data structure is proposed to be applied to the field of robotics.

A “species” of the robotic system is used to evaluate the performance of this artificial genotype. The artificial genotype for each species’ individual is obtained, and it is used to check the **GEP** model proposed. The reaction norm of the species’ individuals is estimated to do this. The proposed artificial genotype shows the same behaviour that a biological genotype does concerning phenotype plasticity.

5.9.2 Genotype model

In computer science, the concept of data structure allows the modelling of complex systems where the data and the relationships between them are considered. Thus, If the genotype is considered from this point of view, it is possible to define a data structure that manages the information that describes the anatomy and behaviours of an individual member of a species.

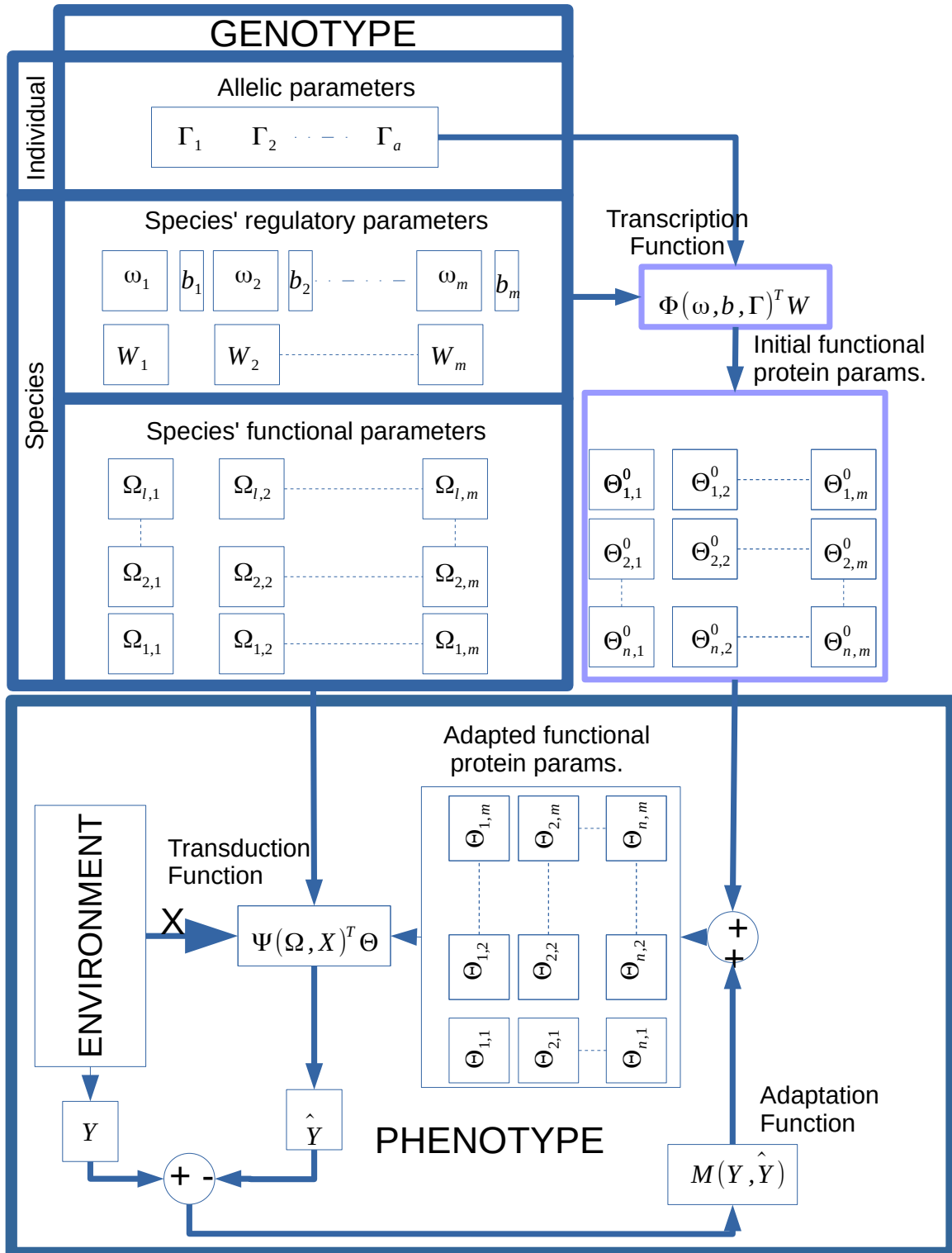


Figure 5.20: Schema of the proposed genotype model and the relationship with the environment and phenotype

The biochemical information stored in the DNA strings is converted in some way in living organisms with their anatomy, physiology and behaviours. In the proposed model, different types of information are defined according to how this information is encoded in living organisms. The genotype model proposed transfers this information to parameter space.

- **Allelic information.** It is the information stored in DNA which encode amino acids and proteins directly, so some phenotypes can be determined directly by this information. This information is encoded by *allelic parameters*: $\Gamma = \{\Gamma_1, \Gamma_2, \dots, \Gamma_a\} \mid \Gamma \in \mathbb{R}^a$, where a is the number of encoded alleles.
- **Species' regulatory information.** It is related to information stored in DNA which does not encode amino acids directly but is shared by all individuals of the species. This kind of data is encoded by the *species' regulatory parameters* (SRP). There are two kinds of SRP parameters:
 - The first class is the *regulatory parameters of transcription function* (RPTF): $\mathbf{W} = \{\mathbf{W}_1, \mathbf{W}_2, \dots, \mathbf{W}_m\} \mid W_i \in \mathbb{R}^t$ where t is the number of combinations of the encoded proteins from allelic genes, to encode a functional protein.
 - The second class is the *combination parameters from allelic* (CPFA) and *species' regulatory information* (SRI): $\omega = \{\omega_1, \omega_2, \dots, \omega_m\} \mid \omega_i \in \mathbb{R}^{a,t}$, $\mathbf{b} = \{b_1, b_2, \dots, b_n\} \mid b_i \in \mathbb{R}^t$.
- **Species' functional information.** It is the information encoded in DNA which is transformed into specific species' phenotypes. The function of this information is regulatory and regards the control of the functional combination of synthesized proteins from allelic information which is encoded by the *species' functional parameters* (SFP): $\Omega \in \mathbb{R}^{l,m}$, where l is the number of the environment modifiers and m is the number of proteins which adjust the obtained phenotype.
- **Functional protein configuration.** It is a sequence of proteins that are obtained from the translation function and regulatory species information. These proteins represent a certain phenotype that the environment can modify. This kind of information is transferred into parametric space:
 - *Initial functional protein parameters* (FPP⁰), these parameters represents the initial proteins synthesized from species' genes, but they are going to be modified by the interaction with the environment. $\Theta^0 = \{\Theta_1^0, \Theta_2^0, \dots, \Theta_m^0\} \mid \Theta_i^0 \in \mathbb{R}^n$ where n is the number of parameters that define a phenotype. The successive modified sets of functional parameters depend on the consecutive

environments where the individual has been adapted. This kind of parameters are named

- *adapted functional protein parameters*: $\Theta = \{\Theta_1, \Theta_2, \dots, \Theta_m\} \mid \Theta_i^0 \in \mathbb{R}^n$ where n is the number of parameters that define a phenotype.

The parameter space defined above can be considered as a data structure. Several operations can be established for modeling GEP mapping (figures 5.20 and 5.21).

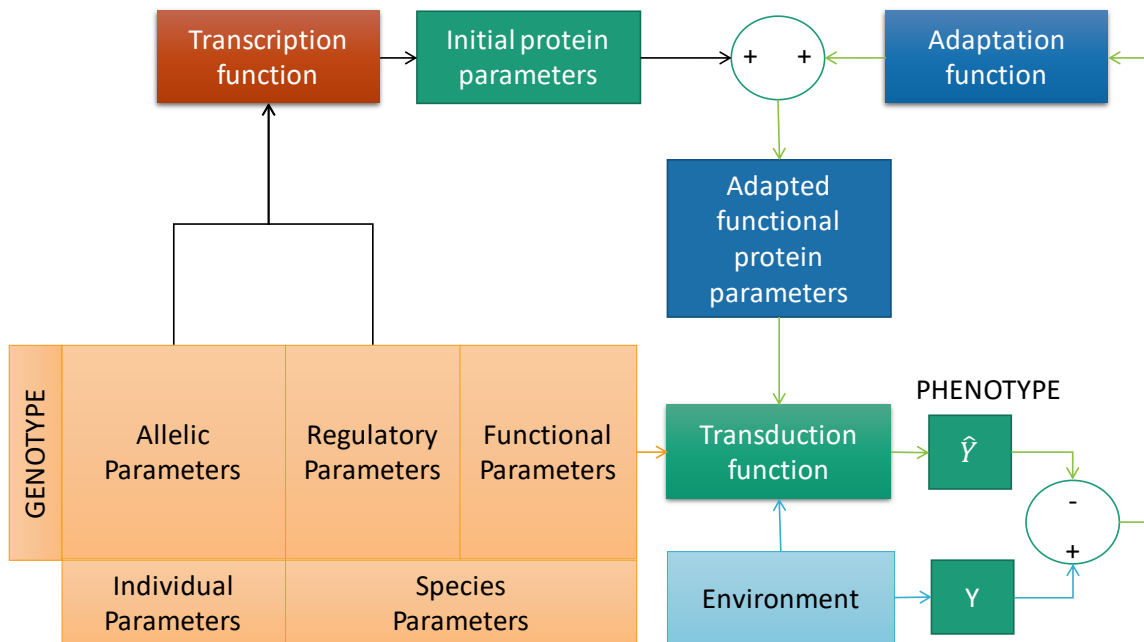


Figure 5.21: Block schema of genotype model and the relationship with the environment and phenotype

Hence, a species' individual has got a genotype defined by the previous structure. One part is specific for this individual (allelic information). In the biological case, this information is represented by allelic genes converted into proteins. The transcription of one gene may be turned on or off by other genes called regulatory genes Griffiths (2005). In the proposed system, this transcription process is modeled by the *transcription function* (equation (5.27)).

$$\Theta^0 = \Phi(\omega, b, \Gamma)^T W \quad (5.27)$$

Mathematically, the transcription function is a regression model that relates the allelic information with initial functional parameters. The SRP fit this model and represent the information shared with every member of this species that accounts for the phenotypic behaviour.

Furthermore, proteins encoded by one gene may modify the proteins encoded by a second gene to activate or deactivate protein function. The equivalent of the latter proteins is the **SFP** in the proposed model. The environment can also modify these proteins through signal transduction. Moreover, proteins encoded by one gene may bind to proteins from other genes to form an active complex that performs some functions. This schema of functioning is modelled by a *transduction function*.

$$\hat{Y} = \Psi(\Omega, \mathbf{X})^T \Theta \quad (5.28)$$

This function finally generates a phenotype. From a mathematical point of view, the transduction function is a recursive regression model, where there are some input cues (\mathbf{X}) that are combined with common fixed parameters (Ω) into a nonlinear function Ψ for all the species individuals and the regression parameters are Θ .

So far, environment adaptation has not been considered. In 1930, Ronald A. Fisher emphasised (Fisher, 1930) that adaptation is characterised by the movement of a population towards a phenotype that best fits the present environment. However, this evolution is produced by changes in the individuals in this population. In the proposed model, this is considered in the *adaptation function*.

$$\Theta_e = \Theta_{e-1} + M(Y, \hat{Y}) \quad (5.29)$$

where e is the number of interacting successive environments. This equation has to accomplish these limit restraints: when $e = 0 \rightarrow \Theta_e = \Theta^0$ and the difference between $\Theta_e - \Theta_{e-1}$ has to tend to zero. The value Y is the optimal phenotype and \hat{Y} is the current individual phenotype.

Equation (5.29) expresses a sequence of changes in functional proteins modifying the phenotype shown by the individual. The motor for these changes is the gap between the optimal phenotype and the current phenotype expressed in a functional way in the adaptation function. A first-degree dynamic system has been defined as where the initial functional protein parameters determine its initial condition. When the individual is adapted, the gap between Y and \hat{Y} has to be minimum (Θ_e^*). Once at this point, there might be another adaptation stage for the individual, so Θ_e^* generates epigenetic changes in **SRP**. These changes are propagated to the descendants improving Θ^0 estimation. From an information point of view, environment adaptation is a learning procedure whose goal is to learn the environment model to better predict the response to environmental cues.

Function in GEP	GEP model	Saccadic Behaviour Model
Transcription	$\Theta^0 = \Phi(\omega, b, \Gamma)^T W$ (5.27)	$\{\Omega_m; \{\Gamma^{(p)} = \Phi(\Gamma^{(m)})^T \mathbf{W} + \xi\}\}$ (5.24)
Transduction	$\hat{Y} = \Psi(\Omega, \mathbf{X})^T \Theta$ 5.28	$\mathbf{Y}_i = \phi(\mathbf{X}_i)^T \Gamma_i^{(p)} + \epsilon_i$ (5.3)
Adaptation	$\Theta_e = \Theta_{e-1} + M(Y, \hat{Y})$ (5.29)	$\hat{\omega}_i = \mathbf{R}_i^{-1} (\mathbf{R}_i^T)^{-1} \mathbf{b}_i$ (3.57)

Table 5.8: Equivalence between the proposed GEP model and the equations used to predict the internal parameters of a robotic system from its morphology.

5.9.3 Applying the GEP model to the robotic systems

The comparison between the biological individual-species binomial with the individual-family of a robotic system introduced in section 5.7.3, has led to the generation of a model based on how the information stored in the genotype of living beings can be converted into anatomy and behaviours. In this section, we discuss how the two fields are connected.

To match the GEP model with the robot families proposed, it is necessary to identify the transcription, activation and adaptation function. In the case of saccadic behaviour studied in this work, the robot must learn to change the camera's position to gaze at the object. The environment transduction cues are the projection of the visual stimulus in the robot camera images. They, combined with the proprioception of the robot, must generate the saccadic behaviour. Thus, the transduction function is the system controller. If the robotic system had perfectly adapted to the environment, the projection of the visual point in the images of the cameras would be in the centre, exactly. So the distance between the actual projection and the image centre could be considered a gap between the optimal phenotype and the showed phenotype. In the GEP proposed model, the system controller represented by a transduction function is modified by an adaptation function depending on the phenotype gap, so the proposed system controller represented by a transduction function is an adaptive controller.

In the proposed FEL model (section 4.4.2.1), there are two controllers, a fixed one (\mathbf{B}) and adaptive (\mathbf{C}_f), both contributions are the system controller. As \mathbf{B} is independent of the environment, it is possible to apply the $G \rightarrow P$ model and \mathbf{B} can be estimated from allelic information (Γ), directly. The \mathbf{C}_f controller is implemented by a single-layer neural network, with seven inputs and three outputs. The environment cues are defined by these seven inputs ($l=7$). Gaussian activations using random space features were used

for the hidden layer. If these random space features are the same for every species' individual, they are the species' functional parameters (Ω) because they regulate the phenotype function. The weights in this network combine the activation functions, as the transduction function is modified by functional proteins parameters in the proposed model; hence these weights are Θ . The dimensions of Θ are the number of units in the hidden layer (n) and the number of outputs (m). The adaptation function is equivalent to adapt the weights in the neural network.

Finally, the transcription function is another regression model that relates the allelic information with the initial functional parameters (Θ^0). Hence, the regression parameters can be obtained if Γ and Θ^0 are known. The problem is to fix the Θ^0 value for each individual in the species. The summary and comparison of all these equations in this chapter and the previous one can be seen in table 5.8. Given these considerations, the genotype model described in figure 5.20 is transformed for the case of the saccadic behaviour model as shown in figure 5.22.

As shown in figure 5.21, there is a set of parameters that are not specific to an individual belonging to a species. In nature, these parameters are determined by millennia of evolution and are encoded in DNA. So the first question is what information is common to all individuals from a given family of robotic systems. It has emerged throughout this chapter that how robotic systems adapt to a given environment seems to be the current nexus of all of them. This information is stored in the weights of the neural network that has learned the relationship between the morphology and the internal model of all the individuals that compose the family. It, therefore, seems logical to assume that the weights of the pnn_{20} network, which have been learned using the 44271 robotic heads in our dataset (section 5.6), constitute the parameters of the species. In this way, using all the defined parameters, it is possible to define the artificial genotype of each individual belonging to a robotic family.

In this way, the procedure to determine the artificial genotype of a family of robotic systems is represented in figure 5.23.

From the morphological parameters (Allelic Parameters) and the values of the network centres of the adaptive controller—shared by all the individuals of the family (Functional parameters)—the relationship between the morphological and internal model of the family of robotic systems should be determined. For this purpose, each of the individuals generated is exposed to the same environment.

In this case, the adaptation process is considered successful when several iterations have taken place. In figure 5.23, it corresponds to the final part of the training curve. This point represents the average visual error shown by the robotic system

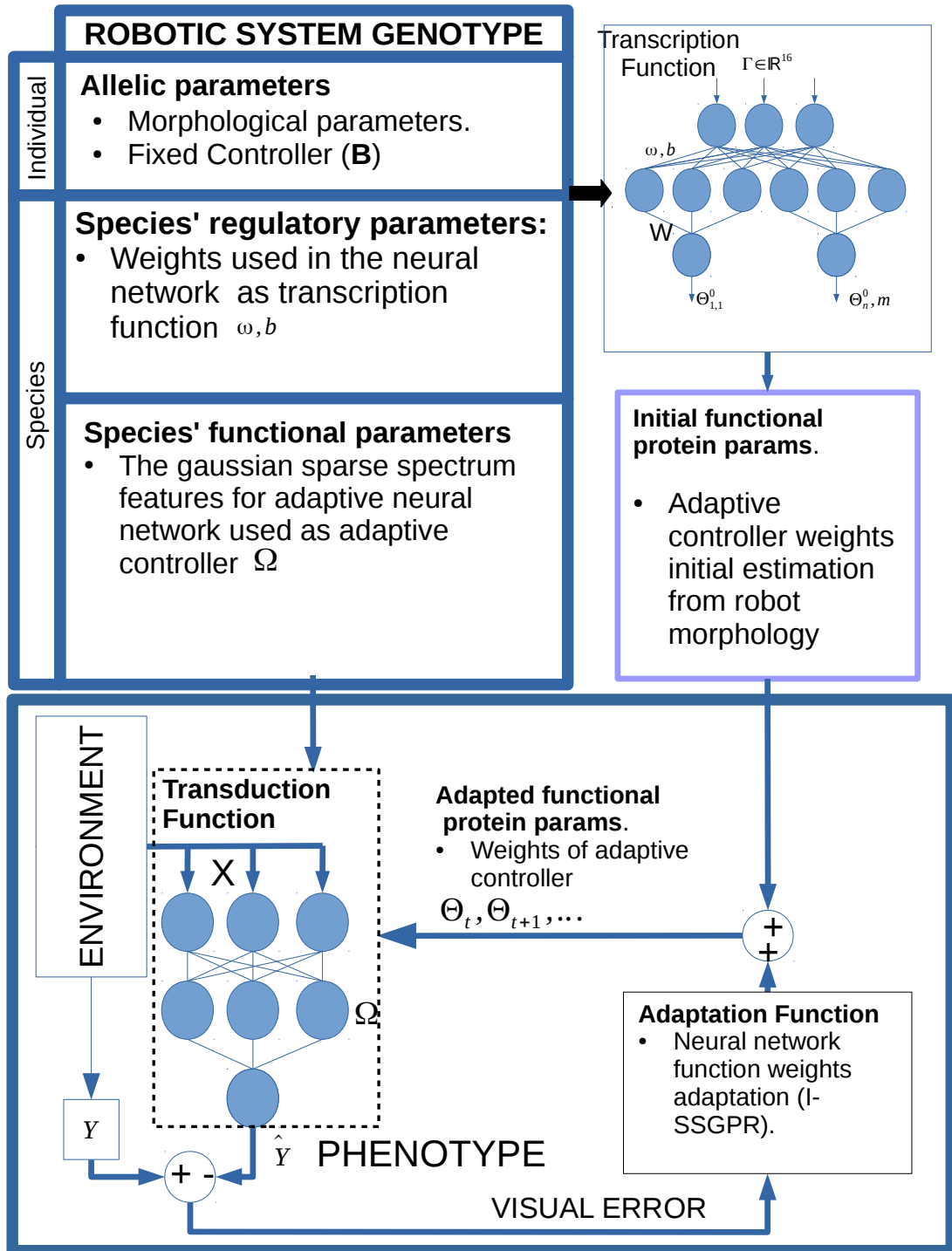


Figure 5.22: Schema of the proposed genotype model and the relationship with the environment and phenotype

when iterating with the environment and should remain constant if the environment does not change or no new adaptations are produced. A visual error can be considered a measure of system behaviour, i.e. it is a phenotypic trait of the system.

Finally, using the established procedure to learn the relationship between morphology and internal parameters, it is possible to determine the characteristic regulatory parameters of the species. As a result, 44271 genotypes are generated that describe each individual of the considered family of robotic systems.

5.9.4 Experiments to evaluate GEP model

In order to evaluate the proposed artificial genotype model, it is necessary to study how the **GEP** model performs in the biological environment. When an individual is placed in one distribution of environments, it produces a resulting distribution of phenotypes. The relationship between them is regulated by the **norm of reaction**. Therefore, each individual in a control population is regulated by the norm of reaction. The relation between genotype and phenotype is complex: a single genotype may produce different phenotypes, and the same phenotype may be produced by different genotypes depending on the environment (Griffiths, 2005).

In any case, the norm of reaction is the function describing this relationship. For example, studies evaluate how a given trait such as the size and shape of the eyes of *Drosophila* fly and the environment temperature can modify them. To estimate the norm of reaction in genetic analysis, a population with a given genotype, such as cuttings of the same plant, is exposed to different environments. For example, variations in the growth place of the plant so that the resulting phenotype—in this case, the height of the plant—has a direct relationship to the elevation of the land on which the plant has grown. Figure 5.24 shows the norm of reaction obtained for three individuals of three different types of *Drosophila* flies.

In the case of a population or family of robotic systems in which the parameters composing its artificial genotype shared by all individuals of this species have been determined, and given an individual belonging to this population (morphological characteristics), it is possible to generate a starting point by employing the transcription function; so that, once this individual interacts with an environment, it produces an adaptation process that generates a final phenotype. As seen in section 5.7.2.4, given a robotic head belonging to a family of robotic systems, it is possible to predict the initial values of the weights of the adaptive controller. When this head is exposed to an environment, the neural network comprising the adaptive controller starts to update the network's weights based on the interaction with the environment by performing saccadic

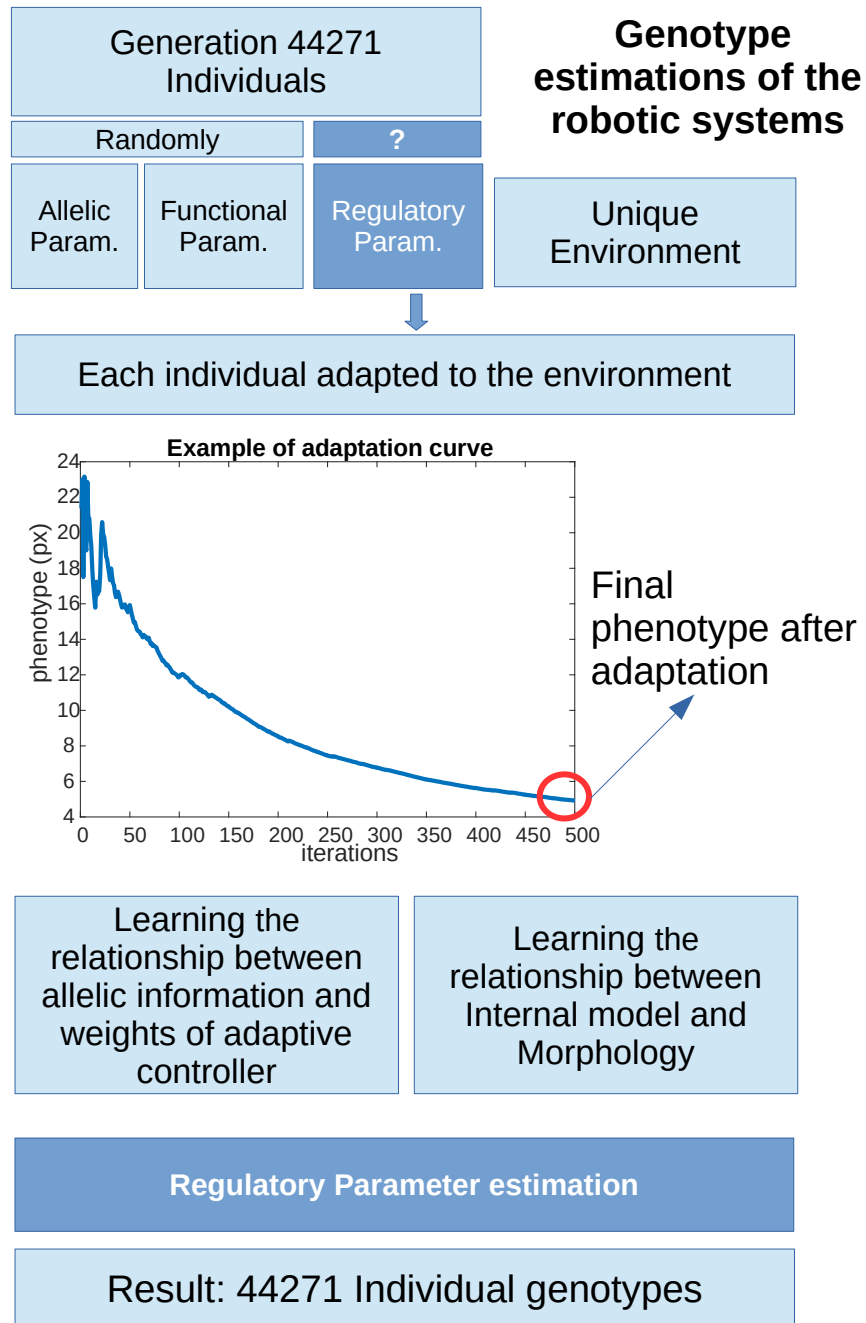


Figure 5.23: Procedure to determine the artificial genotype of the family of robotic system described in section 5.6

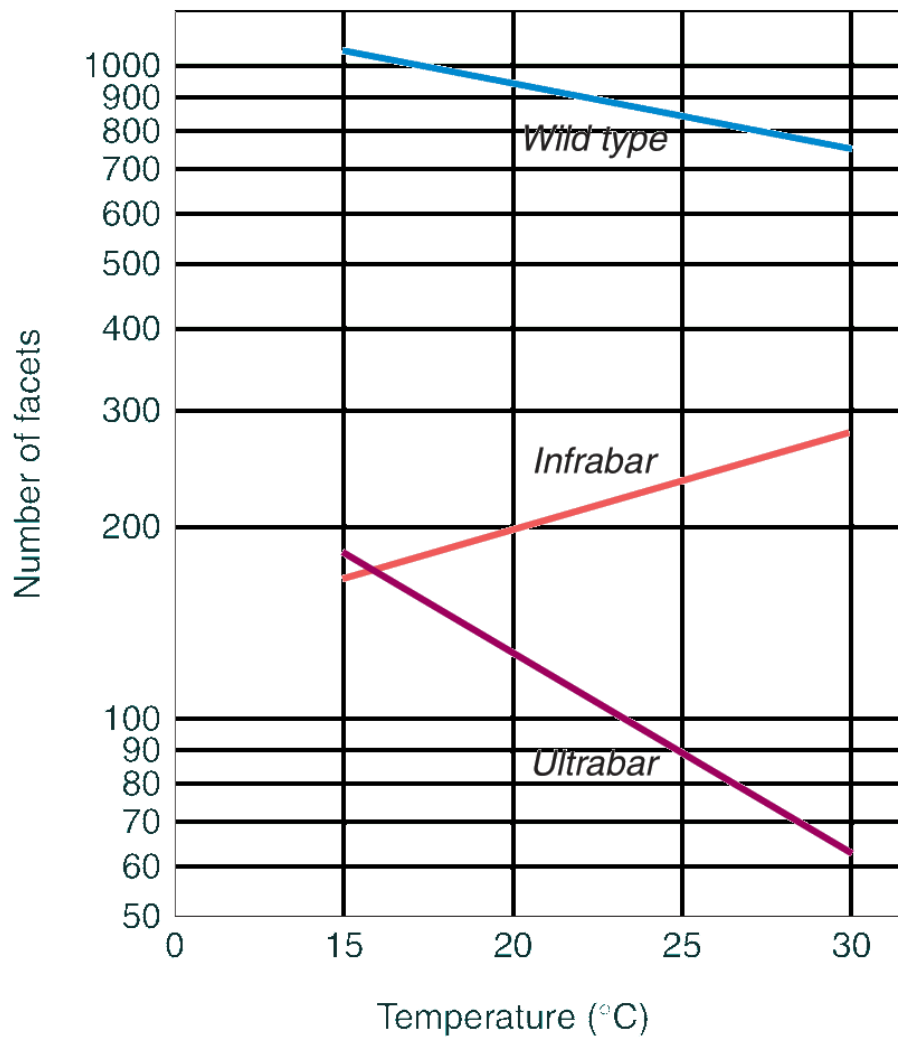


Figure 5.24: Example of norm of reaction obtained for three types of *Drosophila* flies (Griffiths, 2005)[Chapter 1, pp 19]. The facets are related to the size and shape of the fly's eyes, the x-axis shows the constant temperature at which the flies develop.

movements. The result of this adaptation is a learning curve as shown in figure 5.14, where the visual error progressively decreases, reaching a final mean value considered as a phenotype trait.

In order to determine the reaction norm of this family of robotic systems, it is necessary to have several robotic systems with the same artificial genotype and expose them to different environments so that they experience a process of adaptation to generate a final phenotypic trait.

As the environment was defined in section 4.5.2, the geometric centre of the 3D space involved is a good characteristic for representing it. Therefore, it is possible to generate different environments by varying just this property and keeping the others unchanged, e.g. the outer dimensions.

Under these conditions, if the same robotic head with the same artificial genotype is subjected to an adaptation process in each of the generated environments, it is possible to determine the norm of reaction of the system from the different final phenotypes (Visual Error) obtained. This process can be repeated for the 44271 individuals from the family of robotic systems. For the sake of clarity, three individuals from the control group are randomly selected, and the obtained pairs of values (phenotype and environment traits) are represented in figure 5.25. It can be observed that there is a linear correlation. The mean squared correlation coefficient for the control group is $R^2 = (0.930 \pm 0.034)$ for linear regression of their norms of reaction.

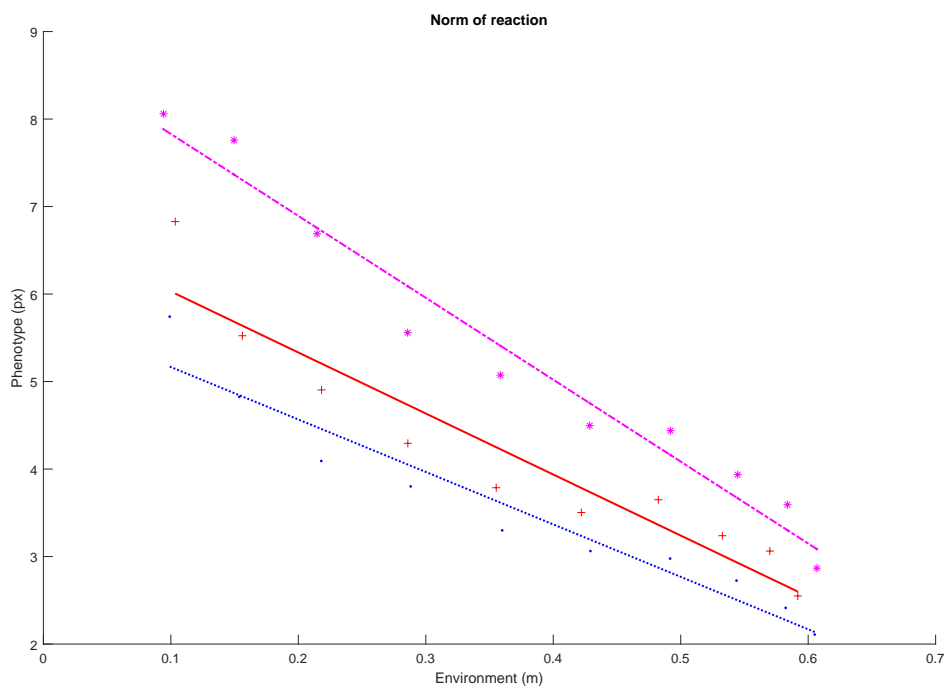


Figure 5.25: Obtained norm of reaction for three randomly robotic heads

5.9.5 Conclusion

As can be seen from the comparison between figures 5.24 and 5.25, variation of the environment causes different phenotypes in the biological systems. In addition, this behaviour is also replicated to some extent by the robotic systems where an artificial genotype was defined.

These samples (figure 5.25) represent three robotic systems from the same species, which show a specific phenotype after an adaptation process using the explained model. These results are similar for every robot in the control group, as the value of R^2 shows. Beyond the shape of the curves, this experiment shows:

- The proposed genotype model can show different behaviours (different curves) for each robotic system.
- The relationship between environments and phenotype can be handled (linearly in this case) by the proposed model.
- In a certain way, phenotype plasticity is achieved by the proposed model. This model differs from classical genetic and evolutionary algorithms, which only consider allelic information for determining an individual's phenotype. This key point is possible because the plasticity of a robot system is achieved without changing the individual genotype.

5.10 Conclusion

In this chapter, the relationship between the three elements involved in or being part of the perceptual loop of a robotic system has been studied.

First, the problem has been addressed from the point of view of a robotic system but considering the environment as a part of it. Then, once the information flows were identified, finding the relationship among the internal parameters and the morphology is converted into a regression problem. The aim has been to change the optics of the problem, i.e. a set of robotic systems is considered sharing morphological parameters (even if they do not have the same value), trying to adapt their behaviour to the same environment. This change has allowed the regression problem to be tackled.

We show how, given the morphological parameters of the system, its internal model can be estimated, as suggested by equation (5.1), and following the hypothesis that the internal model is an approximation of the environment model. Moreover, if the

internal model parameters are learned as a MLE, each one of these parameters is statistically independent (equation (5.7)).

Learning the internal model of a robotic system from its morphology can be a high-dimensional regression problem. This situation was indeed the case for a robotic head implementing a saccadic behaviour we use to verify how the modelled relationship can be learned in a real problem. To address this problem, the internal model parameters were decomposed into two sets. The first one is a set of interdependent parameters for the proposed fixed controller. The estimation for this set was done using a single layer feedforward neural network. The second one is an independent set of parameters corresponding to the weights of the adaptive controller, which was implemented using an online neural network with the I-SSGPR algorithm for its adaptation. In this second case, three different neural network architectures were proposed. Then, the best model was selected according to different performance indicators, such as MSE and indexes for model selection based on information criteria. Also, an experimental evaluation was conducted using the predictions of the proposed models to improve the online training process for a new robotic system. We conclude that the parallel neural network had better performance in all cases than the other two with a smaller number of weights. The independence condition of the learned weights (equation (5.7)) has proved to be essential for selecting the proper architecture for the learning tool.

The concept of family and individual applied to robotic systems has been introduced to address the possible applications of this work. It has been shown that the knowledge of the relationship between the morphological parameters and the internal model can improve the adaptation of a new robotic system to the environment (section 5.8.1). Even though the morphology information is not complete, this improvement is produced according to the results obtained in section 5.8.2.

In the last part of the chapter, a further step in bio-inspiration has been taken. The parallel between family species and individuals from the robotics and natural world has been introduced. Defining a relationship between morphology and internal modelling in a robotic system has allowed us to establish an artificial genotype model that identifies each individual and shares common family (species) traits. Furthermore, it has been shown that this artificial genotype has made it possible to replicate the phenotypic plasticity of living beings by estimating a norm of reaction of the proposed robotic system.

In section section 5.1 , machines and self-reconfigurable robots that can modify their morphology to change their functionality were introduced (Pfeifer et al., 2007). For them, knowing to rapidly generate their updated internal models after a change in morphology will greatly enhance their performance, adaptivity and versatility.

Similarly, in the current trend towards benchmarking and reproducible research (Bonsignorio and del Pobil, 2015), getting the internal model directly from the morphology can significantly contribute to effectively implement the same solutions in robot designs that, being similar, differ in their configuration parameters.

Finally, the greatest potential can be expected in the context of Industry 4.0. and *Cloud Robotics*. *Cyber-Physical* Robotic Systems will have access to big data in the form of libraries of global datasets; cloud computing for statistical analysis and learning; as well as collective robot learning (Kehoe et al., 2015). Given the large variety of existing robot designs –and variations in the parameters of similar designs– for shared knowledge in the Cloud to be fully operational, internal models should be readily available there, as pre-computed datasets or as computing service on demand, so that different robots can take advantage of that knowledge and exhibit a rational behaviour without extensive learning.

5.11 Publications supporting this chapter

- **Duran, A.J., del Pobil, A.P., 2017**, "Discovering the Relationship Between the Morphology and the Internal Model in a Robot System by Means of Neural Networks", in *Intelligent Autonomous Systems 14*, edited by W. Chen, K. Hosoda, E. Menegatti, M. Shimizu and H. Wang, *Advances in Intelligent Systems and Computing* Vol. 531, Springer, Berlin, pp. 839-852. ISBN: 978-3-319-48035-0. Doi: 10.1007/978-3-319-48036-7.
- **Duran, A.J., del Pobil, A.P., 2016**, "Initial weight estimation for learning the internal model based on the knowledge of the robot morphology", in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2016)*, Daejeon, Korea. pp. 3941-3946.
- **Duran, A.J., del Pobil, A.P., 2016**, "A model of artificial genotype and norm of reaction in a robotic system", in *From Animals to Animats 14*, edited by Elio Tuci, A. Giagkos, M. Wilson and J. Hallam , *Lecture Notes in Artificial Intelligence* vol. 9825, Springer, Heidelberg, pp. 267–279. ISBN: 978-3-319-43487-2. DOI: 10.1007/978-3-319-43488-9_24
- **Duran, A.J., del Pobil, A.P., 2018**, "Predicting the internal model of a robotic system from its morphology", *Robotics and Autonomous Systems*, Vol. 110, pp. 33-43. DOI: 10.1016/j.robot.2018.08.014

Chapter 6

Saccadic behaviour for depth estimation

Beauty is no quality in things themselves: It exists merely in the mind which contemplates them; and each mind perceives a different beauty.

David Hume, Of the Standard of Taste and Other Essays (1757)

6.1 Introduction

The saccadic movement has been defined, in chapter 4, from an implementation point of view in a robotic system as a means of exploring the world around us. The chapter 5 addressed the problem of learning this behaviour by a robotic system, not exclusively understood as an individual but as a family of robotic systems interacting with an environment. This chapter presents how small saccadic movements within bio-inspired behaviour can enhance the perception of the environment by a robotic system. In particular, this work is focused on the estimation of a cognitive trait such as depth. Depth perception is the visual ability to perceive the world in three dimensions, along with the ability to measure how far away an object is. Unfortunately, monocular vision is poor at determining depth and usually requires stereo vision.

Depth perception is a characteristic that allows individuals of different species to better adapt to their environment. This issue is crucial in the field of robotics.

The human visual-oculomotor system is a source of inspiration for solving visual perception problems in robotics. Many species exhibit behaviours that require accurate depth estimation in their environments. In particular, primates solve this problem by the concurrent use of multiple estimators deriving from different visual cues (Chinellato et al., 2012a).

In robotics, the most popular sensors used to obtain this information are arguably RGB-D sensors, such as Microsoft Kinect (Han and et al., 2013) and any of its variations (Liu et al., 2019). They are typically based on the known infrared pattern projection. This pattern is deformed depending on the environment's depth round. Then, estimation of depth is computed, employing these deformations.

In computer vision, several methods and algorithms have been established to determine a scene's depth using a single camera or image. For instance, by applying patches to determine the pose of planes in a single image, it is possible to generate the depth map with a single image (Saxena et al., 2009). In addition, if the velocity of the camera is known, the depth map can be deduced (Matthies et al., 1993) from a stream of images. Recent results about obtaining structure from motion with a monocular camera are based on feature tracking and triangulation methods (Petersl and Gabriele, 2010), (Schonberger and Frahm, 2016).

In addition to these methods, novel deep learning approaches use complex neural network architectures to learn the correlation between an RGB image and its equivalent RGB-D in an unsupervised way (Poggi and et. al., 2018), (Eigen et al., 2014a). However, all of these procedures have in common: they only consider the visual cues as inputs, ignoring the camera's motion and sometimes even computing it from the images.

RGB-D sensors and deep learning techniques have certain drawbacks. In the former, objects with absorption in the infrared range are not detected, and these sensors also have problems outdoors and with reflective and transparent objects. In the case of deep learning, long training processes along with vast and pertinent datasets are necessary; moreover, several specific problems arise when this technique is applied in robotics (Sünderhauf et al., 2018). A summary of all these proposals can be seen in figure 6.1.

A cognitive process called fixation is common in certain living beings, including humans. In this process, the visual attention is focused on a point. At first glance, saccadic movements may not seem to be involved. However these movements play a controversial role in this process.

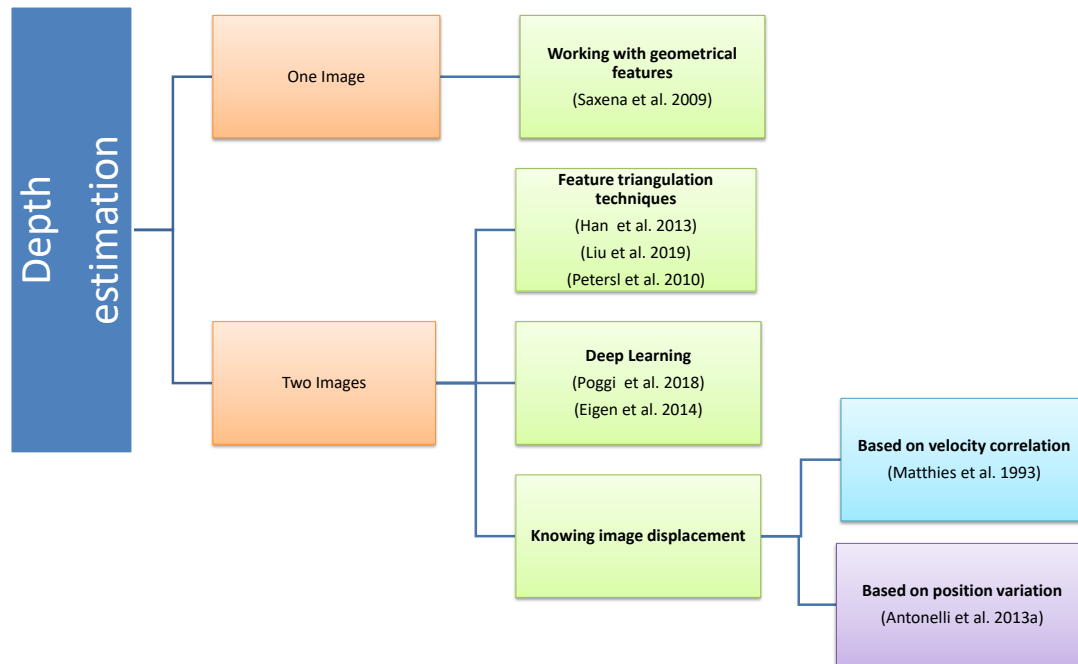


Figure 6.1: Summary of the different approaches referenced and employed in robotics for depth estimation using a monocular camera.

As discussed in previous chapters, saccadic movements direct the gaze towards the area of interest so that the information is maximised in the ocular area where there are more receptors (fovea). Fixation movements are commonly used to understand and explain cognitive tasks such as reading or visual exploration of a scene. During the second part of the 20th century, several authors have studied the attention mechanisms used by human beings. For example, [Yarbus \(2013\)](#) used a famous painting “*An Unexpected Visitor*” by Repin to see what features of the scene the subjects under study were gazing at, attempting to determine the age of the people in the painting. The result was that the subjects were focused on the faces of the people. In addition, using photographs, [Mackworth and Morandi \(1967\)](#) concluded fixations are produced in the areas where complete information is available.

Subsequently, more precise work has been done to introduce the influence of semantic content in the exploration of the scene to study the processes of fixation systematically. [Loftus and Mackworth \(1978\)](#); [Gordon \(2004\)](#) suggest that during the first moments of visual exploration of a scene, an initial fixation is systematically produced in order to identify features globally. These first fixations seem to indicate they are directed towards analysis processes of visual properties beyond semantic search. Afterwards, the scene is subjected to a detailed foveal analysis by the subject.

This work has been developed in this context of the global fixation process since

one of the desired visual properties of the scene is its depth. However, if there is no retinal shift, it is known that the perception fades immediately away. Therefore, small fixational eye movements are suggested to maintain active perception (Pritchard et al., 1960). These eye movements can be microsaccades, tremors and eye drifting. However, it has been considered that this retinal image motion is not enough to determine the depth sign in reference to the fixation plane, and the direction of the image movement relative to the observer motion is decisive to obtain this depth sign (Nadler et al., 2008).

Several species use eye movements in coordination with small displacements of the head during the process of visual fixation to obtain depth information of the gazed scene (Aytekin and , 2012).

The 'fixation' process is anything but fix since tiny intersaccadic eye movements around the gazed location are produced during maintained fixation. A large fraction of these movements are smooth, but seemingly random changes in eye position occur so-called ocular drift and ocular tremor, respectively (Krauzlis and et al., 2017). Moreover, very small saccades (microsaccades) are generated with variable frequency and amplitude during maintained fixation. Even though microsaccades and saccades exhibit similar motor characteristics and share a common neural substrate (Ko et al., 2010), there has been a long controversy over the visual functions of these movements.

Recent studies show that these microsaccades are precisely directed and play a fundamental role in enhancing visual acuity (Intoy and Rucci, 2020).

Other authors suggest that during maintained fixation, very small saccades (microsaccades) are generated at regular intervals (Martinez-Conde et al., 2004). As is also the case with regular saccades, these microsaccades produce visual cognitive suppression during the movement (Irwin, 2003), and therefore they generate strong onset transients in the visual input stream. These transients benefit from visual processing due to a generation of a coordinated and synchronised input signal.

These ideas from biology-inspired earlier work in robotics for distance estimation based on the parallax produced by camera rotations (Santini and Rucci, 2007) and compensatory head/eye movements (Kuang et al., 2012). In later works, the concept is extended to depth estimation (Antonelli et al., 2013a). However, although Antonelli and co-workers based their work on the coordination of the neck and the oculomotor system to maintain the fixation point, they did not consider microsaccadic movements (Antonelli et al., 2014), (Antonelli et al., 2016).

In this chapter, a depth estimation algorithm is presented, considering all these works. This algorithm is based on a series of hypotheses that can be applied to the fixation process produced by human beings.

As with many scientific developments, the algorithm originated from the necessity to determine depth in a particular scenario: A robotic system with an eye in hand camera operating in a warehouse to pick up objects. The guiding line of research was to mimic the fixation movements of the human ocular system with a robotic arm to enrich the visual perception of the environment captured by the robot's hand camera and then to use this information to locate and plan the grasping of objects on the stage.

As can be seen throughout this chapter, the use of saccadic or, in this particular case, microsaccadic movements play an essential role in the development of this algorithm. Therefore, in this chapter, all the concepts introduced in the previous chapters are integrated, implementing a behaviour transcending saccadic movement for simple exploration.

A model is proposed based on a geometric model developed from the decomposition of the eye movements susceptible to be generated in the fixation process. Depth estimation is based on the equivalent of the retinal displacement (measured as optic flow) and the perception itself.

The algorithm derived from the proposed model is tested in simulation, analysing its limitations and the influence on the environment.

Finally, a robotic system is designed and built to allow the implementation and testing of the algorithm in a real environment.

6.2 Objectives

The main objective of this chapter is to develop an algorithm for depth estimation based on the movements that humans produce in the fixation process.

In summary, the objectives to be achieved in this chapter are:

- Based on the geometric study of fixation movements involving microsaccades, generate a mathematical model to determine the depth of a scene.
- Implement the algorithm developed in simulation and study its performance, as well as the factors involved in its execution.
- Try to improve with the proposed algorithm the weak points of the RGBD camera in simulation with transparent objects.
- Evaluate whether the perceptual behaviour originated by the algorithm can be similar to the human being perception in specific circumstances.

- Check the proposed algorithm for depth estimation in a real robotic system.

6.3 Model

6.3.1 Model hypothesis

The human fixation mechanisms are the source of inspiration to develop the proposed model, which is sustained by these hypotheses:

- i) An initial fixation process is systematically produced in order to identify visual features globally in a scene exploration task [Loftus and Mackworth \(1978\)](#); [Gordon \(2004\)](#).
- ii) During the fixation process, head-eye movements can be considered as perturbations around an initial pose. A complex set of coordinated movements implicating the head and the oculomotor system are generated in the fixation process ([Aytekin and , 2012](#)). These movements aim to maintain the gaze point despite random displacements of the head and eyes during fixation.
- iii) So-called visual suppression occurs during microsaccadic movements ([Hafed and Krauzlis, 2010](#); [Irwin, 2003](#)) to the effect that only in the intersaccadic gaps is visual information accessible. In consequence, the fixation process can be regarded as spatial image sampling.
- iv) the primary cue for estimating depth and 3D perception is the optical flow produced by the observer. When it is not the result of external movements, optic flow and motion parallax are consistent when other depth cues are not available ([Fantoni et al., 2010](#)).
- v) The contribution generated by ego-motion signal makes it possible to clarify the inherent ambiguity associated with the optical flow ([Jain and Backus, 2010](#)).

6.3.2 Mathematical model

When the fixation process starts, there is no information available about the depth of the scene. In any case, this process must start at an initial stimulus point detection on the retina. Probably, by executing a saccade, the oculomotor system is adapted to focus this initial stimulus on the fovea. The visual perception (image) that is received by the visual system at this moment is taken as a reference. At this point begins a series

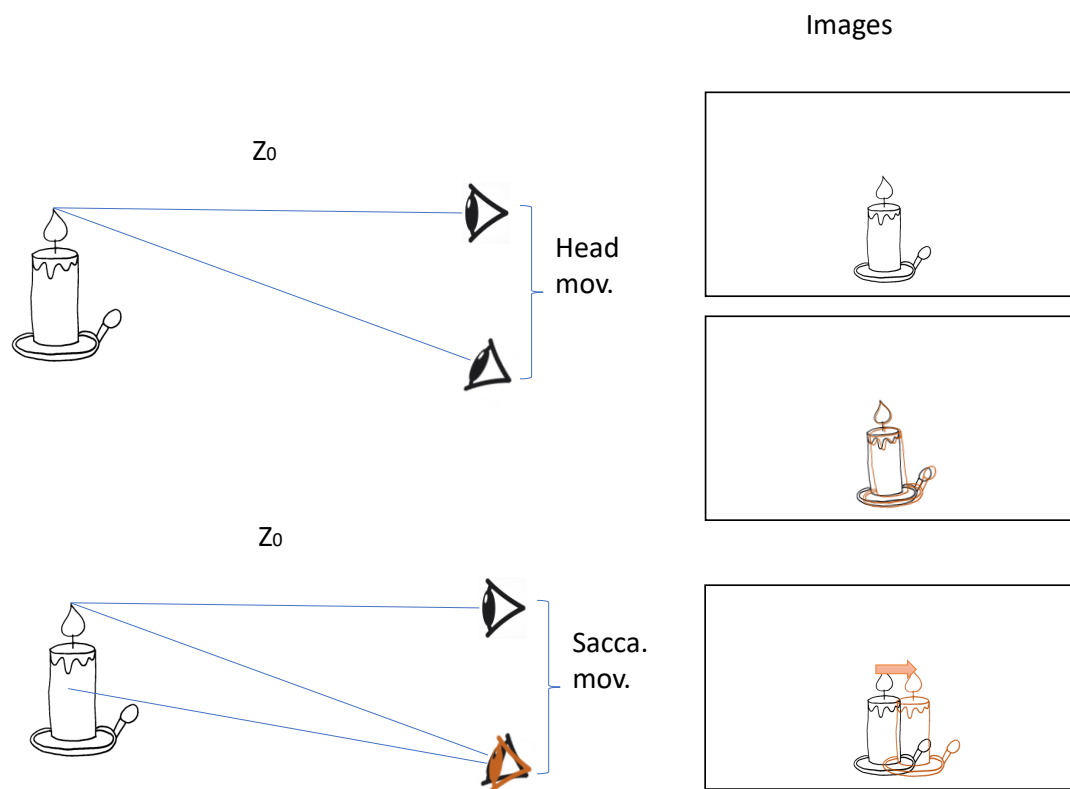


Figure 6.2: Example of how microsaccades can affect perception in the fixation process.

of small movements producing active retinal displacement to help maintain the visual perception stream. Among these movements are microsaccades, which, as indicated in section 6.3.1, hypothesis ii, the role given to them in this model is not only to generate displacements in the retina but also to provoke a sampling of the image stream.

In this case, we consider that the shift induced in the retina is produced exclusively by the generation of eye movements. Therefore, the stage where the fixation process is taking place remains static.

Due to perception suppression, images are not considered during saccades. However, when the saccadic movement has just finished, the image received by the visual system is compared with the reference image and depth perception is updated with this new information.

In reality, multiple types of movements are involved in the fixation process; modelling them together would complicate the problem excessively. For simplicity, only microdisplacements of the head and those produced by microsaccades are considered. In order to focus the problem, the example in figure 6.2 is shown. The desired goal

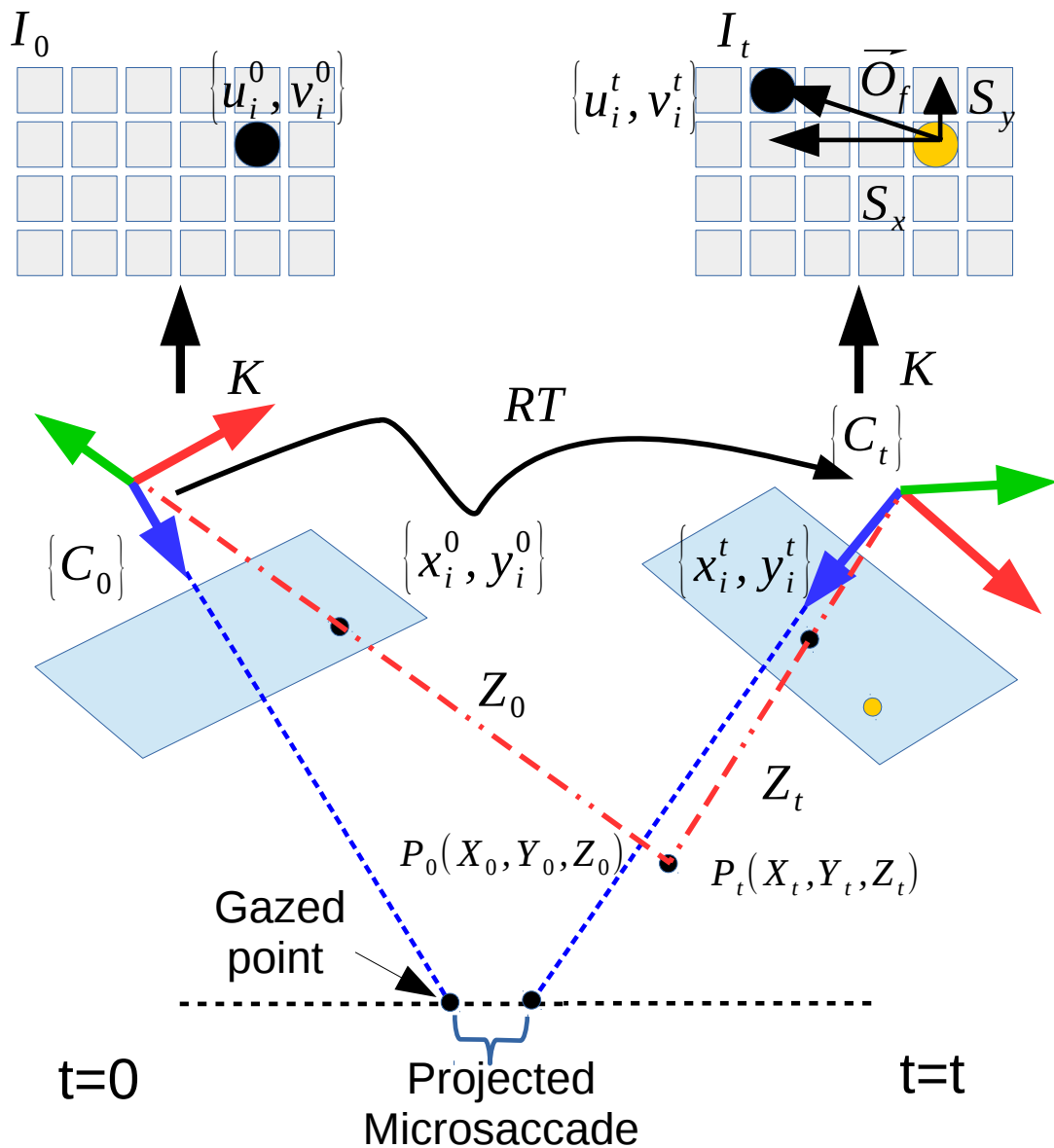


Figure 6.4: Schema of the considered camera movements. Initially, the camera is represented by the $\{C_0\}$ frame of reference. A point P_0 , with coordinates in $\{C_0\}$ given by $\{X_0, Y_0, Z_0\}$, projects onto the image plane with coordinates $\{x_i^0, y_i^0\}$ and pixel coordinates $\{u_i^0, v_i^0\}$, which are computed using the projection matrix K . A roto-translation (RT) of $\{C_0\}$ results in a new frame $\{C_t\}$ and the projection of that point changes to $\{x_i^t, y_i^t\}$ and $\{u_i^t, v_i^t\}$. Its apparent displacement on the image is given by $O_f = \{S_x, S_y\}$

point of the fixation process with the initial image of reference I_0 . Given a point of the scene (P_0) that generates an intensity value in that image and projecting it onto the image plane, the pixel $\{x_i^0, y_i^0\}$ is obtained. The Z-axis of the visual system is aligned with the gaze point, and Z_0 is the value of the depth in P_0 , understanding depth here as the distance from the camera frame of reference $\{C_0\}$ to the perpendicular plane to the

camera Z-axis containing P_0 .

After a head movement and a microsaccade ($t=t$), the gazed point has been displaced, and the new camera pose is aligned with this new gazed point. The depth concerning the new camera frame $\{C_t\}$ has changed. After the microsaccade, visual perception suppression disappears, and a new image I_t is obtained. The original P_0 is now P_t with respect to the new camera frame and its projection on the image plane corresponds to a new pixel position in the image $\{x_i^t, y_i^t\}$. An optical displacement has taken place in the image plane $O_f = \{Sx, Sy\}$. This value can be estimated by computing the optical flow between both images.

The aim is to determine the value of Z_0 that corresponds to depth sensation in the fixation point. In order to reach this goal, we define several matrices and vectors in homogeneous coordinates.

- The pixel coordinates in I_t and I_0 are defined by vectors $m_t = [u_t, v_t, 1, 1]^T$ and $m_0 = [u_0, v_0, 1, 1]^T$, where u and v are expressed in the centred image coordinates system.
- A projection matrix is defined: \mathbf{K} that is a function of the camera parameters, mainly of the focal lengths. To simplify the model $\mathbf{K} = \{k_{i,j}, \forall i, j \in \{1, \dots, 4\}, i \neq j \implies k_{i,j} = 0 \wedge \text{diag}(\mathbf{K}) = \{f, f, 1, 1\}$ where f is the focal length of the camera.
- To work in homogeneous coordinates, two matrices are defined depending on the depth value: $\mathbf{H}(Z) = \{h_{i,j}, \forall i, j \in \{1, \dots, 4\}, i \neq j \implies h_{i,j} = 0 \wedge \text{diag}(\mathbf{H}) = \{1/Z, 1/Z, 1/Z, 1\}$. Thus, there are two such matrices, one for the initial camera pose $\mathbf{H}(Z_0)$ and the other one for the other pose $\mathbf{H}(Z_t)$.
- Finally, regarding roto-translation matrix between the frames, we consider that the angular variation is small enough to approximate the rotation by using the skew matrix \mathbf{M} ; in addition, the translation matrix \mathbf{T} is given by the Cartesian difference between $\{C_0\}$ and $\{C_t\}$. These matrices are defined in (equation (6.1)).

$$\mathbf{M} = \left(\begin{bmatrix} 0 & -\Delta W_z & \Delta W_y \\ \Delta W_z & 0 & -\Delta W_x \\ -\Delta W_y & \Delta W_x & 0 \end{bmatrix} \right); \mathbf{T} = \left(\begin{bmatrix} \Delta X \\ \Delta Y \\ \Delta Z \end{bmatrix} \right) \quad (6.1)$$

Where $\Delta W_{(x,y,z)}$ is the angular variation in each axis. The roto-translation matrix \mathbf{RT} is defined as a composition in (6.2).

$$\mathbf{RT} = \left(\begin{bmatrix} 1 + \mathbf{M} & \mathbf{T} \\ 0 & 1 \end{bmatrix} \right) \quad (6.2)$$

If the ego-motion signal is known by means of T and M , the new pixel position in the image plane m_t can be computed by using expression (equation (6.3)) from the reference image pixel position.

$$m_t = \mathbf{H}(Z_t) \cdot \mathbf{K} \cdot \mathbf{RT} \cdot \mathbf{K}^{-1} \cdot \mathbf{H}(Z_0)^{-1} \cdot m_0 \quad (6.3)$$

The value of Z_t can be obtained from the expression: $P_t = \mathbf{RT} \cdot P_0$, and taking into account that $P_0 = \{u_0 \cdot Z_0/f, v_0 \cdot Z_0/f, Z_0\}$, the value of Z_t can be calculated with (equation (6.4)).

$$Z_t = Z_0 + \Delta Z - X_0 \Delta W_y + Y_0 \Delta W_x \quad (6.4)$$

From equations (6.3) and (6.4), it can be concluded that m_t is only a function of the camera parameters (f), the ego-motion components ($\{\Delta X, \Delta Y, \Delta Z, \Delta W_x, \Delta W_y, \Delta W_z\}$) and the initial depth Z_0 . When the scene is considered static, the apparent displacement produced in the image of pixel m_0 is only originated by ego-motion, therefore (equation (6.5)) must be satisfied.

$$m_0 = m_t - O_f \quad (6.5)$$

However, given that both the ego-motion and the optic flow (O_f) can have an error in their estimations, we can write:

$$m_0 = \hat{m}_t - \hat{O}_f + \epsilon \longrightarrow \epsilon = m_0 - \hat{m}_t + \hat{O}_f \quad (6.6)$$

Where ϵ represents the accumulative error resulting from computing m_t using (equation (6.3)) and also includes the optic flow estimation error. m_0 is known since it is the initial pixel position in the reference image, whereas m_t can be calculated from (equations (6.3) and (6.4)). ϵ is a vectorial magnitude, and thus, a cost function based on its module can be defined as (equation (6.7)).

$$L = \frac{1}{2} \|\epsilon\|^2 = \frac{1}{2} (\epsilon_u^2 + \epsilon_v^2) = \frac{1}{2} ((u_0 - \hat{u}_t + S_x)^2 + (v_0 - \hat{v}_t + S_y)^2) \quad (6.7)$$

If it is assumed that the value of Z_0 is not correct and the errors corresponding to optic flow components $\{S_x, S_y\}$ and ego-motion estimation are approximately constant; then, the greatest contribution to the value of ϵ is the undetermined knowledge about Z_0 . If Z_0 were the optimum value for the cost function defined in (equation (6.7)), it could be computed using (equation (6.8)).

$$Z_0^* = \arg \min_{Z_0^* \in [Z_n, Z_f]} L = \{Z_0^* \mid \forall \alpha \in [Z_n, Z_f] : L(\alpha) \geq L(Z_0^*)\} \quad (6.8)$$

Deriving (equation (6.7)) with respect to Z_0 , the equation (6.9) is obtained:

$$\frac{\partial L}{\partial Z_0} = -[m_0 - \hat{m}_t + \hat{O}_f]^T \cdot \frac{\partial \hat{m}_t}{\partial Z_0} \quad (6.9)$$

It is useful to define these expressions to implement (6.9):

$$\begin{aligned} f_u &= u_o/f; & f_v &= v_o/f \\ V_z &= (1 - \Delta W_y f_u + \Delta W_x f_v); & A_z &= \Delta Z + Z_0 V_z \\ V_y &= (\Delta W_x - f_v - \Delta W_z f_u); & A_y &= \Delta Y - Z_0 V_y \\ V_x &= (\Delta W_y + f_u - \Delta W_z f_v); & A_x &= \Delta X + Z_0 V_x \end{aligned} \quad (6.10)$$

Then, \hat{m}_t , m_0 and \hat{O}_f in (6.9) can be expressed as:

$$\begin{aligned} \hat{m}_t &= \left[f \frac{A_x}{A_z}, f \frac{A_y}{A_z}, 1, 1 \right]^T; \\ m_0 &= [u_0, v_0, 1, 1]^T; \\ \hat{O}_f &= [S_x, S_y, 0, 0]^T \end{aligned} \quad (6.11)$$

The derivative of \hat{m}_t with respect to Z_0 can be written as:

$$\frac{\partial \hat{m}_t}{\partial Z_0} = [M_x, M_y, 0, 0]^T \quad (6.12)$$

where

$$\begin{aligned} M_x &= \frac{fV_x}{A_z} - fV_z \frac{A_x}{A_z^2}; \\ M_y &= \frac{fV_y}{A_z} - fV_z \frac{A_y}{A_z^2} \end{aligned} \quad (6.13)$$

From the above equations, it can be concluded that the value of the derivative of the cost function depends only on: the initial point coordinates (u_0, v_0) , the variation of the camera pose $(\Delta X, \Delta Y, \Delta Z, \Delta W_x, \Delta W_y, \Delta W_z)$ and the measured optical flow (S_x, S_y) in the initial image pixel.

Returning to the example at the beginning (figure 6.2), the mathematical model indicates that it is possible to estimate an optical flow value in each pixel of the image by

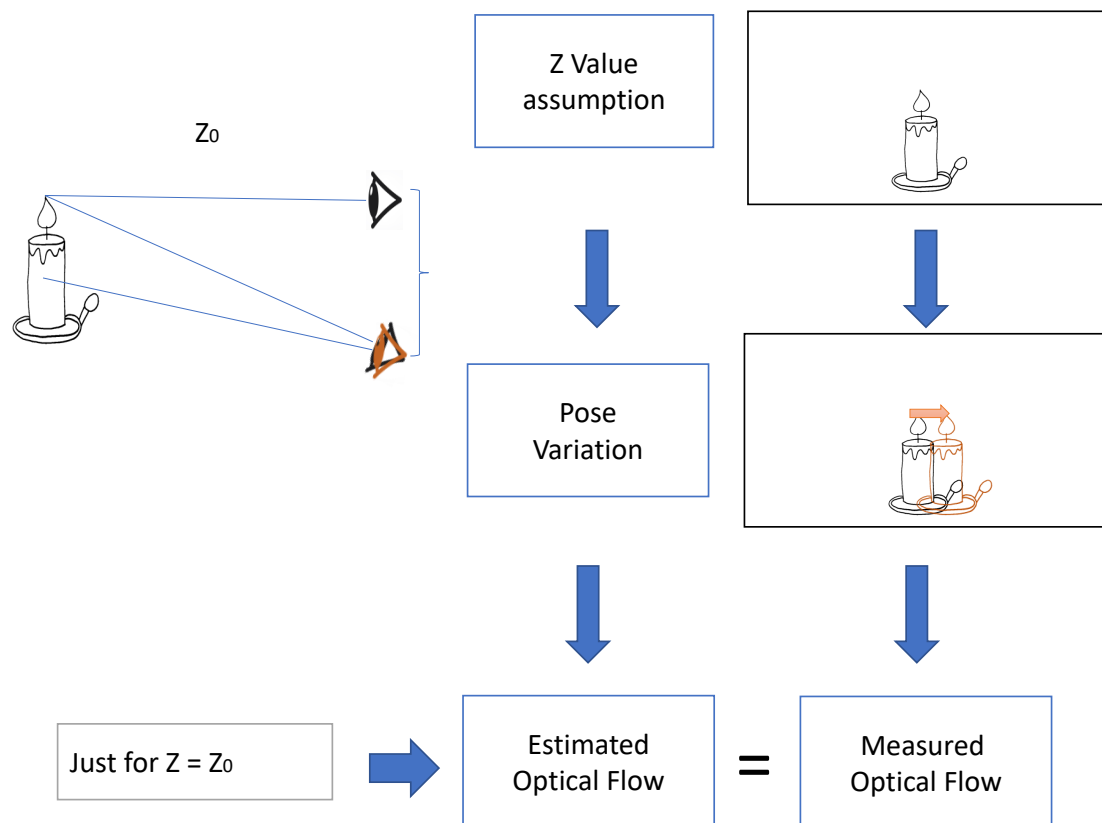


Figure 6.5: This scheme represents the idea behind the mathematical development. A depth is assumed to be correct in the case that the optical flow predicted by the position variation is equal to the optical flow measured from comparing the current image with the reference image to estimate the depth value .

varying the camera position, assuming a certain distance to the fixation point. In turn, it is possible to estimate a pixel-by-pixel optical flow employing the visual difference between the current image and the reference image. It must be fulfilled that the estimated optical flow at one pixel must be equal to the measured optical flow, assuming that both position variation and optical flow estimates are sufficiently accurate. In addition, only when the assumed distance is equal to the actual distance (figure 6.5).

Under these conditions, depth estimation has been converted into many independent optimization problems (one for each image pixel). This fact conditions the method of optimization to use:

- Even though straightforward stochastic gradient descent (SGD) could solve it; it would be necessary to define a different learning rate for each optimization problem since each pixel from the initial image is independent of the rest. Moreover,

probably this learning rate could depend on the real Z value corresponding to each pixel. Consequently, gradient-based methods that work at a constant learning rate are discarded. Instead, the learning ratio must be adapted in each iteration for each pixel.

- Another aspect to consider is the noise in the signals for the gradient calculation. Due to the estimation method, the optical flow has inherent variability, especially in areas with no texture. In addition, the position increase is estimated from self-perception data which may also present some noise.

A gradient descent method that can deal with these two issues to successfully compute Z_0^* , is the ADADELTA method (Zeiler, 2012). This algorithm is based on SGD, but it also introduces several filters in the estimation of the gradient and second derivatives. These filters can reduce the noise influence.

6.3.3 Depth estimation algorithm

Algorithm 6 Depth estimation

Require: $I_0, Z_n, \rho, \sigma, \mathbf{c}_0$

```

1:  $\mathbf{Z}^{(0)} \leftarrow Z_n, t \leftarrow 0, h, w \leftarrow \text{size}(\mathbf{I}_0)$ ;
2:  $\mathbf{G}_0 \leftarrow \text{zeros}(h, w); \Delta\mathbf{G}_0 \leftarrow \text{zeros}(h, w)$ 
3: loop
4:    $\mathbf{c} \leftarrow \text{HeadEyeMovement}()$ 
5:    $\mathbf{I}_t \leftarrow \text{getNewImage}()$ 
6:    $\mathbf{OF} \leftarrow \text{OpticFlow}(\mathbf{I}_0, \mathbf{I}_t)$ 
7:    $\mathbf{M}, \mathbf{T} \leftarrow \text{getEgomotion}(\mathbf{c}_0, \mathbf{c})$ 
8:   for  $i = 1$  to  $h$  do
9:     for  $j = 1$  to  $w$  do
10:       $v \leftarrow i - h/2; u \leftarrow j - w/2$ 
11:       $g_t(u, v, \mathbf{S}, \mathbf{T}, \mathbf{OF}(i, j)) \leftarrow \frac{\partial L}{\partial Z_0}$  {equation (6.9)}
12:       $\mathbf{G}_t(i, j) \leftarrow \rho\mathbf{G}(i, j)_{t-1} + (1 - \rho)g_t^2$ 
13:       $\tau_t \leftarrow \sqrt{\Delta\mathbf{G}(i, j)_{t-1} + \sigma} / \sqrt{\mathbf{G}(i, j)_t + \sigma}$ 
14:       $\Delta\mathbf{G}(i, j)_t = \rho\Delta\mathbf{G}(i, j)_{t-1} + (1 - \rho)\tau_t^2$ 
15:       $\mathbf{Z}^{(t)}(i, j) = \mathbf{Z}^{(t-1)}(i, j) + \Delta\mathbf{G}_t(i, j)$ 
16:       $t \leftarrow t + 1$ 
17:     end for
18:   end for
19: end loop

```

The above mathematical formulations inspired by the fixation process are implemented by algorithm 6. The starting point is the reference image (I_0), and camera pose (c_0) captured at the time of the initial fixation process. Initially, no depth information is available; therefore, all pixels in the image are assigned the same value Z_n . When the fixation process has begun, the movements of the head and the oculomotor system generate displacements in the image (I_t) and the camera pose (c_t). That is, the microsaccades used to carry out the sampling. The initial image I_0 is correlated with each newly obtained image I_t using the Lucas-Kanade method (Lucas and Kanade, 1981). The goal of the Lucas-Kanade algorithm is to minimize the sum of the squared errors between two images, the reference image I_0 and the current image I_t .

From here, the depth estimation algorithm iterates for each image pixel, updating the gradient descent computation with the ADADELTA equations.

As the algorithm advances, the received information increases the sense of depth in the image that corresponds to the initial fixation point. Ultimately, this increase in information is represented in the algorithm by the term $\Delta G_t(i, j)$, which in turn depends on the cost function according to equation (6.9). Thus, if there is no optical shift between the current image and the reference image ($I = I_0$), there is no improvement in depth estimation knowledge.

From a computational complexity point of view, each pixel is visited once in each iteration, as shown in algorithm 6. Moreover, the computations made on each pixel only depend on the state of that pixel in the previous step, the optical flux estimated on this point and the camera displacement. Therefore, the temporary asymptotic cost in this part of the algorithm is $\mathcal{O}(\mathcal{N})$, where \mathcal{N} is the total number of pixels in the image. Regarding the asymptotic spatial cost, the complete algorithm must store the resulting depth image, the optical flux components in each iteration, the initial image, and the current image. Therefore, the spatial cost has a magnitude of $\Theta(5\mathcal{N})$. This algorithm is amenable to parallel computing since each pixel is independent of the previous and current states of the rest of the pixels. This trait allows it to be implemented using parallel computing techniques on either GPUs or CPUs.

6.3.4 Algorithm parameters

As it can be seen in the description of algorithm 6, it is necessary to set several parameters for its proper execution: First, Z_0 is the initial distance for all image pixels; second, ρ acts as a low pass filter coefficient for the gradient adaptation and its derivative. Finally, σ regulates the gain of the gradient variation in each step. Since gradient descent techniques do not differentiate between local and global minima, selecting

these parameters is relevant for good quality results.

In addition, if the span of the work area is known, the search limits can be defined a priori; if the sought minimum lies outside these limits, the algorithm will not converge. Also, the noise factor affects its performance because the values it generates may be outside these limits. In such cases, it is necessary to define an action policy for the pixels in which this phenomenon occurs.

6.4 Experiments in simulation

6.4.1 Simulation setup

Evaluation tests are carried out with the Baxter robot in the Gazebo/ROS simulator. Given the degrees of freedom of Baxter's head, it is impossible to replicate the movements of the primate's oculomotor system. Instead, we use the 7-DOF arm of this robot with an eye-in-hand camera.

Although some robotic systems described in the literature could perform this task correctly (Santini et al., 2009; Kuang et al., 2012), the design of the experiments based on this specific platform was developed in the context of the RoboPicker (Del Pobil et al., 2017) project for which a low-cost robot is called for, and manipulation takes place in a confined space. Therefore, the primary function of the arm in our experiments is to move the camera in such a way that it maintains orientation, and it positions itself in the same way that a human eye would perform fixational movements.

Baxter's wrist camera can be configured in several ways. Of all the possible ways, a resolution of 900x600 pixels and a focal length of 405.7 was chosen. The camera simulation was set up with the same parameters. In addition, white Gaussian noise was applied to the image to introduce uncertainty in the optical flow computation. The standard deviation of this gaussian noise is common to all performed experiments and is equal to 0.01 pixels.

The space of movements for the camera is specified as a sphere defined by two parameters: the central point and the radius of movements r_m (see figure 6.3). r_m is considered constant to reduce the number of experiments, with a value of 0.015 m according to the order of magnitude in the experiments of (Aytekin and , 2012). These authors suggest r_m is not uniform, and its value depends on the distance to the fixation point). A controversial point in the literature is the maximum radius of a microsaccade. Some studies set this value between 1° and 2° . However, most microsaccades have a magnitude smaller than 0.5° for many tasks (Rolf, 2009). The parameter r_g is defined

by the microsaccade amplitude as shown in figure 6.3. r_g varies with the fixation point distance as shown equation (6.14) taking an amplitude of 0.5° .

$$r_g \approx 0.0088 \cdot d \quad (6.14)$$

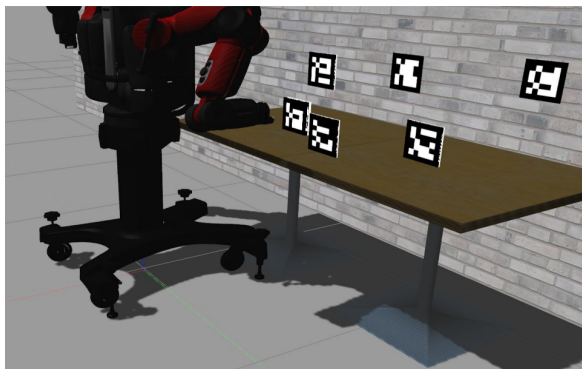
6.4.2 Experimental procedure

Based on the fixation process, the next procedure is used in all experiments:

- An artificial scenario is placed in front of the wrist camera of the simulated Baxter robot (figure 6.6a) and a starting point of the camera for the fixation process is selected.
- In order to simplify, three distances are selected for the fixation point in the scenario, all of them on the same axis Z from the camera. In addition, the microsaccade radius r_g is computed as a function of that distance (equation (6.14)).
- The initial image I_0 (figure 6.6c) and pose C_0 are saved.
- The depth image of the scene is captured with a simulation of an RGBD camera placed in precisely the same initial position and with identical resolution as the RGB camera figure 6.6b.
- The camera starts to move randomly within a sphere of radius r_m , maintaining the fixation point projected onto the image plane within the circle of radius r_g . These displacements are reflected in the generation of successive images (I_t) differing slightly from the reference image (figure 6.6d).
- The successive images (I_t) and poses (C_t) are compared with the initial image and pose by applying algorithm 6. As can be seen in the description of the algorithm, it is necessary to calculate the optical flow between the reference image and the current image (figure 6.7a). The algorithm generates depth estimations for each pixel in each iteration (figure 6.7b). Finally, it converges to a resulting depth image (figure 6.7c) corresponding to the best estimation in all the regions of the proposed scenario (figure 6.7d).

6.4.3 Evaluation methods

Two kinds of scenarios are tested to evaluate the performance of algorithm 6. The first scenario is used to study the accuracy of depth estimation and the influence of the



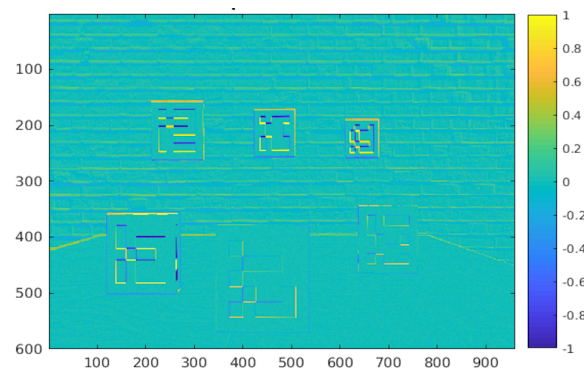
(a) Baxter robot setup scenario on starting point of the camera for the fixation process



(b) Depth image used as background directly captured from a depth camera sensor simulation in the same pose that RGB camera

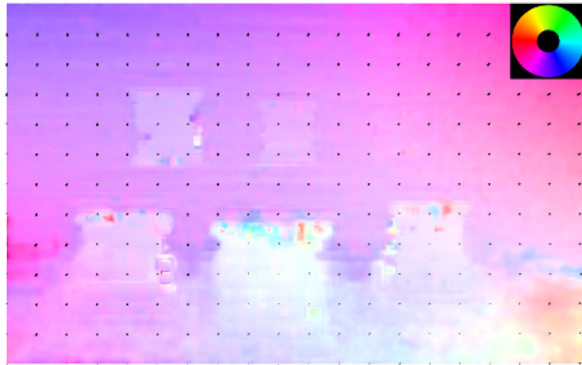


(c) RGB initial image captured as reference (I_0)

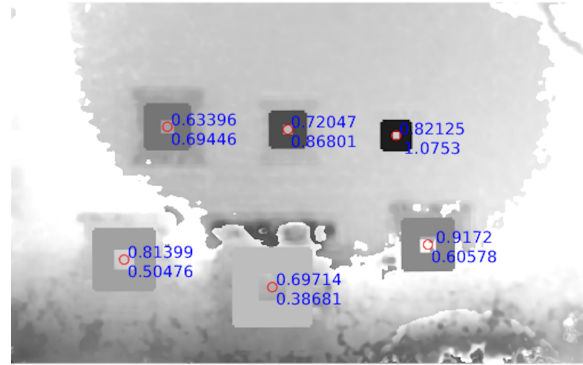


(d) Pixel by pixel intensity difference between reference image (I_0) and another captured after micro head random and microsaccadic movements (I_t)

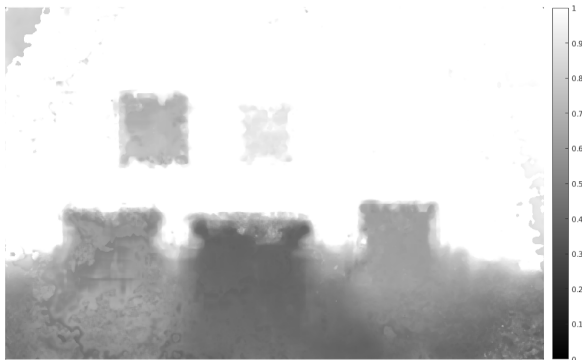
Figure 6.6: Different images captured during the execution of the proposed algorithm.



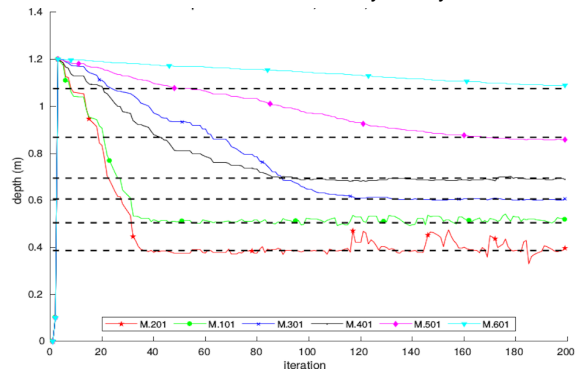
(a) Estimation of the optical flow between the reference image (I_0) and the current image (I_t). The colours indicate the displacement vector angle between the two images.



(b) Example of intermediate estimation of the proposed algorithm. In this image the different depth zones begin to differentiate but some of them are already clearly visible.



(c) Example of a final depth estimation obtained after running the algorithm



(d) Example of a representation showing the progress of the algorithm in reference to the average distance of each of the objects in the initial image.

Figure 6.7: Different partial results were obtained during the execution of the algorithm to visualize its evolution.

algorithm parameters.

To quantitatively estimate the performance of the algorithm, two types of backgrounds are used. The first one uses the depth image generated by the simulator (figure 6.6b) as the most accurate depth estimation ground truth. The second background utilises six squared plates placed in the simulated scenario on which 6 Aruco Markers are printed [et al. \(2014\)](#). The type of markers and their relative position with respect to the initial camera location are shown in table 6.1 and figure 6.6c. The error is estimated from the standard deviation after 30 measures for each marker position using the Aruco markers detector algorithm. These error values provide information about the repeatability of the measurement, not its accuracy concerning the background. Using the Aruco markers, the depth of each marker plane can be estimated. In addition, each marker encloses an image area where the depth should be approximately the same. Therefore, applying this mask to the obtained depth image from the algorithm and computing the mean and standard deviation for each marker area, the result must be comparable to the distance estimated by the Aruco detector.







Marker id=101 	Marker id=201 	Marker id=301 
$(0.509 \pm 0.006)m$	$(0.386 \pm 0.005)m$	$(0.605 \pm 0.004)m$
Marker id=401 	Marker id=501 	Marker id=601 
$(0.694 \pm 0.003)m$	$(0.866 \pm 0.002)m$	$(1.071 \pm 0.009)m$

Table 6.1: Aruco markers used and estimations of the depth from the camera with the Aruco detector.

The second evaluation scenario comprises several simulated objects with different shapes and textures in the same setup. Then, the obtained depth image is compared in each iteration with the real one using the mean square error between them. Several Aruco markers are also introduced in this scenario to be used as control points.

6.4.4 Experimental tests

The primary objective of the experimental tests is the evaluation of the proposed algorithm. In addition, secondary goals are intended:

- i) Study the influence of the choice of parameters on the performance and results of the adaptive process.

- ii) Evaluate the effect of a plausible Gaussian error in the inputs of the algorithm.
- iii) Validate the algorithm in an environment with ordinary objects.

To avoid shifts in the image due to changes in perspective and to keep the set of control markers within the scene in all images, three virtual fixation points were selected at different distances from the initial position of the camera, which is the same for all experiments $d = \{0.3, 0.6, 0.9\}(m)$. In addition, any interference produced by choice of fixation points within the environment is tried to be avoided. Furthermore, it can be assured that all Aruco markers will appear in almost all images, and therefore it is possible to track and compare with them in each iteration.

Considering that the final objective is to obtain a depth estimation as similar as possible to the image generated by the simulation of the depth camera, two of the criteria used to evaluate the results are the structural similarity index (SSIM) Zhou Wang et al. (2004) and the global mean square error (MSE), along with the standard deviation between the depth images in each iteration.

In addition, to check whether differences exist in the algorithm's performance depending on the depth, the comparison between the estimation of the distance in the planes defined by the aruco markers and the one estimated by the algorithm in each iteration is used. Moreover, the exact position of each plane corresponding to each marker is known. Thus, this value can be compared to the algorithm's results for this region of the environment.

The markers are a redundant way of confirming the algorithm's performance; perhaps in simulation, it does not make much sense to use them, but in a real robot where it is necessary to have physical depth references to evaluate the execution. Therefore, the results will be comparable in a real robot if these markers are used as a linking factor.

Finally, a policy regarding the procedure is defined when the estimated value of the distance lies outside the defined limits of the work area. For example, this can occur when there is an error in the optical flow estimation or the position variation. One option was to reset its value to the initial distance or decide not to update the value of Z_t^* . After several tentative tests, this second policy was implemented.

6.4.4.1 Influence of the choice of parameters

It can be observed from the adaptive part of the proposed algorithm that ρ acts as the smoothing coefficient in an exponential mean filter, both for the gradient square

and $\Delta G(i, j)_t$ adaptation. Therefore, the possible noise must modulate the choice of ρ that the estimation of the gradient and its derivative may present. Furthermore, it can be assumed that this noise has a similar effect on all depth image pixels; therefore, the value of ρ is taken as the same for all of them. The σ parameter ((Zeiler, 2012)) has a regularisation function to prevent a zero value for the denominator of the τ_t estimate. Its importance changes depending on the relative value of the estimation of the gradient square concerning the σ value. The rest of the system variables are defined to study the influence of both parameters:

- The fixation point is placed at 0.6 m;
- 0.1 m is assigned as the initial value of Z for all pixels in the depth image;
- The displacements of the camera and RGB images are the same for all variations of the studied parameters.

Under these conditions, the value of ρ is fixed and varied σ and vice versa. The considered parameter values are $\sigma = \{0.001, 0.005, 0.01, 0.05\}$ and $\rho = \{0.4, 0.5, 0.7, 0.9, 0.99\}$. Examples of the obtained results for these tests are shown in figures 6.8 to 6.11

In examining these figures, several observations can be made:

- Comparing all figures, it can be seen that the algorithm converges to a value of both MSE with an approximately constant standard deviation. Using the SSIM as a reference also this convergence is produced.
- Comparing figures 6.8 and 6.10, it can be observed that when the value of σ is constant for any value of ρ , the final convergence point is very similar for all cases (figure 6.10). However, there is a small variation in variable σ and constant ρ . Therefore this result suggests that the value of σ affects the final value of MSE and variance obtained with the algorithm. This result is confirmed by the evolution plot of the SSIM index where the most stable convergence between all algorithm runs occurs when sigma is constant.
- Considering the moment when the algorithm reaches the equilibrium point, comparing figures 6.8 and 6.10 it is possible intuitively to understand how the value of ρ affects this point, while in figure 6.8 when ρ is constant this point is more or less the same in all cases, and when ρ is varied and σ is kept constant this point changes.
- From the observation of these plots, it can also be seen that the algorithm's behaviour is similar in a range of σ and ρ . Beyond this range, the algorithm does

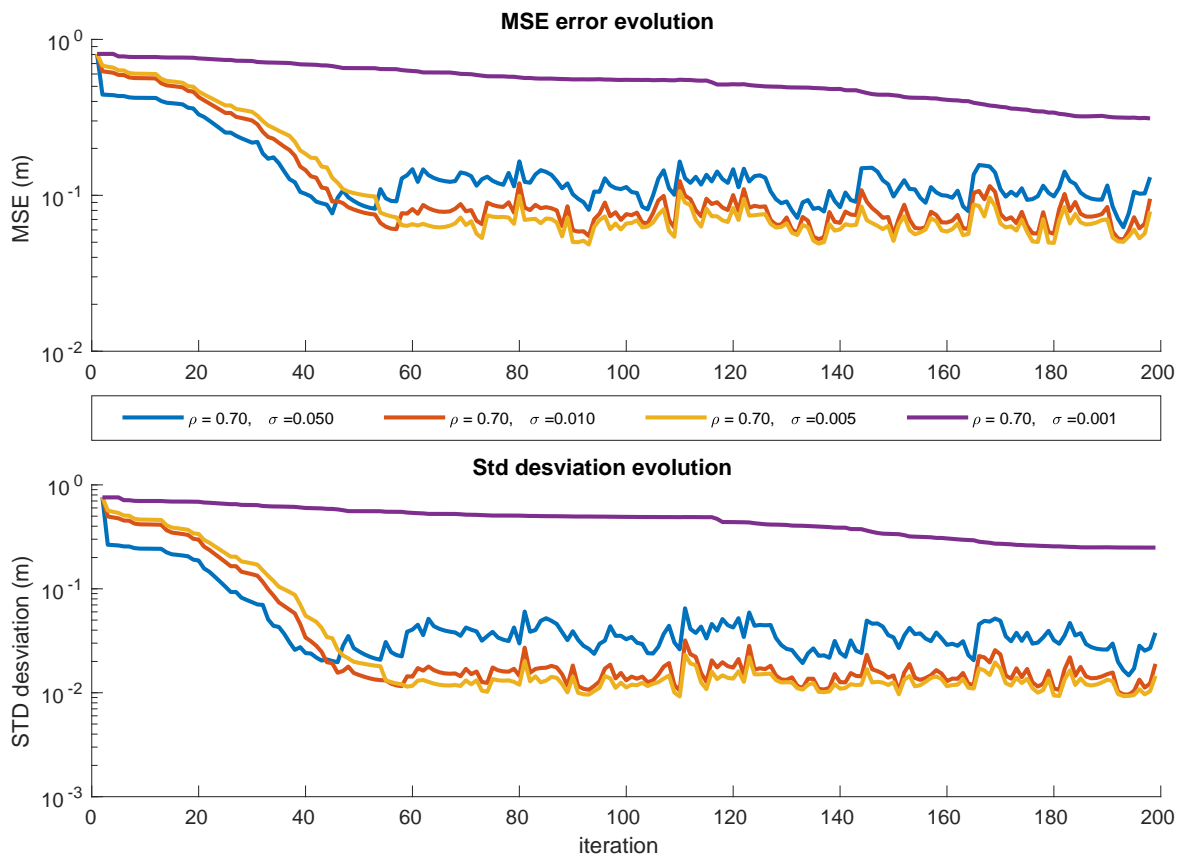


Figure 6.8: MSE and standard deviation between the background depth image and the estimation made by the proposed algorithm for 60 cm fixation point and different values of σ for a constant value of $\rho = 0.70$

not give stable results. For instance, in the case of figure 6.8, it can be seen that for minimal σ values, the result does not converge in the same way as the rest. The same behaviour occurs in figure 6.10, where for very high ρ values, the convergence point is significantly delayed.

As a conclusion, in figure 6.10, the mean of the last 50 iterations is $0.0169 \pm 0.0787\text{m}$. Also, these results suggest a behaviour for the influence of ρ for a constant σ , in the sense that the lower ρ is, the faster the algorithm converges (around 30 iterations for $\rho = 0.4$). In principle it seems that the lower σ is, the better the obtained results are (figure 6.8). This trend, however, has a limit and for a very low σ , the results are poor.

As expected, ρ parameter acts as a filter causing the stabilisation of the final results in exchange for the number of iterations to reach them. On the other hand, the behaviour of σ is more complex. As it can be seen in figure 6.8, for a given ρ the lower the value of σ the better the overall result. However, if σ becomes too small, the algorithm gets *frozen* (purple line in figure 6.8). This is also the case for too high values

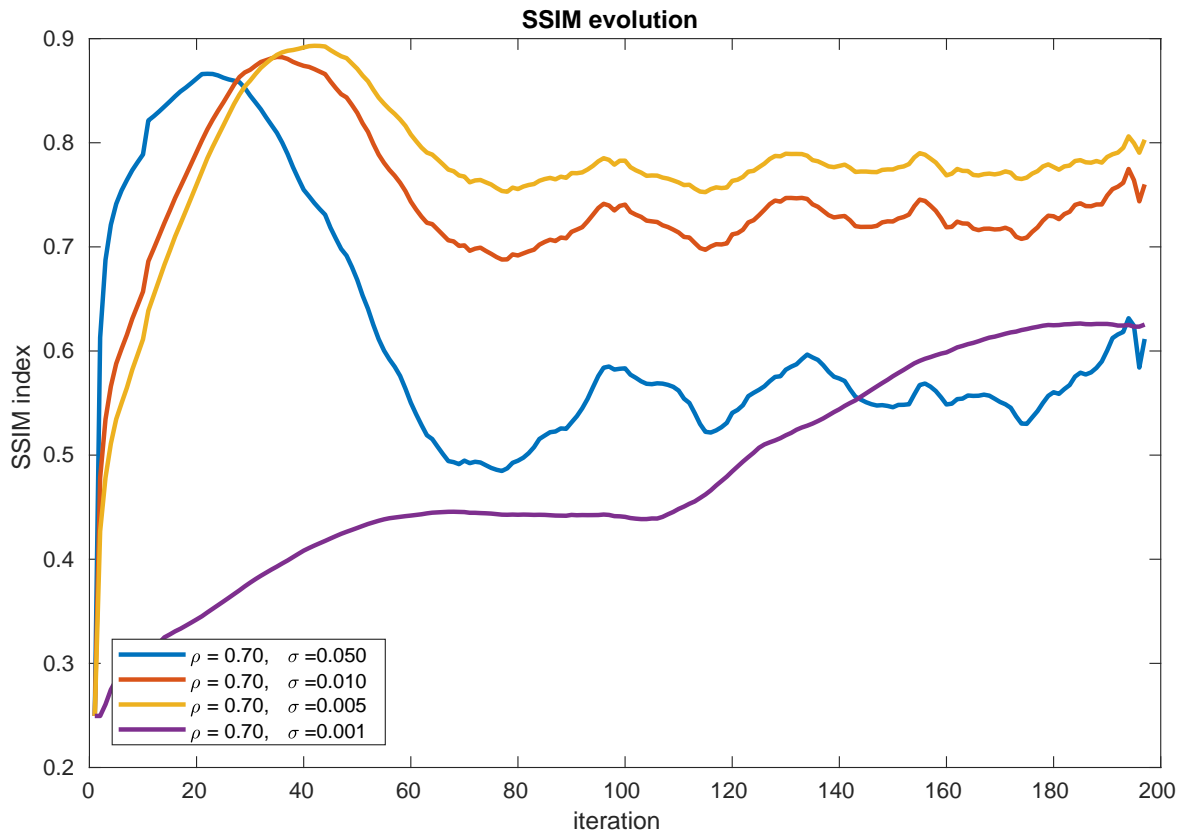


Figure 6.9: SSIM index estimation between the background depth image and the estimation made by the proposed algorithm for 60 cm fixation point and different values of σ for a constant value of $\rho = 0.70$

of ρ (green line in figure 6.10). In any case, the choice of σ and ρ should be made jointly since the closer ρ is to 1 –and, therefore, filters more– the higher the value of σ should be.

In addition, the influence of the initial value of Z_n on the algorithm results is studied. Thus, the rest of parameters are fixed and the value of Z_n is varied. The obtained results are shown in figure 6.12. It is apparent that the convergence to the final result seems faster the higher the value of Z_n .

Figures 6.12 and 6.13 suggest the initial value Z_0 only conditions the moment of reaching a more or less stable result, but it does not seem to affect the final depth image.

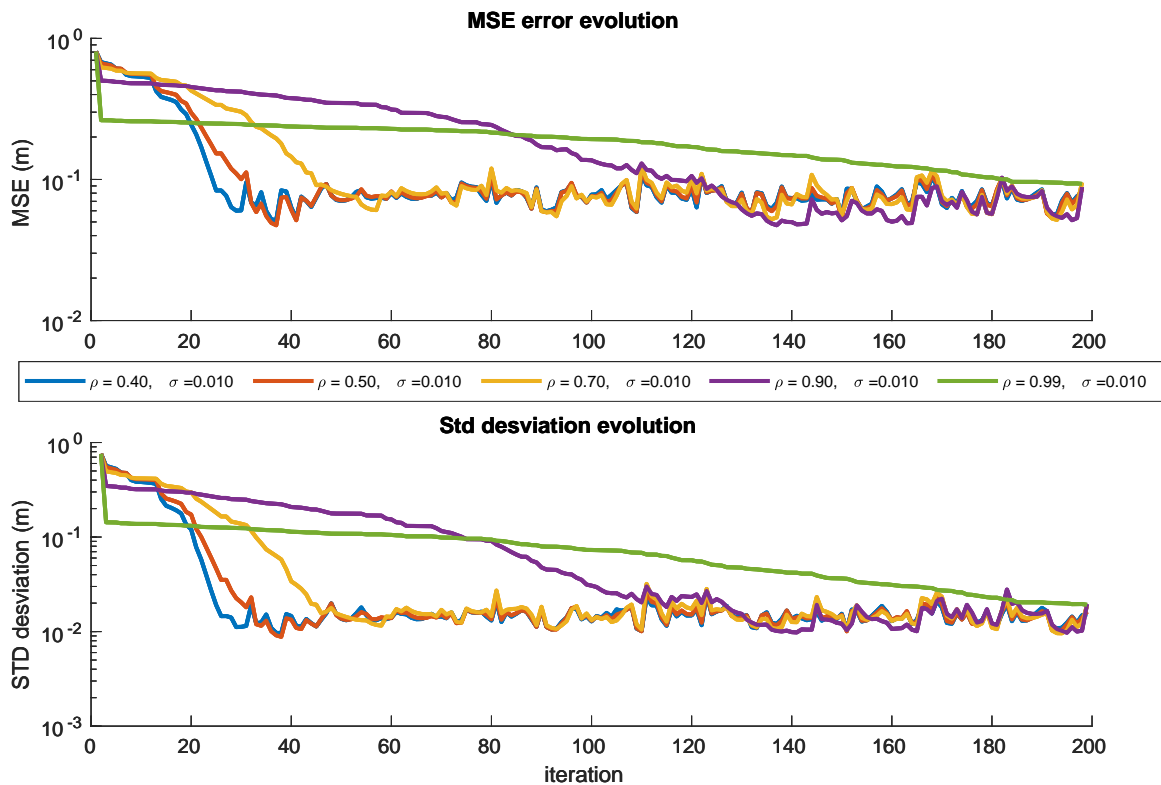


Figure 6.10: MSE and standard deviation between the background depth image and the estimation made by the proposed algorithm for 60 cm fixation point and different values of ρ for a constant value of $\sigma = 0.01$

6.4.4.2 Additive error influence

The gradient descent algorithm takes advantage of parameters ρ and σ to filter the noise in the input signals. However, the estimation of the gradient and its derivative is severely affected by this noise. Therefore, a gaussian error in the image is introduced in the experiments that directly affects the precision in obtaining the optical flow to assess this issue's impact. Furthermore, this error acts on each pixel individually. On the contrary, estimating the displacement error of the camera affects the calculation of depth in all pixels.

From this point of view, the same experimental conditions is used, that is,

- Radius of movements ($r_m = 0.015\text{m}$)
- Fixation point at distance 0.6 m
- Same captured RGB images and camera displacements. However, in each itera-

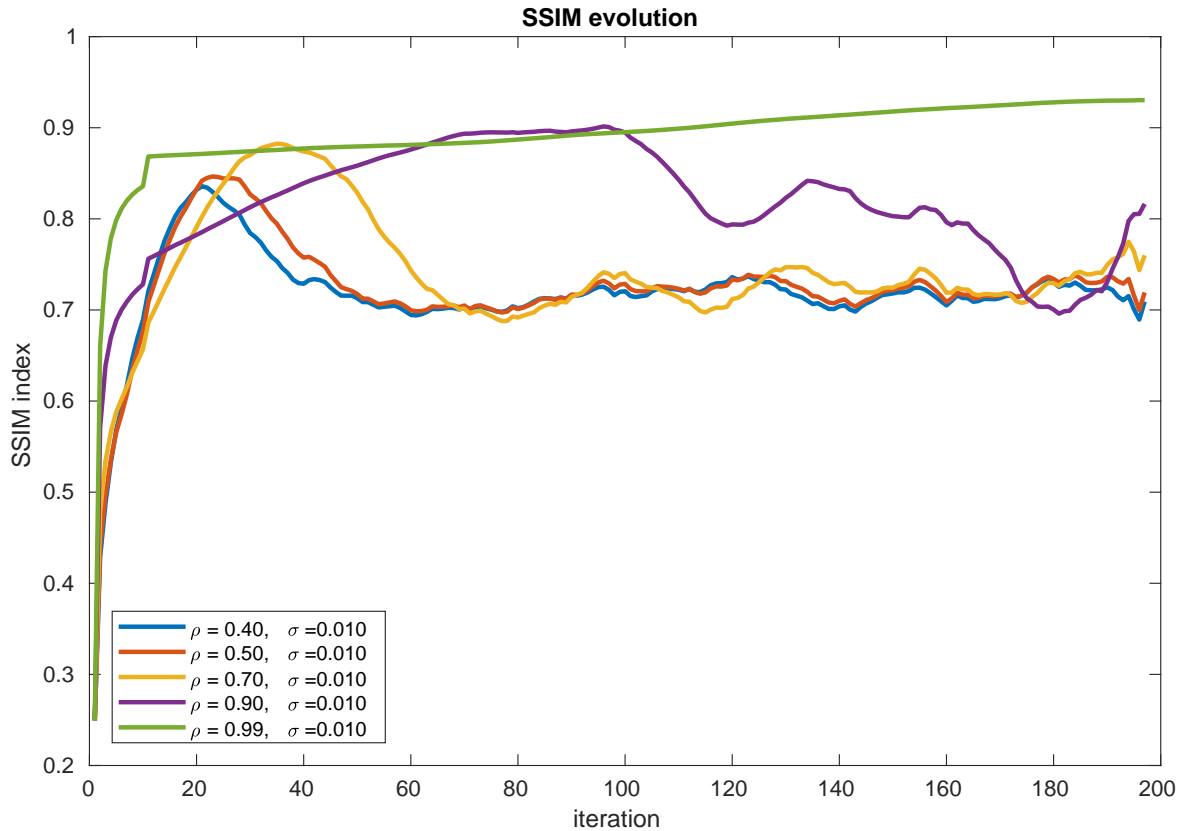


Figure 6.11: SSIM index estimation between the background depth image and the estimation made by the proposed algorithm for 60 cm fixation point and different values of ρ for a constant value of $\sigma = 0.010$

tion we disturb the camera displacement computations with white gaussian noise affecting its rotational and translational components, and characterised by standard deviations ϕ_r and ϕ_t . The chosen values for ϕ_t are $\phi_t = \{0.0001; 0.00050; 0.0010\}$ m that represent $\{1.2\%, 6.6\%, 13.2\%\}$ of the maximum possible displacement respectively.

- The selected values for ϕ_r are $\phi_r = \{0.005^\circ, 0.1^\circ, 0.3^\circ\}$.

After the execution of the algorithm, the obtained results are shown in figure 6.14. In addition, grey-scale representations of the final depth images for the best and worst cases are shown in figure 6.15.

Adding a Gaussian error to the estimation of the camera position affects each pixel of the final depth image equally; the application of the algorithm is pushed to the limit. Notwithstanding, in this case, the effects of σ and ρ become more apparent. These experiments also establish the error limits when applying the algorithm to a real robot.

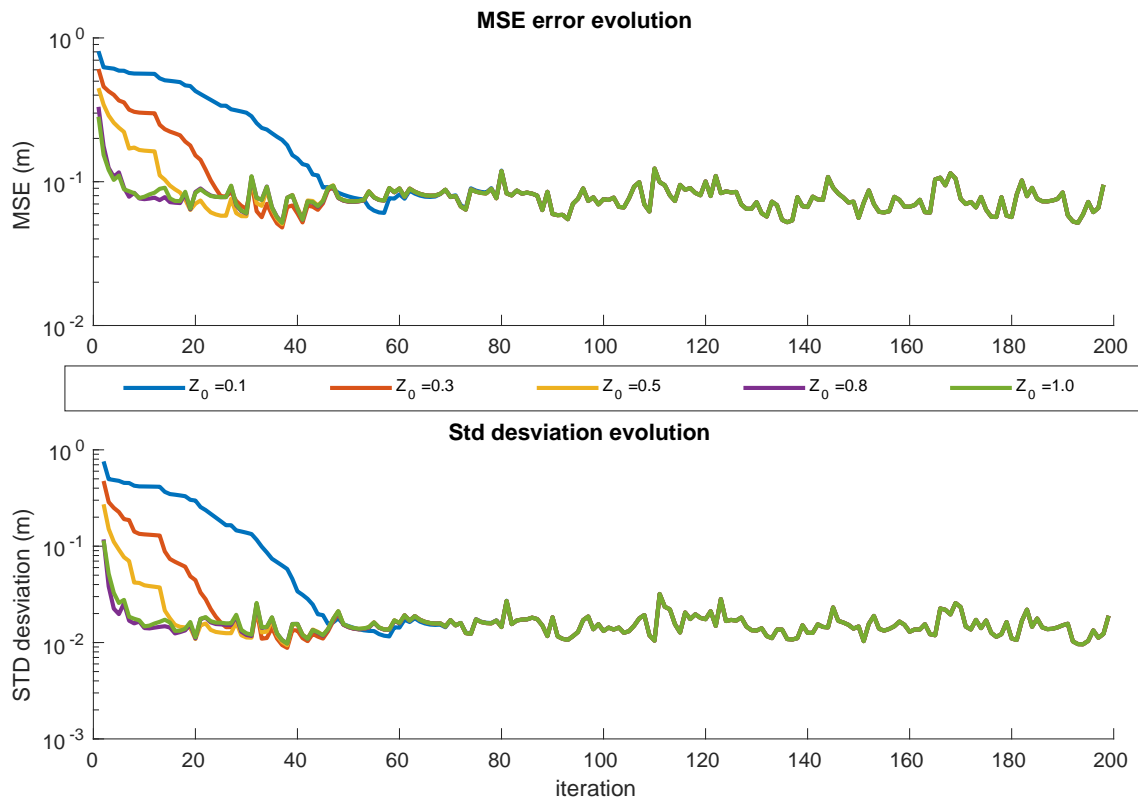


Figure 6.12: MSE and standard deviation between the background depth image and the estimation made by the proposed algorithm for 60 cm fixing point and different values of Z_0 for constant values of $\rho = 0.7$ and $\sigma = 0.01$

The obtained results raise some points for discussion.

- i) The lower σ is, it seems that the more robust the performance is in all cases.
- ii) Increasing ρ tends to stabilise the algorithm results in some cases, depending on the value of σ , and with a limit: as in figure 6.10 the algorithm hardly progresses, for a value of $\rho = 0.99$.
- iii) The uncertainty when the added noise error is too high generates non-valid results. As it can be seen in figure 6.15 qualitatively, the added error has a manifest influence on the quality of the results.

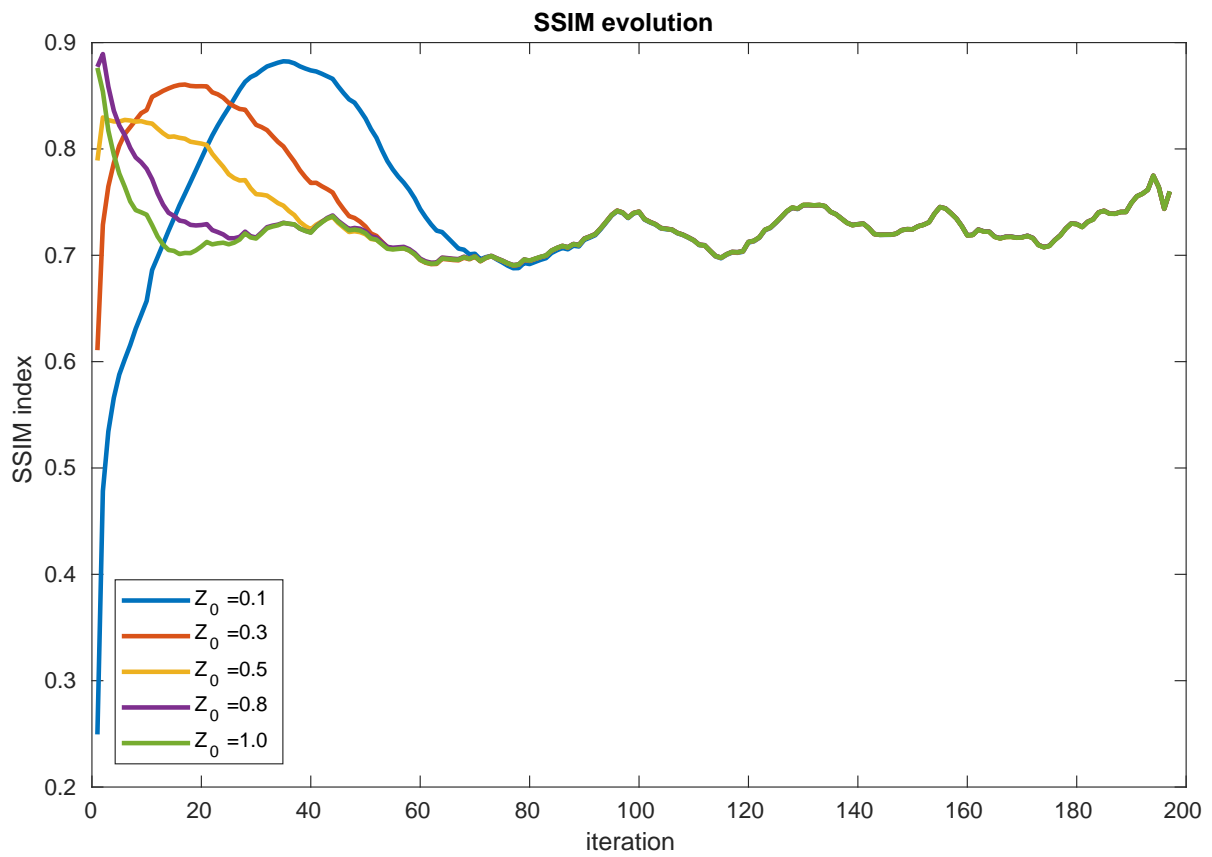


Figure 6.13: SSIM index estimation between the background depth image and the estimation made by the proposed algorithm for 60 cm fixing point and different values of Z_0 for constant values of $\rho = 0.7$ and $\sigma = 0.01$

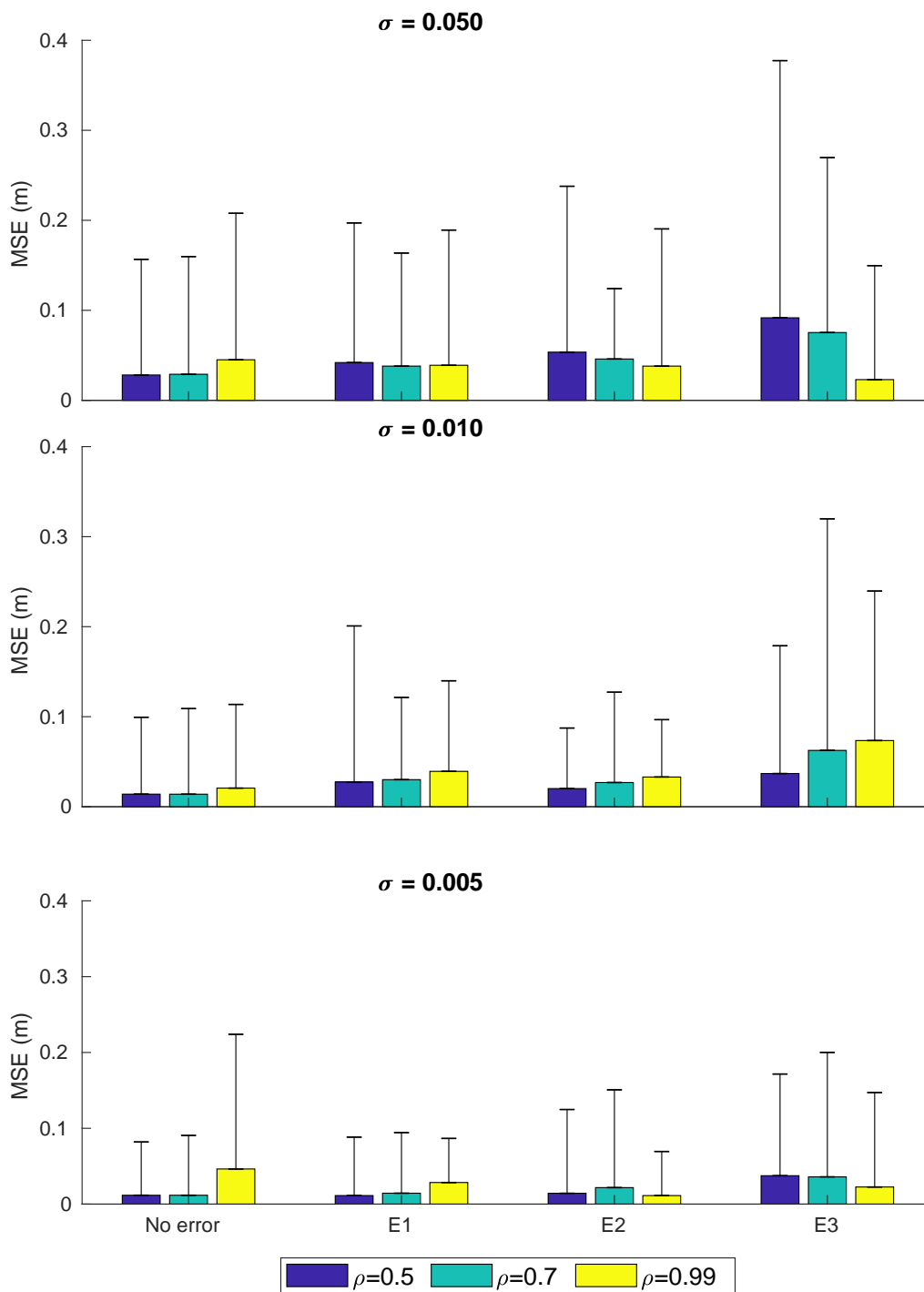


Figure 6.14: MSE with error bars representing standard deviation of the last 20 iterations. The bars are grouped by the added white noise. $E_i = \{\phi_t(i), \phi_r(i)\}$.

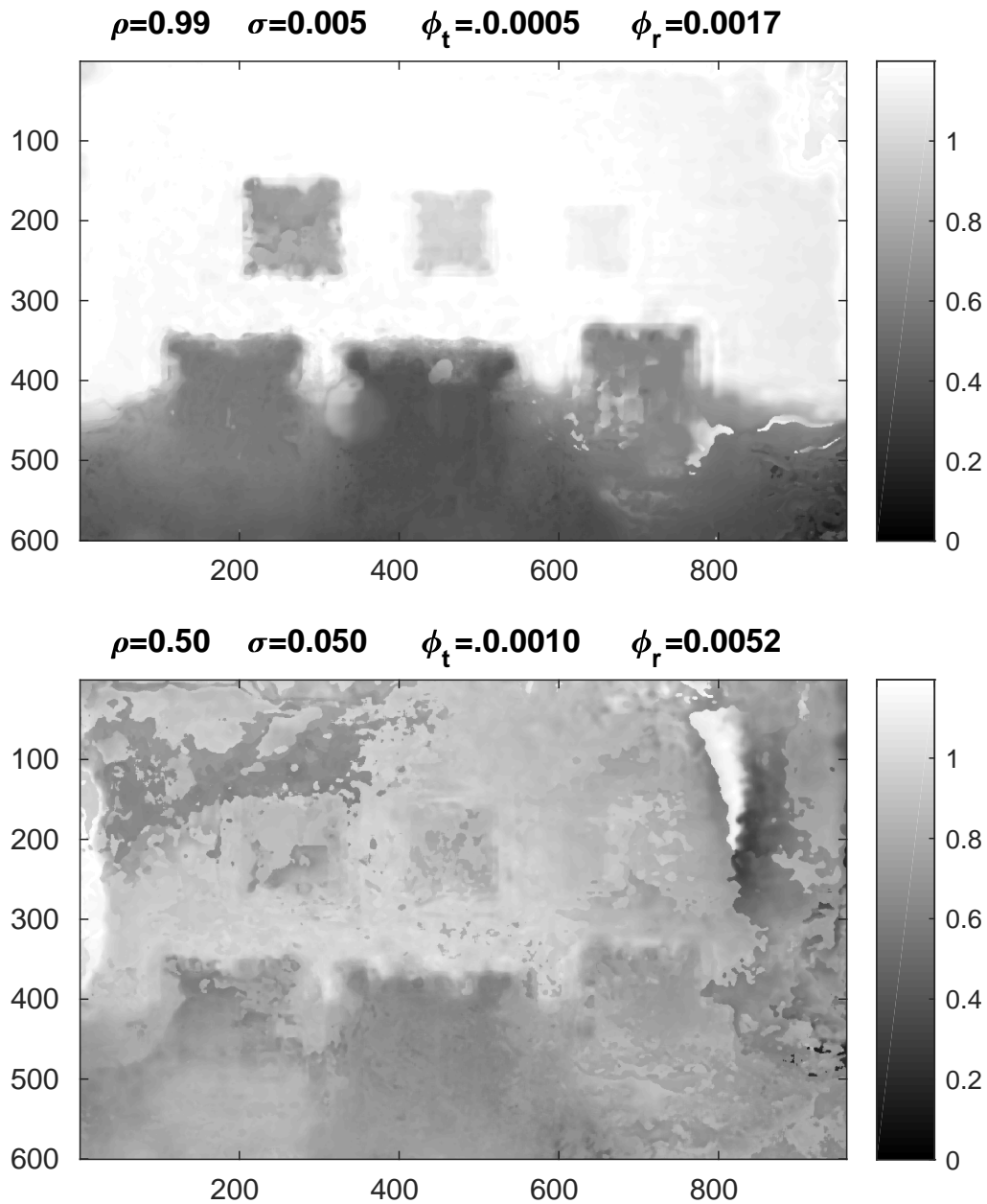


Figure 6.15: Image depth, showing the distance for each pixel scaled in 0-255 range. The upper image corresponds to the best obtained **MSE** and on the bottom the worst **MSE** for all experiments where white noise error was added to the camera displacement. ϕ_r is expressed in radians and ϕ_t in meters

6.4.4.3 Aruco marker comparison

The purpose of using Aruco markers in the simulation is twofold. First, create surfaces where the distance to the camera is known, and second, build a scenario that is easy to test when moving from simulation to reality.

It is straightforward to compare the results in the simulation since the distances from the markers to the camera frame are perfectly known. It is also possible to check the predictions that the Aruco algorithm makes for these known distances. In order to test these errors, the proposed depth estimation algorithm is applied in the same scenario but only varying the distance of the fixation point and keeping the rest of the parameters constant for all the tests.

In table 6.2, the obtained results are shown. The first column lists the relative errors between the estimation of the algorithm and the distance predicted from Aruco markers. The second column compares the relative errors between the estimation of the algorithm and the distance given by the simulator.

Representing the difference between the estimated average distance in the area defined in the image by each Aruco marker and the estimation of the marker position using the Aruco algorithm, the evolution of the proposed algorithm can be obtained considering the comparison with the position of the markers. These graphs allow to study the evolution of the algorithm in a real scenario. An example of this type of plot can be seen figure 6.16

As suggested in (López-Cerón and Canas, 2016), the accuracy of the Aruco Markers decays with distance. Table 6.2 shows that the greater the distance from a given marker, the results generated by the algorithm are closer to the simulator ground truth than to the values provided by the markers. This result suggests that the proposed algorithm is less sensitive to error variation with distance than the Aruco markers for this particular case.

6.4.4.4 Environment with ordinary objects

So far, only a simplified scenario has been considered, which has made it possible to evaluate the accuracy of depth estimation and the influence of the algorithm parameters and noise. All the surfaces involved were planes perpendicular to the camera. Models of several objects have been chosen (Rasouli and Tsotsos, 2017) in order to test the effectiveness of the algorithm in other more complex environments. They are arranged in front of the camera in the same way as the markers.

Table 6.2: Relative errors between depth estimated by the algorithm and the distance predicted from Aruco markers, and between the estimation of the algorithm and the distance given by the simulator, for the six Aruco markers and three fixation distances.

		d=30 cm	
		Aruco difference	Simulator difference
M.201		(0.57 ± 0.60) %	(0.83 ± 0.68) %
M.101		(0.57 ± 0.42) %	(1.64 ± 0.42) %
M.301		(0.35 ± 0.07) %	(0.33 ± 0.07) %
M.401		(0.97 ± 0.10) %	(0.12 ± 0.09) %
M.501		(0.61 ± 0.07) %	(1.04 ± 0.07) %
M.601		(2.79 ± 0.07) %	(1.41 ± 0.08) %
		d=60 cm	
		Aruco difference	Simulator difference
M.201		(2.30 ± 0.24) %	(0.52 ± 0.22) %
M.101		(0.46 ± 0.17) %	(1.16 ± 0.17) %
M.301		(0.62 ± 0.09) %	(0.38 ± 0.09) %
M.401		(1.05 ± 0.04) %	(0.28 ± 0.04) %
M.501		(1.82 ± 0.03) %	(0.24 ± 0.03) %
M.601		(5.10 ± 0.13) %	(2.98 ± 0.14) %
		d=90 cm	
		Aruco difference	Simulator difference
M.201		(1.75 ± 1.61) %	(1.70 ± 1.54) %
M.101		(0.77 ± 0.21) %	(0.75 ± 0.21) %
M.301		(0.59 ± 0.59) %	(0.59 ± 0.59) %
M.401		(0.56 ± 0.09) %	(0.09 ± 0.04) %
M.501		(0.48 ± 0.17) %	(0.95 ± 0.17) %
M.601		(2.36 ± 0.11) %	(0.10 ± 0.08) %

The RGB image for this scenario is shown in figure 6.17a. There are various types of objects in terms of shape, texture and transparency. The corresponding depth image as generated by the simulator is shown in figure 6.17b; it will be used as a reference ground truth image.

The design of these objects has been established to study the behaviour of the algorithm against a set of shapes, sizes and transparencies that can exist in a real scenario. For example, choosing the semi-transparent bottle in the simulator but not in the RGBD image obtained might compromise the proposed algorithm's performance.

As can be seen in figure 6.17a, three Aruco markers have also been inserted in the scenario to serve as a control point to check if the distance determined by the algorithm is in accordance with the distance estimated by the markers. Aruco markers are placed closer to the camera to avoid estimation error by the variation of the precession of the

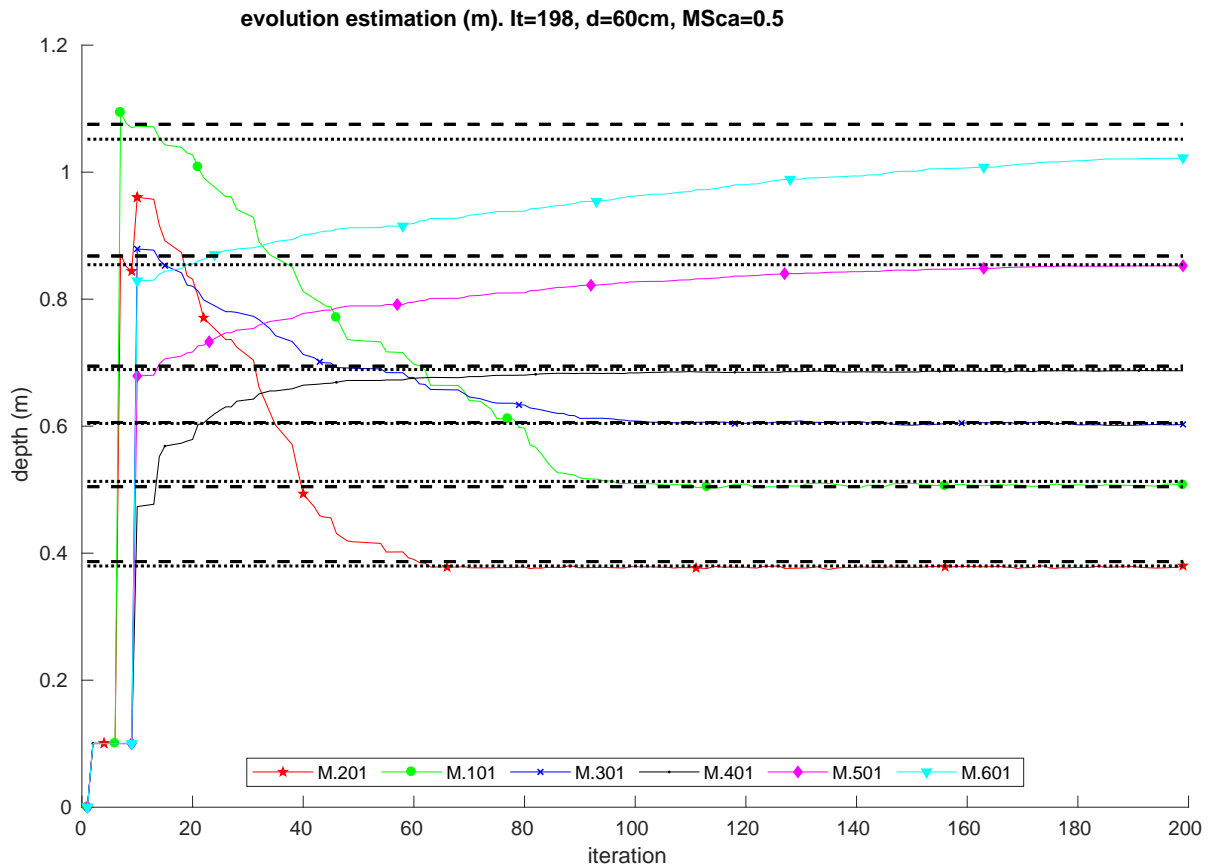


Figure 6.16: Example of the evolution of depth estimate mean for the Aruco regions only, with a fixation point at 60 cm. The dotted lines are the distances of the markers given by the simulator, and the dashed lines are the distances estimated by the Aruco algorithm.

markers with the distance.

MSE is used to evaluate the results. In addition, since **MSE** can yield misleading results in certain circumstances, **SSIM** is also employed. Finally, **SSIM** provides us with information about the structural similarity between the depth image generated by the simulator and that estimated by the algorithm.

For these tests, the parameters were given the values for which the best results in figure 6.15 were obtained without white noise error. Namely, $\rho = 0.5$ and $\sigma = 0.005$.

The evolution of **MSE** and standard deviation are shown in figure 6.18 with the expected behaviour. Figure 6.19 illustrates the evolution of **SSIM** index along the whole adaptive process. The range of possible values for **SSIM** extends from 0 to 1, being more similar the closer it is to 1. In this case, the variability of that index oscillates between 0.75 and 0.85 at the end of the algorithm iteration for all selected fixation points, comparing with the ideal depth image represented in Figure 6.17b.

Both the numerical results obtained from the analysis of figure 6.19 and figure 6.18 as well as the qualitative results derived from figure 6.20 show that the proposed algorithm is able to determine the depth image of a more complex scenario.

Thus, the evolution of **SSIM** and **MSE** is analogous and reaches the convergence value at iteration 40 for all cases. Remarkably, **SSIM** reaches a value of about 0.80, which indicates a very high structural similarity with the reference image.

It is also primary to highlight the behaviour of the algorithm concerning non-textured objects such as the night lamp or the camera, for which the determination of the optic flow involves more important difficulties. It is also noteworthy the behaviour for semi-transparent objects —such as the wine bottle or the beer mug handle— where the algorithm gives good results which could not be obtained, for instance, with standard depth sensors.



(a) RGB image of the scenario.



(b) Ground truth depth image.

Figure 6.17: Layout for the experiments with ordinary objects.

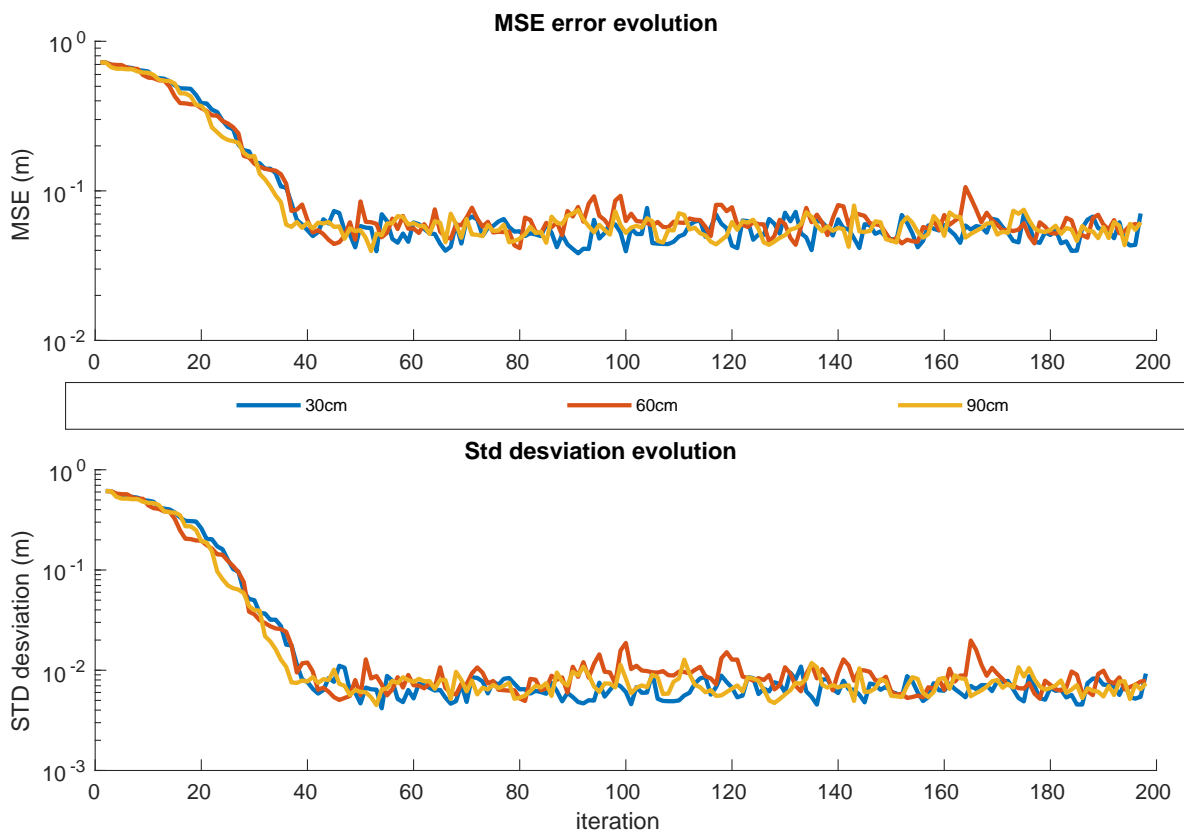


Figure 6.18: MSE and standard deviation evolution with real objects scenario for three different fixation points at 30, 60, 90 cm.

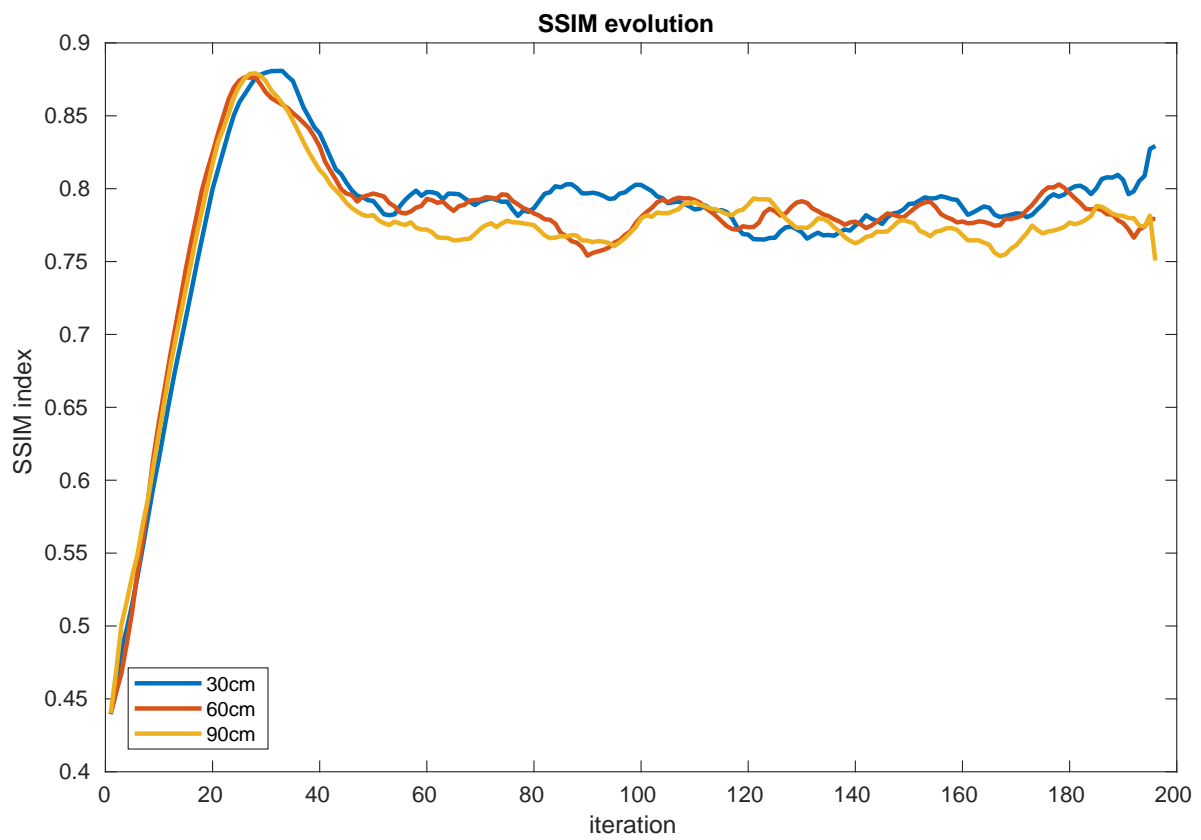


Figure 6.19: SSIM evolution with real objects scenario for three different fixations points at 30, 60, 90 cm.

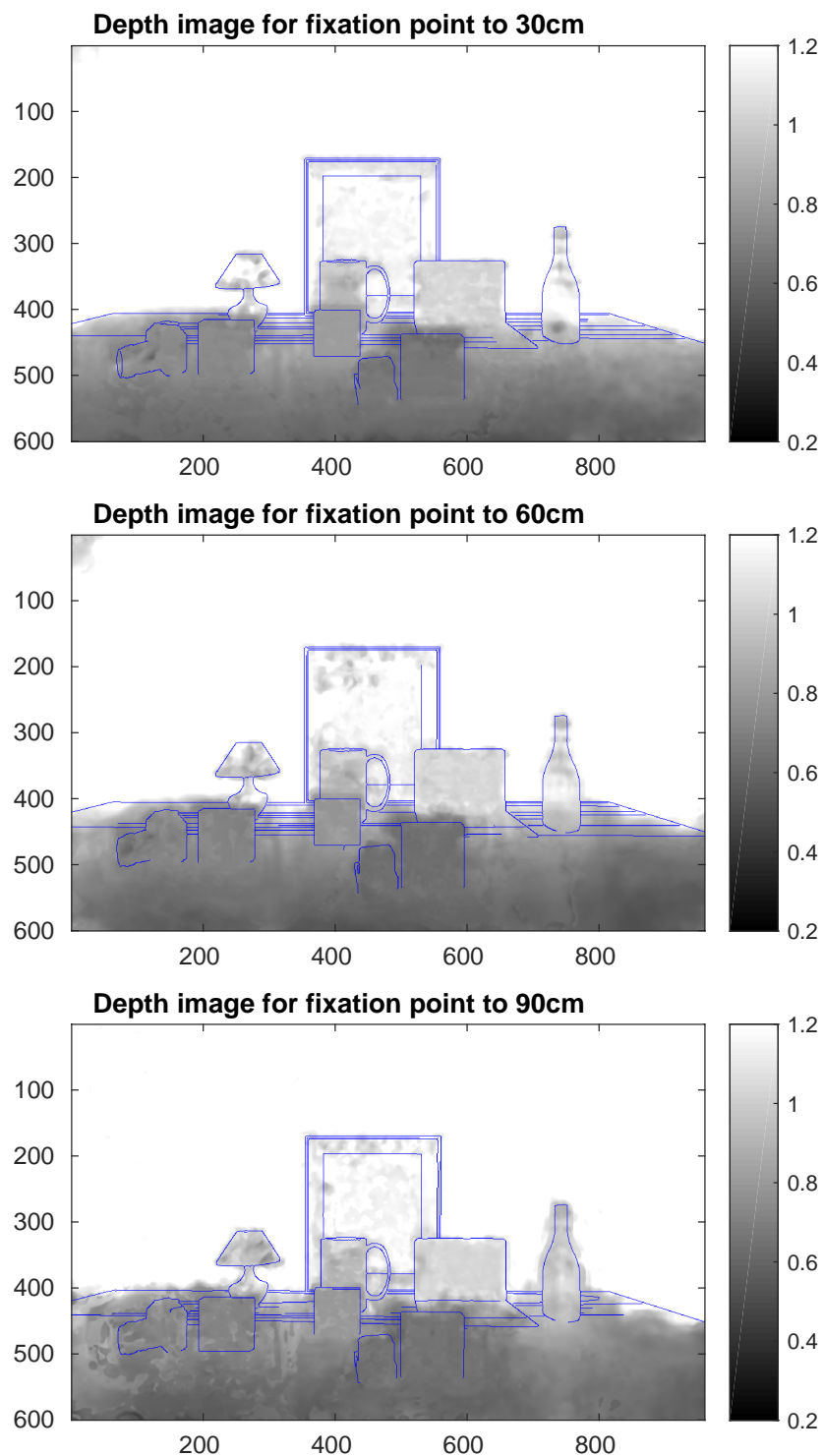


Figure 6.20: Gray-scale experimental results for the scenario with ordinary objects and fixation points at 30, 60 and 90 cm. The units of the gray-scale bar are in meters. The blue lines are the objects contours computed over background image using an edge detection algorithm. The aim of these lines is to delimit clearly the different depth areas generated by the objects in the algorithm results.

6.4.4.5 Influence of microsaccade movements in the depth algorithm execution

For studying the influence of microsaccades on the algorithm precision, the value of the fixation distance is fixed at 60 cm, and maintaining $\rho = 0.99$, $\sigma = 0.01$, the depth algorithm is tested using a $r_g = 1.5$ cm (figure 6.3) and $r_g = 1.0$ cm which is equivalent to consider just micro-displacements of the head and no saccadic movements of the ocular system.

The scenario with the Aruco markers is considered (figure 6.6c) to test the influence of microsaccades on the algorithm experimentally. Although both MSE and SSIM are measures providing information on the numerical and structural similarity of the depth image generated by the algorithm and the background image, to see what local influence can exist as a function of depth, it is better to use the Aruco markers directly, knowing these present precision problems as the depth increases.

In this case, a new method of comparison is introduced. The Aruco markers delimit an area in the image. If this area is overlaid on the depth image generated by the algorithm, it is possible to determine a depth distribution for the area enclosed by each marker. Thus, it is possible to obtain a depth profile for the image considering all these distributions. In addition, using a mixture of 6 Gaussians to approximate them, we can obtain numerical values describing the mean position and variance of depth in each marker.

The results can be found in figure 6.21b keeping the fixation point at $d=60$ cm, the algorithm parameters and executing the algorithm without microsaccadic displacements ($r_g = 0$). A comparison of these findings with the equivalent ones in figure 6.21a suggests that, apart from improving the time to reach a fine depth estimation, microsaccade displacements produce a stabilisation of the depth estimation for the further objects. This issue is in agreement with classical results to the effect that the probability of microsaccade occurrence increases with the distance of the viewing point from the mean eye position Rolfs (2009).

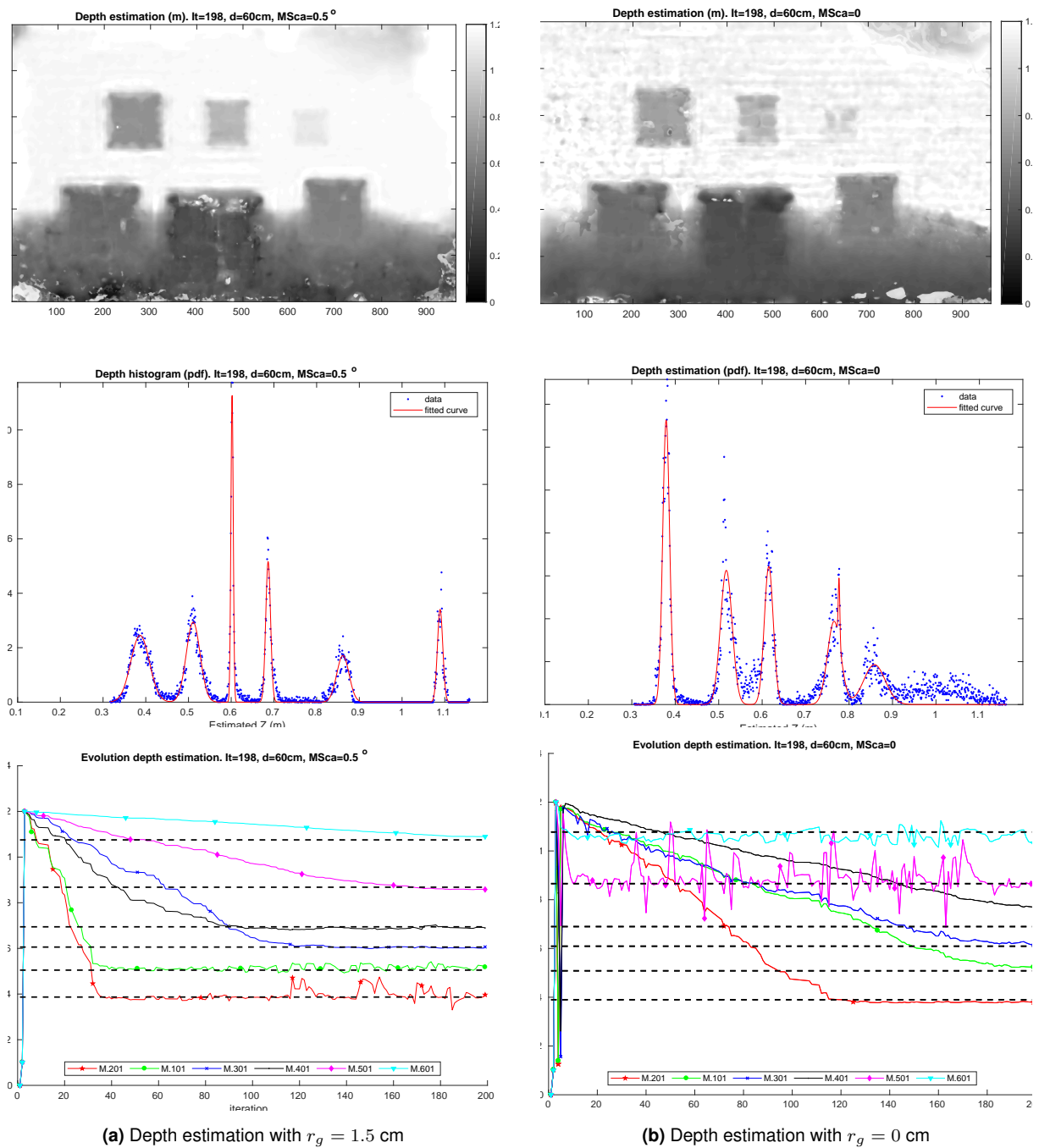


Figure 6.21: The first row is the obtained depth map after 198 iterations. The second one shows the depth histogram in the area of the markers. A mixture of 6 Gaussians to obtain the mean and standard deviation values is used. Each of these Gaussians should be centred at the marker distance. The last row represents the evolution of the depth mean for each area during the algorithm execution.

6.4.4.6 Ouchi Illusion

The graphic artist H. Ouchi designed in 1977 a static image provoking an alteration of visual perception in humans (Ouchi, 2013). This image can be seen in figure 6.22. Small retinal displacements that occur during the fixation process generate a segmentation of the internal pattern and the movement of this frame. Some observers report an apparent depth discontinuity, with the centre floating as it moves atop the background (Spillmann et al., 1993).

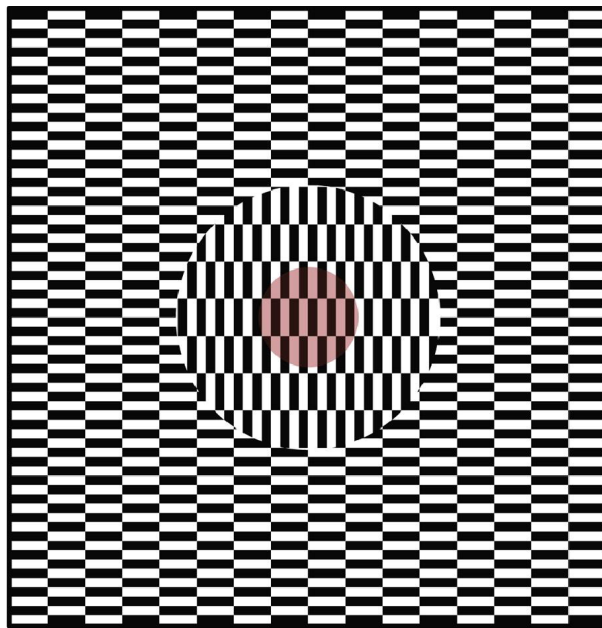
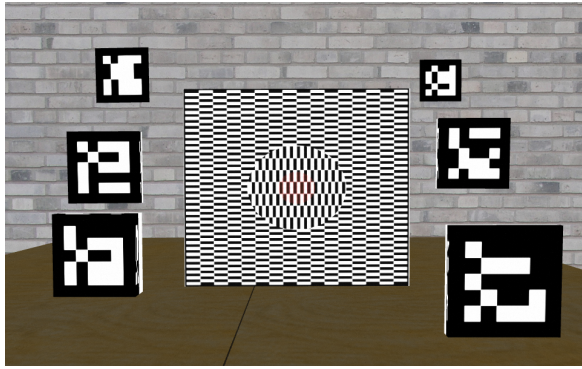


Figure 6.22: Ouchi illusion

This region segmentation occurring in depth should be reproduced to some extent by the proposed algorithm. Therefore, the depth image generated by the simulator and the one generated by the algorithm when the fixation point is in the centre of the illusion image should be different.

A scenario is designed as shown in figure 6.23b to test this hypothesis in simulation. The image with the Ouchi illusion is placed in the centre of the scenario figure 6.23a. Some Aruco markers are distributed around it as a reference to the algorithm's performance outside the Ouchi pattern. The image generated by the simulator of the depth of the proposed scene can be seen in figure 6.23b.

This experiment aims to determine whether the Ouchi pattern can generate a differentiation by applying the proposed algorithm for depth estimation between the inner circular area and the rest of the pattern as it occurs in humans. The image obtained should be similar to the one shown in figure 6.23b using a standard depth estimation



(a) Scenario distribution to test the Ouchi Illusion with depth algorithm



(b) Depth image generated by the simulation. As can be seen, the Ouchi marker is recognized by the simulator as a uniform plane. The Arco markers are placed in several depth levels.

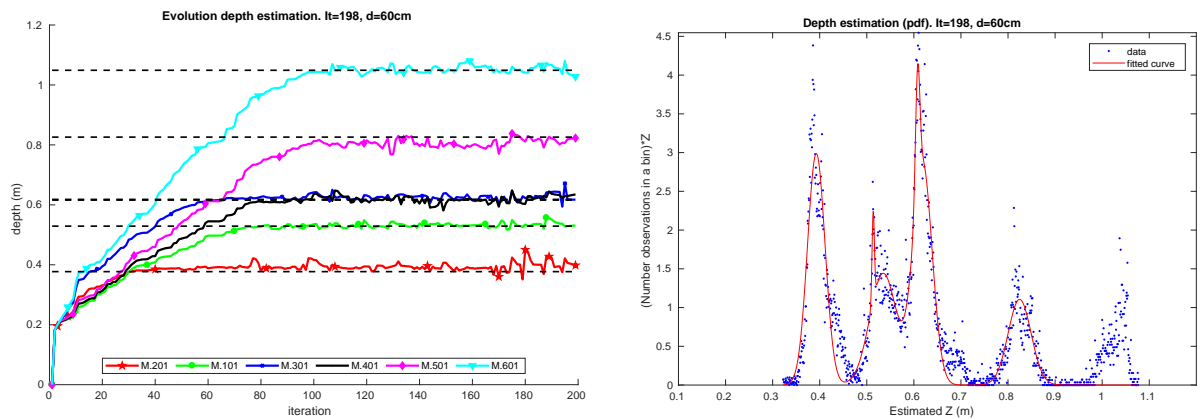
Figure 6.23: Ouchi image experiment setup

system, such as an RGBD camera.

In this case, the Gaussian error added to the generation of the images by the simulator is preserved, as explained in section 6.4.2. However the value of the camera pose is not distorted. In these conditions, different values of ρ and σ parameters are tested. Thus, in figure 6.24a, it can be observed the evolution of the mean distance in each control area defined by the markers for $\rho = 0.85$ and $\sigma = 0.001$.

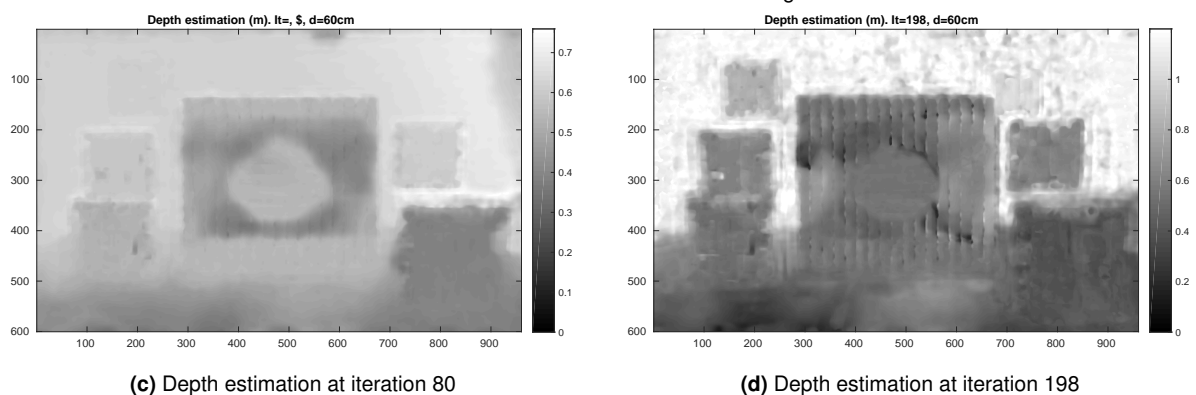
This result suggests that the algorithm execution is correct since all the positions of the markers are indicated in figure 6.24b. However, from the first iterations, a differentiation between the inner circular region of the Ouchi figure is detected. This differentiation is maintained throughout the algorithm execution (figures 6.24c and 6.24d). Although the estimation in the areas determined by the markers is progressive and conforms to the curve of the figure 6.24b, the area around the central circle is constantly changing, maintaining the differentiation.

The explanation for this effect is based on the proven influence of this perceptual illusion in determining the optic flow by humans and primates (Fermüller et al., 2000). The computation of the depth is made based on the estimation of the optical flow at each point. Therefore, the result is in accordance with the distortion of the optical flow caused by this illusion.



(a) Marker area depth average evolution over algorithm execution

(b) Depth distribution of the Aruco marker areas in the last iteration of the algorithm.



(c) Depth estimation at iteration 80

(d) Depth estimation at iteration 198

Figure 6.24: Ouchi experiment results

6.5 Experiments in a real robot

6.5.1 Introduction

The experiments developed so far in this chapter have been conducted in simulation to validate the proposed algorithm for estimating depth from modelling the fixation process in humans. This algorithm is based on capturing RGB images and estimating the camera's position variation considering its perception. For this purpose, the simulator of a robotic arm integrated into a commercial platform such as Baxter is employed.

Due to the good results obtained during the simulations and their similarity with the biological case, we present its implementation with a robotic system to simulate the fixation process.

In order to develop the robotic system to test the algorithm, the following design principles are considered:

Table 6.3: Oculomotor system setup according to morphology definition in section 4.3

Left				Right			
q_2	q_3	q_4	q_6	q_2	q_3	q_4	q_6
2.2 cm	5.05 cm	0.0 cm	0.7 cm	2.2 cm	-5.05 cm	0.0	0.7 cm
f	s	w	h	f	s	w	h
12 mm	1.4 μm	800 px	600 px	12 mm	1.4 μm	800 px	600 px

- i) The robot should have a two-camera system and be able to execute saccadic movements with sufficient accuracy to determine the variation of the position of the cameras.
- ii) As described in section 6.3.1, the fixation process involves both eye movements and microdisplacements of the head; the neck performs this task in humans. Therefore, the proposed system should generate these movements that affect the whole oculomotor system equally.
- iii) The baseline between the cameras should not be too large for future applications combining the monocular depth estimation algorithm in each camera together with the disparity between the two images in order to improve the depth image information. For this reason, Tombatossals' head (figure 4.15b) is not considered for this application since its baseline is 27 cm.

Based on these principles, it was decided to design and build a robotic head consisting of two elements working in unison for the axis of the fixation process: a two camera oculomotor system with a Helmholtz setup (figure 4.1) and a modified Stewart platform to perform the functions of the neck.

6.5.2 Oculomotor system

6.5.2.1 Mechanical and optical design

In chapter 4, several parameters were defined to establish a robotic system with a Helmholtz setup. Table 6.3 shows the values of these design parameters used to build the oculomotor system.

As seen in section 6.4.4.2, the accuracy and stability of the camera position estimation is a critical factor for the correct performance of the proposed algorithm. This issue

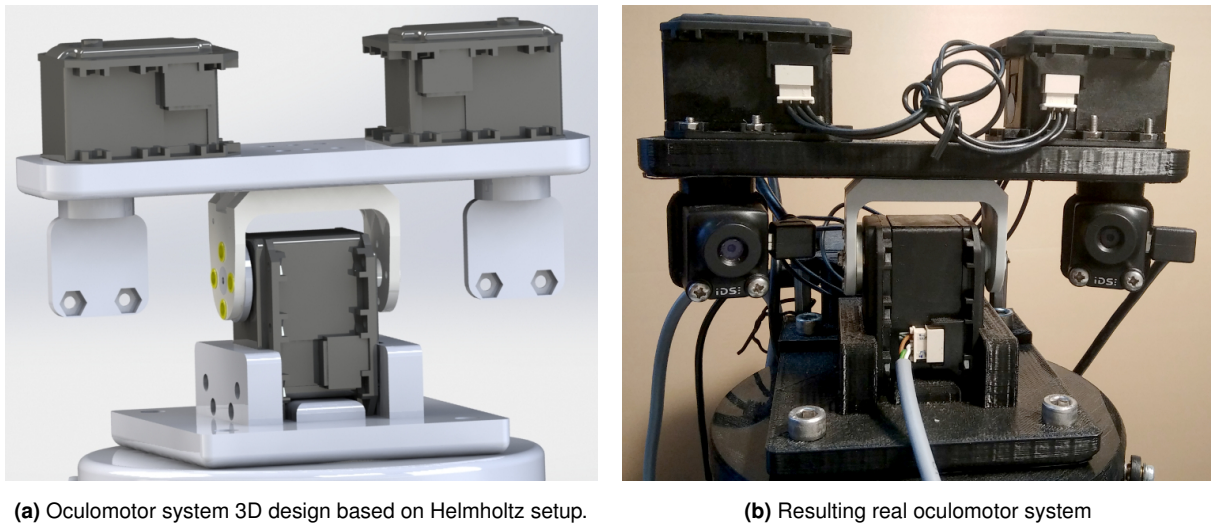


Figure 6.25: Oculomotor system detail

motivated the selection of motors for the oculomotor system allowing for a theoretical resolution of 0.0015 rad. Three Dynamixel MX28T motors (appendix D, table D.1) are used to implement the joints of the system.

The main idea is to create a prototype robotic system that can emulate the movements in the human fastening process. The connecting parts between the joints of the oculomotor system are produced exploiting the capabilities of 3D printing, bearing in mind the dimensions of the motors and cameras, including the cable connections and fixations. Taking advantage of the fixing possibilities offered by this type of motor, it is only necessary to design three types of parts: the base of the tilt joint, the base common to the pan joints, and the connections to the cameras.

The optical part of the oculomotor system consists of two IDS UEye Xs (appendix D, table D.3) cameras, characterized by their small size, allowing them to be adequately integrated into the designed structure without requiring an increase in the baseline of the system. The cameras are attached to the structure shown in figure 6.25a.

The main advantage of using this system compared to other arrangements is practical since it is possible to directly apply all saccadic movement generation systems developed in chapter 4.

The picture of the system with all the assembled components can be seen in figure 6.25b.

6.5.2.2 Oculomotor system control for saccade generation

For controlling the oculomotor system of this mechanical head in order to execute saccadic movements, the FEL controller described in section 4.4.2.1 is used. As mentioned, this architecture is composed of two controllers: a fixed B and an adaptive C_f . Both must be determined in order to perform the saccadic movement.

However, the training procedure to estimate B and C_f , both in the simulation and Tombatossals robot described in section 4.4.2.5, was performed by placing a stimulus in front of the oculomotor system in such a way the perception of the stimulus triggers a first estimation of the control action by the fixed controller which the adaptive controller corrects. Therefore, the oculomotor system is placed in front of a screen on which Aruco markers are projected to solve this problem. When the system is as close to the screen as possible, the field of vision is effectively covered by the projection. Figure 6.26 shows the layout of environment and the oculomotor system and how the Aruco markers are projected to perform the estimation of the controllers.

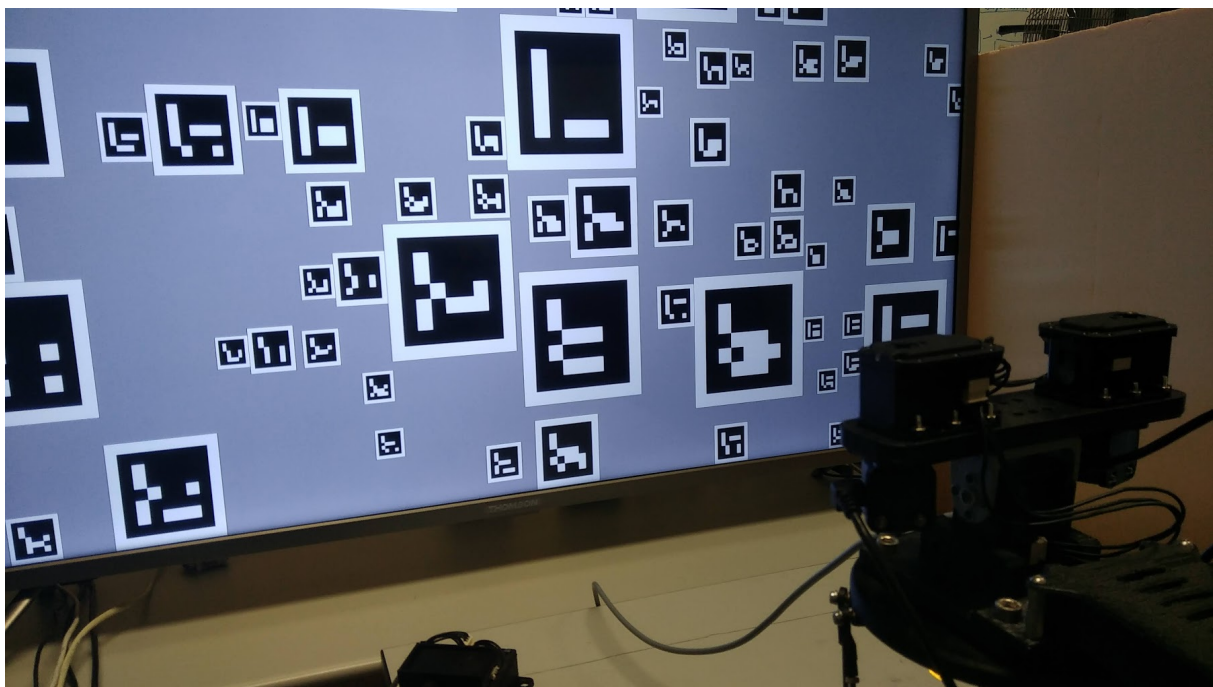


Figure 6.26: Oculomotor system and environment setup to train the fixed and adaptive controller from stimuli projected on a TV monitor.

Once it is solved how the stimuli are generated, the fixed controller (B) is determined as indicated in section 4.4.2.3.

For this purpose, two Aruco markers (one for the left and one for the right camera) are projected in such a way that their projected centre approximately corresponds to the

centre of the image generated by the cameras.

From this point, the motor babbling process begins, and random shifts are produced in the three motors that make up the oculomotor system. These displacements, as indicated in section 4.4.2.3, produce changes in the relative position of the stimulus in the camera image. In this case, instead of gathering data and performing the adjustment indicated by equation (4.8), the recursive least square (RLS) was used, as described in section 3.5. The aim is to be able to evaluate how the controller estimation evolves in each iteration.

It is possible to appreciate that in the case of the y-axis of the image, there is a slight deviation from this linear model for the more significant angle variations. However, it should also be noted that this variation is accurate at the centre of the image, which is the starting point for the displacements, and therefore it is possible that, as more extreme points of the image are considered, this variation will deviate from the linear model.

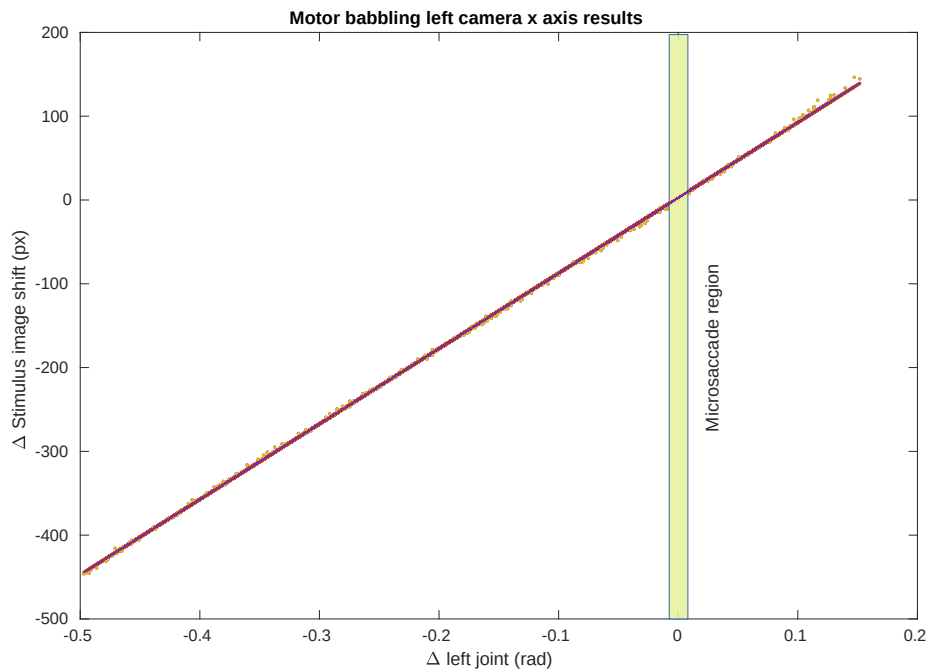
The results obtained are in accordance with the design of the oculomotor system developed, in which the optical centre of the image was intended to be at the geometric centre of the camera rotations.

The maximum angular displacement should be 0.0088 rad to execute the movement considering the maximum angle of the microsaccades used in the experiments performed in the simulation section 6.4.1,

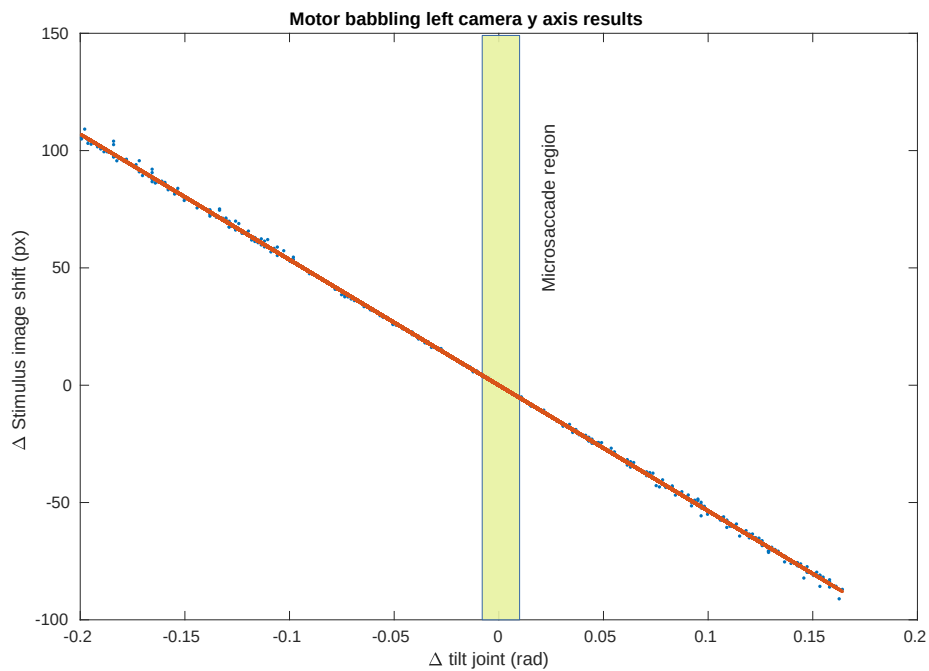
As can be seen in figure 6.27, the projection of this interval over the range of movements in which the fixed controller has been estimated is a tiny region. Consequently, it is possible to consider that the behaviour in this interval could be very close to the linear model, and therefore the estimation of the fixed controller could be a satisfactory approximation to execute microsaccades.

However, this approximation was made without considering the initial angular increment of the motors that conform to the oculomotor system's joints and considering the image's centre as a reference for the calculation. In figure 6.27b, it can be seen that as the angular increment increases, the dispersion in the variation in pixels of the image grows.

The numerical result (equation (6.15)) of B estimation also shows which are the values that contribute more to the pixel variation in the image:



(a) Variation of the number of pixels on x axis of the image respect to the increase or decrease of the joint value in the left camera of the oculomotor system.



(b) Variation of the number of pixels on y axis of the image respect to the increase or decrease of the joint tilt value of the oculomotor system.

Figure 6.27: Motor babbling results

$$\begin{bmatrix} \Delta_{tilt} \\ \Delta_{left} \\ \Delta_{right} \end{bmatrix} = \begin{bmatrix} -0.014 & \mathbf{1.119} & -0.031 & \mathbf{1.138} \\ \mathbf{-1.114} & -0.007 & 0.002 & -0.003 \\ -0.007 & -0.001 & \mathbf{-1.139} & -0.036 \end{bmatrix} \begin{bmatrix} \Delta x_l \\ \Delta y_l \\ \Delta x_r \\ \Delta y_r \end{bmatrix} \quad (6.15)$$

$B_{3 \times 4}$

Therefore, although B can be a useful approximation to generate the microsaccades, C_f is estimated as described in section 4.4.2.7. However, instead of using a robotic arm to move the stimulus in the robot environment, the TV monitor shown in figure 6.26 is used to provide the stimuli.

The core of the adaptive controller is a single-layer neural network with 300 random features, and it is adapted using the algorithm based Kalman filter (algorithm 1). The inputs and outputs are as defined in the description of the FEL architecture in section 4.4.2.1.

In each iteration, the system displays on the TV monitor twenty Aruco markers in different positions (figure 6.26) each one is different. All of them cover the field of view of the cameras. The system selects one at a time from the Aruco markers, which are visible in both cameras. Then, using the approximation of the fixed controller corrected by the current output of the neural network that forms the adaptive controller, a saccade is triggered that attempts to centre the target marker on the two cameras. The most common situation is that, until the adaptive controller is not trained, there is a difference after saccade movement between the visual position of the stimulus in the camera images and its centre. The adaptive controller is trained using this visual error. An error could occur when, for any reason, either in the stimulus selection or the stimulus search fails. For example, if no marker is detected at the initial position or if the selected marker is not detected after the saccade, the system ignores this iteration and restarts another iteration without updating the adaptive controller. Following this procedure results in a learning curve, as shown in the figure below.

After the training process, a test of 100 iterations was done. In this case, the neural network was not adapted; just the visual error is estimated. The average value of these 100 iterations is 3.589 ± 3.041

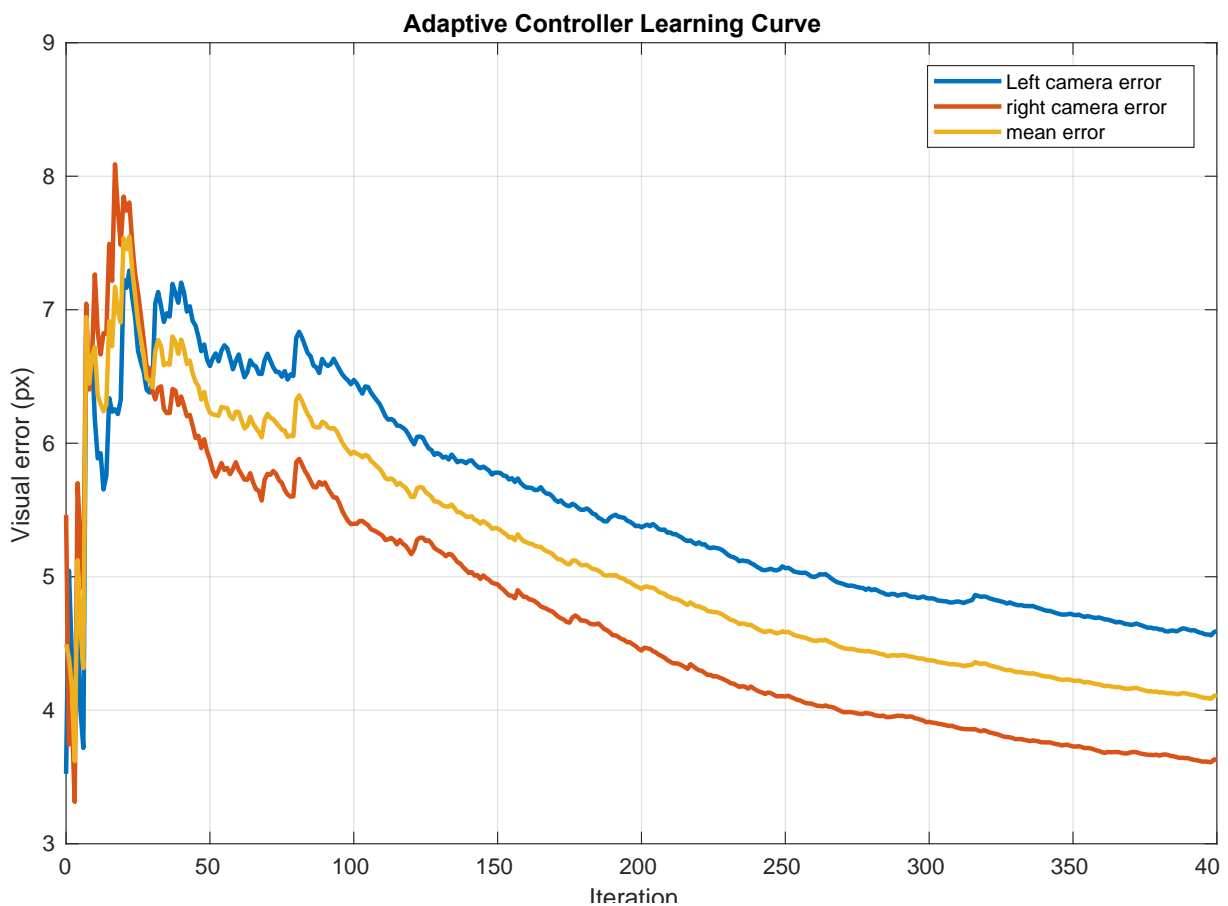


Figure 6.28: Learning curve of the adaptive controller training process for the designed oculomotor system

6.5.3 Neck system

6.5.3.1 Introduction

As indicated in the discussion of the model of fixation movements in humans (section 6.3.1), it is necessary to consider the microdisplacements of the head in order to understand the phenomenon in its totality. Therefore, the robotic platform proposed to implement the depth estimation algorithm must consider some mechanism to allow the displacement of the oculomotor system developed in section 6.5.2. This function is performed in humans by the neck.

In robotics, many mechanisms have been proposed to mimic the movements of the neck of a human being with similar degrees of freedom. A classic example is a neck developed for iCub robot (Metta et al., 2008). This design has a high performance; however, it limits the maximum payload (1.5 kg). Typically, the neck is not a critical factor

in the design of many humanoid robots because it adds complexity and limits the vision system usually mounted on it (see table 4.1). The way the neck is designed in many robotic systems is to add the roll pitch and yaw rotations (figure 6.29, not provided by oculomotor systems based on a Helmholtz setup).

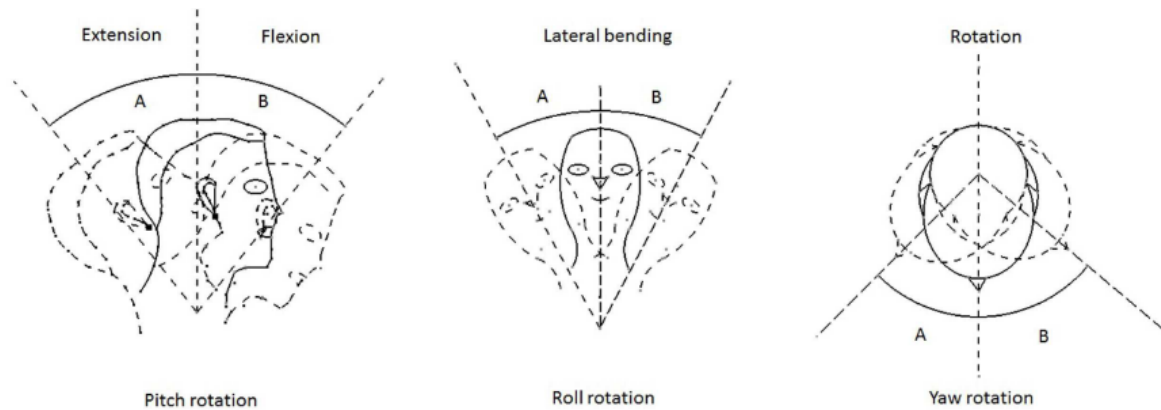


Figure 6.29: Diagram showing different types of neck movement in humans.(Alfayad et al., 2016)

The presence of a robotic neck often involves attempts to mimic human form and expressions rather than any functional utility (Hashimoto et al., 2006). On the other hand, there have been attempts to faithfully reproduce the behaviour of the neck through complex systems (Cronos robot (Holland and Knight, 2006)). This reference and many other works base the control of neck movements on tendons. For instance (Nori et al., 2007) use a central spring-actuated by three tendons which lengths are adjusted employing three motors. However, in this case, the goal is not to mimic the physiology as the functionality of the neck.

In this case, the development of the neck system is being considered separately from the oculomotor system, but systems are integrating the two in a serial motor assembly (Alfayad et al., 2016; Kim et al., 2004).

Finally, apart from the tendons and motors connected in serial, there is a mechanism based on parallel manipulators enabling the neck movements to be reproduced with a certain level of precision and simplicity (Lingampally and Arockia Selvakumar, 2017). This mechanism is precisely the approach that is developed in this work. Specifically, a robotic neck based on a Stewart platform is designed and developed. The oculomotor system described in the previous section is subsequently integrated into the platform.

6.5.3.2 Stewart Platform

A Stewart platform (Furqan et al., 2017) is usually composed of six prismatic actuators grouped in pairs over a platform's baseplate mounted above six universal joints. These actuators are joint to a top plate with other six universal joints. Thus, any point of the top platform can be moved with 6 degrees of freedom. From the control point of view, this kind of system has a peculiarity raising interest: their inverse kinematic can be computed analytically but forward kinematic cannot.

In general, solving the positioning of a system accurately has direct applications to any industry requiring such precision. Due to Stewart Platform properties, they are used in many fields; for example, they are widely used in precision surgery robotics (Wapler et al., 2003), in flight simulator machines, and astronomy to positioning radio telescopes (Su and Duan, 2000). The industrial applications of this kind of system are related to its capability of positioning with 6 DOF in a highly accurate way. As a result, they support classical industrial robots or high precision tasks in electronics or optics industries.

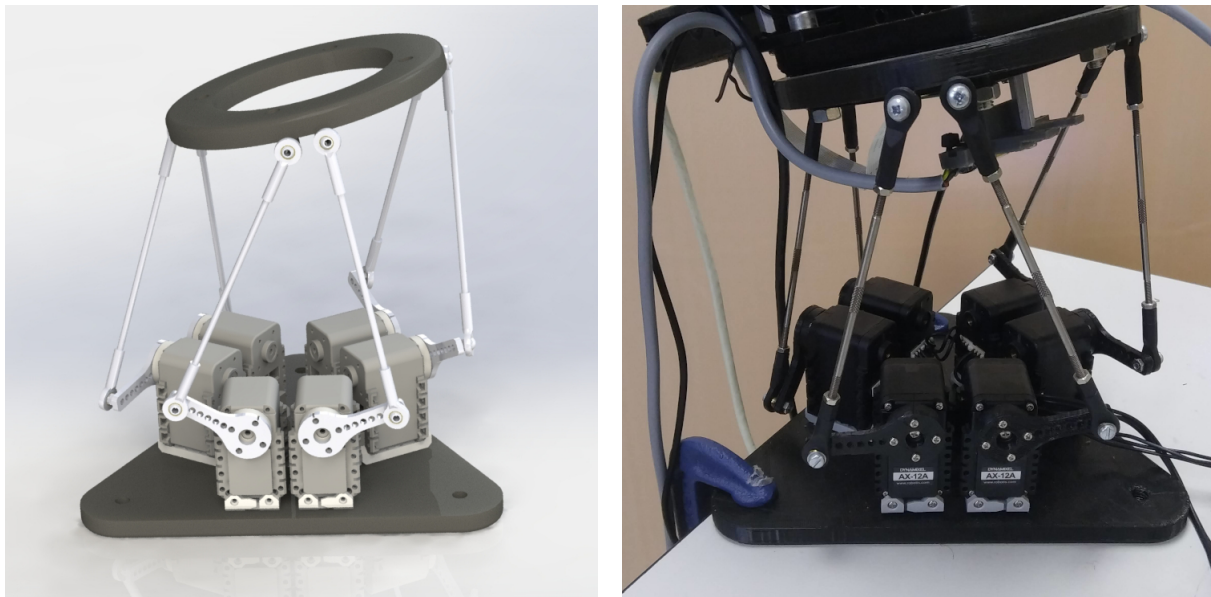
This project aims to develop a cost-effective platform based on off-the-shelf hardware components and 3D-printed elements, allowing us to reproduce the neck movements of human beings in the fixation process. The industrial Stewart platforms often use linear actuators. However, these can be replaced by rotary motors that could imitate their behaviour (Szufnarowski, 2013; Patel et al., 2018).

Following the design equations proposed in (Szufnarowski, 2013), a Stewart platform is assembled. Both the base and the platform are 3D printed parts. Six Dynamixel AX12A servomotors are used for the rotary joints. The servomotors are connected to the platform employing rods with universal joints mounted at their ends. This trait is the most critical part because the quality of these components directly affects the accuracy of platform positioning.

In figure 6.30a, it can be seen the model developed in 3D, where the essential components forming the rotary Stewart platform can be seen: the base (triangular part), the platform (circular part), the six AX12A motors with the piece connected to the rods and the universal joints linking the entire system. The Stewart Platform finally built can be seen on the figure 6.30b

6.5.4 Built oculomotor-neck system

The described oculomotor system (section 6.5.2.2) is mounted and fixed over the neck platform of figure 6.30. As a result, the built robotic system has the 6 DOF



(a) Neck based on rotary Stewart platform 3D model.

(b) Resulting robotic neck

Figure 6.30: Neck system detail

of the neck and three DOF of the oculomotor system; therefore, the total of DOF is nine. Initially, the kinematic control of the position for one of the cameras seems rather complex. However, by exploiting the Stewart platform's ability to compute the joint motor positions numerically to place the platform reference frame in a given pose, the initial problem is greatly simplified, i.e. it is reduced to solving two kinematic chains from the Stewart platform frame to each of the cameras. The result of the union of the oculomotor system and the Stewart platform acting as a neck can be seen in figure 6.31.

After a calibration process of both cameras separately with the traditional chess-board method (Wang et al., 2010), the detection of the Aruco markers is used to estimate the accuracy of the camera pose variation. This parameter is fundamentally involved in the depth estimation algorithm by the fixation process. In this way, the kinematic model of the system is tested. The camera's accuracy depends externally on the resolution of the camera and the precision of the Marker's pose and internally on the precision of the motors and the accuracy of the 3D model used to print the parts composing the system. The pursued objective is to have a variance in the estimation of the camera position of a maximum of 0.5 mm because under these conditions—according to the previous studies performed in simulation (section 6.4.4.2)—the proposed algorithm has a reasonable behaviour.

It was observed that when the movement of the cameras was executed by the oculomotor system, after performing 30 repetitions with different position variations, the variance in the determination of the camera position was 0.3 mm; therefore, it was within

the range required by the algorithm. However, the variance increased to 1 mm when the neck displacement was added, out of the acceptable range.

The inaccuracy origin is to be found in the type of motors used in the design of the platform and the universal joints employed. As a result, future versions of the system will include more precise motors and components.

In order to use the built system to test the depth estimation algorithm, a third camera (Raspicam V1.0, appendix D, table D.4) was incorporated into the system controlled by a Raspberry Pi (Upton and Halfacree, 2014)(see figure 6.31). This camera captures images of a fixed chessboard in a known position with respect to the platform's base. It is possible to determine the chessboard's position accurately using this camera with a 1920x1080 resolution. It can determine the actual variation of the platform position using two consecutive chessboard position readings. It is possible to determine, together with the model of the oculomotor system, the position of the cameras very accurately (with a variance of 0.6 mm) using the chessboard's position as a reference. However, the lack of precision in the positioning of the platform means the absolute displacement cannot be controlled within the range of movements needed to produce the head displacements determined by the parameters of the algorithm.

We decided to verify with the marker reading using the position of the chessboard as a reference if the displacement of the cameras after the execution of the microsaccades and the movement of the neck is within the range defined by the algorithm parameters; otherwise, the movement is repeated.

A description of the software architecture used to control this system is provided in appendix B.

6.5.5 Depth estimation test with the real robot

6.5.5.1 Environment setup

To check the data obtained in simulation, objects similar to those employed in section 6.4.4.4 are used. In addition, transparent objects with different textures and colours are considered.

These objects are distributed on a table so that even though there may be occlusions, the result from the point of view of depth presents multiple surfaces and textures to evaluate the algorithm's behaviour. As commented in section 6.4.1, Aruco Markers are introduced in the scenario for two reasons: i) to give information on how the algorithm evolves along the fixation process; ii) one of the markers is used as a

stimulus to perform the initial saccade.

In figure 6.32, the set of objects and distribution after executing an initial saccade are shown using as reference the central marker. The images for both the right (figure 6.32b) and left (figure 6.32a) cameras are shown. The fact that the marker is not in the centre of both images is due precisely to both images sharing a joint (tilt). Therefore, even if the employed FEL controller returns a value for this joint, the command sent to the head is the average of the tilt values returned by both the right and left camera controllers.

In the case of the cameras employed, they have a smaller field of view than the simulated cameras and this forces to reduce the maximum distance of the objects and the number of markers that can be displayed. In addition, if the marker is tiny, the Aruco detector cannot estimate its position due the resolution of the cameras, or it estimates it very inaccurately. Considering all these limitations, the maximum depth range is 80 cm, and instead of placing six markers on the image as in the simulations, only three markers are deployed.

6.5.5.2 Parameter adjustments and fixation process

The system starts from an initial neck position and the vision of the reference marker in both cameras (not in the centre). The first action is to perform a saccade on both cameras separately using the independently trained FEL controller.

This action is the starting point of the fixation process. Next, the RGB images captured after the first saccade (figure 6.32) are used as references to generate the depth image of the algorithm (I_0 in algorithm 6).

A sphere of motions for each camera is defined as proposed in section 6.3.2 to apply the proposed depth estimation algorithm. First, there is a random increase in the pose of the neck platform such that the position and orientations of the neck platform are varied. Then, the ranges in position are defined by this sphere whose radius is $r_m = 0.03\text{m}$. In the simulation, a radius of 0.015 m was used. This radius is enlarged to minimize the impact on the inaccuracy of the built Stewart platform. Regarding the orientation, the range is 0.02 radians for each of the rotations.

Once the neck movement is produced according to the new chosen pose (directly using the Stewart platform equations to calculate the value of the joints to achieve it), using the model of the oculomotor system and the reference of the chessboard captured by the camera of the Raspberry pi, the position of the oculomotor system cameras is estimated, which must be in the range of the selected sphere. If this is not the case,

another increment of the neck position is produced again until it is achieved.

The experiment is started with values of ρ and σ of 0.85 and 0.01, respectively. This selection is based on the results obtained in section 6.4.4.2. In addition, bearing in mind that the error in the camera pose estimation is significantly affected by the accuracy of the Stewart platform. Therefore, it can be appreciated that it produces a wide variation in estimating the depth at the three markers. In addition, however, it shows some oscillation around a central value that may correspond to the distance estimated by the Aruco algorithm.

The evolution of the estimation produced by the algorithm in reference to the three Aruco Markers can be seen in figure 6.33. The origin of this error can come from two sources: the estimation of the optical flow determining the displacement of the pixels in the image or the estimation of the variation of the camera pose. By observing figure 6.33, it can be seen that the variation occurring affects the three markers equally throughout the whole fixation process. Since the optical flow calculation is performed on a pixel-by-pixel basis, it seems unlikely that all pixels comprising the three marker regions in the image would exhibit the same type of deviation. It, therefore, seems more plausible that these variations are affected by the uncertainty in the shift in position.

6.5.5.3 Results

Fortunately, in order to solve the problem posed by figure 6.33, the algorithm allows to partially filter this noise in the estimation of the camera position. For this purpose, several experiments were performed with the same scenario and the same starting position, but increasing the parameter ρ as shown in the simulation of section 6.4.4.2 to reduce the effect of noise in the depth estimation.

In view of the obtained results shown in figures 6.34 and 6.35, the following conclusions are suggested:

- The results obtained validate the work carried out in simulation. Qualitatively by comparing figure 6.20 with figure 6.35, it is possible to determine the similarity of the results.
- An interesting result is the behaviour of the algorithm in the area of the left camera where the crystal glass is visible. As predicted in the simulation, the algorithm can return a result in this area. As can be seen, the grayscale intensity is at the same level as the marker at its feet. Therefore it suggests that the depth in that area should be more or less the same. This result is a competitive advantage over RGBD cameras which are not able to detect transparent objects.

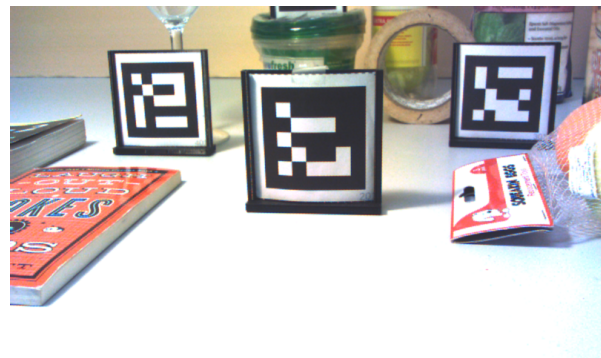
- The introduction of the Aruco markers in the scene has been confirmed as an element for depth estimation control when working in the real world. Otherwise, it would have been necessary to mount on the Stewart platform, in addition to the designed oculomotor system, an RGBD camera to capture the depth image and use it to generate a background.
- Quantitatively, it has been verified that the behaviour of the algorithm is similar to that observed in simulation, albeit the uncertainty in pose estimation of the used robotic system is much greater than considered in the simulation. Comparing figures 6.33 and 6.34e, the prediction of the simulation in section 6.4.4.2 is verified. By increasing the value of ρ in the algorithm, the depth estimation is more stable.
- The robotic system designed and built allows the execution of the movements that likely occur in a fixation process in humans; therefore, with further improvements, it can be a helpful tool for future work in this line of research.



Figure 6.31: Final oculomotor-neck robotic system developed



(a) Left camera



(b) Right camera

Figure 6.32: Initial RGB image from both cameras of the built oculomotor-neck system

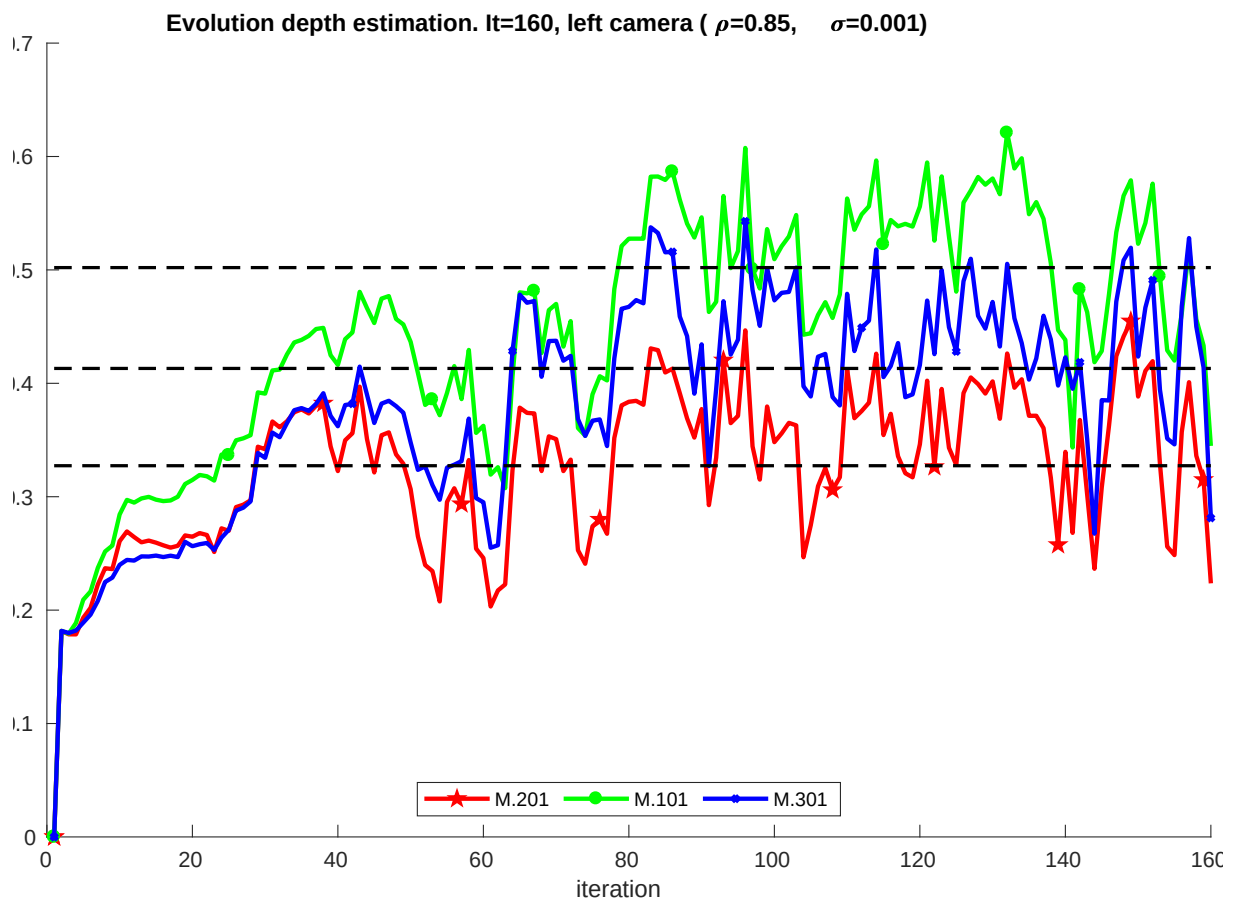
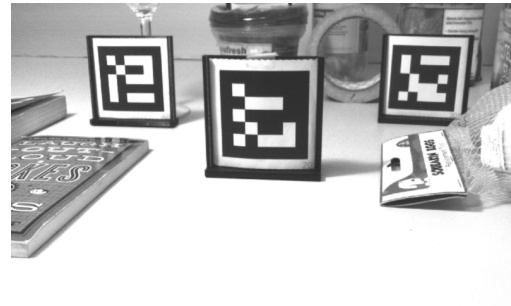


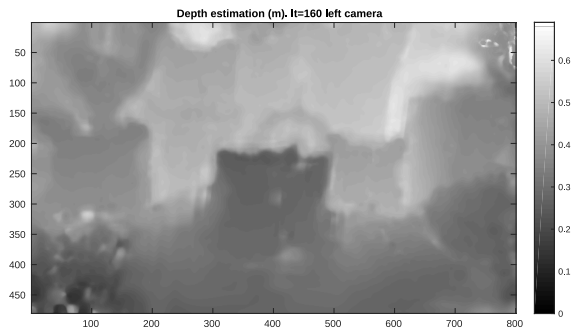
Figure 6.33: Evolution of obtained depth estimation in the Aruco Markers regions of the left camera with the oculomotor-neck robotic system for $\rho = 0.85$ and $\sigma = 0.0001$



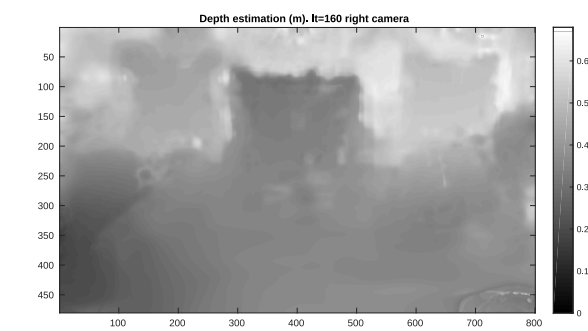
(a) Reference image captured after first saccade for left camera



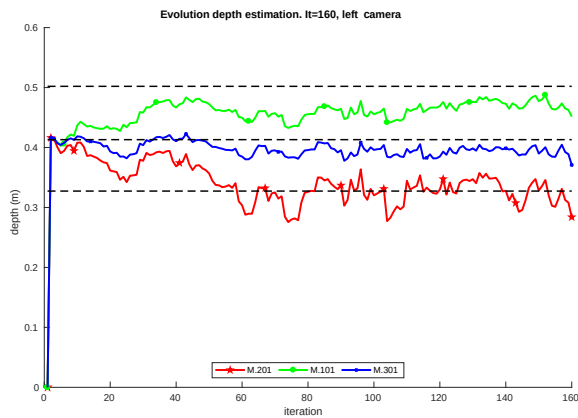
(b) Reference image captured after first saccade for right camera



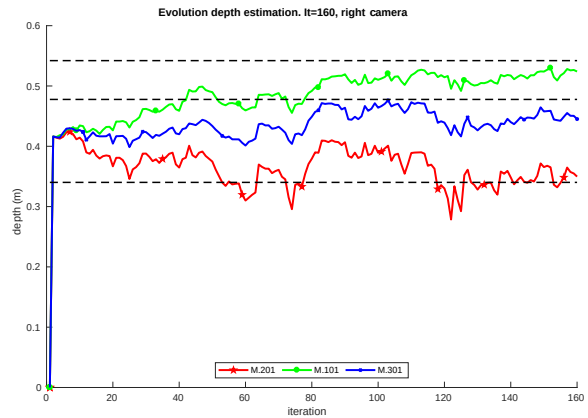
(c) Result generated by the algorithm after 160 iterations for left camera



(d) Result generated by the algorithm after 160 iterations for right camera



(e) Evolution of the depth estimation from left camera in the regions where the markers used as reference are located.



(f) Evolution of the depth estimation from left camera in the regions where the markers used as reference are located.

Figure 6.34: Results obtained by the proposed depth estimation algorithm for both cameras during the fixation process with parameters $\rho = 0.99$ and $\sigma = 0.001$.

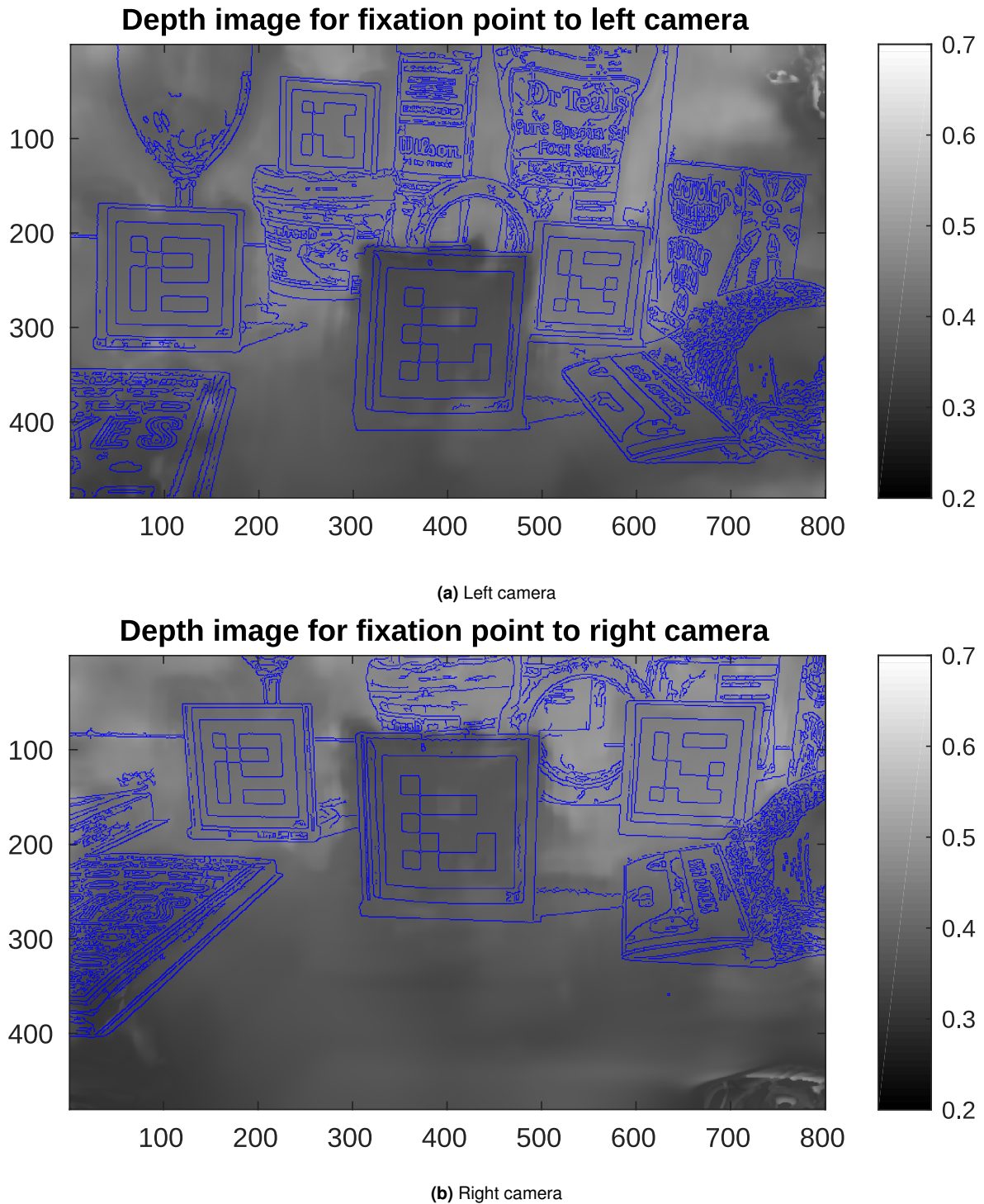


Figure 6.35: Overlaying the edges of the RGB images on the depth estimation obtained for both cameras, in order to delimit the contours in the depth image and thus better appreciate how the algorithm behaves on the different surfaces.

6.6 Conclusions

In this chapter, from several hypotheses based on bio-inspired models, a mathematical development is proposed leading to an algorithm to estimate the depth of a scene with a monocular system. This algorithm is supported by two traits: the proprioception of the movement and the visual displacement between two images estimated through the optical flow.

We assumed that each image pixel corresponds to an environment point located at a particular distance. Given a camera displacement, it is possible to estimate the optical displacement of the pixel in the image from the proposed camera model. This shift should be equal to the measured optical flow if and only if the assumed distance is correct. With this approach, the solution involves solving an optimization problem. Therefore, it is an optimization problem for each pixel in the image. After the first saccade, which starts the fixation process, the reference image is captured. At this point, all the movements occurring in the fixation process increase the information about the scene's depth.

A series of simulation experiments with various evaluation methods have been developed to test the performance of the proposed algorithm. We concluded that the algorithm works well for the conditions imposed in the simulation and has some tolerance to optical flow noise. However, it is more sensitive to the uncertainty in the estimation of the pose variance.

Two methods of comparison were used, the depth image generated by the simulator and the introduction of Aruco Markers in the scene to control the distance estimation to evaluate the results. The first method indicates the deviation of the algorithm from the correct value in the simulation; however, it has the disadvantage that it is hardly feasible in the real-world implementation of the algorithm. The algorithm is influenced by the parameters chosen in the execution time and the noise in the inputs. The accuracy can be checked using the depth image generated by the simulator and the Aruco markers.

From the performance obtained in simulation by the algorithm, it is possible to conclude that the microsaccades have two functions: they allow the sampling of images since this is one of the hypotheses considered when developing the algorithm. They also improve the perception of depth by increasing the information for the stabilization of the proposed algorithm.

Optical flow as an essential part of depth estimation has two consequences evaluated in the experimental test. First, because RGB images are used, the objects appearing in the image can show optical flow and, therefore, their depth can be es-

timated. This phenomenon explains the results obtained for transparent objects in the simulation and later in a real environment. Second, It has been proven that the algorithm is also sensitive to the Ouchi illusion, which misleads the perception in such a way that a non-existent depth separation is produced in an area where it does not really exist.

Although the simulation experiments were performed for a camera mounted on a robotic arm, an oculomotor-neck robotic system was designed and built to replicate the fixation process. The built prototype can mimic the microsaccades and the head displacement occurring during the fixation process with limited accuracy.

Despite using a different robotic system to the one employed in simulation, the results of the algorithm in an environment with real objects were similar to those obtained in simulation; however, it was necessary to adjust the parameters to minimize the noise in the estimation of the pose in the prototype developed.

In conclusion, a monocular depth estimation algorithm was developed by studying fixation movements where microsaccades play a relevant role. In this chapter, this algorithm has been formulated and tested both in simulation and in a robotic system developed ad-hoc, yielding results in accordance with expectations.

6.7 Publications supporting this chapter

- **Duran, A.J., del Pobil, A.P., 2019**, "Improving robot visual skills by means of a bio-inspired model", in *Proc. 9th Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics (ICDL-EPIROB 2019)*, Oslo, Norway, pp. 25-30. DOI: 10.1109/DEVLRN.2019.8850712.
- **Duran, A.J., del Pobil, A.P., 2021**, "Robot depth estimation inspired by fixational movements", *IEEE Transactions on Cognitive and Developmental Systems*. (in press, on line). Doi: 10.1109/TCDS.2020.3025057.
- **Antonelli, M., Duran, A.J., del Pobil, A.P., 2013**, "Application of the Visuo-Oculomotor Transformation to Ballistic and Visual-Guided Eye Movements", in *Proc. International Joint Conference on Neural Networks (IJCNN 2013)*, Dallas, Texas, USA, pp. 813-820. ISBN: 978-1-4673-6129-3/13.

Chapter 7

Conclusions and Future work

People ask me to predict the future, when all I want to do is prevent it. Better yet, build it. Predicting the future is much too easy, anyway. You look at the people around you, the street you stand on, the visible air you breathe, and predict more of the same. To hell with more. I want better.

**Ray Bradbury,
Beyond 1984: The People Machines**

7.1 Conclusions

As a result of applying the methodology presented in chapter 1, various biological models have been briefly described to generate some proposals of mathematical models, allowing us to focus on saccadic movements as the central axis of this work, developing and exploring several applications in the field of robotics. Chapter 2 described the biological substrate, both physiological and neurological, allowing the generation of saccadic movements in humans.

An undeniable quality of living beings is their ability to adapt to their environment. Usually, in robotics, the environment is a nuisance spoiling our perfect algorithms. In the case of living beings, the environment is just another element of their evolution. For this reason, in this work, we kept the environment in mind at all times. We chose to

employ adaptive neural network learning as information becomes available, to have the appropriate tools to simulate the adaptation of an artificial model to a given environment. These mathematical tools and their mathematical foundations are described in chapter 3. In particular, single hidden layer neural networks employing supervised learning are the essential tool chosen as the core of the models developed in chapter 4 to implement the saccadic movements in a robotic system. In chapter 3 two algorithms are proposed to adapt these neural networks, based on the Kalman filter or incremental sparse Gaussian process regressions.

After an analysis of the two most common configurations (Fick and Helmholtz setups) to reproduce human eye movements in a robotic system, we selected the Helmholtz configuration because it allows a sufficiently approximate reproduction of eye movements and has the advantage over the Fick setup that it requires one less motor and therefore one less degree of control. Once the configuration of the robotic system was clear, we proceeded to its parametric characterisation, not for a specific robotic system, rather by generically describing its morphology. In other words, the Helmholtz configuration of robotic systems capable of generating saccadic movements is actually formed by two kinematic chains starting at the base of the system and ending at each of the cameras that perform the vision function. The length of the links joining each chain joint is a parameter that can vary from one robotic system to another and define the robotic system's final morphology. In robotics, the kinematic part is usually differentiated from the rest to describe the robotic system. In this case, the vision system is also included in the morphological definition of the system. This trait means the robotic system is seen as a whole, where its morphology is defined by its perceivable external shape and by the parameters differentiating it from other similar systems.

Once the morphology of the robotic system capable of executing the saccadic movements was defined, we studied how to implement the saccadic behaviour. First, binocular encoding of the stimuli improved the accuracy of the saccades generated by the robotic system. Second, two architectures were proposed to generate the control actions to focus a stimulus in the binocular system through the saccadic movement: Feedback error learning and Recurrent architecture. The results obtained in the simulation and the real robot reach rates below 1% of visual error for both controllers, considering the cameras' resolution. The visual error of the experimental data in the test set is close to a generalised extreme value distribution.

The performance gains of these two architectures over others in the literature suggest them as candidates for implementing saccadic behaviours in our robotic system. While the Recurrent architecture presents better values of precision in the movements, the speed of execution of the Feedback error learning architecture converts it into a

favourite for many of the experiments and algorithms that are described throughout this work.

In this way, we have already defined a robotic system with a particular morphology (Helmholtz setup) and that is capable of executing a saccadic behaviour based on a [FEL](#) or [RA](#) architecture. However, the environment conditions and modifies the behaviour of the robotic system. The region of space delimited by six parallelepiped planes, where virtual punctual stimuli are randomly distributed, is considered as the environment with which the robotic system interacts. Its geometric centroid can characterise this environment beyond its volumetric dimensions.

In this way, as long as the appropriate stimulus appears, the robotic system can develop saccadic behaviour in the defined environment. In order to achieve this, however, it is necessary to train the system from scratch. When a living being is born, several pre-programmed behaviours allow the organism to adapt to the new environment rather than learning. These pre-programmed behaviours are engraved by millions of years of evolution in the genetic code of living beings. In robotics, on the other hand, systems tend to have only a few evolutions at best. They are usually unique entities.

In chapter [5](#), we suggested how the relationship between the internal model, which is capable of generating a particular behaviour, and the morphology of a set of robotic systems can be exploited in order to speed up the adaptation of new robotic systems to a given environment rather than starting from scratch. It has been necessary to use various mathematical tools to learn the relationship between the internal model and the morphology to achieve this goal. In order to have a sufficiently large population of robotic systems to learn this relationship, it has been necessary to train from scratch around 45,000 robotic heads with the same number of morphological parameters but with random values within specific ranges. Of the three neural network architectures tested, the parallel neural network has given the best results in predicting the internal model of a system by knowing the morphological parameters defining it.

A practical application arising from the results is that partial knowledge of morphology can still improve the learning of a new system. Finally, using bioinspiration, an artificial genotype has been formulated that in some way characterises each individual of a species of robotic systems. This artificial genotype is formed by a system-specific part and another common to the belonging species. This last one is defined based on the knowledge obtained from the relationship between the morphology and the internal model of all the individuals of the species. Once initialised, this genotype can evolve to adapt to other environments. This conclusion has direct implications for the development of Industry 4.0, where species of robots can use the knowledge acquired by other robots of the same species about how to interact in a given environment in

order to develop their initial capabilities which the local environment —where they will have to work— can modify through the process of adaptation.

For observing whether this model is analogous to the one proposed by analytical genetics between phenotype, environment and genotype through the concept of reaction norm, the reaction norm of a set of robotic systems that exhibit saccadic behaviour and whose artificial genotype has been previously obtained was determined. The results show a phenotypic variation analogous to that produced by living organisms in a biological environment.

In the last part of this work (chapter 6), the fixation process was modelled where saccadic movements play an essential but biologically unclear role in human beings. From a series of hypotheses involving generated saccadic motions as defined in chapter 4, an algorithm for estimating the depth image around the fixation point was designed.

The basic idea developed by this algorithm is that microsaccadic and head movements generate small retinal displacements that are combined with the eye or camera position variation signal. Thus, each pixel in the image becomes an independent minimisation problem between the actual optical displacement and the one predicted by the retinal displacement calculated from the camera's position variation, considering that the object illuminating that pixel is at a certain distance.

As the fixation process combines neck and microsaccades movements, a simulation was developed to evaluate its performance concerning two references: the depth image estimated in the simulation and a set of markers (Aruco markers) are strategically placed in the field of view to determine the distance to these regions from the camera. The latter is decisive to evaluate the algorithm on the real robot.

Simulation tests have shown that the algorithm works, although it is sensitive to estimating the camera position errors. It has also been observed that the microsaccades inherent to the fixation process seem to give some stability to the depth estimation. The tests performed with semi-transparent objects in simulation gave good results later confirmed on the real robot.

In addition, the behaviour of the algorithm was observed by means of the Ouchi illusion, and how this exhibited a differentiation of depth zones similar to the biological case.

To perform the experiments on a real robot, we needed a system that would allow the execution of saccadic movements as defined in chapter 4. However, we also needed a neck that would allow 6 DOF movements. For this reason, we decided to build a

prototype robotic system based on the conjunction of a rotating Stewart platform and a saccadic system generator as defined in previous chapters.

The experiments conducted on this robotic system confirm the performance of the algorithm and its limitations. The primary constraint lies in the need to have adequate accuracy in the estimation of the camera displacement. On the other hand, one of its greatest strengths is its ability to detect transparent objects, where RGBD cameras are unable of doing so.

In summary, three bio-inspired models based on three biological processes centred on the generation of saccadic movements were developed for implementation in robotic systems, allowing to:

- Increase their capacity of active vision and exploration through interactive perception.
- Employ knowledge of how other robotic systems learn to adapt it to a given environment.
- Increase the perception ability of a robotic system through the combination of saccadic and other types of movements inspired in visual fixation process.

7.2 Contributions

The work described in this thesis contributes to the development of several useful and applicable models in the field of robotics and specifically in active vision:

1. We propose two bio-inspired architectures for generating saccadic movements for active exploration of the environment through interactive perception. In addition, we parametrised a robotic system capable of generating these movements going beyond the mere description of the kinematic chains composing it and considering the specifications of the elements constituting the vision system as part of the set of parameters defining the morphology of the robot.
2. We developed the concept of an artificial genotype to describe a set of robotic systems with common morphological characteristics and interacting with the same environment. This genotype is initiated from the methodology developed to determine a robotic system's internal model and morphology. We can obtain this relationship using single-layer hidden neural networks in parallel.

3. Based on the hypotheses that microsaccadic movements may play a specific role in the fixation process in humans, we developed an algorithm for estimating a depth image using a single camera by minimising the difference between the optical shift in the image due to fixation movements and the optical displacement expected when the camera displacement is known.
4. We designed and built a robotic system that allows saccadic movement and coordinated neck movements. This system is based on a rotating Stewart platform combined with a Helmholtz system. In addition, we developed the software drivers for the control of this system.
5. The developed models based on microsaccadic movements and interactive perception have allowed us to generate and improve neural Deep Learning networks for depth image estimation with a single camera (appendix A).
6. For the implementation of all the proposed algorithms in the employed robotic systems, we developed a software architecture (appendix B) allowing us to use modular programming and the execution of parallel processes.

7.3 Future work

The work exposed in this thesis opens up a number of opportunities for future research lines as well as several open questions that would enhance and expand the research performed throughout this thesis.

1. One of the possible research lines that can be established is based on the concept of artificial genotype introduced in this work. Recent studies in biology corroborate that developmental plasticity plays a fundamental role in the diversification and specialisation of organisms. We are convinced that this emerging area of research in biology can contribute to a new paradigm in evolutionary robotics. The fundamental idea behind it is to overcome the current limitations of evolutionary robotics in terms of genotype-phenotype decoding, which in the vast majority of cases is posited as one-to-one correspondence, i.e. genes uniquely determine phenotypes in such a way that the influence of the environment is limited to the fact that it is used merely as a testbed to assess the fitness of the phenotype. This idea contrasts with current knowledge in evolutionary biology, according to which genotypes of organisms exhibit a capacity to express a range of different phenotypes in response to different environmental conditions. This behaviour is known as phenotypic or developmental plasticity and can be visualised through

norms of reaction, which represent values of a specific phenotypic trait for a set of environments.

From the proposed artificial genotype model, we estimated a reaction norm for a species of robotic systems. The results obtained simulate the process according to a genotype for a robotic system developing in different environments leading to the expression of different phenotypes, showing behaviour similar to that of living organisms in terms of their reaction norm.

A possible starting point, supported by several recent studies in evolutionary biology, is incorporating plasticity mechanisms in the development of robots through an evolutionary approach, which could be beneficial in obtaining autonomous robotic systems with a greater capacity to adapt to the environment. Since natural selection not just selects between genotypes but also between phenotypes, the phenotype and variation between phenotypes can play an essential role in the artificial evolution of robots in such a way that the environment not only can serve to select between the variations produced genetically but also create phenotypic variation and select between that variation. For this purpose, our artificial genotype model can provide a starting point.

The idea is that rather than searching for a robot design (genotype) that scores highest in a specific fitness function under particular environmental conditions, the winning design will be the one that shows the most remarkable plasticity in its phenotype. It hence will be more adaptable to changing environmental circumstances. This result will have important implications for the practical applications of autonomous robotic systems: this plasticity or capacity for adaptation will allow the system to modify its behaviour quickly facing new circumstances in its environment for which it was not explicitly designed or even, in the case of morphological plasticity, to redesign it much more efficiently based on its reaction norm, which will help us predict the optimal traits that the new phenotype should have.

2. Other possible lines of research arise from the contributions made in chapter 6:

- Refinement of the proposed depth estimation algorithm to deal with the problems posed by noise in estimating the variation of the camera's position. It could extend its use to situations where the robotic system experiences uncontrolled oscillations around an initial point, for example, a hovering drone, where we can determine its relative displacement concerning an initial position. Furthermore, use this displacement to generate the depth image.
- The perception of depth in humans comes from the integration of several cues. Experiments with the ad-hoc prototype developed to replicate fixation movements have allowed us to estimate the depth image independently

for each camera. However, this signal could be integrated with the depth estimation using the disparity generated between the left and right images. A possible line of research would be integrating these three signals to obtain a more accurate depth image of the robotic system's environment.

- The use of the geometric approach to modelling the fixation movement has allowed the design and integration of a series of depth neural models to determine the depth image from the variation in the image produced by a displacement of the camera (appendix A). The refinement of these deep learning models from the integration of the binocular system could be a possible research line.
- The result obtained with Ouchi illusion can launch a research line to test the built robotic system with other optical illusions even parametrised them and confirming its behaviour is similar to the biological one.

Appendix A

Deep learning application to monocular depth estimation in warehouse automation

A.1 Introduction

As a consequence of the results obtained in the development of an algorithm for depth estimation based on the oculomotor movements of human beings in the fixation process (chapter 6), two variables have been identified as fundamental when performing this task: the proprioception of the pose variation and the optical displacement in the images. Therefore, a new line of research is proposed to generalise for any circumstance and bear these two variables in mind.

In the last years, deep learning has become the mainstream paradigm in computer vision and machine learning (LeCun et al., 2015). Following this trend, more and more approaches using deep learning have been proposed to address different problems in sensing for robotics. However, robots pose several challenges for this methodology that relate to the fact that robot sensing is inherently active (Sunderhauf et al., 2018). This dynamic nature also offers opportunities that have been exploited for years in the context of active vision (Bajcsy et al., 2018); for instance, more information can be extracted from sensory signals by incorporating knowledge about the stable relationship between them and concurrent motor actions (Bohg et al., 2017). Similarly, spatial and temporal coherence resulting from embodiment can be exploited, for example, by taking advantage of the correlation of consecutive images or those taken from slightly different viewpoints (Sunderhauf et al., 2018). In contrast, data-intensive computer vision relies

primarily on enormous amounts of decontextualised images.

In this appendix, the relationship between optical flow and camera displacement (developed in chapter 6) is used to generate neural network models to estimate the monocular depth of any given scene in a particular context, such as the application of a robot for picking tasks in a warehouse. The tests performed in simulation in chapter 6 already considered this working scenario, but eventually, due to its bio-inspired basis, it was tested on a suitable robotic system.

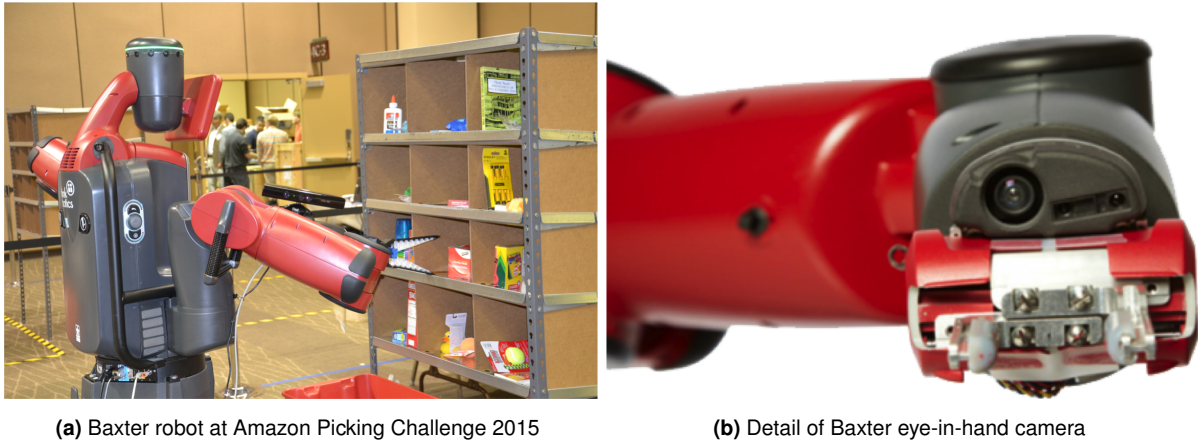
Model-based approaches as employed in chapter 6 are usually opposed to purely data-driven methods -such as deep learning-, but the use of environment models in combination with model-free algorithms is a promising trend towards more efficient reinforcement learning (Yu, 2018). With this aim, the knowledge of the specific parameters of our sensor is incorporated into the generic data-driven deep learning techniques. More specifically, three deep network architectures are proposed in such a way that information from a modelled and parameterised sensor is considered sequentially in their design, namely:

- i) The estimation of image displacements based on the camera model;
- ii) The optical flow estimation based on the correlation of two consecutive images and the subsequent correlation with the change in camera position;
- iii) The estimation of the camera displacement from a depth image.

Consistently incorporating each model improves the results and learning performance with a considerably smaller data consumption cost of training compared to pure data-driven deep learning. As can be seen, all these concepts were applied in the development of the model in chapter 6.

In the case of deep learning for object recognition, some works have taken advantage of active vision (Malmir et al, 2017), and even a dataset has been recently proposed that somehow includes temporal consistency (Lomonaco and Maltoni, 2017). For an up-to-date compilation of the literature on deep learning in robotics and interactive perception, see (Sunderhauf et al., 2018) and (Bohg et al., 2017), respectively.

Related work on monocular depth estimation with convolutional neural networks (CNN) can be categorised according to the number of input images (single or multiple) and the learning approach (supervised or unsupervised). A multi-scaled deep network for supervised learning was proposed to infer the depth map from a single image (Eigen et al., 2014b). Others followed this single-image approach by considering computational random fields (Liu et al., 2015) or using long short-term memory and recurrent neural



(a) Baxter robot at Amazon Picking Challenge 2015

(b) Detail of Baxter eye-in-hand camera

Figure A.1: On the left, Baxter robot with UJI RobInLab team at Amazon Picking Challenge 2015. Manipulating items within the confined space of the shelf poses a number of challenges in terms of visibility and maneuverability. This could not be accomplished with the RGB-D sensor shown in the image that was mounted on the robot's elbow. The right-hand image shows a detail of Baxter's fully integrated built-in eye-in-hand visual sensor that we propose to use for 3D depth estimation as a complement to the RGB-D sensor.

networks (Kumar et al., 2018). However, even though it is possible to reconstruct 3D information from a single image, the performance is not as good as that of networks that consider several images or take into account the camera motion (Ummenhofer et al., 2017).

Unsupervised learning techniques have been recently proposed, such as a network composed of depth and pose networks with a loss function based on warping views to a target (Zhou et al., 2017); or another based on generative adversarial networks (Almalioglu et al., 2018).

These deep learning techniques depend on an undetermined scale factor that converts the generated depth maps into absolute values. Unfortunately, this trait is not practical for most cases in robotics since an absolute depth map of the surrounding environment is needed. Pinard et al. have recently pointed out this issue (Pinard et al., 2018), solving the problem by adding the velocity of the camera as an additional input. On the other hand, the distances that are handled in (Pinard et al., 2018) and other related approaches are very different from the working range that we consider since they are too large and coarse. The reason is that their focus is on localisation tasks, while the aim of this work is dealing with a maximum distance that is determined by the working area of the robot for manipulation within the competition's deep shelves as opposed to nearly a bird's eye view in existing datasets.

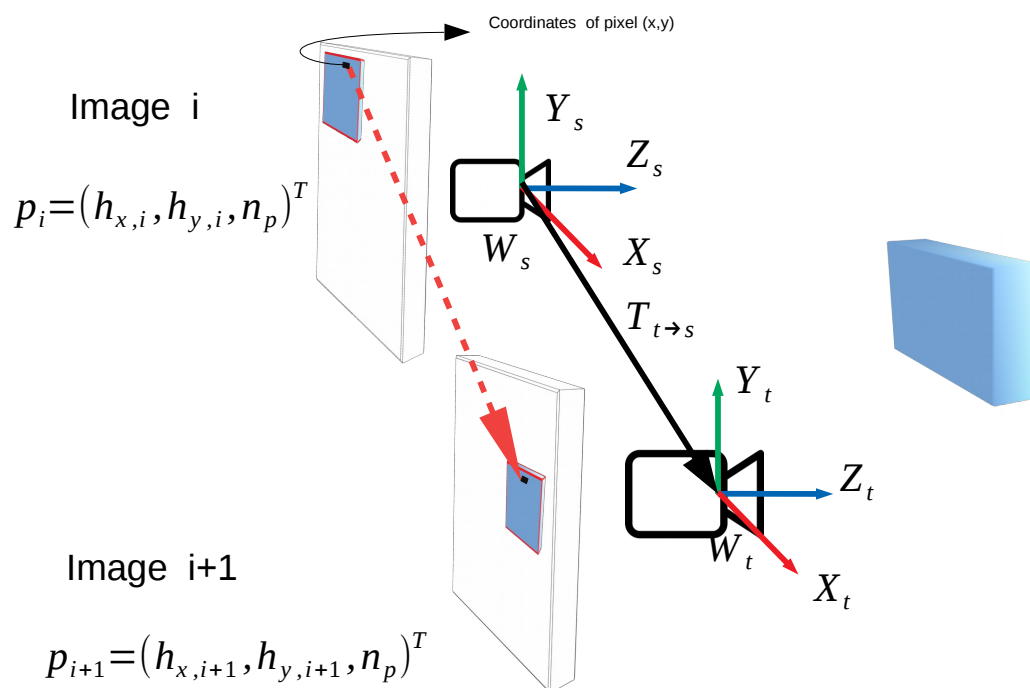


Figure A.2: An object is projected onto the image plane of the camera in two consecutive instants (i, i+1) after a displacement ($T_{s \rightarrow t}$) taking as reference the frame source (s) of the camera (W_s) and considering the target (t) frame of the camera (W_t), the displacement of the pixel (p_i) in homogeneous coordinates h_x , h_y and n_p (near camera plane), is determined by the difference between the position of the pixel in the images (i+1) and (i), i.e. $p_{i+1} - p_i = (T_{t \rightarrow s} p_i) - p_i$. x and y denote the coordinates of the pixel in the image plane.

A.2 Methodology

In order to evaluate the impact of considering prior and external models (as designed in chapter 6) in the design of deep neural network architectures, three deep schemes are developed to solve the depth estimation problem in a robot with an eye-in-hand camera for manipulation in an online shopping warehouse shelf (figure A.1)

The proposed model in chapter 6 based on human fixation movements uses the camera model and the relationship between optical flow and camera position variation. The proposed deep networks integrate this model into their architecture. In figure 6.4, the displacement of the camera was produced considering that the gaze point was maintained to a certain degree. In this case, a more generic case is considered, as shown in figure A.2

While moving the robot's hand towards a target object, first the mounted camera captures a scene of the surroundings (a source image) that is equivalent to capture the initial image in algorithm 6. Then, the hand moves slightly, and the camera captures a new scene (a target image). Simultaneously, a relative pose between those images ($T_{t \rightarrow s}$) is calculated based on the joint angles measured by the encoders embedded in the robot's arm. Besides, the relative pose is converted into a displacement map of each pixel in the image plane. Therefore, it has the same dimensions as the images. Let us denote by p_i and p_{i+1} the homogeneous coordinates of a pixel in the images i and $i+1$ respectively (see figure A.2). Using a transformation matrix from a target view to a source view ($T_{t \rightarrow s}$), the displacement of each pixel is defined as follows:

$$Displacement(x, y) = \begin{bmatrix} \Delta px \\ \Delta py \\ \Delta pz \end{bmatrix} = p_{i+1} - p_i = (T_{t \rightarrow s} p_i) - p_i \quad (\text{A.1})$$

This displacement is a simplified form of the position variation generated by the fixation movements used in chapter 6. This case has been simplified because the camera's roll pitch and yaw angles hardly change since they are considered parallel or perpendicular movements to the image.

Based on this configuration, three different arrangements of deep neural network architecture are proposed:

A.2.1 DepthS neural network

It combines directly on a convolutional neural network the two sources of information to generate the depth image, i.e. the two images and the position variation. The variation of the camera position is transformed using the camera model and the camera's intrinsic and extrinsic parameters to estimate the displacement of each pixel of the image concerning the reference frame of the world. Therefore, the neural network has as input two tensors formed by the two RGB images and a third tensor formed by the variation in the three axes of each image pixel. The neural network's output is the depth image corresponding to the first RGB image in a grayscale of 0 to 255 intensity. This network is illustrated in figure A.3 and we call this architecture *depthS*. DepthS is based on *FlowNetSimple* Dosovitskiy et al. (2015). First, depthS concatenates these inputs and convolves them three times before the contracting part. The contracting part is composed of multiple convolutional layers to abstract feature maps.

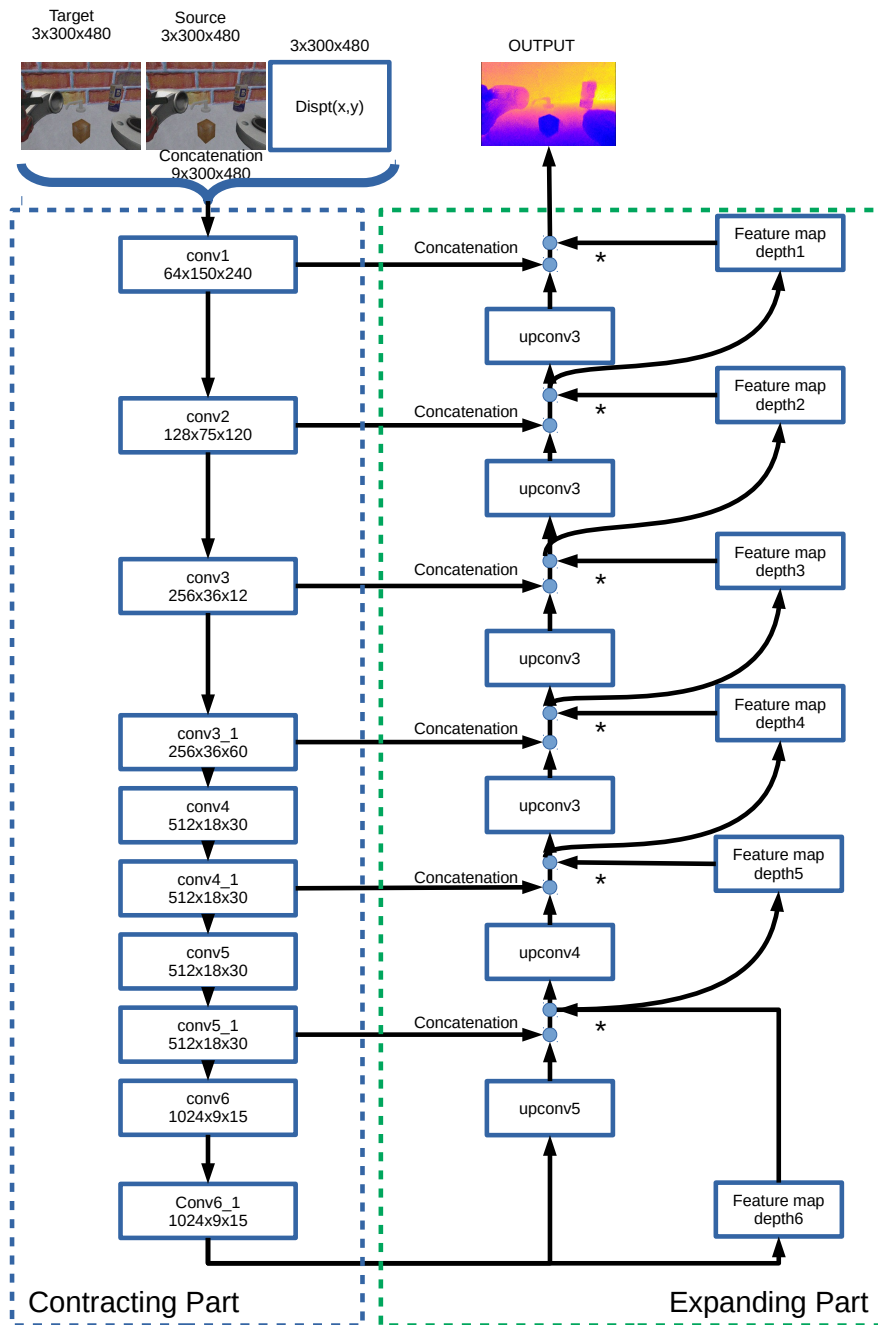


Figure A.3: Detailed architecture of depthS. The three inputs (source image, target image and displacement map) are convoluted to extract the local features, and then the features are deconvoluted to generate the depth image. Each box represents a network layer where convolutional operations occur in the contractive part, and unconvolutional and unpooling operations are performed in the expanding part.

A.2.2 DepthC neural network

: This architecture attempts to go a step further in incorporating the model obtained in chapter 6 into its design. Knowing the existence of a deep neural network capable of estimating the optical flow given two images (*FlowNetCorr* (Dosovitskiy et al., 2015)), the information of the position variation is incorporated into its architecture. In this network, a type of layer called correlation layer is introduced, which is the basis for the performance of *FlowNetCorr*. The fundamental characteristic of this layer is that it performs multiplicative patch comparisons between two feature maps. The overall architecture of *depthC* is shown in figure A.4. First, *depthC* processes three inputs with identical streams, a pair of images and a displacement map. Then, feature maps of the images are combined by the first correlation layer. Subsequently, the product of the first correlation layer is convolved three times, and it is associated with a feature map of the displacement at the second correlation layer. Moreover, the product of the second correlation layer is concatenated with a feature map of a target image denoted by *conv_redir* in figure A.4. In this way, the feature map generated from the source image is combined with the features generated from the correlation between the visual displacement and the target image. Finally, contracting and expanding parts process the product and generate a final depth map as in *depthS*.

A.2.3 DepthCSx neural network

This network is the most complex proposal and combines the two previous ones, and introduces a key factor behind the depth estimation algorithm shown in chapter 6. This algorithm reached the final result by successive iterations searching for depth value minimising the difference between the estimated and predicted optic flow. To transfer this concept is proposed an architecture of a depth neural network, inspired by *FlowNet2* (Ilg et al., 2017), *DeMoN* (Ummenhofer et al., 2017) and *SfM-Learning* (Zhou et al., 2017). This type of network is denoted by *depthCSx*, where *x* is the number of *depthS* networks used in the network design. First, *depthC* processes a pair of images and a displacement map and predicts a first depth map. After obtaining the first depth map from *depthC*, *depthS* processes the following information: a pair of images, a predicted depth map, a warped image and brightness differences. To effectively link the first depth map with a pair of images, a warped image and a brightness error Ilg et al. (2017) Zhou et al. (2017) are introduced. The warped image \tilde{I}_w is obtained from a target view I_t by projecting pixels onto the source view I_s , based on the predicted depth map \hat{D} and relative pose $\hat{T}_{t \rightarrow s}$ and using bilinear interpolation to obtain the value of the warped image \tilde{I}_w at location p_t .

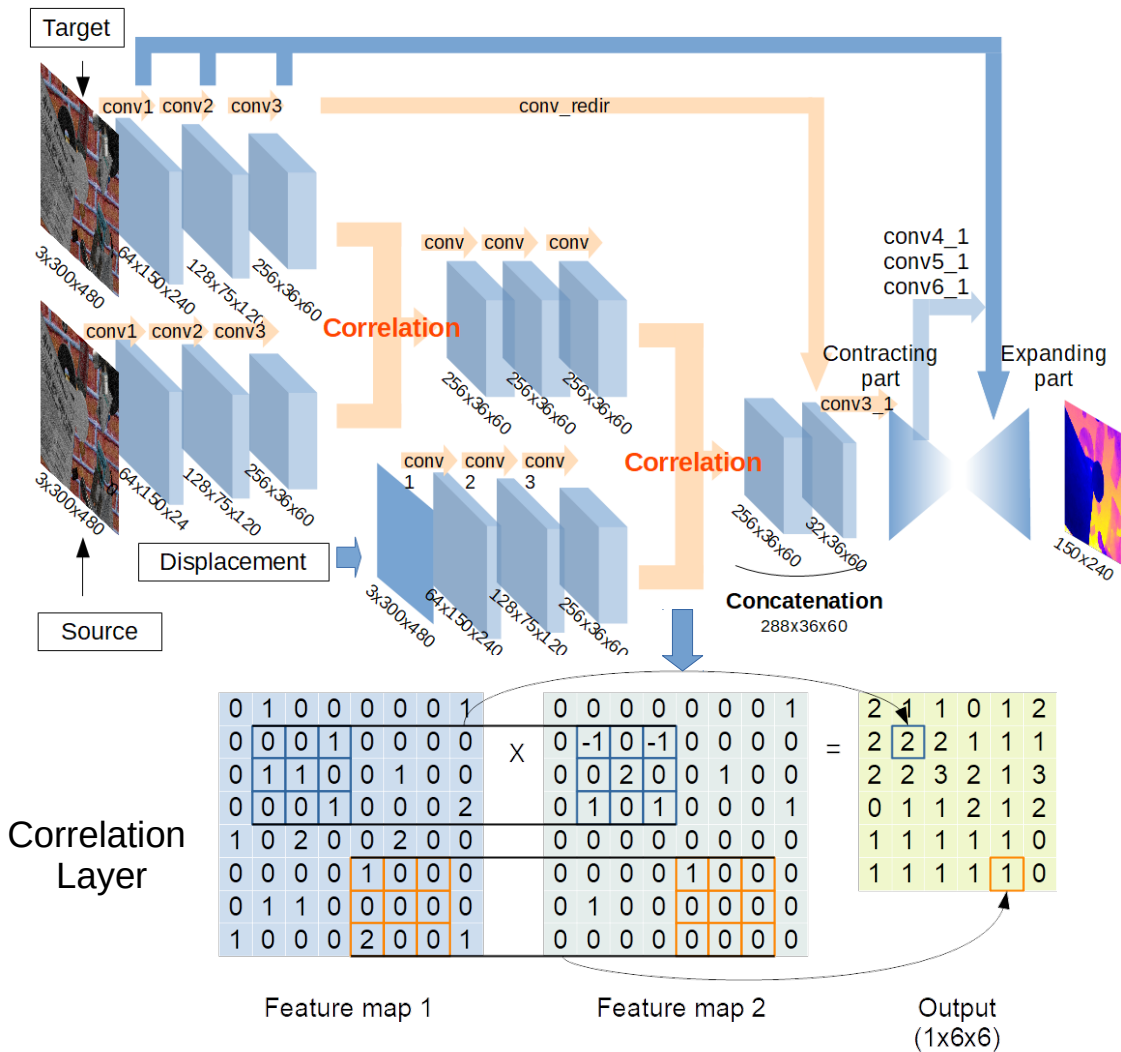


Figure A.4: Architecture of depthC. The features extracted from the images are processed by two convolutional branches merged by a correlation layer. The result is convoluted and correlated with the third stream of features extracted from the displacement map. An example is shown at the bottom, illustrating how the correlation layer operates.

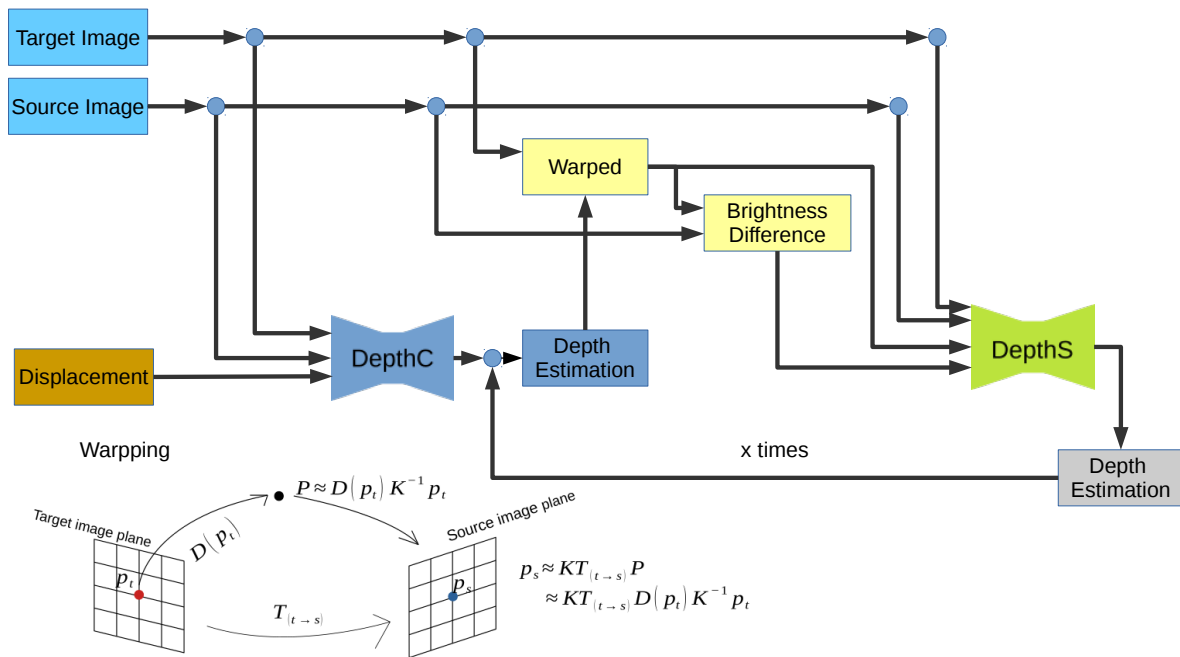


Figure A.5: Architecture of depthCSx. DepthC predicts a first depth map, and then depthS processes a pair of images, the predicted depth map, a warped image and brightness differences to generate an improved depth map. The structure of depthS is repeated x times so that depth maps repeatedly.

A.3 Experimental setup

As in chapter 6, the simulator with the Baxter robot is used. In this case, unlike the scenario described in section 6.4.4.4, no Aruco markers are considered, and the objects are actually arranged in a random and perhaps meaningless way. The aim is to expose as many different surfaces to the network as possible for training. Thirty ordinary objects models are combined and randomly located in front of the robot. A depth camera simulation placed in the location of the RGB camera was used to capture the depth images and generate the ground truth. The arm’s movements are limited so that an initial pose is defined to capture the maximum area of the workspace, and then different poses are generated into a sphere centred in the initial pose and with a maximum radius of 5 cm. This restriction is a notable difference with respect to section 6.4.1 where the radius of movements sphere was 0.015 m. The 6 DoF of the camera are randomly changed within the limits of this sphere in such a way that the end effector is moved within these limits while keeping the orientation of the camera unchanged so that only the translational component will need to be input into the neural networks, reducing in this way their complexity.

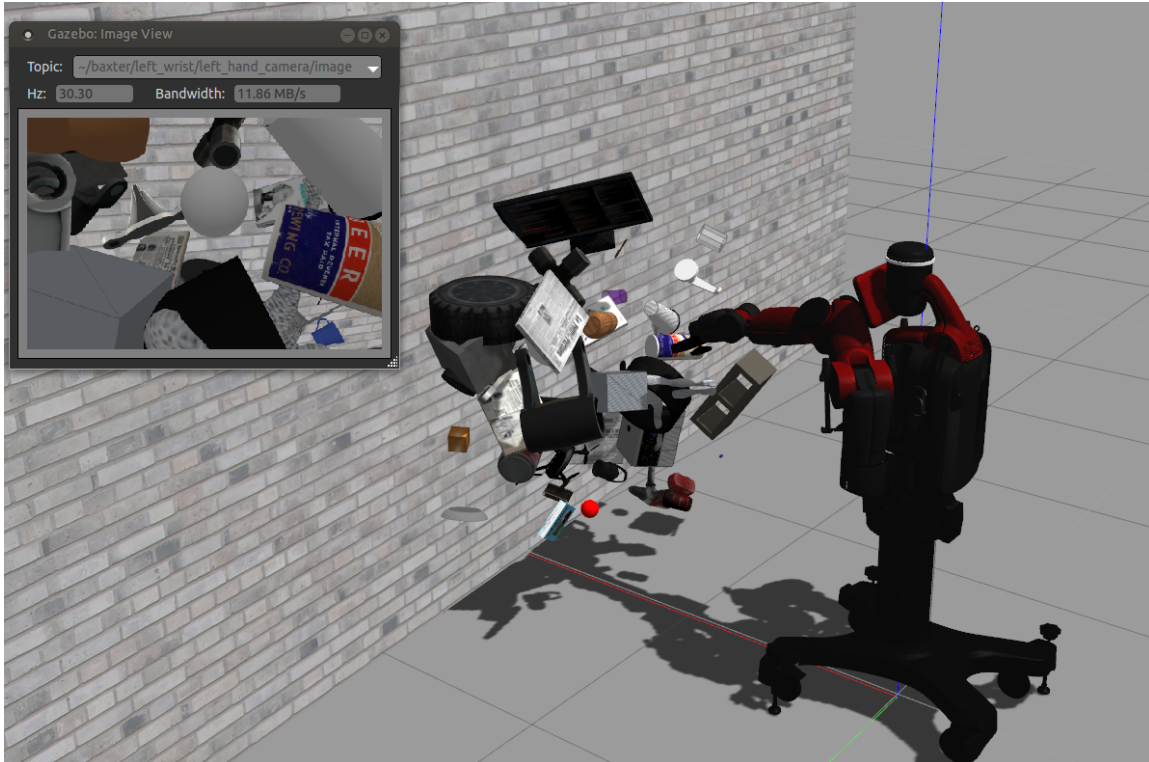


Figure A.6: Examples of generated scenes for the dataset. The insets show the images captured by the eye-in-hand camera.

Finally, the proposed resulting dataset comprises 10,000 such elements (for 5,000 scenes with objects on a table and 5,000 in the workspace).

A.4 Experimental tests

The depthS architecture, the depth C and two variations of the depthCSx: depthCS and depthCSS are considered for the experiments. These networks are trained with the generated dataset. In one epoch of training and validation set, the networks were first trained on the 8,000 scenarios in the training set, and the learning was evaluated on the 2,000 scenarios in the validation set. The best weights in the validation were saved for inference. For the inference, the networks tried to predict depth maps on additional 600 unseen scenarios..

To circumvent overfitting and learn a depth map efficiently, the Adam optimiser [Kingma and Ba \(2014\)](#), and regularisation techniques were applied. A stochastic gradient descent method (SGD) was first tested, but the Adam optimiser converged faster.

As for regularisation in the training process, two techniques were used: L2 reg-

ularisation gives a penalty on a loss function with the coefficient of a sum of squared weights. The selected value for L2 regularisation used for training is 10^{-4} . The second technique is dropout [Srivastava et al. \(2014\)](#). Moreover, a normalisation technique is also used. In particular group normalisation [Wu and He \(2018\)](#). The size of the group is configured as 16.

To evaluate the results, several evaluation metrics are computed ([Zhou et al., 2017](#)) ([Xu et al., 2018](#)) ([Ummenhofer et al., 2017](#)):

- L1-rel calculates a depth error relative to the ground truth.
- L1-inv can relatively increase if there is a large error for small values of depth.
- RMSE, It is the employed evaluation metric in chapter 6
- Finally sc-inv is a scale-invariant error introduced ([Engel et al., 2014](#)).

A.5 Experimental results

The training curves of the considered neural networks can be seen in figure [A.7](#). DepthC outperforms depthS in both training and validation scores. Also, it can be observed that the training and validation scores converge with a minimal oscillation. However, the validation scores did not decrease well after around 15 epochs. The results evaluated with the proposed metrics are shown in table [A.1](#).

Beyond the interest of the proposed approach for our particular application domain, the results in Table [A.1](#) support the hypothesis that integrating prior knowledge and sensor models relative to active robotic vision into deep learning can significantly improve performance. The three neural networks have been trained with the same number of epochs.

Also, leveraging the camera model in depthS allows us to define the absolute units of the obtained depth images (Figure [A.3](#)); a clear advantage over alternative

Table A.1: Results for error metrics

Network	Error metrics			
	RMSE (m)	L1-rel	L1-inv	SC-inv
depthS	0.1173	0.1650	3.1450	0.1767
depthC	0.0856	0.1219	0.9603	0.1240
depthCS	0.0766	0.1089	0.7879	0.1183
depthCSS	0.0732	0.1050	0.7649	0.1119

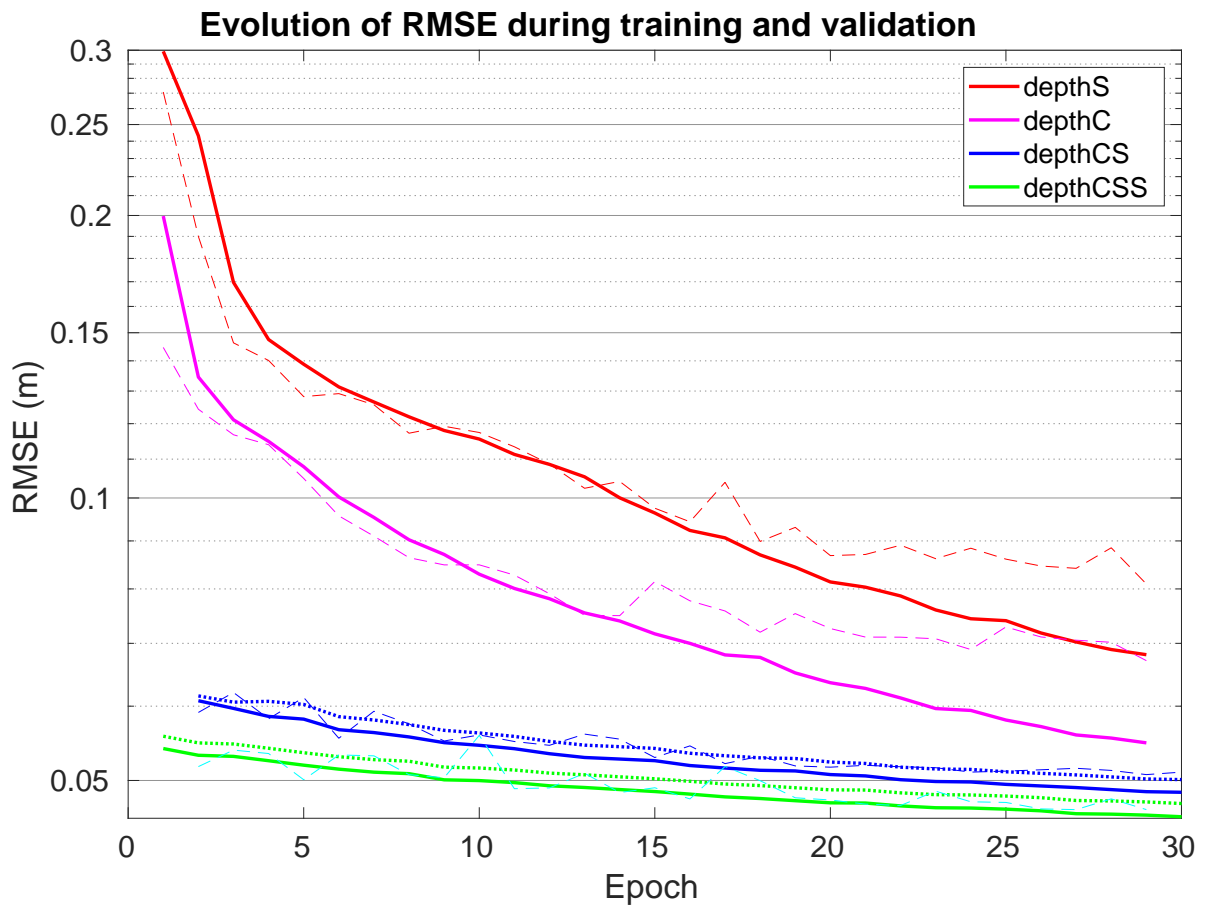


Figure A.7: Evolution of RMSE for the proposed neural networks during training and validation. Dashed lines correspond to validation, solid lines to training, and dotted lines represent the sum of the weighted loss functions.

deep learning methods, for which a scale factor is needed to estimate the real distance. For depthS and depthC, the training parameters and procedures are similar. However, the incorporation in the depthC design of the depth estimation model from optical flow and the displacements of the camera improves its performance concerning depthS around 25% (as measured with RMSE). Embedding prior knowledge in depthC from two previously established rules for the estimation of optical flow from two images and the correlation of this magnitude with the variation of the camera position allowed us to define the correlation rules between these elements. The correlation layers extract features of each input before the layer and link them patch by patch in the layer.

One representative example of the depth maps predicted by DepthS, DepthC, DepthCS and DepthCSS is displayed in figure A.8 along with the ground truth and the target image. The monocular camera captured the target image at a given moment. The ground truth was provided by a simulated depth camera located at the same coordinates as the eye-in-hand camera. The depth maps are coloured to visualise the distance from the camera.

A.6 Conclusions

This result is the first approach as the conclusions drawn from the model proposed in chapter 6 for estimating the depth of a scene by a robotic system through the execution of fixation movements can be indirectly applied to the design of deep neural networks performing the same function. Moreover, as can be seen when all the developed concepts are considered: two images and the camera position variation of the images (depthS), the optical flow estimation (depthC) and the iteration in the design of the neural networks (depthCS and depthCSS), an evident improvement in the results is obtained.

The advantage of using this type of network instead of the algorithm proposed in chapter 6 is that a trained network only requires the knowledge of two images and the camera variation to estimate the depth. However, the main limitation that did not occur in that algorithm is implementing the training in a real system, where capturing the depth image of the scene from the same point of view as the RGB camera is problematic in a robotic system like Baxter. Therefore, the background to feed the network is challenging to obtain, and the effort to create hundreds of different scenarios to have sufficient variability in the samples is vast.

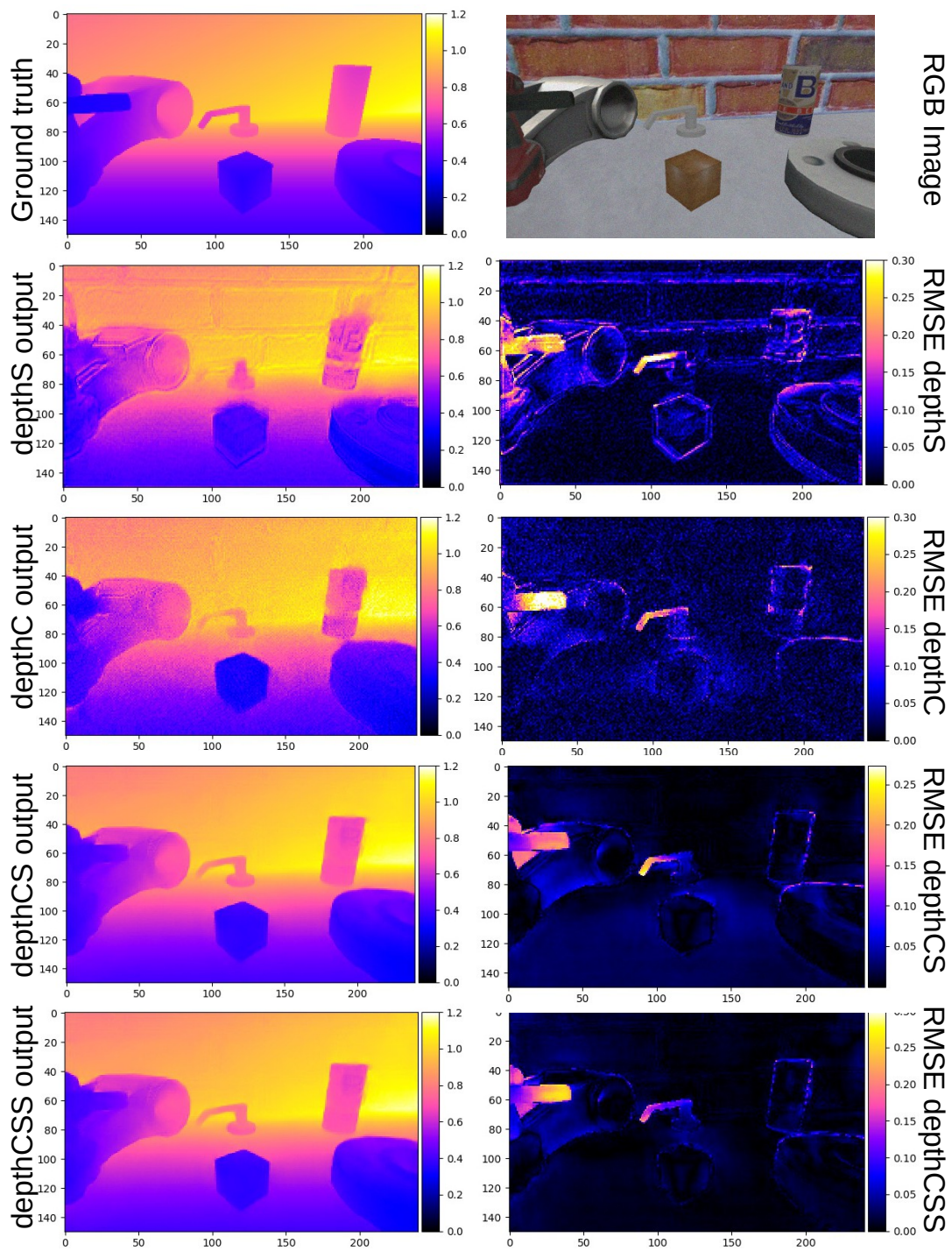


Figure A.8: Representative example of depth maps predicted by depthC, depthS, depthCS and depthCSS. The ground truth and RGB image are also shown. The units of the color bars on the right are meters. Closer distance is colored in blue and farther distance in red and yellow. The RMSE value for each pixel is shown in the images in the right column.

A.7 Publications supporting this appendix

- **Duran, A.J., del Pobil, A.P.**, 2019, "Improving robot visual skills by means of a bio-inspired model", in *Proc. 9th Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics (ICDL-EPIROB 2019)*, Oslo, Norway, pp. 25-30. DOI: 10.1109/DEVLRN.2019.8850712.
- **Duran, A.J., del Pobil, A.P.**, 2021, "Robot depth estimation inspired by fixational movements", *IEEE Transactions on Cognitive and Developmental Systems*. (in press, on line). Doi: 10.1109/TCDS.2020.3025057.
- **Yoneyama, R., Duran, A.J., del Pobil, A.P.**, 2021, "Integrating Sensor Models in Deep Learning Boosts Performance: Application to Monocular Depth Estimation in Warehouse Automation", *Sensors*, Vol. 21, No. 4, 1437. DOI: 10.3390/s21041437

Appendix B

Software architecture to implement the developed algorithms

B.1 Introduction

Several algorithms have been presented to control saccadic movements and determine the depth of a scene from fixation movements in this work. In both cases, a validation on real robotic systems (sections) has been performed. In the case of the simulations, Matlab was employed for their implementation. However, to deploy the algorithms in real robotic systems, ROS (Robot Operating System) ([Quigley et al., 2009](#)) was used instead.

ROS can be considered a communications middleware enabling communication between processes running on the same or remote machines. In addition to this functionality, a broad set of supervisory utilities allow monitoring the communication between processes and facilitating the programming of these processes. Among all the possible middlewares developed for robotics, it seems that ROS has become the standard for software development in this field. ([Elkady and Sobh, 2012](#)).

For ROS to operate as a communications Middleware, it is necessary to define a computer on which an application (roscore) is launched to control all the connections between the different machines configuring the ROS network. The different processes (which in ROS are called nodes) running on different machines connected to the roscore when they are launched to identify themselves and register the port and IP address identifying them on the network (figure [B.1](#)).

The underlying communication protocol is TCP/IP. However, how messages are

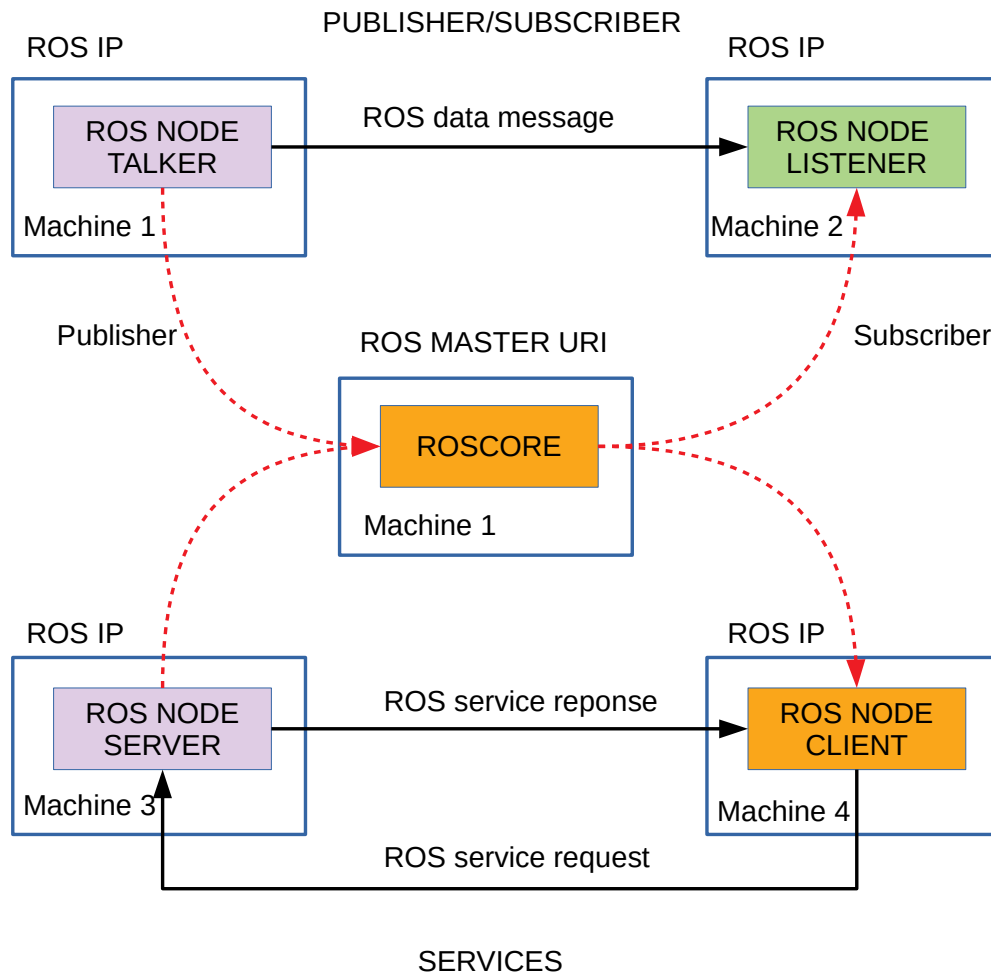


Figure B.1: ROS network connections schema

sent between nodes lead to two forms of transmission schemes:

- **Publisher/subscriber transmission:** This system is based on the producer/consumer model, in which the producer (publisher) posts messages on the network which may be read by more than one consumer (subscriber). A node (publishers) advertise its IP and port through the roscore, which assigns it a name (topic) where it writes messages if any other node (subscriber) is connected to it in reading form. In figure B.1, the talker node on machine one can send messages to the listener node on machine 2. This transmission is done directly once the roscore has informed both of the transmission's source and destination IP and port.
- **Server/client transmission:** This system is based on the client/server communication model. One of the nodes acts as a server, where a client sends requests. The server performs operations and prepares a response that it sends back to the client. Unlike the previous model, the communication is completed here until there is a new request from the client.

In addition to being based on ROS, the developed architecture requires some features such as:

- It should be modular to reduce the impact of changes in one part of the system (module) and make more accessible partial tests of the system. However, Considering this could present several drawbacks as the communication system overload or the difficulty to synchronize tasks.
- **Distributed processing.** This feature allows taking the computation out of the robot, which results in a higher computational capacity. A vivid example is the utilization of the Raspberry Pi with ROS in the robotic system developed in chapter 6 for image capture and estimation of the Stewart platform variation independent of the computer where the rest of the architecture is running, although it is integrated into it. The main disadvantage of this is that the slowest system or process determines the system's speed.

In this appendix, the software architecture used in implementing the different algorithms in this work is described along with other cases such as the Amazon Robotic Challenge 2015 and 2017. it was in these two competitions that this architecture was really pushed to the limit.

B.2 Software architecture description

The developed architecture is based on the module concept. A module is more than a ROS node or a library of functions. The main goal for using this type of modular architecture is to facilitate the decomposition of complex tasks in several small tasks, which are solved independently by each module, in such a way the architecture allows the coordination in the execution of these simple tasks and results in more or less complex behaviour.

A module is a ROS node fitting the black-box model, with a specific structure, not only in the code but also in the parameters and launchers, that allows the standardization and processing of inputs coming from the environment or other modules and generates outputs to other modules or actuators (figure B.2). The inputs and outputs could be: control I/O streams or data I/O streams. The processes and algorithms inside a module are indifferent to the rest of the system. However, module inputs and outputs must be well defined. These are the interface with the rest of the system.

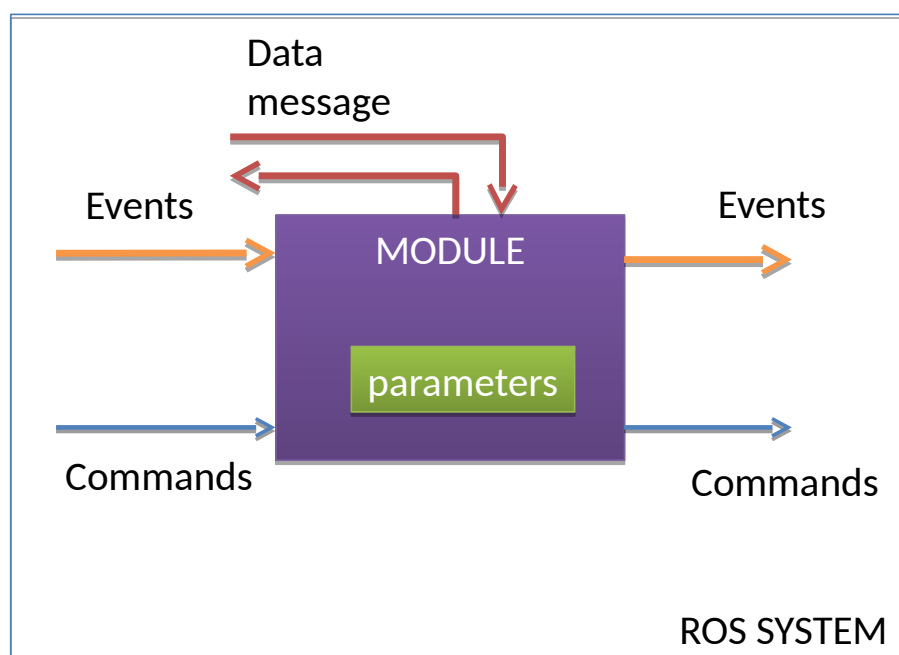


Figure B.2: Module description in software architecture

The control I/O streams in a module could be of two classes: commands or events. A module can receive commands and events. These could affect the process that is executing in the module, changing its internal state. In turn, a module can emit events or send commands to other modules in response to the process it is executing or requesting information from other modules. In essence, the commands that the module

can process or the events it can emit are part of the description and definition of the module and affect the rest of the system since they define the communication interface of the control signals between the modules.

Non-control data needed by the module to execute the process for which it is responsible are received or sent through the ROS standard publisher/subscriber communication. A module could be launched independently of the rest of the system. Therefore it can be tested simulating its inputs, outputs commands and events.

B.2.1 Module commands

A module command is a remote function call, where the function parameters have a fixed type, and the number of them is limited. The data structure of a module command is an identifier of the string type and three vectors of integers, float and string values.

The idea is that this data structure covers most of the possible data or parameters that can be passed to the remote function. The existence of the structure does not imply it is necessary to use it entirely, except in the case of the identifier that always has to be defined since it names the command.

For example, sending an integer and two floats to another module, together with two tags that identify them, is possible. In addition, if it is desired to send more complicated data structures, it is possible to encode them in text strings using the JSON format and send them through the string vector. The response of the remote function to the command executed on another module different from the one calling has the same data structure, i.e. an identifier and three vectors. It is a remote function call.

The remote call executed by the command can be of two types, blocking or non-blocking, so that in the first case, the execution of the calling module is suspended until a response is received or a timeout is exceeded. In the second case, the sending module only receives confirmation of the reception of the command by the receiving module and does not suspend its execution. In figure [B.3](#) you can see how a command works to control action between two system modules, in this case, a module A that is in charge of planning a gripping task and a module B that controls how much a gripper is opened or closed. Module A sends the command with an identifier that must be accepted by module B and an integer parameter. When the command is sent, module B suspends its execution while waiting for module B to process the function assigned to it by module A. When the B process is finished, it sends the response to A, continuing its execution.

Two default commands have to be implemented for all modules:

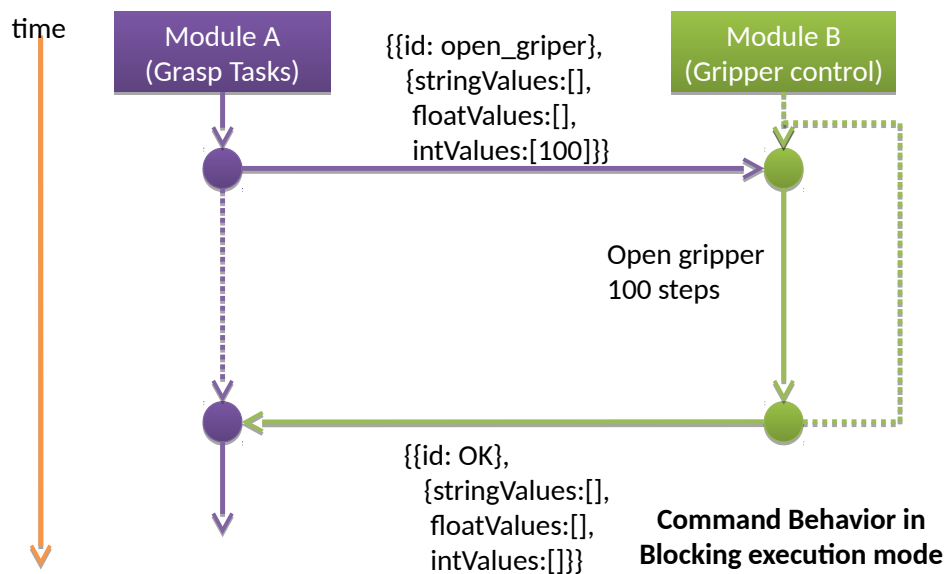


Figure B.3: Time sequencing example of sending a command in blocking execution mode between two modules

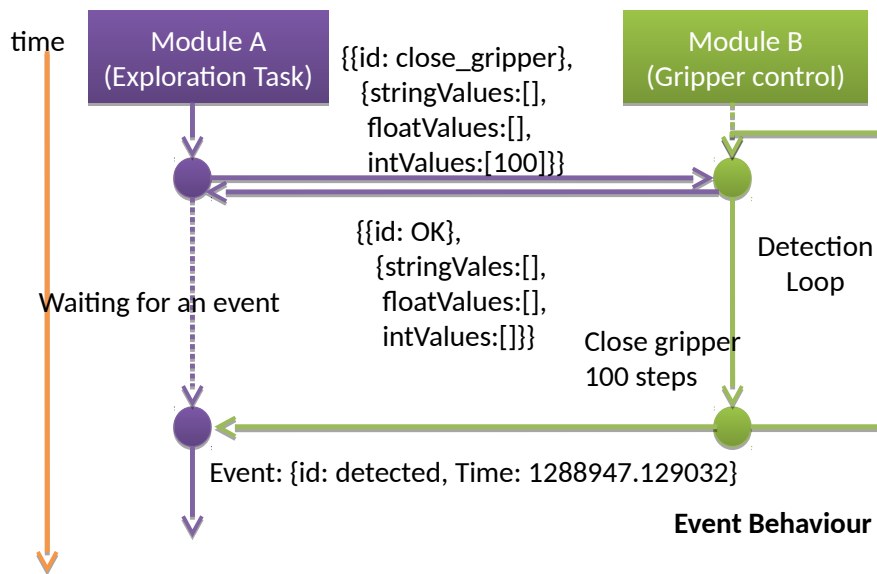
- **Activation command:** This command enables the module functionality. Even though modules can be launched in active mode by default, they can be sent an activation command after deactivation.
- **Deactivation command:** This command suspends the module functionality without closing the module

Therefore, a module always must have at least two functions to handle these commands from other modules.

B.2.2 Module events

A module event is a change state signal. It is implemented as a string that identifies the event and a timestamp. An event is generated by a module when an internal condition is accomplished, and it is emitted. It is broadcast as many times as the emitter module wants. In turn, every module can capture events sent by other modules. The events are stored in a priority queue sorted by timestamp in order of arrival. It is possible to clear this queue when the module handling the event needs to do it. In essence, an event is a kind of interrupt signal for the receiver module.

As can be seen in figure B.4, the combination of commands and events enables the execution of tasks in parallel, and the events act as interrupt signals to synchronize



1

Figure B.4: Time sequencing example of command and event combination

them. Thus, a module can have the following behaviours that allow the synchronization or parallel execution of two processes on the same machine or remote machines:

- A module can wait for an event to arrive before resuming its execution. In this state, the module can receive information but not process it.
- A module can suspend its execution until several events from different modules arrive. This procedure allows synchronizing different execution lines launched in a parallel way.
- The module may be executing its algorithm, and the receipt of an event changes the algorithm's state in such a way that the execution thread is changed.

B.2.3 Module parameters

A module parameter is a variable that can modify the execution of the algorithm or process. However, the module parameters must be defined before the execution of the module. It is necessary to use a command to modify a parameter at run time. Each parameter should have a brief description to make easier debugging tasks.

- The loop thread: The server thread is created by sharing the parameter workspace with it. It executes an algorithm in a loop, processing the data messages and generating events. It can send commands from this thread to other modules. However, it is not recommended that they are in blocking mode.

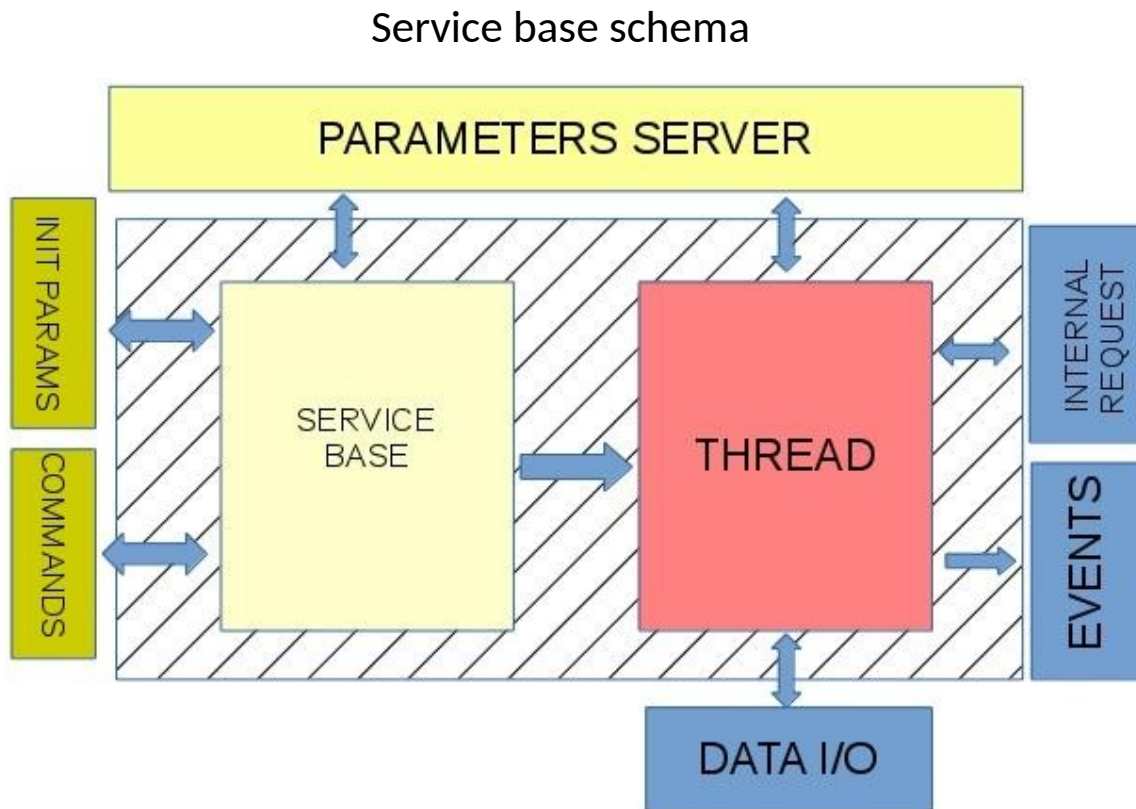


Figure B.6: Implementation schema of the server base module

The service base modules can be considered processing modules in the system architecture. The executing algorithm in the loop thread is processed at a fixed frame rate. This loop can suspend its execution or re-start using deactivation and activation commands, respectively. The service base module is launched in activation mode by default.

These kinds of modules are helpful for processing data streams, such as hardware data streaming. If any, the module's output would be at the same frequency at which the algorithm is executed. As indicated in appendix B.2.1, the commands can modify the behaviour of the algorithm in the loop thread.

When a service base module is launched in the system, immediately a ROS node is created. In this way, it is possible to parse the configuration parameters defined in

the module and create a memory space to be shared by the main thread and the loop thread (figure B.7)

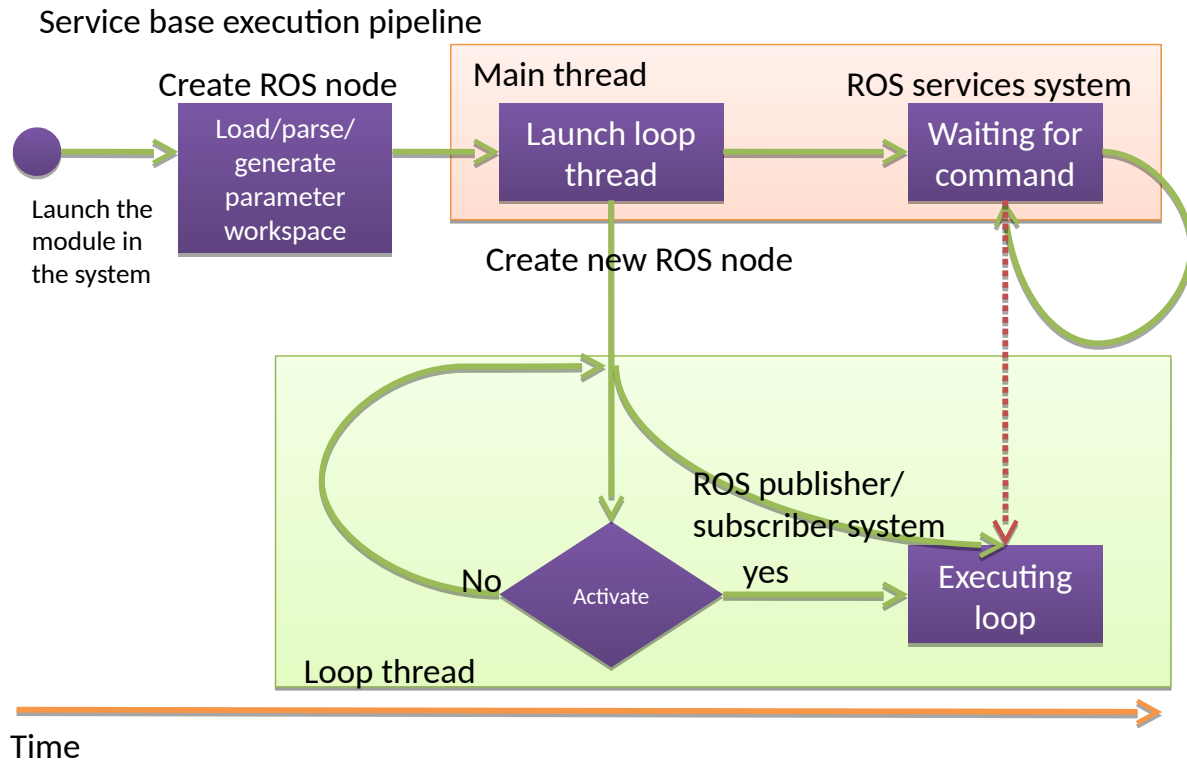


Figure B.7: Service base module execution pipeline

The main thread then creates and launches the thread loop, which in turn generates another ROS node. Finally, using the ROS server/client system, the main thread is kept waiting to receive commands, which, when they arrive, execute a callback in the thread loop.

In turn, in the thread loop, if the activated flag is enabled, a function is executed with a constant frequency. Inside this function, it is possible to write the code that processes the messages and publishes the replies using the ROS publisher/subscriber system.

B.3.2 Action primitive module

The elements that compose an action primitive module are the same as in the case of the service base described in figure B.7. The difference is in the execution pipeline. While the service base creates a loop thread and maintains it until the module is shut down, in the case of the action primitive module, the created thread is executed only once, and when it is finished, the main thread is maintained so that it can be executed

as many times as desired.

A state machine system is integrated into the action primitive modules, so when the secondary thread is created and executed, the execution of this finite state machine is started.

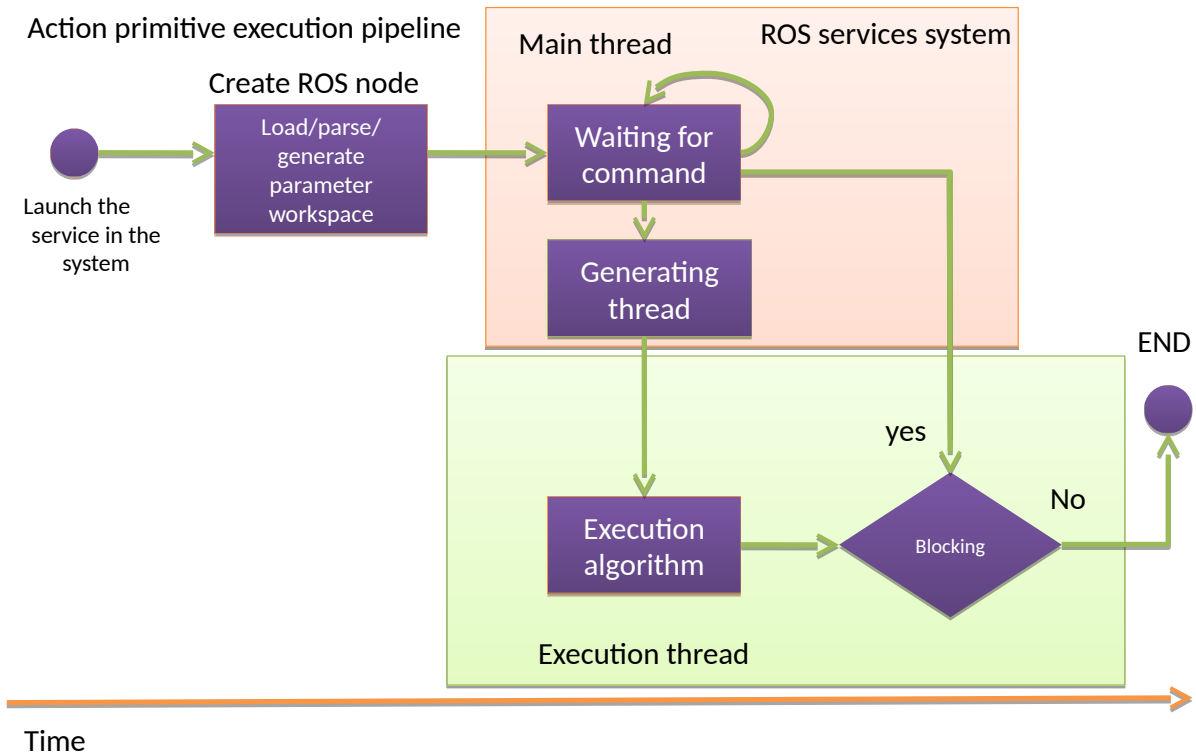


Figure B.8: Action primitive module execution pipeline

A feature to keep in mind about action primitive modules is that the thread (or the associated state machine) can be created and launched in blocking mode so that the executing command blocks the response to the requesting module. In non-blocking mode, the command handler returns a confirmation that the thread has been successfully created but does not block the response to the module that requested the execution.

The action primitive can be considered as a control program module in the system architecture. It is like executing a program on demand. While the execution thread is active, the arriving commands can not be handled (except the interrupt command). So, commands could be used to configure thread execution. There is an option to interrupt the execution thread defining the interruptions points.

In figure B.9, it can be seen an example of a state machine defined in a module whose function is to scan the bins of a shelf to find a certain number of objects and compute the approach vectors for the execution of the grasps. This example is a real

case used in the Amazon Robotic Challenge in 2015, where this architecture was successfully used to integrate all the operations performed by a Baxter robot to pick up objects from a shelf in a warehouse.

B.3.3 Module combination: generating the architecture

Combining the service base and action primitive modules is necessary to execute a certain task or establish a certain behaviour in a robotic system based on this architecture. Each one is in charge of solving small parts of the problem enabling complex behaviour to emerge.

Usually, as shown in figure [B.10](#), the module architecture can be organized in the form of two layers: the services layer and the primitives' layer. Typically, the sensors and actuators of robotic systems emit or receive constant streams of information that can be handled and processed by base services. These service based modules modify their behaviour according to the commands they receive from the upper layer composed of action primitive modules. The services can sample or capture information and send it to the upper layer employing events or in response to commands.

These command responses or events cause state changes in the active state machines in the upstream layers, generating new commands and causing an advance in the desired behaviour.

Exploration action primitive module

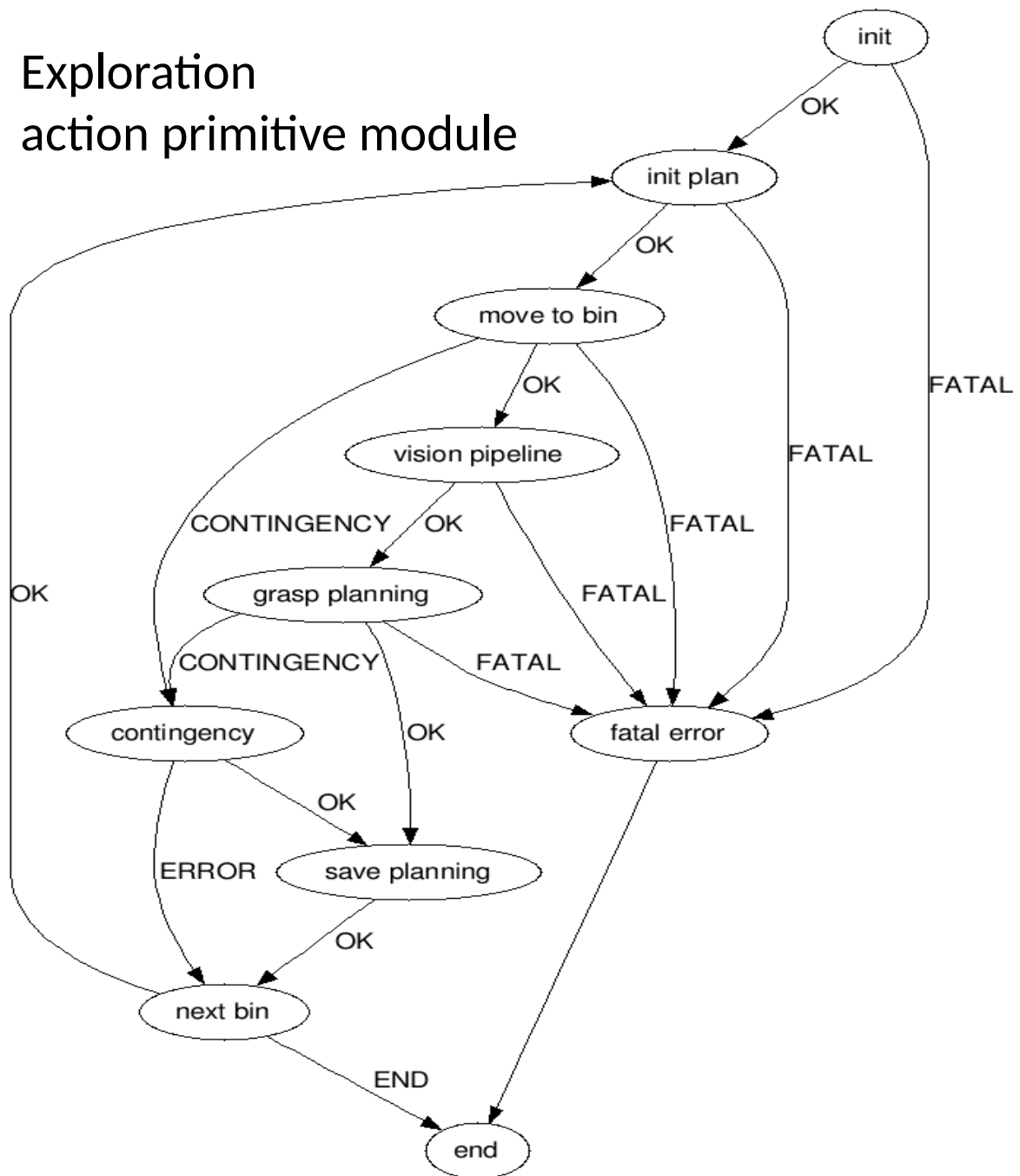


Figure B.9: Finite state machine defined in an exploration action primitive used in the Amazon Robotic Challenge in 2015

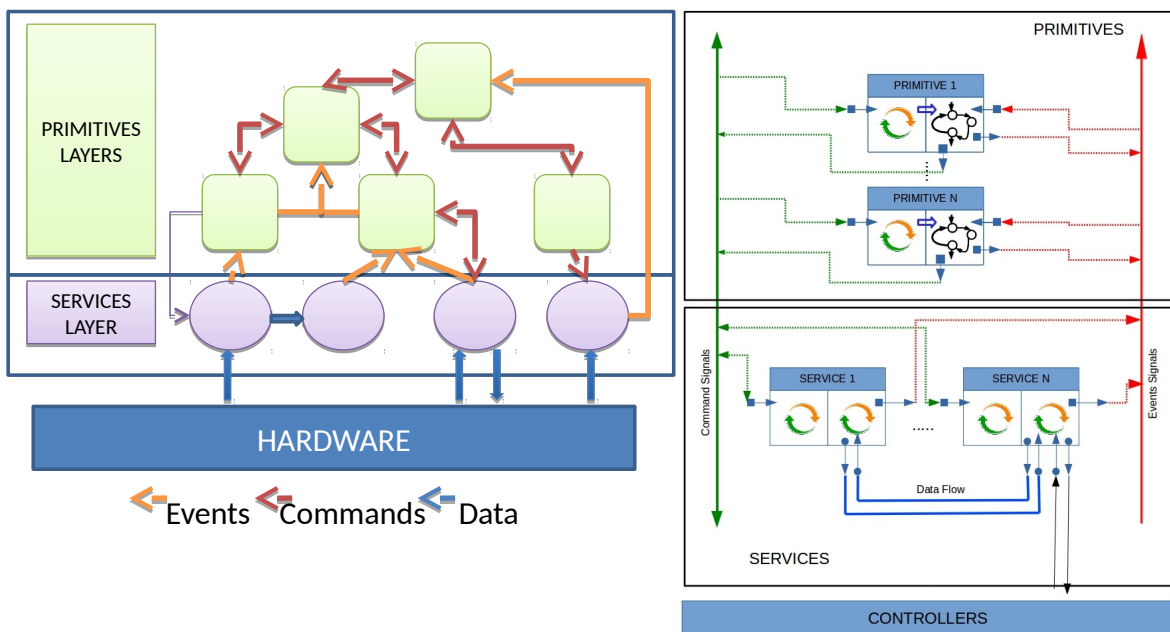


Figure B.10: Schema of the proposal combination of service base and action primitive modules

B.4 Conclusions

This architecture was developed to initially implement the saccadic controllers described in chapter 4. In addition, it allowed unifying criteria when programming ROS nodes, which facilitated the development.

Subsequently, it has been successfully used in two large projects involving a wide range of programmers, such as the Amazon Robotic Challenge in 2015 and 2017.

Finally, it has been implemented in the robotic system developed in this work to replicate the fixation movements of humans (chapter 6). As can be seen in figure B.11, to execute this algorithm, it is necessary to deploy several modules controlling each of the parts of the process, having as coordinator an action primitive module (fixation process in figure B.11) executing a state machine as shown in figure B.12.

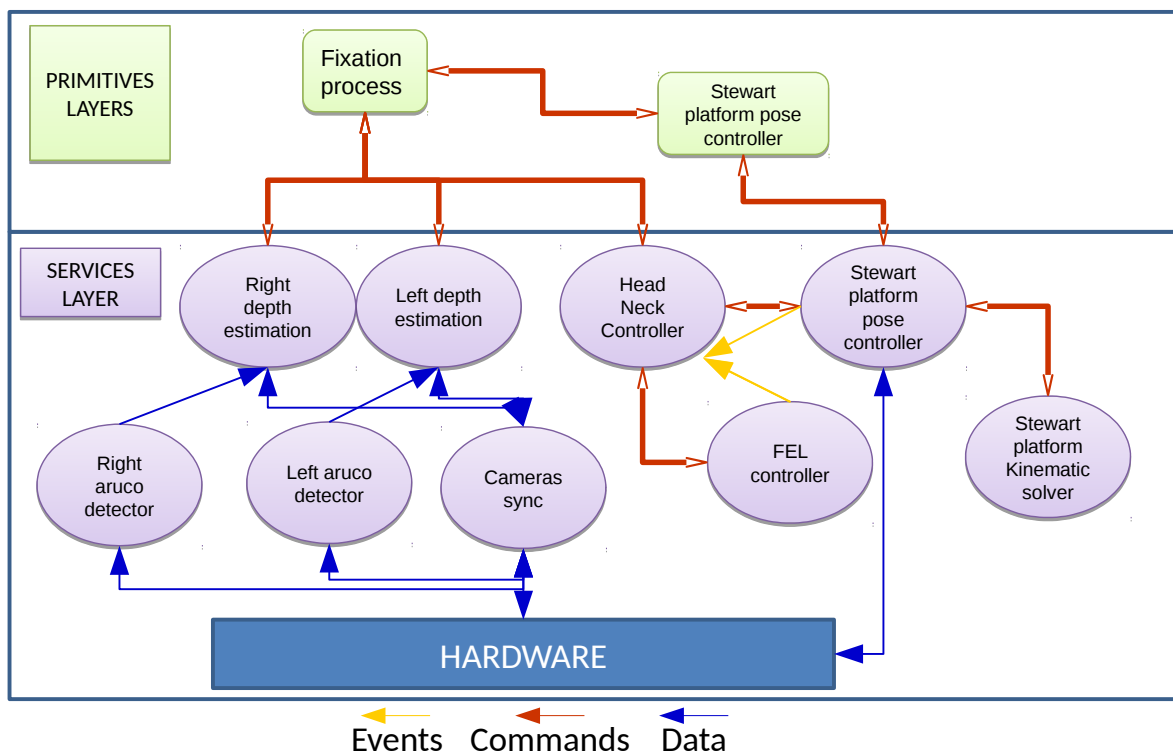


Figure B.11: Schema of the developed software architecture for depth estimation based on fixation process (chapter 6)

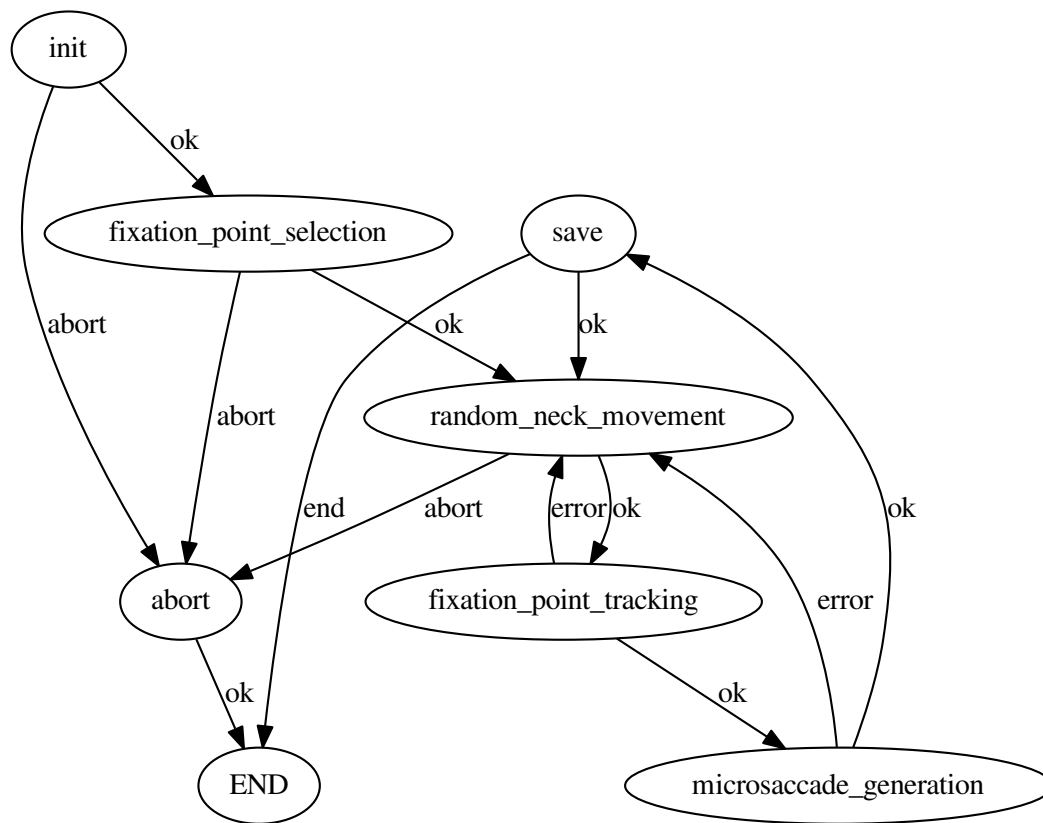


Figure B.12: Finite state machine integrated in action primitive module for depth estimation based on fixation process

B.5 Publications supporting this appendix

- **del Pobil, A.P., Duran, A.J., Antonelli, M., Felip, J. Morales, A., Prats, M., Chinellato, E., 2013,** "Integration of Visuomotor Learning, Cognitive Grasping and Sensor-Based Physical Interaction in the UJI Humanoid Torso", in *Designing Intelligent Robots: Reintegrating AI*, edited by B. Boots et al., AAAI Press, Palo Alto, California, pp. 6-11. ISBN: 978-1-57735-601-1
- **Felip, J., Duran, A.J., Antonelli, A., Morales, A., del Pobil, A.P., 2015,** "Tombatossals: A humanoid torso for autonomous sensor-based tasks", in *Proc. 15th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2015)*, Seoul, Korea. pp. 475-481, DOI: 10.1109/HUMANOIDS.2015.7363592
- **del Pobil, A.P., M. Kassawat, A. J. Duran, M.A. Arias, N. Nechyporenko, A.**

Mallick, E. Cervera, D. Subedi, I. Vasilev, D. Cardin, E. Sansebastiano, E. Martinez-Martin, A. Morales, G.A. Casañ, A. Arenal, B. Goriatcheff, C. Rubert and G. Recatala, 2017, "UJI RobInLab's Approach to the Amazon Robotics Challenge 2017", in *Proc. IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI 2017)*, Daegu, Korea. pp. 318-323.

Appendix C

Mathematical fundamentals

C.1 Introduction

This appendix aims to define and develop several practical mathematical concepts used in various parts of this work. The starting point is Bayesian learning which allows the development of probabilistic concepts that will lead to the introduction of the Gaussian distribution and its properties.

C.2 Mathematical notation of scalars, vectors and matrices

In order to make it easier to follow and reproduce the algorithms presented in this work, the same notation has been kept to identify the essential elements of algebra, i.e. scalar values, vectors, matrices and tensors. In this section, an example of each of them is proposed in order to identify their notation.

- **Scalars:** Scalars are single numbers and are represented by lowercase Latin or Greek letters in italics. For example: $(x_1, x_2, \dots, x_n) \mid x_i \in \mathbb{R}$.
- **Vectors:** Vectors are ordered arrays of single numbers. If all of the scalars in a vector are real-valued then the notation states that the (boldface lowercase) vector value is a member of the n -dimensional vector space of real numbers, \mathbb{R}^n . To explicitly identify the components of a vector, the scalar notation is used. By default, in this work, a vector is assumed to be a set of column-ordered scalar and

therefore these notations are equivalent:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \{x_1, x_2, \dots, x_n\}^T$$

- **Matrices:** are rectangular arrays consisting of numbers. if m and n are positive, that is $m, n \in \mathbb{N}$, then an $m \times n$ matrix contains mn numbers, with m rows and n columns. A matrix is denoted with uppercase, boldface Latin and Greek letters: $A \in \mathbb{R}^{m \times n}$. Each component of A is identified by $a_{i,j}$.

This is the general notation that has been followed throughout the work; in some cases, it has not been maintained or has been slightly changed to avoid confusion with elements not described in it. In these cases, each element has been explicitly defined.

C.3 Probability

Probability is an attempt to measure uncertainty. Often, situations of uncertainty arise when random experiments or phenomena are performed. The sample space Ω is the set of all the possible results of the experiment. The elements of Ω are called elementary events. An event S is any subset of the sample space. Two events S_1 and S_2 are incompatible if they do not have any element in common. σ is defined as the set of all possible events that can occur, and algebraic operations can be used between these sets. P is said to be a probability function over (Ω, σ) if it meets the Kolmogorov's axioms:

1. $P(S) \geq 0; \forall S$. The value of the probability of an event is always positive.
2. $P(\Omega) = 1$.
3. if S_1, S_2, \dots, S_n are incompatible events:

$$P(S_1 \cup S_2 \cup \dots \cup S_n) = \sum_{i=1}^n P(S_i) \quad (\text{C.1})$$

C.4 Probability Rules

C.4.1 Conditional Probability

Let us think of a random experiment and an event of such an experiment, S_1 . Initially, a degree of belief or probability is associated with this event $P(S_1)$. If previous knowledge about that experiment had been given, for example, another event S_2 occurred, the information about the test result changes our degree of belief in S_1 . This new value associated to the event A , $P(S_1|S_2)$ is defined as conditional probability. The conditional probability of event S_1 , given that another event S_2 has already occurred, denoted by $P(S_1|S_2)$, is given by the ratio:

$$P(S_1|S_2) = \frac{P(S_1 \cap S_2)}{P(S_2)} \quad (\text{C.2})$$

Two events S_1 and S_2 are independent if $P(S_1|S_2) = P(S_1)$ or $P(S_2|S_1) = P(S_2)$ or equivalently $P(A \cap B) = P(A)P(B)$.

C.4.2 Total Probability Rule

When numerous events are involved in an experiment, calculating the probabilities of all of them increases its complexity excessively. Fortunately, the Total Probability Rule allows the problem to be broken up into partial calculations. Therefore, let P be a probability function in a sample space. Let $\{S_1, S_2, \dots, S_n \subset F\}$ be a partition of the sample space and let S^* be any event. Then:

$$P(S^*) = P(S^*|S_1)P(S_1) + \dots + P(S^*|S_n)P(S_n) = \sum_{i=1}^n P(S^*|S_i)P(S_i) \quad (\text{C.3})$$

C.4.3 Bayes' Theorem

Assume an event S_i ; it is uncertain whether it has occurred. We desire to estimate the probability of this event $P(S_i)$. Therefore, $P(S_i)$ does not represent the probability that S_i occurs but the degree of belief that S_i has occurred. In principle, there may be no data to know the exact value of the probability of S_i . Even so, an estimation of that probability can be given because the context where the event takes place is partially known, and there is "expert knowledge" to approximate it. This initial value $P(S_i)$ is called a *a priori* probability. Let us assume that there is now new event information S_i through event S^* . In this case, the probability of S_i should be updated based on this

new information and providing a new probability of S_i that takes into account S^* , that is, $P(S_i|S^*)$. This probability is known as a *posteriori* probability.

This probability update is performed through Bayes' theorem, which in its general form is given by the following expression:

$$P(S_i|S^*) = \frac{P(S^*|S_i)P(S_i)}{P(S^*)} \quad (\text{C.4})$$

Bayes' theorem is a way of reversing the conditional probabilities. In combination with Total Probability Rule, the equation (C.4), can be written as:

$$P(S_i|S^*) = \frac{P(S^*|S_i)P(S_i)}{\sum_{i=1}^n P(S^*|S_i)P(S_i)} \quad (\text{C.5})$$

C.4.4 Chain Rule

Given two events S_i and S^* , the probability of both occurring at the same time is expressed by the chain rule:

$$P(S_i \cap S^*) = P(S_i, S^*) = P(S^*|S_i)P(S_i) = P(S_i|S^*)P(S^*) \quad (\text{C.6})$$

The chain rule relates joint probability, conditional probability and marginal probability.

C.5 Inference and Bayesian learning

When a phenomenon is observed, a series of hypotheses are established from these observations. A probability value can be associated with each hypothesis. As more information is acquired about the phenomenon, the hypotheses that describe its behaviour could change.

These initial values come from speculation. For example, if we wish to learn how to detect a pedestrian in a street image, one hypothesis could be whether a specific region is a pedestrian or not (h). When the first image is received, all the areas defined could have the same probability of being a pedestrian $P(h)$. However, if it is somehow possible to identify one of these regions as a possible pedestrian (d) with a probability $P(d|h)$ that the established hypothesis is fulfilled, the features that define this region should condition future hypotheses. Therefore, the initial hypothesis h has been modified by the hypothesis d , i.e. the a posteriori probability $P(h|d)$ can be calculated employing Bayes' theorem as follows:

$$P(h|d) = \frac{P(d|h)P(h)}{P(d)} \quad (\text{C.7})$$

The information increases with the addition of new hypotheses of what a pedestrian is, for example, with more images of pedestrians. The a posteriori probability of the initial hypothesis will be modified by all these new data following Bayes's theorem:

$$P(h|d) = \frac{P(d|h)P(h)}{P(d, h')} \quad (\text{C.8})$$

where $P(d, h')$ is the joint probability and represents the previous knowledge acquired about what is or is not a pedestrian. In a discrete way, this probability can be calculated using the expression:

$$P(d, h') = \sum_{h' \in H} P(d|h')P(h') \quad (\text{C.9})$$

H is the set of all previous hypotheses about what is or is not a pedestrian, and the size of H grows exponentially with each new hypothesis incorporated. The resolution of a problem to relate the new d hypothesis to all the elements from the initial set can be solved using combinatorics. For this reason, this method is not usually applied directly.

Bayesian inference uses a numerical estimator of the degree of belief in a hypothesis before having observed the $P(h)$ hypothesis and calculates the degree of belief in the hypothesis after having observed evidence.

C.6 Random variable and probability distribution models

A probabilistic experiment is considered in a sample space Ω where a probabilistic function has been defined $P(\cdot)$. A random variable is a function that assigns a value, usually numerical, to the result of that probabilistic experiment. $X : \Omega \rightarrow \mathbb{R}$. Depending on the numerical type of these values, the random variable can be discrete or continuous.

A discrete random variable X can take a countable set of discrete values (x). The mapping function between these values and their probability is called mass probabilistic function: $p(x) = P(X = x)$. For discrete variables, this mass function can be estimated experimentally from samples and using frequency calculations.

A random variable is continuous when its domain is a set of infinite and countless elements that can only be defined by intervals.

In continuous random variables as opposed to discrete ones, it is impossible to determine the mass function directly because intervals have an infinite number of elements. However, the histogram is a representation analogous to the empirical mass

function that allows us to approximate the probabilities of the values of a continuous variable.

Therefore, the probability density function of a continuous random variable is defined as the function $f(x)$ such that for any $a, b \in \mathbb{R}$ or $a, b = \pm\infty$:

$$P(a < X < b) = \int_a^b f(x)dx \quad (\text{C.10})$$

Probabilistic models estimate mass functions or density functions from several empirically calculated parameters, assuming that the random variable fits the model.

These model parameters are usually estimated from a set of samples. Therefore, given a set of data and the assumption that these data follow a particular model defined by a set of parameters, ultimately, these parameters must be learned to describe the model.

C.7 Likelihood function

Let x_1, x_2, \dots, x_n be a random sample of a random variable X with a probabilistic mass function p_θ (or with density function f_θ). For each particular sample (x_1, x_2, \dots, x_n) , the likelihood function $\mathcal{L}(\theta)$ is defined as the joint probability (or density) function of X evaluated in (x_1, x_2, \dots, x_n) , with θ denoting a specific parameter value in the parameter space Θ .

In the case of X being a discrete variable:

$$\mathcal{L}(\theta) = \mathcal{L}(X|\theta) = \mathcal{L}(x_1, x_2, \dots, x_n|\theta) = P_\theta(X = x_1, X = x_2, \dots, X = x_n) \quad (\text{C.11})$$

When X is a continuous random variable:

$$\mathcal{L}(\theta) = \mathcal{L}(X|\theta) = \mathcal{L}(x_1, x_2, \dots, x_n|\theta) = \mathcal{L}(f_\theta(x_1), f_\theta(x_2), \dots, f_\theta(x_n)) \quad (\text{C.12})$$

The notation $\mathcal{L}(\theta)$, indicates that it depends exclusively on the parameters of the model and not on the data (x_1, x_2, \dots, x_n) .

C.8 Maximum likelihood estimation

Let \hat{X} be a sample of random variable X that follows a likelihood function $\mathcal{L}(\theta)$ defined by a parameter value θ in the parameter space. For each particular sample

$\{x_1, x_2, \dots, x_n\} \in \hat{X}$, the maximum likelihood estimation (ML) is the value $\hat{\theta}_{MLE}$ that maximises the likelihood function:

$$\mathcal{L}_{ML}(x_1, \dots, x_n | \hat{\theta}_{MLE}) = \operatorname{argmax}_{\theta} \mathcal{L}(x_1, \dots, x_n | \theta) \quad (\text{C.13})$$

The specific value of θ that maximises the likelihood function is called maximum likelihood estimator (MLE), $\hat{\theta}_{MLE}(\hat{X})$.

Let $\theta = (\theta_1, \dots, \theta_k)$ denote a parameter value θ in a k -dimensional parameter space that is involved in defining the likelihood function of a random variable X , the procedure for computing the maximum likelihood estimator of θ given a particular sample (\hat{X}) is:

1. Write down the likelihood function: $\mathcal{L}(\theta) = \mathcal{L}(x_1, \dots, x_n | \theta)$.
2. Define the support or log-likelihood function: $l(\theta) = \ln \mathcal{L}(\theta)$.
3. Compute $\hat{\theta}_j$ so that:

$$\frac{\partial}{\partial \theta_j} l(\theta) = 0$$

4. Check that it really is a maximum, i.e:

$$\frac{\partial^2}{\partial \theta_j^2} l(\theta) |_{\theta_j = \hat{\theta}_j} = 0$$

In other words, from a set of data or samples, the values of the likelihood function parameters are being pursued so that the probability of generating these data is maximum.

C.9 Bayesian learning to estimate model parameters

To learn the parameters of a given model from a set of data that in theory fit that model, the procedure to be followed is based on Bayesian inference and is always the same. Given a set of observations $\mathbf{D} = \{\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_n\}$ that fits a model defined by the parameters θ :

1. Define the likelihood function: $\mathcal{L}(\mathbf{D} | \theta)$
2. Specify the a priori probability for the model parameters. A degree of uncertainty or belief must be defined about the initial values of the model parameters $P(\theta)$.

3. *Estimate the a posteriori probability.* Using Bayes' theorem, this probability could be computed as:

$$P(\theta|\mathbf{D}) = \frac{\mathcal{L}(\mathbf{D}|\theta)P(\theta)}{P(\mathbf{D})} \quad (\text{C.14})$$

However, taking into account that the term $P(\mathbf{D})$ is a normalisation factor that keeps the probability in the range 0-1, it is possible to write:

$$P(\theta|\mathbf{D}) \propto \mathcal{L}(\mathbf{D}|\theta)P(\theta) \quad (\text{C.15})$$

Let X_* be an unobserved value, the posterior predictive distribution of X_* is the conditional probability of X_* given the observed values in the training dataset: $P(X_*|\mathbf{D})$. The probability distribution of the estimated model parameters is conditioned by the observed values of the training dataset: $P(\theta|\mathbf{D})$. Therefore the posterior predictive distribution of X_* given \mathbf{D} is calculated by marginalizing the distribution of X_* given θ over the posterior distribution of θ given \mathbf{D} . In the case of a continuous random variable:

$$f(X_*|\mathbf{D}) = \int_{\theta} f(X_*|\theta, \mathbf{D})f(\theta|\mathbf{D})d\theta \quad (\text{C.16})$$

C.10 Covariance

Given two random variables, the covariance is the degree of relationship between them. Thus, the covariance, considering the concept of probabilistic independence, can measure the degree of relationship between two random events characterised by their random variables and, therefore, their dependence.

If a discrete random variable X with a finite number of outcomes x_1, x_2, \dots, x_n , with associated probabilities p_1, p_2, \dots, p_n , are considered, the expectation of X is defined as:

$$\mathbb{E}[X] = \sum_{i=1}^n x_i p_i \quad (\text{C.17})$$

In the continuous random variable case:

$$\mathbb{E}[X] = \int_{\mathbb{R}} x f(x) dx \quad (\text{C.18})$$

where $f(x)$ is the probability density function of the random variable X .

Under these conditions, given two random variables X_1 and X_2 , the covariance is defined as the expected value of the product of their difference between their values and their individual expectations:

$$\text{cov}(X_1, X_2) = \mathbb{E}[(X_1 - \mathbb{E}[X_1])(X_2 - \mathbb{E}[X_2])] \quad (\text{C.19})$$

An attractive property of the covariance is its relationship with the dot product because it has the same mathematical properties. This fact allows the interpretation of the covariance as a measure of the similarity between the variables considered.

C.11 Gaussian distribution

C.11.1 Definition

Given a random continuous variable X , with a probability density distribution described by equation (C.20), it is called Gaussian or Normal distribution.

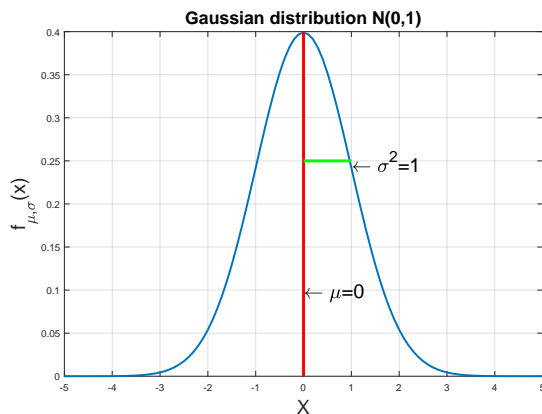


Figure C.1: Gaussian distribution for $\mu = 0.0$ and $\sigma = 1.0$.

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2} \quad (\text{C.20})$$

Where the model parameters (θ) are:

- μ : center of masses or mean.
- σ^2 : variance.

The shape of this function for the case where $\mu = 0$ and $\sigma = 1.0$, is shown in figure C.1. Many natural phenomena follow this probability distribution, generally all

kinds of physical measurements. The Gaussian curve is centred on the μ parameter, a scalar, because a univariate Gaussian is considered. By definition of probability density function, the area enclosed under the curve should be equal to one.

$$\frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{\infty} e^{-\frac{1}{2\sigma^2}(x-\mu)^2} = 1 \tag{C.21}$$

In order to schematise that the independent variable X follows a normal probability

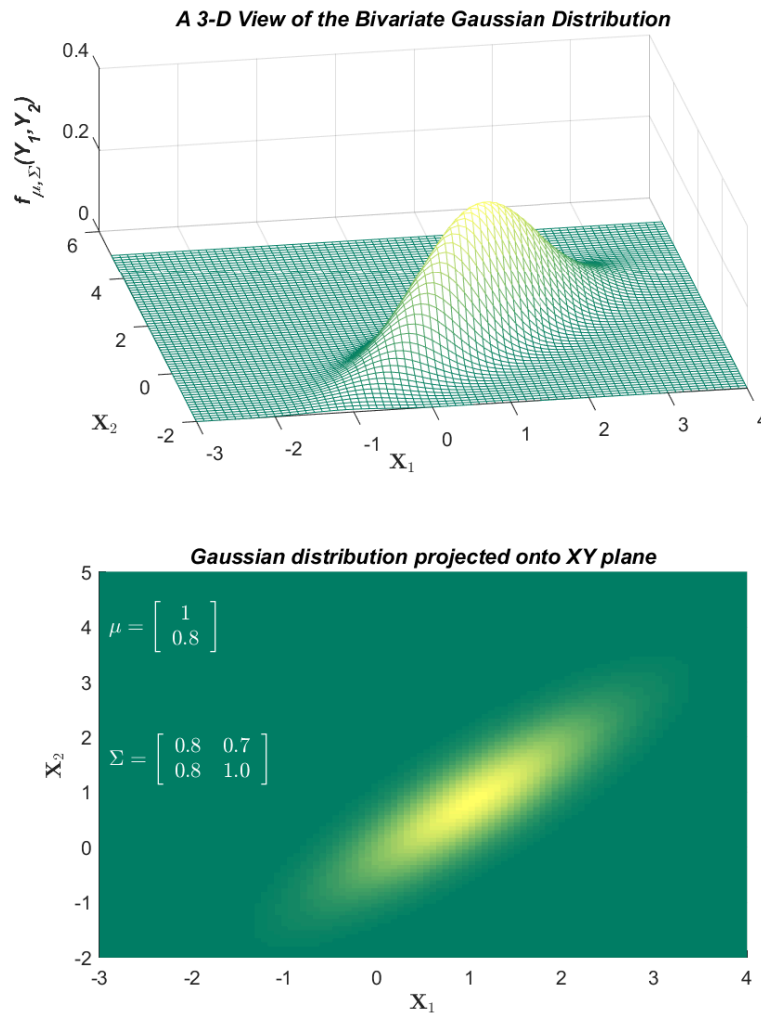


Figure C.2: Bivariate Gaussian distribution example.

distribution with parameters μ and σ^2 , the following notation is used:

$$X \sim \mathcal{N}(\mu, \sigma^2) \tag{C.22}$$

If more than one dimension is considered, e.g. $\mathbf{x} \in \mathbb{R}^n$, the probability function of an n-dimensional Gaussian distribution is given by this expression:

$$f(\mathbf{x}) = |2\pi\Sigma|^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1}(\mathbf{x}-\mu)} \tag{C.23}$$

Where:

$$\boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_n \end{pmatrix}; \boldsymbol{\Sigma} = \begin{pmatrix} \sigma_{1,1}^2 & \cdots & \sigma_{1,n}^2 \\ \vdots & \ddots & \vdots \\ \sigma_{n,1}^2 & \cdots & \sigma_{n,n}^2 \end{pmatrix} = \begin{pmatrix} \Sigma_{1,1} & \cdots & \Sigma_{1,n} \\ \vdots & \ddots & \vdots \\ \Sigma_{n,1} & \cdots & \Sigma_{n,n} \end{pmatrix} \quad (\text{C.24})$$

An example of bi-variate Gaussian distribution is shown in figure C.2. $\boldsymbol{\Sigma}$ is a Gaussian covariance matrix, according to equation (C.19) given a vector $\boldsymbol{x} \in \boldsymbol{X}$: $\boldsymbol{\Sigma} = \mathbb{E} [(\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^T]$. This matrix has several important properties:

1. $\boldsymbol{\Sigma}$ is symmetric
2. It is positive semi-definite, i.e. for any \boldsymbol{y} vector, $\boldsymbol{y}^T \boldsymbol{\Sigma} \boldsymbol{y}$ is always non-negative.
3. It is positive definite for a multivariate Gaussian distribution. In other words, for any positive \boldsymbol{y} vector, $\boldsymbol{y}^T \boldsymbol{\Sigma} \boldsymbol{y}$ is always positive.

Considering these properties, the exponent term in equation (C.23) is always greater than 0, and its exponential is always less than 1.

There are three properties of the Gaussian multivariate distribution that are important for understanding operations with random variables that follow a Gaussian probability density function:

1. The linear combination of two independent Gaussian random variables is Gaussian.
2. The marginal of a joint Gaussian distribution is Gaussian
3. The conditional of a joint Gaussian distribution is Gaussian.
4. The linear transformation of a Gaussian random variable is a Gaussian. Given $\boldsymbol{x} \in \boldsymbol{X}$ and a matrix \boldsymbol{A} and a vector \boldsymbol{b} the expected value of the transformation $\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$ and its covariance can be computed as:

$$\begin{aligned} \mathbb{E} [\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}] &= \boldsymbol{A}\mathbb{E} [\boldsymbol{x}] + \boldsymbol{b} \\ \text{cov}(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}) &= \boldsymbol{A}\text{cov}(\boldsymbol{x})\boldsymbol{A}^T \end{aligned} \quad (\text{C.25})$$

This means that for gaussian distributed variables:

$$\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \Rightarrow \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b} \sim \mathcal{N}(\boldsymbol{A}\boldsymbol{\mu} + \boldsymbol{b}, \boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^T) \quad (\text{C.26})$$

An interesting property of the probability density distribution of a Gaussian model is when the concept of independence is introduced among the random variables on which

it is defined. For example, if two independent random variables are assumed to follow a univariate Gaussian distribution: $x_1 \sim \mathcal{N}(\mu_1, \sigma^2)$ and $x_2 \sim \mathcal{N}(\mu_2, \sigma^2)$. Using the conditional probability rules expressed in appendix C.4, the joint probability of these random variables can be written as:

$$\begin{aligned} P(x_1, x_2) &= \mathcal{N}(\mu_1, \sigma^2)\mathcal{N}(\mu_2, \sigma^2) \\ &= (2\pi\sigma^2)^{-0.5} e^{-\frac{1}{2\sigma^2}(x_1-\mu_1)^2} (2\pi\sigma^2)^{-0.5} e^{-\frac{1}{2\sigma^2}(x_2-\mu_2)^2} \\ &= \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2}((x_1-\mu_1)^T(\sigma^2)^{-1}(x_1-\mu_1)+(x_2-\mu_2)^T(\sigma^2)^{-1}(x_2-\mu_2))} \end{aligned}$$

grouping terms:

$$P(x_1, x_2) = (2\pi\sigma^2)^{-1} \exp \left[\frac{1}{2} \begin{pmatrix} (x_1 - \mu_1) \\ (x_2 - \mu_2) \end{pmatrix}^T \begin{pmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{pmatrix}^{-1} \begin{pmatrix} (x_1 - \mu_1) \\ (x_2 - \mu_2) \end{pmatrix} \right]$$

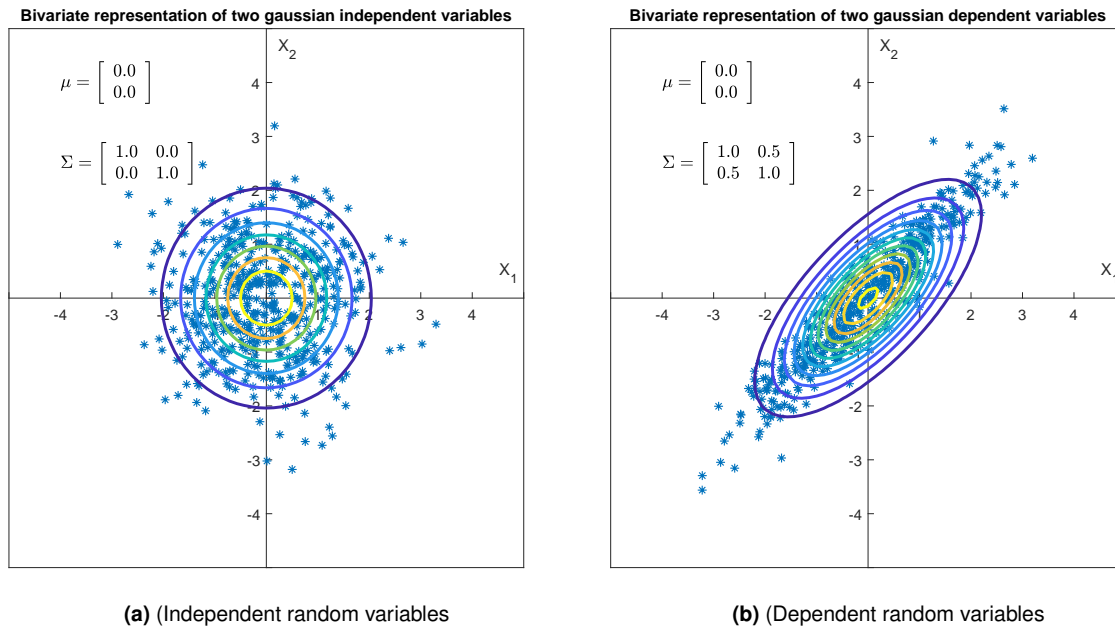


Figure C.3: Representation of two independent and dependent random variables following Gaussian distributions

Comparing with equation (C.23), the value of Σ corresponds to a diagonal matrix: $\sigma^2\mathbf{I}$. Therefore, given n independent variables that follow a univariate Gaussian distribution, their joint probability distribution is also a Gaussian distribution whose mean is the vector of means of each of the distributions, and the variance is a diagonal matrix formed by the variances of each distribution. An example of this distribution can be seen on the left of figure C.3.

The fact that $\Sigma_{2,1} = \Sigma_{1,2} = 0$ implies that knowing x_1 does not condition the value of x_2 (figure C.3a). On the other hand —as it is apparent on figure C.3b— if a value for x_1 is

taken, x_2 is limited; therefore, they are correlated values. In this case, $\Sigma_{2,1} = \Sigma_{1,2} = 0.5$, and thus, there is a difference in the contribution of each Gaussian variable to the final joint Gaussian distribution.

C.11.2 Sampling from a Gaussian distribution

Given a Gaussian distribution $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, the goal is to obtain values of \mathbf{x} that belong to this distribution. In the univariate case, x can be written as: $x \sim \mu + \sigma\mathcal{N}(0, 1)$ and from this expression values of x can be generated. In the multivariate case, the mean is a vector, and the variance is a square matrix. If this covariance matrix is decomposed into two triangular matrices using Cholesky factorisation $\boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}^T$, samples that fit this distribution can be generated using the following expression:

$$\mathbf{x} \sim \boldsymbol{\mu} + \mathbf{L}\mathcal{N}(0, \mathcal{I}) \quad (\text{C.27})$$

Where $\boldsymbol{\mu} = [\mu_1, \mu_2, \dots, \mu_n]^T$, and \mathbf{L} is a lower triangular matrix product from Cholesky decomposition. This operation can be done because the requirements to apply this decomposition are that the matrix must be square and positive. By definition of the covariance matrix, both are fulfilled.

C.11.3 Conditional Gaussian distribution

As mentioned above, one of the properties of the Gaussian distribution is that the conditional of a joint Gaussian distribution is also Gaussian. Thus, given two random variables \mathbf{x}_1 and \mathbf{x}_2 , whose marginal probability density functions are defined as: $f(\mathbf{x}_1) = \mathcal{N}(\mathbf{x}_1|\mu_1, \Sigma_{1,1})$ and $f(\mathbf{x}_2) = \mathcal{N}(\mathbf{x}_2|\mu_2, \Sigma_{2,2})$. The joint probability of \mathbf{x}_1 and \mathbf{x}_2 follows a Gaussian with $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ where:

$$\boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}; \boldsymbol{\Sigma} = \begin{pmatrix} \Sigma_{1,1} & \Sigma_{1,2} \\ \Sigma_{2,1} & \Sigma_{2,2} \end{pmatrix} \quad (\text{C.28})$$

Sometimes it is useful to employ the concept of the inverse covariance matrix called the precision matrix:

$$\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1} = \begin{pmatrix} \Lambda_{1,1} & \Lambda_{1,2} \\ \Lambda_{2,1} & \Lambda_{2,2} \end{pmatrix} \quad (\text{C.29})$$

Where:

$$\begin{aligned}
\Lambda_{1,1} &= (\Sigma_{1,1} - \Sigma_{1,2}\Sigma_{2,2}^{-1}\Sigma_{2,1})^{-1} \\
\Lambda_{1,2} &= -(\Sigma_{1,1} - \Sigma_{1,2}\Sigma_{2,2}^{-1}\Sigma_{2,1})^{-1}\Sigma_{1,2}\Sigma_{2,2}^{-1} \\
\Lambda_{2,1} &= -(\Sigma_{2,2} - \Sigma_{2,1}\Sigma_{1,1}^{-1}\Sigma_{1,2})^{-1}\Sigma_{2,1}\Sigma_{1,1}^{-1} \\
\Lambda_{2,2} &= (\Sigma_{2,2} - \Sigma_{2,1}\Sigma_{1,1}^{-1}\Sigma_{1,2})^{-1}
\end{aligned} \tag{C.30}$$

We wish to calculate the conditioned probability of obtaining \mathbf{x}_1 given \mathbf{x}_2 which, as previously defined, must also be a Gaussian distribution of the form $\mathcal{N}(\mu_{1|2}, \Sigma_{1|2})$, where it can be demonstrated that (Bishop, 2006):

$$\mu_{1|2} = \mu_1 + \Sigma_{1,2}\Sigma_{2,2}^{-1}(x_2 - \mu_2) = \Sigma_{1|2}(\Lambda_{1,1}\mu_1 - \Lambda_{1,2}(x_2 - \mu_2)) \tag{C.31}$$

$$\Sigma_{1|2} = \Sigma_{1,1} - \Sigma_{1,2}\Sigma_{2,2}^{-1}\Sigma_{2,1} = \Lambda_{1,1}^{-1} \tag{C.32}$$

C.11.4 Marginal and conditional Gaussians

Let \mathbf{x} be a continuous random variable following a gaussian model for its marginal probability density function $f(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x)$. Given another continuous random variable \mathbf{y} conditioned by \mathbf{x} , the conditional probability of \mathbf{y} and \mathbf{x} is fitted to a Gaussian Normal distribution: $f(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\mathbf{x} + \mathbf{b}, \boldsymbol{\Sigma}_y)$, where \mathbf{A} and \mathbf{b} are parameters of the model and $\boldsymbol{\Sigma}_y$ is the co-variance matrix of \mathbf{y} . The *a posteriori* probability according to equation (C.4) is:

$$f(\mathbf{x}|\mathbf{y}) = \frac{\mathcal{N}(\mathbf{y}|\mathbf{A}\mathbf{x} + \mathbf{b}, \boldsymbol{\Sigma}_y)\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x)}{f(\mathbf{y})} \tag{C.33}$$

It is fulfilled that the obtained conditional distribution is Gaussian too:

$$f(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_{x|y}, \boldsymbol{\Sigma}_{x|y})$$

where:

$$\boldsymbol{\Sigma}_{x|y}^{-1} = (\boldsymbol{\Sigma}_x^{-1} + \mathbf{A}^T\boldsymbol{\Sigma}_y^{-1}\mathbf{A})^{-1} \tag{C.34}$$

$$\boldsymbol{\mu}_{x|y} = \boldsymbol{\Sigma}_{x|y}^{-1} [\mathbf{A}^T\boldsymbol{\Sigma}_y^{-1}(\mathbf{y} - \mathbf{b}) + \boldsymbol{\Sigma}_x^{-1}\boldsymbol{\mu}_x] \tag{C.35}$$

In addition, the marginal distribution of the random variable \mathbf{Y} is a Gaussian:

$$f(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\boldsymbol{\mu}_x + \mathbf{b}, \boldsymbol{\Sigma}_y + \mathbf{A}\boldsymbol{\Sigma}_x\mathbf{A}^T) \tag{C.36}$$

The mathematical deduction of these expressions can be found at (Bishop, 2006). An intermediate interesting result that is achieved during the derivation of the above equations, is the one that concerns the joint probability of $f(\mathbf{X})$ and $f(\mathbf{Y}|\mathbf{X})$:

$$\begin{bmatrix} f(\mathbf{x}) \\ f(\mathbf{y}|\mathbf{x}) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_x \\ \mathbf{A}\boldsymbol{\mu}_x \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_x & \boldsymbol{\Sigma}_x\mathbf{A}^T \\ \mathbf{A}\boldsymbol{\Sigma}_x & \mathbf{A}\boldsymbol{\Sigma}_x\mathbf{A}^T + \boldsymbol{\Sigma}_y \end{bmatrix}\right) \tag{C.37}$$

C.12 Introduction to the kernel concept

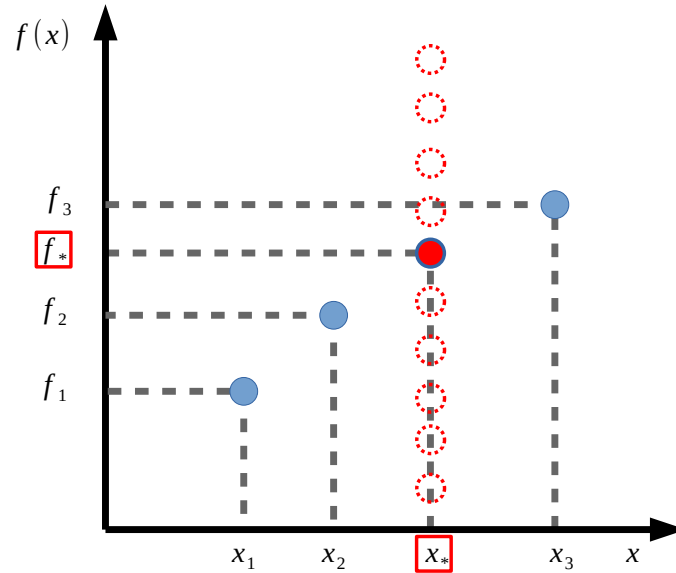


Figure C.4: Example to introduce the concept of the kernel.

A practical case consisting of three points $\{x_1, x_2, x_3\}$ following a certain function or model $f(x)$ is to be assumed. A possible representation of these points can be seen in figure C.4. We wish to find out the model $f(x)$ assuming that the joint distribution of the three points is a Gaussian distribution as follows:

$$\begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} \sim \mathcal{N} \left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} k_{1,1} & k_{1,2} & k_{1,3} \\ k_{2,1} & k_{2,2} & k_{2,3} \\ k_{3,1} & k_{3,2} & k_{3,3} \end{bmatrix} \right) \quad (\text{C.38})$$

If f follows a Gaussian distribution, two close values of x should have close values of f . As indicated in appendix C.10 the covariance captures this similarity correlation between the three points. An extended concept of covariance could be to assume a function $k(x_i, x_j)$ such that $k(x_i, x_j) \rightarrow 0$ when the similarity between x_i and x_j is very small. Instead, $k(x_i, x_j) \rightarrow 1$ when x_i and x_j are very similar. The expression $k(x_i, x_j)$ is a kernel function. An example of a kernel function can be the radial function:

$$k(x_i, x_j) = e^{-\lambda \|x_i - x_j\|} \quad (\text{C.39})$$

Therefore, equation (C.38) can be written using kernel notation as: $f \sim \mathcal{N}(0, k(x_i, x_j))$. If the dataset $D = \{(x_1, f_1), (x_2, f_2), (x_3, f_3)\}$ is considered, given a new value of x_* we

could estimate its corresponding value of f_* which is supposed to belong to the same model that describes the dataset points. Therefore the new joint probability could be defined from equation (C.38) as:

$$\begin{bmatrix} f \\ f_* \end{bmatrix} \sim \mathcal{N} \left(0, \begin{bmatrix} k_{1,1} & k_{1,2} & k_{1,3} & k_{1,*} \\ k_{2,1} & k_{2,2} & k_{2,3} & k_{2,*} \\ k_{3,1} & k_{3,2} & k_{3,3} & k_{3,*} \\ k_{*,1} & k_{*,2} & k_{*,3} & k_{*,*} \end{bmatrix} \right) \quad (\text{C.40})$$

The above matrix can be summarized using the following equations:

$$\mathbf{K} = \begin{bmatrix} k_{1,1} & k_{1,2} & k_{1,3} \\ k_{2,1} & k_{2,2} & k_{2,3} \\ k_{3,1} & k_{3,2} & k_{3,3} \end{bmatrix}; \mathbf{k}_* = \begin{bmatrix} k_{1,*} \\ k_{2,*} \\ k_{3,*} \end{bmatrix}; \quad (\text{C.41})$$

$$\begin{bmatrix} f \\ f_* \end{bmatrix} \sim \mathcal{N} \left(0, \begin{bmatrix} \mathbf{K} & \mathbf{k}_* \\ \mathbf{k}_*^T & k_{*,*} \end{bmatrix} \right) \quad (\text{C.42})$$

This composition can also be seen in this way:

$$\begin{bmatrix} f_* \\ f \end{bmatrix} \sim \mathcal{N} \left(0, \begin{bmatrix} k_{*,*} & \mathbf{k}_*^T \\ \mathbf{k}_* & \mathbf{K} \end{bmatrix} \right) \quad (\text{C.43})$$

In these conditions the probability of obtaining f_* from the values of f ($p(f_*|f) = \mathcal{N}(\mu_{f_*|f}, \Sigma_{f_*|f})$) can be calculated from equation (C.31) and equation (C.32):

$$\mu_{f_*|f} = \cancel{\mu_{x_*}}^0 + \mathbf{k}_*^T \mathbf{K}^{-1} \left(f - \cancel{\mu_x}^0 \right) = \mathbf{k}_*^T \mathbf{K}^{-1} f \quad (\text{C.44})$$

$$\Sigma_{f_*|f} = k_{*,*} - \mathbf{k}_*^T \mathbf{K}^{-1} \mathbf{k}_* \quad (\text{C.45})$$

Therefore, the most likely value expected for f_* given a set of f values from a set of correlated x and x_* variables is given by the expression:

$$\mu_{f_*|f} = \mathbb{E}(f_*) = \mathbf{k}_*^T \mathbf{K}^{-1} f \quad (\text{C.46})$$

Appendix D

Specifications of the components of the oculomotor and neck robotic system

Table D.1: MX28T motor specifications. These data have been provided by the motor manufacturer.

Item	Specifications
MCU	ARM CORTEX-M3 (72 [MHz], 32Bit)
Position Sensor	Contactless absolute encoder (12Bit, 360 [°]) Maker : ams(www.ams.com), Part No : AS5045
Motor	Coreless(Maxon)
Baud Rate	8,000 [bps] ~4.5 [Mbps]
Control Algorithm	PID control
Resolution	4096 [pulse/rev]
Backlash	20 [arcmin] (0.33 [°])
Operating Mode	Joint Mode (0 ~360 [°]) Wheel Mode (Endless Turn)
Weight	MX-28AR/AT : 77 [g], MX-28R/T : 72 [g]
Dimensions (W x H x D)	35.6 x 50.6 x 35.5 [mm]
Gear Ratio	193 : 1
Stall Torque	2.3 [Nm] (at 11.1 [V], 1.3 [A])
	2.5 [N.m] (at 12 [V], 1.4 [A])
	3.1 [Nm] (at 14.8 [V], 1.7 [A])
No Load Speed	50 [rev/min] (at 11.1 [V])
	55 [rev/min] (at 12 [V]) 67 [rev/min] (at 14.8 [V])
Radial Load	1 30 [N] (10 [mm] away from the horn)
Axial Load	1 15 [N]
Operating Temperature	-5 ~+80 [°C]
Input Voltage	10.0 ~14.8 [V] (Recommended : 12.0 [V])
Command Signal	Digital Packet
Protocol Type	TTL Half Duplex Asynchronous Serial Communication with 8bit, 1stop, No Parity RS485 Asynchronous Serial Communication with 8bit, 1stop, No Parity
Physical Connection	RS485 / TTL Multidrop Bus
ID	254 ID (0 ~253)
Feedback	Position, Temperature, Load, Input Voltage, etc Full Metal Gear
Material	Engineering Plastic(Front, Middle, Back) 1 Metal(Front)
Standby Current	100 [mA]

Table D.2: AX12A motor specifications. These data have been provided by the motor manufacturer.

Item	Specifications
Baud Rate	7843 bps ~1 Mbps
Weight	53.5g(AX-12, AX-12+), 54.6g(AX-12A)
Dimensions (W x H x D)	32mm x 50mm x 40mm 1.26 X 1.97 X 1.57 [inch]
Resolution	0.29 [°]
Running Degree	0 [°] ~300 [°] Endless Turn
Motor	Cored
Gear Ratio	254 : 1
Stall Torque	1.5 N*m (at 12V, 1.5A)
No Load Speed	59rpm (at 12V)
Operating Temperature	-5 [°C] ~+70 [°C]
Input Voltage	9.0 ~12.0V (Recommended : 11.1V)
Command Signal	Digital Packet
Protocol Type	Half Duplex Asynchronous Serial Communication (8bit, 1stop, No Parity)
Physical Connection	TTL Level Multi Drop Bus
ID	254 ID (0~253)
Feedback	Position, Temperature, Load, Input Voltage, etc
Gear Material	Engineering Plastic(Full)
Case Material	Engineering Plastic(Front, Middle, Back)

Table D.3: Ueye XS camera specifications. These data have been provided by the camera manufacturer.

Item	Specifications
Sensor type	CMOS color
Shutter System	Rolling Shutter
Characteristic	Linear
Sensor reading method	Progressive scan
Pixel class	QSXGA
Resolution	5.04 Mpx
Resolution (h xv)	2592 x 1944 Pixel
Aspect ratio	4:3
DAC	10 bit
Color Depth	8 bit
Optical sensor class	1/4"
Optical surface	3.629 mm x 2.722 mm
Diagonal of optical sensor	4.54 mm (1/3.53")
Pixel size	1.4 μm

Table D.4: Rapicam v1.0 camera specifications. These data have been provided by the camera manufacturer.

Specification	Camera Module v1
Size	Around 25 × 24 × 9 mm
Weight	3g
Still resolution	5 Megapixels
Video modes	1080p30, 720p60 and 640 × 480p60/90
Linux integration	V4L2 driver available
C programming API	OpenMAX IL and others available
Sensor	OmniVision OV5647
Sensor resolution	2592 × 1944 pixels
Sensor image area	3.76 × 2.74 mm
Pixel size	1.4 μm × 1.4 μm
Optical size	1/4"
Full-frame SLR lens equivalent	35 mm
S/N ratio	36 dB
Dynamic range	67 dB @ 8x gain
Sensitivity	680 mV/lux-sec
Dark current	16 mV/sec @ 60 C
Well capacity	4.3 Ke-
Fixed focus	1 m to infinity
Focal length	3.60 mm +/- 0.01
Horizontal field of view	53.50 +/- 0.13 degrees
Vertical field of view	41.41 +/- 0.11 degrees
Focal ratio (F-Stop)	2.9

Index of figures

- 1.1 Outline of the methodology followed in this research work. 8

- 2.1 Perception and action cycle 12
- 2.2 Schema of eye movements 15
- 2.3 Schema of the eye vergence and version movements 16
- 2.4 Areas of the brain involved in the sensory part of the visuo-oculomotor system 18
- 2.5 Schema of the main brain areas involved in saccadic behaviour 20

- 3.1 Schema of the machine learning types depending on the kind of problem to be solved. 24
- 3.2 Biological and artificial neural network comparison 27
- 3.3 Different schematic representations of a single-layer network. 28
- 3.4 Linear single layer neural network schema. 31
- 3.5 Schema of linear single layer neural network with generic basis function. 37

- 4.1 Conceptual design of a robot oculomotor system based on Helmholtz configuration 48
- 4.2 Conceptual design of a robot oculomotor system based on Fick configuration 49
- 4.3 Example of the head model 53
- 4.4 Two examples of how the morphologies of the robot heads change by varying their morphological parameters. 55

4.5	2D model of the robot head.	58
4.6	Radial basis function space partition in monocular transformation neural network	60
4.7	Generated dataset to train the RBFNNs.	61
4.8	Learning neural networks curves for the proposed neural networks comparing the binocular and monocular training behaviour.	62
4.9	Monocular versus binocular results	63
4.10	Birnbaum–Saunders distribution for binocular and monocular comparison	64
4.11	Gazing error of the monocular and binocular encoding as a function of the target distance.	65
4.12	FEL schema	66
4.13	RA schema	67
4.14	Tombatossals Robot.	72
4.15	Tombatossals' visual robotic system details.	73
4.16	Distribution of the initial fixation points (black crosses) and target points (black dots) used for creating the dataset.	73
4.17	Diagram of the process followed to generate the data used to compare the performance of the FEL and RA controllers.	75
4.18	Training and testing curves fro FEL controller	77
4.19	Training and testing curves fro RA controller	78
4.20	Comparative between FEL and RA controllers	79
4.21	The UJI humanoid torso: Tombatossals	80
4.22	Different types of real stimulus	81
4.23	Histograms generated from the obtained visual error	83
4.24	Probability density distribution and cumulative probability curves of the visual error estimated with the robot results	84
4.25	Number of loops (mean and standard deviation) in the RA as a function of the gain	86
4.26	Example of how the environment can be established in various forms.	87

4.27 Parallelepiped region defined as environment.	88
5.1 Schema showing the information fluxes and transformations among the different system components and the environment.	95
5.2 Different morphologies of robotic systems generated from the ranges defined in table 5.1	102
5.3 Environment and frustrum of the cameras intersecting	103
5.4 Schema of robotic head generation	106
5.5 Mean square error for \mathbf{B}_i estimation with 1000 robot head setups versus the number of iterations	107
5.6 Visual error probability density functions	109
5.7 Schema of the single layer feedforward neural network proposed for learning the relation between the \mathbf{B} matrix of the fixed controller and the morphological parameters.	111
5.8 Mean square error obtained by different neural network hidden layer setups. The red line represents the standard deviation for three repetitions of the training	111
5.9 The learning curve for \mathbf{B} and $\Gamma^{(m)}$ regression	112
5.10 Schema of single-layer neural network for learning adaptive weights from morphological parameters	113
5.11 Proposed stack of the autoencoders	116
5.12 Proposed regression problem decomposition into many straightforward regressions.	118
5.13 Mean square error obtained for different neural network setups. The red line represents the standard deviation over the three repetitions of the training.	120
5.14 Mean of the visual error for 500 robot head setups	122
5.15 Standard deviation of the visual error for 500 robot head setups	123
5.16 Comparison of the three models proposed with the 500 testing robotic heads	124

5.17 The MSE $\{\theta_0, \hat{\theta}_0\}$ vs mean visual error resulting from the θ training for pnn_{20} architecture.	125
5.18 Average of the learning curves for 500 systems in the test dataset that has been initialised with the prediction made by the internal model estimation from the morphology.	134
5.19 Schema of the genetic concepts from an abstract point of view	137
5.20 Schema of the proposed genotype model and the relationship with the environment and phenotype	139
5.21 Block schema of genotype model and the relationship with the environment and phenotype	141
5.22 Schema of the proposed genotype model and the relationship with the environment and phenotype	145
5.23 Procedure to determine the artificial genotype of the family of robotic system described in section 5.6	147
5.24 Example of norm of reaction obtained for three types of Drosophila flies	148
5.25 Obtained norm of reaction for three randomly robotic heads	149
6.1 Summary of the different approaches referenced and employed in robotics for depth estimation using a monocular camera.	155
6.2 Example of how microsaccades can affect perception in the fixation process.	159
6.3 Simplified scheme to show the sphere of eye movements.	160
6.4 Schema of the considered camera movements for modelling visual fixation process	161
6.5 Schema representing the ideas behind the mathematical development .	165
6.6 Different images captured during the execution of the proposed algorithm.	170
6.7 Different partial results were obtained during the execution of the algorithm to visualize its evolution.	171
6.8 MSE and standard deviation between the background depth image and the estimation made by the proposed algorithm for 60 cm fixation point and different values of σ for a constant value of $\rho = 0.70$	175

6.9 SSIM index estimation between the background depth image and the estimation made by the proposed algorithm for 60 cm fixation point and different values of σ for a constant value of $\rho = 0.70$	176
6.10 MSE and standard deviation between the background depth image and the estimation made by the proposed algorithm for 60 cm fixation point and different values of ρ for a constant value of $\sigma = 0.01$	177
6.11 SSIM index estimation between the background depth image and the estimation made by the proposed algorithm for 60 cm fixation point and different values of ρ for a constant value of $\sigma = 0.010$	178
6.12 MSE and standard deviation between the background depth image and the estimation made by the proposed algorithm for 60 cm fixing point and different values of Z_0 for constant values of $\rho = 0.7$ and $\sigma = 0.01$	179
6.13 SSIM index estimation between the background depth image and the estimation made by the proposed algorithm for 60 cm fixing point and different values of Z_0 for constant values of $\rho = 0.7$ and $\sigma = 0.01$	180
6.14 MSE with error bars representing standard deviation of the last 20 iterations. The bars are grouped by the added white noise. $E_i = \{\phi_t(i), \phi_r(i)\}$	181
6.15 Image depth, showing the distance for each pixel scaled in 0-255 range.	182
6.16 Example of the evolution of depth estimation mean for the Aruco regions	185
6.17 Layout for the experiments with ordinary objects.	187
6.18 MSE and standard deviation evolution with real objects scenario for three different fixations points at 30, 60, 90 cm.	188
6.19 SSIM evolution with real objects scenario for three different fixations points at 30, 60, 90 cm.	189
6.20 Gray-scale experimental results for the scenario with ordinary objects and fixation points at 30, 60 and 90 cm	190
6.21 Results obtained for depth estimation algorithm	192
6.22 Ouchi illusion	193
6.23 Ouchi image experiment setup	194
6.24 Ouchi experiment results	195

6.25 Oculomotor system detail	197
6.26 Oculomotor system and environment setup to train the fixed and adaptive controller from stimuli projected on a TV monitor.	198
6.27 Motor babbling results	200
6.28 Learning curve of the adaptive controller training process for the designed oculomotor system	202
6.29 Diagram showing different types of neck movement in humans.(Alfayad et al., 2016)	203
6.30 Neck system detail	205
6.31 Final oculomotor-neck robotic system developed	210
6.32 Initial RGB image from both cameras of the built oculomotor-neck system	210
6.33 Evolution of obtained depth estimation in the Aruco Markers regions of the left camera with the oculomotor-neck robotic system for $\rho = 0.85$ and $\sigma = 0.0001$	211
6.34 Results obtained by the proposed depth estimation algorithm for both cameras during the fixation process with parameters $\rho = 0.99$ and $\sigma = 0.001$	212
6.35 Overlaying the edges of the RGB images on the depth estimation obtained for both cameras	213
A.1 Baxter robot with UJI RobInLab team at Amazon Picking Challenge 2015.	226
A.2 Geometrical scheme of a camera displacement	227
A.3 Detailed architecture of depthS.	229
A.4 Architecture of depthC.	231
A.5 Architecture of depthCSx.	232
A.6 Examples of generated scenes for the dataset. The insets show the images captured by the eye-in-hand camera.	233
A.7 Evolution of RMSE for the proposed neural networks during training and validation.	235
A.8 Representative example of depth maps predicted by depthC, depthS, depthCS and depthCSS.	237

B.1	ROS network connections schema	240
B.2	Module description in software architecture	242
B.3	Time sequencing example of sending a command in blocking execution mode between two modules	244
B.4	Time sequencing example of command and event combination	245
B.5	Example of the yaml file with module parameters definition	246
B.6	Implementation schema of the server base module	247
B.7	Service base module execution pipeline	248
B.8	Action primitive module execution pipeline	249
B.9	Finite state machine defined in an exploration action primitive used in the Amazon Robotic Challenge in 2015	251
B.10	Schema of the proposal combination of service base and action primitive modules	252
B.11	Schema of the developed software architecture for depth estimation based on fixation process (chapter 6)	253
B.12	Finite state machine integrated in action primitive module for depth estimation based on fixation process	254
C.1	Gaussian distribution for $\mu = 0.0$ and $\sigma = 1.0$	264
C.2	Bivariate Gaussian distribution example.	265
C.3	Representation of two independent and dependent random variables following Gaussian distributions	267
C.4	Example to introduce the concept of the kernel.	270

Acronyms

AE	Autoencoder
AIC	Akaike information criterion
BG	Basal ganglia
BS	Brainstem
CAE	Contractive autoencoders
CCD	Charge Coupled Device
CMOS	Complementary Metal Oxide Semiconductor
CPFA	Combination parameters from allelic
EAs	Evolutionary algorithms
ER	Evolutionary Robotics
FEF	Frontal eye field
FEL	Feedback error learning
GDF	Generalized degrees of freedom
GDF	Generalized degrees of freedom
GEP	Genotype-Environment-Phenotype
GVA	Generalized extreme value distribution
ISSGPR	Incremental sparse spectrum Gaussian process regression
IT	Inferior Temporal Cortex
LIP	Lateral intraparietal cortex

MFN	Multilayer feedforward network
ML	Maximun likelihood estimation
MLE	Maximum likelihood estimator
MSE	Mean square error
PCA	Principal component analysis
PPC	Posterior parietal cortex
RA	Recurrent architecture
RBF	Radial basis function
RBFNN	Radial basis function neural networks
RLS	Recursive least Square
RPTF	Regulatory Parameters of transcription function
SAE	Sparse autoencoder
SBG	Saccadic burst generator
SC	Superior colliculus
SCG	Scaled conjugate gradient back-propagation
SFP	Species' funtional parameters
SGD	Stochastic gradient descent (SGD)
SN	Substantia nigra
SRI	Species' regulatory information
SRP	Species' regulatory parameters
SSIM	Structural similarity Index
V1	Primary Visual Cortex
V2	Secondary Visual Cortex
V4	Visual area V4

Publications supporting this thesis

Publications in international journals with JCR impact factor

1. **Antonelli, M., Gibaldi, A., Beuth, F., Duran, A.J., Canessa, A., Chessa, M., Solari, F., del Pobil, A.P., Hamker, F., Chinellato, E., Sabatini, S.P., 2014**, "A hierarchical system for a distributed representation of the peripersonal space of a humanoid robot", *IEEE Transactions on Autonomous Mental Development*, Vol. 6, No. 4, pp. 259-273. DOI: 10.1109/TAMD.2014.2332875.
2. **Antonelli, M., Duran, A.J., Chinellato, E., del Pobil, A.P., 2015**, "Learning the Visual-Oculomotor Transformation: Effects on Saccade Control and Space Representation", *Robotics and Autonomous Systems*, Vol. 71, pp. 13-22. DOI: 10.1016/j.robot.2014.11.018.
3. **Duran, A.J., del Pobil, A.P., 2018**, "Predicting the internal model of a robotic system from its morphology", *Robotics and Autonomous Systems*, Vol. 110, pp. 33-43. DOI: 10.1016/j.robot.2018.08.014
4. **Duran, A.J., del Pobil, A.P., 2021**, "Robot depth estimation inspired by fixational movements", *IEEE Transactions on Cognitive and Developmental Systems*. (in press, on line). Doi: 10.1109/TCDS.2020.3025057.
5. **Yoneyama, R., Duran, A.J., del Pobil, A.P., 2021**, "Integrating Sensor Models in Deep Learning Boosts Performance: Application to Monocular Depth Estimation in Warehouse Automation", *Sensors*, Vol. 21, No. 4, 1437. DOI: 10.3390/s21041437

Published chapters

1. **del Pobil, A.P., Duran, A.J., Antonelli, M., Felip, J. Morales, A., Prats, M., Chinellato, E., 2013**, "Integration of Visuomotor Learning, Cognitive Grasping

and Sensor-Based Physical Interaction in the UJI Humanoid Torso”, in *Designing Intelligent Robots: Reintegrating AI*, edited by B. Boots et al., AAAI Press, Palo Alto, California, pp. 6-11. ISBN: 978-1-57735-601-1

2. **Antonelli, M., Duran, A. J., Chinellato, E., & Del Pobil, A. P., 2013**, “Speeding-up the Learning of Saccade Control” in *Biomimetic and Biohybrid Systems*, edited by N.F. Lepora et al., *Lecture Notes in Artificial Intelligence* Vol. 8064, Springer-Verlag, Berlin, pp. 12-23. ISBN: 978-3-642-39801-8. DOI 10.1007/978-3-642-39802-5_2.
3. **Duran, A.J., del Pobil, A.P., 2016**, “A model of artificial genotype and norm of reaction in a robotic system”, in *From Animals to Animats 14*, edited by Elio Tuci, A. Giagkos, M. Wilson and J. Hallam , *Lecture Notes in Artificial Intelligence* vol. 9825, Springer, Heidelberg, pp. 267–279. ISBN: 978-3-319-43487-2. DOI: 10.1007/978-3-319-43488-9_24
4. **Duran, A.J., del Pobil, A.P., 2017**, “Discovering the Relationship Between the Morphology and the Internal Model in a Robot System by Means of Neural Networks”, in *Intelligent Autonomous Systems 14*, edited by W. Chen, K. Hosoda, E. Menegatti, M. Shimizu and H. Wang, *Advances in Intelligent Systems and Computing* Vol. 531, Springer, Berlin, pp. 839-852. ISBN: 978-3-319-48035-0. Doi: 10.1007/978-3-319-48036-7.

Publications in conference proceedings

1. **Antonelli, M., Duran, A.J., del Pobil, A.P., 2013**, “Application of the Visuo-Oculomotor Transformation to Ballistic and Visual-Guided Eye Movements”, in *Proc. International Joint Conference on Neural Networks (IJCNN 2013)*, Dallas, Texas, USA, pp. 813-820. ISBN: 978-1-4673-6129-3/13.
2. **Antonelli, M., Duran, A.J., Chinellato, E., del Pobil, A.P., 2015**, “Adaptive saccade controller inspired by the primates’ cerebellum”, in *Proc. IEEE International Conference on Robotics and Automation (ICRA 2015)*, Seattle, USA, pp. 5048-5053. DOI: 10.1109/ICRA.2015.7139901
3. **Felip, J., Duran, A.J., Antonelli, A., Morales, A., del Pobil, A.P., 2015**, “Tombatozals: A humanoid torso for autonomous sensor-based tasks”, in *Proc. 15th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2015)*, Seoul, Korea. pp. 475-481, DOI: 10.1109/HUMANOIDS.2015.7363592

4. **Duran, A.J., del Pobil, A.P.**, 2016, "Initial weight estimation for learning the internal model based on the knowledge of the robot morphology", in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2016)*, Daejeon, Korea. pp. 3941-3946.
5. **del Pobil, A.P., M. Kassawat, A. J. Duran**, M.A. Arias, N. Nechyporenko, A. Mallick, E. Cervera, D. Subedi, I. Vasilev, D. Cardin, E. Sansebastiano, E. Martinez-Martin, A. Morales, G.A. Casañ, A. Arenal, B. Goriatcheff, C. Rubert and G. Recatala, 2017, "UJI RobInLab's Approach to the Amazon Robotics Challenge 2017", in *Proc. IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI 2017)*, Daegu, Korea. pp. 318-323.
6. **Kassawat, M., Duran, A. J., Cervera, E.** 2018, "Simplificación de la tarea de picking industrial para el diseño de un sistema robótico". XXXIX Jornadas de Automática, Almeria, Spain pp. 364–371.
7. **Duran, A.J., del Pobil, A.P.**, 2019, "Improving robot visual skills by means of a bio-inspired model", in *Proc. 9th Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics (ICDL-EPIROB 2019)*, Oslo, Norway, pp. 25-30. DOI: 10.1109/DEVLRN.2019.8850712.

Bibliography

- AKAIKE, HIROTOGU (1998). *Information Theory and an Extension of the Maximum Likelihood Principle*. pp. 199–213. Springer New York, New York, NY. ISBN 978-1-4612-1694-0. doi: 10.1007/978-1-4612-1694-0_15
- ALBERCH, P. (1991). «From genes to phenotype: dynamical systems and evolvability». *Genetica*, **84(1)**, pp. 5–11. ISSN 1573-6857. doi: 10.1007/BF00123979
- ALDRIN, BUZZ and ABRAHAM, KEN (2009). *Magnificent Desolation: The Long Journey Home from the Moon*. ISBN 9780307463456
- ALFAYAD, SAMER; ASSWAD, MOHAMAD EL; ABDELLATIF, A.; OUEZDOU, FETHI B.; BLANCHARD, ARNAUD; BEAUSSÉ, NILS and GAUSSIER, PHILIPPE (2016). «HYDROïD humanoid robot head with perception and emotion capabilities: Modeling, design, and experimental results». *Frontiers Robotics AI*, **3(APR)**, pp. 1–16. ISSN 22969144. doi: 10.3389/frobt.2016.00015
- ALMALIOGLU, YASIN; SAPUTRA, MUHAMAD RISQI U; DE GUSMAO, PEDRO PB; MARKHAM, ANDREW and TRIGONI, NIKI (2018). «GANVO: Unsupervised Deep Monocular Visual Odometry and Depth Estimation with Generative Adversarial Networks». *arXiv preprint arXiv:1809.05786*
- ANDERS, ULRICH and KORN, OLAF (1999). «Model selection in neural networks». *Neural Networks*, **12(2)**, pp. 309–323. ISSN 08936080. doi: 10.1016/S0893-6080(98)00117-8
- ANTONELLI, M.; DEL POBIL, A. P. and RUCCI, M. (2013a). «Depth estimation during fixational head movements in a humanoid robot». In: *International Conference on Computer Vision Systems*, pp. 264–273. Springer
- ANTONELLI, M.; DEL POBIL, A. P. and RUCCI, M. (2014). «Bayesian multimodal integration in a robot replicating human head and eye movements». In: *2014 IEEE International Conference on Robotics and Automation*, pp. 2868–2873. ISSN 1050-4729. doi: 10.1109/ICRA.2014.6907271

- ANTONELLI, M.; RUCCI, M. and SHI, B. (2016). «Unsupervised learning of depth during coordinated head/eye movements». In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5199–5204. ISSN 2153-0866. doi: 10.1109/IROS.2016.7759764
- ANTONELLI, MARCO; CHINELLATO, ERIS and DEL POBIL, ANGEL P. (2013b). *On-Line Learning of the Visuomotor Transformations on a Humanoid Robot*. volume 193 of *Advances in Intelligent Systems and Computing*, pp. 853–861. Springer Berlin Heidelberg. ISBN 978-3-642-33925-7. doi: 10.1007/978-3-642-33926-4_82
- ANWAR, SAJID; HWANG, KYUYEON and SUNG, WONYONG (2017). «Structured pruning of deep convolutional neural networks». *ACM Journal on Emerging Technologies in Computing Systems*, **13(3)**, pp. 1–18. ISSN 15504840. doi: 10.1145/3005348
- ASFOUR, T.; REGENSTEIN, K.; AZAD, P.; SCHRÖDER, J.; BIERBAUM, A.; VAHRENKAMP, N. and DILLMANN, R. (2006). «ARMAR-III: An integrated humanoid platform for sensory-motor control». *Proceedings of the 2006 6th IEEE-RAS International Conference on Humanoid Robots, HUMANOIDS*, pp. 169–175. doi: 10.1109/ICHR.2006.321380
- AYTEKIN, M. and , M. (2012). «Motion parallax from microscopic head movements during visual fixation». *Vision Research*, **70(617)**, pp. 7–17. ISSN 00426989. doi: 10.1016/j.visres.2012.07.017
- BAJCSY, RUZENA; ALOIMONOS, YIANNIS and TSOTSOS, JOHN K. (2018). «Revisiting active perception». *Auton Robot*, **521**, pp. 436—444
- BALDI, PIERRE (2012). «Autoencoders, Unsupervised Learning, and Deep Architectures». *ICML Unsupervised and Transfer Learning*, pp. 37–50. ISSN 0899-7667
- BELL, ANDREW H. and MUNOZ, DOUGLAS P. (2008). «Activity in the superior colliculus reflects dynamic interactions between voluntary and involuntary influences on orienting behaviour». *European Journal of Neuroscience*, **28(8)**, pp. 1654–1660. ISSN 0953816X
- BENGIO, YOSHUA; LAMBLIN, PASCAL; POPOVICI, DAN and LAROCHELLE, HUGO (2007). «Greedy Layer-Wise Training of Deep Networks». *Advances in Neural Information Processing Systems*, **19(1)**, p. 153. ISSN 01628828. doi: citeulike-article-id:4640046
- BIRNBAUM, Z W and SAUNDERS, S C (1969). «A new family of life distributions». *Journal of Applied Probability*, **6(2)**, pp. 319–327. doi: 10.2307/3212003
- BISHOP, CHRISTOPHER M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg. ISBN 0387310738

- BOHG, J.; HAUSMAN, K.; SANKARAN, B.; BROCK, O.; KRAGIC, D.; SCHAAL, S. and SUKHATME, G. S. (2017). «Interactive Perception: Leveraging Action in Perception and Perception in Action». *IEEE Transactions on Robotics*, **33(6)**, pp. 1273–1291. doi: 10.1109/TRO.2017.2721939
- BOMPOS, NIKOLAOS A; ARTEMIADIS, PANAGIOTIS K; OIKONOMOPOULOS, APOLLON S and KYRIAKOPOULOS, KOSTAS J (2007). «Modeling, full identification and control of the mitsubishi pa-10 robot arm». In: *2007 IEEE/ASME international conference on Advanced intelligent mechatronics*, pp. 1–6. IEEE
- BONGARD, JOSH; ZYKOV, VICTOR and LIPSON, HOD (2006). «Resilient machines through continuous self-modeling». *Science*, **314(5802)**, pp. 1118–1121
- BONSIGNORIO, FABIO and DEL POBIL, ANGEL P (2015). «Toward replicable and measurable robotics research [from the guest editors]». *IEEE Robotics & Automation Magazine*, **22(3)**, pp. 32–35
- BOSCO, ANNALISA; BREVEGLIERI, ROSSELLA; CHINELLATO, ERIS; GALLETTI, CLAUDIO and FATTORI, PATRIZIA (2010). «Reaching activity in the medial posterior parietal cortex of monkeys is modulated by visual feedback». *Journal of Neuroscience*, **30(44)**, pp. 14773–14785
- BROOMHEAD, D and LOWE, D. (1988). «Multivariable functional interpolation and adaptive networks». *Complex Systems*, **2**, pp. 321–355
- BRUSKE, JÖRG; HANSEN, MICHAEL; RIEHN, LARS and SOMMER, GERALD (1997). «Biologically inspired calibration-free adaptive saccade control of a binocular camera-head». *Biological Cybernetics*, **77(6)**, pp. 433–446. ISSN 0340-1200. doi: 10.1007/s004220050403
- BUNEO, CHRISTOPHE A.; JARVIS, MURRAY R.; BATISTA, AARON P. and ANDERSEN, RICHARD A. (2002). «Direct visuomotor transformations for reaching». *Nature*, **416(6881)**, pp. 632–636. ISSN 00280836. doi: 10.1038/416632a
- BURNHAM, K.P. and ANDERSON, D.R. (2002). *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach (2nd ed)*. volume 172. ISBN 978-0-387-22456-5. doi: 10.1016/j.ecolmodel.2003.11.004
- CHALUPA, LEO M and WERNER, JOHN S (2004). *The visual neurosciences, Vols. 1 & 2*. MIT press
- CHAO, F; LEE, MH and LEE, JJ (2010). «A developmental algorithm for ocular-motor coordination». *Robotics and Autonomous Systems*, **58(3)**, pp. 239–248

- CHEN, G. and DONG, X. (1998). *From Chaos to Order: Methodologies, Perspectives, and Applications*. World scientific series on non-linear science: Monographs and treatises. World Scientific. ISBN 9789810225698
- CHEN-HARRIS, HAIYIN; JOINER, WILSAAN M.; ETHIER, VINCENT; ZEE, DAVID S. and SHADMEHR, REZA (2008). «Adaptive control of saccades via internal feedback». *Journal of Neuroscience*, **28(11)**, pp. 2804–2813. ISSN 02706474. doi: 10.1523/JNEUROSCI.5300-07.2008
- CHINELLATO, E.; ANTONELLI, M.; GRZYB, B. J. and DEL POBIL, A. P. (2011). «Implicit Sensorimotor Mapping of the Peripersonal Space by Gazing and Reaching». *IEEE Transactions on Autonomous Mental Development*, **3(1)**, pp. 43–53. ISSN 1943-0612. doi: 10.1109/TAMD.2011.2106781
- CHINELLATO, E.; GRZYB, B.J. and DEL POBIL, A.P. (2012a). «Pose Estimation through Cue Integration: a Neuroscience-Inspired Approach». *IEEE Transactions on Systems, Man and Cybernetics -Part B: Cybernetics*, **424(2)**, pp. 530–538
- CHINELLATO, ERIS; ANTONELLI, MARCO and DEL POBIL, ANGEL P (2012b). «A Pilot Study on Saccadic Adaptation Experiments with Robots». In: TonyJ. Prescott; NathanF. Lepora; Anna Mura and PaulF.M.J. Verschure (Eds.), *Biomimetic and Biohybrid Systems*, volume 7375 of *Lecture Notes in Computer Science*, pp. 83–94. Springer Berlin Heidelberg. ISBN 978-3-642-31524-4. doi: 10.1007/978-3-642-31525-1_8
- CHINELLATO, ERIS and DEL POBIL, ANGEL P. (2016). *The Visual Neuroscience of Robotic Grasping. Achieving Sensorimotor Skills through Dorsal-Ventral Stream Integration*. Cognitive Systems Monographs, Vol. 28. Springer. ISBN 978-3-319-20302-7
- COHEN, RONALD A (2011). *Optokinetic Reflex*. pp. 1823–1824. Springer New York, New York, NY. ISBN 978-0-387-79948-3
- CREMER, SVEN; MASTROMORO, LAWRENCE and POPA, DAN O. (2016). «On the performance of the Baxter research robot». *2016 IEEE International Symposium on Assembly and Manufacturing, ISAM 2016*, pp. 106–111. doi: 10.1109/ISAM.2016.7750722
- DE HAAN, LAURENS and FERREIRA, ANA (2007). *Extreme Value Theory: An Introduction*. volume 49. ISBN 9780387239460. doi: 10.1198/tech.2007.s684
- DE LEEUW, JAN (2009). «Journal of statistical software». *Wiley Interdisciplinary Reviews: Computational Statistics*, **1(1)**, pp. 128–129. ISSN 19395108. doi: 10.1002/wics.10

- DEL POBIL, A. P.; KASSAWAT, M.; DURAN, A. J.; ARIAS, M. A.; NECHYPORENKO, N.; MALLICK, A.; CERVERA, E.; SUBEDI, D.; VASILEV, I.; CARDIN, D.; SANSEBASTIANO, E.; MARTINEZ-MARTIN, E.; MORALES, A.; CASAN, G. A.; ARENAL, A.; GORIATCHEFF, B.; RUBERT, C. and RECATALA, G. (2017). «UJI RobInLab's approach to the Amazon Robotics Challenge 2017». *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, **2017-Novem(Mfi)**, pp. 318–323. doi: 10.1109/MFI.2017.8170448
- DEL POBIL, ANGEL P. (2018). «Robots as Cyberphysical Systems: The Challenges Ahead». keynote speech at the 12th ACM International Conference on Ubiquitous Information Management and Communication, Langkawi, Malaysia
- DIANKOV, ROSEN (2010). *Automated Construction of Robotic Manipulation Programs*. Ph.D. thesis, Carnegie Mellon University, Robotics Institute
- DONCIEUX, STEPHANE and MOURET, JEAN-BAPTISTE (2014). «Beyond black-box optimization: a review of selective pressures for evolutionary robotics». *Evolutionary Intelligence*, **7(2)**, pp. 71–93. ISSN 1864-5917. doi: 10.1007/s12065-014-0110-x
- DOSOVITSKIY, ALEXEY; FISCHER, PHILIPP; ILG, EDDY; HAUSSER, PHILIP; HAZIRBAS, CANER; GOLKOV, VLADIMIR; VAN DER SMAGT, PATRICK; CREMERS, DANIEL and BROX, THOMAS (2015). «FlowNet: Learning optical flow with convolutional networks». In: *Proceedings of the IEEE international conference on computer vision*, pp. 2758–2766
- EBERHART, RUSSELL C and SHI, YUHUI (2011). *Computational intelligence: concepts to implementations*. Elsevier
- EIGEN, DAVID; PUHRSCHE, CHRISTIAN and FERGUS, ROB (2014a). «Depth map prediction from a single image using a multi-scale deep network». In: *Advances in neural information processing systems*, pp. 2366–2374
- EIGEN, DAVID; PUHRSCHE, CHRISTIAN and FERGUS, ROB (2014b). «Depth map prediction from a single image using a multi-scale deep network». In: *Advances in neural information processing systems*, pp. 2366–2374
- ELKADY, AYSSAM and SOBH, TAREK (2012). «Robotics Middleware: A Comprehensive Literature Survey and Attribute-Based Bibliography». *Journal of Robotics*, **2012**, pp. 1–15. ISSN 1687-9600. doi: 10.1155/2012/959013
- ENGEL, JAKOB; SCHÖPS, THOMAS and CREMERS, DANIEL (2014). «LSD-SLAM: Large-scale direct monocular SLAM». In: *European conference on computer vision*, pp. 834–849. Springer

- ENRIGHT, J. T. (1984). «Changes in vergence mediated by saccades.» *The Journal of Physiology*, **350(1)**, pp. 9–31. ISSN 14697793. doi: 10.1113/jphysiol.1984.sp015186
- ESPOSITO, F.; FERILLI, S.; FANIZZI, N.; BASILE, T. M. A. and DI MAURO, N. (2004). «Incremental Learning and Concept Drift in INTHELEX». *Intell. Data Anal.*, **8(3)**, p. 213–237. ISSN 1088-467X
- ET AL., S. GARRIDO-JURADO (2014). «Automatic generation and detection of highly reliable fiducial markers under occlusion». *Pattern Recognition*, **47(6)**, pp. 2280 – 2292
- FANTONI, C.; CAUDEK, C. and DOMINI, F. (2010). «Systematic distortions of perceived planar surface motion in active vision». *Journal of Vision*, **10(5)**, pp. 12–12
- FAUSETT, LAURENE (Ed.) (1994). *Fundamentals of neural networks: architectures, algorithms, and applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA. ISBN 0-13-334186-0
- FELIP, J; DURÁN, A J; ANTONELLI, M; MORALES, A and DEL POBIL, A P (2015). «Tombatossals: A humanoid torso for autonomous sensor-based tasks». In: *IEEE-RAS International Conference on Humanoid Robots*, . doi: 10.1109/HUMANOIDS.2015.7363592
- FELIP, JAVIER; LAAKSONEN, JANNE; MORALES, ANTONIO and KYRKI, VILLE (2013). «Manipulation primitives: A paradigm for abstraction and execution of grasping and manipulation tasks». *Robotics and Autonomous Systems*, **61(3)**, pp. 283–296
- FERMÜLLER, CORNELIA; PLESS, ROBERT and ALOIMONOS, YIANNIS (2000). «The Ouchi illusion as an artifact of biased flow estimation». *Vision Research*, **40(1)**, pp. 77–95. ISSN 00426989. doi: 10.1016/S0042-6989(99)00162-5
- FISHER, RONALD AYLMER (1930). *The genetical theory of natural selection: a complete variorum edition*. Oxford University Press
- FORSSÉN, PER ERIK (2007). «Learning saccadic gaze control via motion prediction». *Proceedings - Fourth Canadian Conference on Computer and Robot Vision, CRV 2007*, pp. 44–51. doi: 10.1109/CRV.2007.42
- FUCHS, M.; BORST, CH; GIORDANO, P. ROBUFFO; BAUMANN, A.; KRAEMER, E.; LANGWALD, J.; GRUBER, R.; SEITZ, N.; PLANK, G.; KUNZE, K.; BURGER, R.; SCHMIDT, F.; WIMBOECK, T. and HIRZINGER, G. (2009). «Rollin' Justin - Design considerations and realization of a mobile platform for a humanoid upper body». *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 4131–4137. ISSN 10504729. doi: 10.1109/ROBOT.2009.5152464

- FURQAN, MOHD; SUHAIB, MOHD and AHMAD, NAZEER (2017). «Studies on Stewart platform manipulator: A review». doi: 10.1007/s12206-017-0846-1
- GANDHI, NEERAJ J and KATNANI, HUSAM A (2011). «Motor functions of the superior colliculus». *Annual review of neuroscience*, **34**, pp. 205–231. ISSN 1545-4126
- GIJSBERTS, ARJAN and METTA, GIORGIO (2013). «Real-time model learning using Incremental Sparse Spectrum Gaussian Process Regression.» *Neural networks : the official journal of the International Neural Network Society*, **41**, pp. 59–69. ISSN 1879-2782. doi: 10.1016/j.neunet.2012.08.011
- GIRARD, B. and BERTHOZ, A. (2005). «From brainstem to cortex: Computational models of saccade generation circuitry». *Progress in Neurobiology*, **77(4)**, pp. 215–251. ISSN 03010082. doi: 10.1016/j.pneurobio.2005.11.001
- GORDON, ROBERT D. (2004). «Attentional allocation during the perception of scenes». *Journal of Experimental Psychology: Human Perception and Performance*, **30(4)**, pp. 760–777. ISSN 00961523. doi: 10.1037/0096-1523.30.4.760
- GRIFFITHS, A.J.F. (2005). *An Introduction to Genetic Analysis*. W. H. Freeman, 8 edition. ISBN 9780716749394
- GRIFFITHS, A.J.F. (2008). *Introduction to Genetic Analysis*. W. H. Freeman, 10 edition. ISBN 9780716768876
- GURNEY, KEVIN (1997). *An introduction to neural networks*. CRC press
- HAFED, ZIAD M. (2011). «Mechanisms for generating and compensating for the smallest possible saccades». *European Journal of Neuroscience*, **33(11)**, pp. 2101–2113. ISSN 0953816X. doi: 10.1111/j.1460-9568.2011.07694.x
- HAFED, ZIAD M and KRAUZLIS, RICHARD J (2010). «Microsaccadic suppression of visual bursts in the primate superior colliculus». *The Journal of neuroscience*, **30(28)**, pp. 9542–9547. ISSN 1529-2401. doi: 10.1523/JNEUROSCI.1137-10.2010
- HAMKER, FRED (2005a). «The emergence of attention by population-based inference and its role in distributed processing and cognitive control of vision». *Computer Vision and Image Understanding*, **100(1-2 SPEC. ISS.)**, pp. 64–106. ISSN 10773142
- HAMKER, FRED H. (2005b). «The reentry hypothesis: The putative interaction of the frontal eye field, ventrolateral prefrontal cortex, and areas V4, IT for attention and eye movement». *Cerebral Cortex*, **15(4)**, pp. 431–447. ISSN 10473211

- HAN, JUNGONG and ET AL. (2013). «Enhanced computer vision with Microsoft Kinect sensor: A review». *IEEE transactions on cybernetics*, **43(5)**, pp. 1318–1334
- HASHIMOTO, TAKUYA; HITRAMATSU, SACHIO; TSUJI, TOSHIAKI and KOBAYASHI, HIROSHI (2006). «Development of the face robot SAYA for rich facial expressions». *2006 SICE-ICASE International Joint Conference*, pp. 5423–5428. doi: 10.1109/SICE.2006.315537
- HASSOUN, M.H. (1995). *Fundamentals of Artificial Neural Networks*. A Bradford Book. MIT Press. ISBN 9780262082396
- HASTIE, T.; TIBSHIRANI, R. and FRIEDMAN, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. Springer Series in Statistics. Springer New York. ISBN 9780387848587
- HAYKIN, ET AL., S. (2001). *Kalman Filtering and Neural Networks*. ISBN 0471221546
- HAYKIN, SIMON S et al. (2009). «Neural networks and learning machines/Simon Haykin.»
- HERMANN, MARIO; PENTEK, TOBIAS and OTTO, BORIS (2016). «Design principles for industrie 4.0 scenarios». In: *System Sciences (HICSS), 2016 49th Hawaii International Conference on*, pp. 3928–3937. IEEE
- HOFFMANN, H; SCHENCK, W and MÖLLER, R (2005). «Learning visuomotor transformations for gaze-control and grasping». *Biological Cybernetics*, **93(2)**, pp. 119–130
- HOLDEN, A J and ET AL. (2006). «Reducing the Dimensionality of Data with Neural Networks». *Science (New York)*, **313(July)**, pp. 504–507. ISSN 0036-8075
- HOLLAND, OWEN and KNIGHT, ROB (2006). «The anthropomimetic principle». *Proceedings of AISB'06: Adaptation in Artificial and Biological Systems*, **2**, pp. 115–122
- HUDSON, D L and COHEN, M E (2000). *Neural networks and artificial intelligence for biomedical engineering*. ISBN 9780470545355\n0470545356
- ILG, EDDY; MAYER, NIKOLAUS; SAIKIA, TONMOY; KEUPER, MARGRET; DOSOVITSKIY, ALEXEY and BROX, THOMAS (2017). «Flownet 2.0: Evolution of optical flow estimation with deep networks». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2462–2470
- INTOY, JANIS and RUCCI, MICHELE (2020). «Finely tuned eye movements enhance visual acuity». *Nature Communications*, **11(1)**, pp. 1–11. ISSN 20411723. doi: 10.1038/s41467-020-14616-2

- IRWIN, D.E. (2003). «Eye movements and visual cognitive suppression». In: *Psychology of Learning and Motivation*, volume 42, pp. 265–293. Elsevier
- ISIDORI, ALBERTO; MARCONI, LORENZO and SERRANI, ANDREA (2003). «Fundamentals of internal-model-based control theory». In: *Robust Autonomous Guidance*, pp. 1–58. Springer
- JAIN, A. and BACKUS, B. T. (2010). «Experience affects the use of ego-motion signals during 3D shape perception». *Journal of vision*, **10(14)**, p. 30. ISSN 1534-7362. doi: 10.1167/10.14.30
- KANDEL, E.; SCHWARTZ, J. and JESSELL, T. (2000). *Principles of Neural Science, Fourth Edition*. McGraw-Hill Companies, Incorporated. ISBN 9780838577011
- KANEKO, KENJI; KANEHIRO, FUMIO; MORISAWA, MITSU HARU; AKACHI, KAZUHIKO; MIYAMORI, GO; HAYASHI, ATSUSHI and KANEHIRA, NORIYUKI (2011). «Humanoid robot HRP-4 - Humanoid robotics platform with lightweight and slim body». *IEEE International Conference on Intelligent Robots and Systems*, pp. 4400–4407. doi: 10.1109/IROS.2011.6048074
- KAWATO, MITSUO (1990). «Feedback-error-learning neural network for supervised motor learning». *Advanced neural computers*, **6(3)**, pp. 365–372
- KEHOE, BEN; PATIL, SACHIN; ABBEEL, PIETER and GOLDBERG, KEN (2015). «A survey of research on cloud robotics and automation». *IEEE Transactions on automation science and engineering*, **12(2)**, pp. 398–409
- KHAITAN, SIDDHARTHA KUMAR and MCCALLEY, JAMES D (2015). «Design techniques and applications of cyberphysical systems: A survey». *IEEE Systems Journal*, **9(2)**, pp. 350–365
- KIM, HYUNDO; YORK, GEORGE; BURTON, GREG; MURPHY-CHUTORIAN, ERIK and TRIESCH, JOCHEN (2004). «Design of an anthropomorphic robot head for studying autonomous development and learning». *Proceedings - IEEE International Conference on Robotics and Automation*, **2004(4)**, pp. 3506–3511. ISSN 10504729. doi: 10.1109/robot.2004.1308796
- KINGMA, DIEDERIK P and BA, JIMMY (2014). «Adam: A method for stochastic optimization». *arXiv preprint arXiv:1412.6980*
- KO, HEE KYOUNG; POLETTI, MARTINA and RUCCI, MICHELE (2010). «Microsaccades precisely relocate gaze in a high visual acuity task». *Nature Neuroscience*, **13(12)**, pp. 1549–1554. ISSN 10976256. doi: 10.1038/nn.2663

- KOHLER, MICHAEL; KRZYZAK, ADAM and WALK, HARRO (2009). «Optimal global rates of convergence for nonparametric regression with unbounded data». *Journal of Statistical Planning and Inference*, **139(4)**, pp. 1286–1296. ISSN 03783758. doi: 10.1016/j.jspi.2008.07.012
- KRAUZLIS, RICHARD J. and ET AL. (2017). «Neuronal control of fixation and fixational eye movements». *Philosophical Transactions of the Royal Society B: Biological Sciences*, **372(1718)**
- KUANG, XUTAO; GIBSON, MARK; SHI, BERTRAM E. and RUCCI, MICHELE (2012). «Active vision during coordinated head/eye movements in a humanoid Robot». *IEEE Transactions on Robotics*, **28(6)**, pp. 1423–1430. ISSN 15523098. doi: 10.1109/TRO.2012.2204513
- KUBAT, MIROSLAV; GAMA, JOAO and UTGOFF, PAUL (2004). «Incremental learning and concept drift: Editors introduction». *Intelligent Data Analysis*, **8(3)**, pp. 211–212. ISSN 1088467X. doi: 10.3233/ida-2004-8301
- KULLBACK, S. and LEIBLER, R. A. (1951). «On Information and Sufficiency». *Ann. Math. Statist.*, **22(1)**, pp. 79–86. doi: 10.1214/aoms/1177729694
- KUMAR, ARAN C.S.; BHANDARKAR, SUCHENDRA M. and PRASAD, MUKTA (2018). «Depthnet: A recurrent neural network architecture for monocular depth prediction». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 283–291
- KUPERSTEIN, M. (1988). «Neural Model of Adaptive Hand-Eye Coordination for Single Postures». *Science*, **239**, pp. 1308–1311
- LAPPE, M (2008). «What is adapted in saccadic adaptation?» *The Journal of Physiology*, **587(1)**, pp. 5–5. doi: 10.1113/jphysiol.2008.166900
- LAPPE, MARKUS (2009). «What is adapted in saccadic adaptation?» *Journal of Physiology*, **587(1)**, p. 5. ISSN 00223751. doi: 10.1113/jphysiol.2008.166900
- LÁZARO-GREDILLA, MIGUEL; QUIÑONERO-CANDELA, JOAQUIN; RASMUSSEN, CARL EDWARD and FIGUEIRAS-VIDAL, ANÍBAL R (2010). «Sparse spectrum Gaussian process regression». *The Journal of Machine Learning Research*, **11**, pp. 1865–1881
- LECUN, YANN; BENGIO, YOSHUA and HINTON, GEOFFREY (2015). «Deep learning». *Nature*, **521**, pp. 436—444

- LENZ, A.; BALAKRISHNAN, T.; PIPE, A. G. and MELHUISE, C. (2008). «An adaptive gaze stabilization controller inspired by the vestibulo-ocular reflex». *Bioinspiration and Biomimetics*, **3(3)**, pp. 1–18. ISSN 17483182. doi: 10.1088/1748-3182/3/3/035001
- L'HEUREUX, A; GROLINGER, K; ELYAMANY, H F and CAPRETZ, M A M (2017). «Machine Learning With Big Data: Challenges and Approaches». *IEEE Access*, **5**, pp. 7776–7797. ISSN 2169-3536. doi: 10.1109/ACCESS.2017.2696365
- LINGAMPALLY, PAVAN KALYAN and AROCKIA SELVAKUMAR, A. (2017). «A humanoid neck using parallel manipulators». *International Conference on Robotics and Automation for Humanitarian Applications, RAHA 2016 - Conference Proceedings*. doi: 10.1109/RAHA.2016.7931868
- LIU, BANGLI; CAI, HAIBIN; JU, ZHAOJIE and LIU, HONGHAI (2019). «RGB-D sensing based human action and interaction analysis: A survey». *Pattern Recognition*, **94**, pp. 1–12. ISSN 00313203. doi: 10.1016/j.patcog.2019.05.020
- LIU, FAYAO; SHEN, CHUNHUA and LIN, GUOSHENG (2015). «Deep convolutional neural fields for depth estimation from a single image». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5162–5170
- LOFTUS, GEOFFREY R and MACKWORTH, NORMAN H (1978). «Cognitive determinants of fixation location during picture viewing.» *Journal of Experimental Psychology: Human perception and performance*, **4(4)**, p. 565
- LOMONACO, V. and MALTONI, D. (2017). «Core50: a new dataset and benchmark for continuous object recognition». *arXiv preprint arXiv:1705.03550*
- LÓPEZ-CERÓN, ALBERTO and CANAS, JOSÉ M (2016). «Accuracy analysis of marker-based 3D visual localization». In: *XXXVII Jornadas de Automatica Workshop*,
- LUCAS, BRUCE D. and KANADE, TAKEO (1981). «An Iterative Image Registration Technique with an Application to Stereo Vision». In: *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2, IJCAI'81*, pp. 674–679. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA
- MACKWORTH, NORMAN H and MORANDI, ANTHONY J (1967). «The gaze selects informative details within pictures». *Perception & psychophysics*, **2(11)**, pp. 547–552
- MALLOT, H.A.; ALLEN, J.S. and ALLEN, R.S.J.S. (2000). *Computational Vision: Information Processing in Perception and Visual Behaviour*. A Bradford book. MIT Press. ISBN 9780262133814

- MALMIR ET AL, M. (2017). «Deep active object recognition by joint label and action prediction». *Computer Vision and Image Understanding*, **156**, pp. 128–137
- MANDELBROT, BENOIT (1960). «The Pareto-Lévy Law and the Distribution of Income». *International Economic Review*, **1(2)**, pp. 79–106. ISSN 00206598, 14682354. doi: 10.2307/2525289
- MARJANOVIC, MJ; SCASSELLATI, B and WILLIAMSON, MM (1996). «Self-taught visually guided pointing for a humanoid robot». *From Animals to Animats 4: Simulation of Adaptive Behavior*, pp. 35–44
- MARTINEZ-CONDE, S.; MACKNIK, S. L. and HUBEL, D. H. (2004). «The role of fixational eye movements in visual perception». *Nature Reviews Neuroscience*, **5(3)**, pp. 229–240. ISSN 1471003X. doi: 10.1038/nrn1348
- MARZOCCHI, NICOLETTA; BREVEGLIERI, ROSSELLA; GALLETI, CLAUDIO and FATTORI, PATRIZIA (2008). «Reaching activity in parietal area V6A of macaque: Eye influence on arm activity or retinocentric coding of reaching movements?» *European Journal of Neuroscience*, **27(3)**, pp. 775–789. ISSN 0953816X
- MATTHIES, LARRY; SZELISKI, RICHARD and KANADE, TAKEO (1993). «Kalman Filter-based Algorithms for Estimating Depth from Image Sequences». *Multisensor Fusion for Computer Vision*, **236**, pp. 87–130. ISSN 1573-1405. doi: 10.1007/978-3-662-02957-2_6
- MCBRIDE, S; LAW, J and LEE, M (2010). «Integration of Active Vision and Reaching from a Developmental Robotics Perspective». *Autonomous Mental Development, IEEE Transactions on*, **2(4)**, pp. 355–366
- MENG, LINGHENG; DING, SHIFEI and XUE, YU (2017). «Research on denoising sparse autoencoder». *International Journal of Machine Learning and Cybernetics*, **8(5)**, pp. 1719–1729. ISSN 1868808X. doi: 10.1007/s13042-016-0550-y
- METTA, GIORGIO; SANDINI, GIULIO; VERNON, DAVID; NATALE, LORENZO and NORI, FRANCESCO (2008). «The iCub humanoid robot: an open platform for research in embodied cognition». In: *Proceedings of the 8th workshop on performance metrics for intelligent systems*, pp. 50–56
- MOLCHANOV, PAVLO; TYREE, STEPHEN; KARRAS, TERO; AILA, TIMO and KAUTZ, JAN (2019). «Pruning convolutional neural networks for resource efficient inference». *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, (2015), pp. 1–17

- MØLLER, MARTIN F (1993). «A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning Supervised Learning». *Neural Networks*, **6**, pp. 525–533. ISSN 08936080. doi: 10.1016/S0893-6080(05)80056-5
- MOODY, JOHN and DARKEN, CHRISTIAN J (1989). «Fast learning in networks of locally-tuned processing units». *Neural computation*, **1(2)**, pp. 281–294
- MOSCHOVAKIS, ADONIS K. (1996). «The superior colliculus and eye movement control». *Current Opinion in Neurobiology*, **6(6)**, pp. 811–816. ISSN 09594388. doi: 10.1016/S0959-4388(96)80032-8
- MUNOZ, DOUGLAS P and WURTZ, ROBERT H (1995). «Saccade-related activity in monkey superior colliculus. I. Characteristics of burst and buildup cells». *Journal of neurophysiology*, **73(6)**, pp. 2313–2333
- MURRAY, D; DU, F; MCLAUCHLAN, P; REID, I; SHARKEY, PAUL and BRADY, J (1992). «Design of Stereo Heads», pp. 155–174
- NADLER, JACOB W; ANGELAKI, DORA E and DEANGELIS, GREGORY C (2008). «A neural representation of depth from motion parallax in macaque visual cortex». *Nature*, **452(7187)**, p. 642
- NG, ANDREW (2011). «Sparse autoencoder». *CS294A Lecture notes*, **72**, pp. 1–19
- NGUYEN-TUONG, DUY and PETERS, JAN (2011). «Model learning for robot control: a survey». *Cognitive Processing*, **12(4)**, pp. 319–340. ISSN 1612-4782. doi: 10.1007/s10339-011-0404-1
- NORI, F; NATALE, L; SANDINI, G and METTA, G (2007). «Autonomous learning of 3d reaching in a humanoid robot». *IEEE/RSJ IROS, International Conference on Intelligent Robots and Systems*, pp. 1142–1147
- OHZAWA, IZUMI; DEANGELIS, GREGORY C and FREEMAN, RALPH D (1990). «Stereoscopic depth discrimination in the visual cortex: neurons ideally suited as disparity detectors». *Science*, **249(4972)**, pp. 1037–1041
- OUCHI, HAJIME (2013). *Japanese optical and geometrical art*. Courier Corporation
- PARK, JOOYOUNG and SANDBERG, IRWIN W (1991). «Universal approximation using radial-basis-function networks». *Neural computation*, **3(2)**, pp. 246–257
- PATEL, VATSAL; KRISHNAN, SANJAY; GONCALVES, AIMEE and GOLDBERG, KEN (2018). «SPRK: A low-cost stewart platform for motion study in surgical robotics». *2018 International Symposium on Medical Robotics, ISMR 2018, 2018-Janua(Figure 2)*, pp. 1–6. doi: 10.1109/ISMR.2018.8333300

- PÉREZ-SÁNCHEZ, BEATRIZ; FONTENLA-ROMERO, OSCAR and GUIJARRO-BERDIÑAS, BERTHA (2018). «A review of adaptive online learning for artificial neural networks». *Artificial Intelligence Review*, **49(2)**, pp. 281–299. ISSN 15737462. doi: 10.1007/s10462-016-9526-2
- PETERSL, KLAUS HAVING and GABRIELE (2010). «the Structure-From-Motion Reconstruction Pipeline – a Survey With Focus on Short Image Sequences». *Kybernetika*, **46(5)**, pp. 926–937. ISSN 00235954
- PFEIFER, R; LUNGARELLA, M and IIDA, F (2007). «Self-Organization, Embodiment, and Biologically Inspired Robotics». *Science*, **318(November)**, pp. 1088–1093. ISSN 0036-8075. doi: 10.1126/science.1145803
- PIGLIUCCI, MASSIMO (2001). *Phenotypic plasticity: beyond nature and nurture*. JHU Press
- PIGLIUCCI, MASSIMO (2010). «Genotype–phenotype mapping and the end of the ‘genes as blueprint’ metaphor». *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, **365(1540)**, pp. 557–566. ISSN 0962-8436. doi: 10.1098/rstb.2009.0241
- PINARD, CLÉMENT; CHEVALLEY, LAURE; MANZANERA, ANTOINE and FILLIAT, DAVID (2018). «Learning structure-from-motion from motion». In: *Proceedings of the European Conference on Computer Vision (ECCV)*,
- POGGI, MATTEO and ET. AL. (2018). «Towards real-time unsupervised monocular depth estimation on cpu». In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5848–5854. IEEE
- POLETTI, MARTINA; INTOY, JANIS and RUCCI, MICHELE (2020). «Accuracy and precision of small saccades». *Scientific Reports*, **10(1)**, pp. 1–13. ISSN 20452322
- PORRILL, JOHN and DEAN, PAUL (2007). «Recurrent cerebellar loops simplify adaptive control of redundant and nonlinear motor systems». *Neural computation*, **19(1)**, p. 170—193. ISSN 0899-7667. doi: 10.1162/neco.2007.19.1.170
- PORRILL, JOHN; DEAN, PAUL and STONE, JAMES V. (2004). «Recurrent cerebellar architecture solves the motor-error problem». *Proceedings of the Royal Society B: Biological Sciences*, **271(1541)**, pp. 789–796. ISSN 14712970. doi: 10.1098/rspb.2003.2658
- POWELL, M. J. D. (1987). «Radial Basis Functions for Multivariable Interpolation: A Review». In: *Algorithms for Approximation*, pp. 143–167. Clarendon Press, Oxford

- PRINCE, SJD; CUMMING, BG and PARKER, AJ (2002). «Range and mechanism of encoding of horizontal disparity in macaque V1». *Journal of Neurophysiology*, **87(1)**, pp. 209–221
- PRITCHARD, R. M.; HERON, W. and HEBB, D. O. (1960). «Visual perception approached by the method of stabilized images.» *Canadian journal of psychology*, **14(Ap 17)**, pp. 67–77. ISSN 00084255. doi: 10.1037/h0083168
- QUIGLEY, MORGAN; CONLEY, KEN; GERKEY, BRIAN; FAUST, JOSH; FOOTE, TULLY; LEIBS, JEREMY; WHEELER, ROB and NG, ANDREW Y (2009). «ROS: an open-source Robot Operating System». In: *ICRA workshop on open source software*, volume 3, p. 5. Kobe, Japan
- RASMUSSEN, CARL EDWARD (2003). «Gaussian processes in machine learning». In: *Summer School on Machine Learning*, pp. 63–71. Springer
- RASOULI, AMIR and TSOTSOS, JOHN K (2017). «The Effect of Color Space Selection on Detectability and Discriminability of Colored Objects». *arXiv preprint arXiv:1702.05421*
- RIFAI, SALAH and MULLER, XAVIER (2011). «Contractive Auto-Encoders : Explicit Invariance During Feature Extraction». *Icml*, **85(1)**, pp. 833–840
- ROHDE, MARIEKE; NARIOKA, KENICHI; STEIL, JOCHEN J.; KLEIN, LINA K. and ERNST, MARC O. (2019). «Goal-related feedback guides motor exploration and redundancy resolution in human motor skill acquisition». *PLoS Computational Biology*, **15(3)**, pp. 1–27. ISSN 15537358. doi: 10.1371/journal.pcbi.1006676
- ROLFS, MARTIN (2009). «Microsaccades: Small steps on a long way». *Vision Research*, **49(20)**, pp. 2415–2441. ISSN 00426989
- ROLLS, E. and DECO, G. (2001). *Computational Neuroscience of Vision*. OUP Oxford. ISBN 9780198524885
- RUCCI, M.; EDELMAN, G. M. and WRAY, J. (1999). «Adaptation of orienting behavior: from the barn owl to a robotic system». *IEEE Transactions on Robotics and Automation*, **15(1)**, pp. 96–110. ISSN 1042-296X. doi: 10.1109/70.744606
- RUCCI, M.; WRAY, J. and EDELMAN, G. M. (2000). «Robust localization of auditory and visual targets in a robotic barn owl». *Robotics and Autonomous Systems*, **30(1)**, pp. 181 – 193. ISSN 0921-8890

- RUCCI, MICHELE; BULLOCK, DANIEL and SANTINI, FABRIZIO (2007). «Integrating robotics and neuroscience: Brains for robots, bodies for brains». *Advanced Robotics*, **21(10)**, pp. 1115–1129. ISSN 01691864. doi: 10.1163/156855307781389428
- SAEGUSA, R; METTA, G; SANDINI, G and SAKKA, S (2008). «Active motor babbling for sensorimotor learning». *IEEE International Conference on Robotics and Biomimetics*, pp. 794–799. doi: 10.1109/ROBIO.2009.4913101
- SAKAGAMI, YOSHIAKI; WATANABE, RYUJIN; AOYAMA, CHIAKI; MATSUNAGA, SHINICHI; HIGAKI, NOBUO and FUJIMURA, KIKUO (2002). «The intelligent ASIMO: System overview and integration». *IEEE International Conference on Intelligent Robots and Systems*, **3(October)**, pp. 2478–2483. doi: 10.1109/irids.2002.1041641
- SANTINI, FABRIZIO; NAMBISAN, ROHIT and RUCCI, MICHELE (2009). «Active 3D vision through gaze relocation in a humanoid robot». *International Journal of Humanoid Robotics*, **6(3)**, pp. 481–503
- SANTINI, FABRIZIO and RUCCI, MICHELE (2007). «Active estimation of distance in a robotic system that replicates human eye movement». *Robot. Auton. Syst.*, **55(2)**, pp. 107–121. ISSN 0921-8890. doi: 10.1016/j.robot.2006.07.001
- SAXENA, ASHUTOSH; SUN, MIN and NG, ANDREW Y. (2009). «Make3D: Learning 3D scene structure from a single still image». *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **31(5)**, pp. 824–840. ISSN 01628828. doi: 10.1109/TPAMI.2008.132
- SCHALL, JD (1991). «Neuronal Activity related to Visually Guided Saccades e Frontal Eye Fields s Monkeys :». *Journal of neurophysiology*, **66(2)**, pp. 559–579
- SCHENCK, W. and MÖLLER, R. (2004). «Staged Learning of Saccadic Eye Movements With a Robot Camera Head», pp. 82–91. doi: 10.1142/9789812702784_0008
- SCHENCK, W and MÖLLER, R (2006). «Learning strategies for saccade control». *Künstliche Intelligenz*, **(3/06)**, pp. 19–22
- SCHONBERGER, JOHANNES L. and FRAHM, JAN-MICHAEL (2016). «Structure-from-Motion Revisited». *2016 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4104–4113. ISSN 10636919. doi: 10.1109/CVPR.2016.445
- SHAMSUDDIN, SYAMIMI; ISMAIL, LUTHFFI IDZHAR; YUSSOF, HANAFIAH; ZAHARI, NUR ISMARRUBIE; BAHARI, SAIFUL; HASHIM, HAFIZAN and JAFFAR, AHMED (2011). «Humanoid robot NAO: Review of control and motion exploration». *Proceedings - 2011 IEEE International Conference on Control System, Computing and Engineering, ICCSCE 2011*, pp. 511–516. doi: 10.1109/ICCSCE.2011.6190579

- SHARKEY, P. M.; MURRAY, D. W.; VANDEVELDE, S.; REID, I. D. and McLAUCHLAN, P. F. (1993). «A modular head/eye platform for real-time reactive vision». *Mechatronics*, **3(4)**, pp. 517–535. ISSN 09574158. doi: 10.1016/0957-4158(93)90021-S
- SHIBATA, T.; VIJAYAKUMAR, S.; CONRADT, J. and SCHAAL, S. (2001). «Biomimetic Oculomotor Control». *Adaptive Behavior*, **9(3-4)**, pp. 189–207. ISSN 1059-7123. doi: 10.1177/10597123010093005
- SICILIANO, B and ET AL. (2008). *Robotics: Modelling, Planning and Control*. Advanced Textbooks in Control and Signal Processing. Springer London. ISBN 9781846286421
- SIGAUD, OLIVIER; SALAN, CAMILLE and PADOIS, VINCENT (2011). «On-line regression algorithms for learning mechanical models of robots: A survey». *Robotics and Autonomous Systems*, **59(12)**, pp. 1115–1129. ISSN 09218890. doi: 10.1016/j.robot.2011.07.006
- SPILLMANN, L; TULUNAYKEESEY, U and OLSON, J (1993). «Apparent floating motion in normal and stabilized vision». In: *Investigative Ophthalmology & Visual Science*, volume 34, pp. 1031–1031. LIPPINCOTT-RAVEN PUBL 227 EAST WASHINGTON SQ, PHILADELPHIA, PA 19106
- SQUIRE, LARRY; BERG, DARWIN; BLOOM, FLOYD E; DU LAC, SASCHA; GHOSH, ANIRVAN and SPITZER, NICHOLAS C (2012). *Fundamental neuroscience*. Academic Press, 3rd edition
- SRINIVAS, M. and PATNAIK, L. M. (1994). «Genetic algorithms: a survey». *Computer*, **27(6)**, pp. 17–26. ISSN 0018-9162. doi: 10.1109/2.294849
- SRIVASTAVA, NITISH; HINTON, GEOFFREY; KRIZHEVSKY, ALEX; SUTSKEVER, ILYA and SALAKHUTDINOV, RUSLAN (2014). «Dropout: a simple way to prevent neural networks from overfitting». *The Journal of Machine Learning Research*, **15(1)**, pp. 1929–1958
- STASSE, O.; FLAYOLS, T.; BUDHIRAJA, R.; GIRAUD-ESCLASSE, K.; CARPENTIER, J.; MIRABEL, J.; DEL PRETE, A.; SOUERES, P.; MANSARD, N.; LAMIRAUX, F.; LAUMOND, J. P.; MARCHIONNI, L.; TOME, H. and FERRO, F. (2017). «TALOS: A new humanoid research platform targeted for industrial applications». *IEEE-RAS International Conference on Humanoid Robots*, pp. 689–695. ISSN 21640580. doi: 10.1109/HUMANOIDS.2017.8246947
- SU, V. X. and DUAN, B. Y. (2000). «Application of the Stewart platform in large spherical radio telescopes». *Journal of Robotic Systems*, **17(7)**, pp. 375–383. ISSN 07412223. doi: 10.1002/1097-4563(200007)17:7<375::AID-ROB3>3.0.CO;2-7

- SÜNDERHAUF, NIKO; BROCK, OLIVER; SCHEIRER, WALTER; HADSELL, RAIA; FOX, DIETER; LEITNER, JÜRGEN; UPCROFT, BEN; ABBEEL, PIETER; BURGARD, WOLFRAM; MILFORD, MICHAEL and CORKE, PETER (2018). «The limits and potentials of deep learning for robotics». *International Journal of Robotics Research*, **37(4-5)**, pp. 405–420. ISSN 17413176. doi: 10.1177/0278364918770733
- SUNDEHAUF ET AL., NIKO (2018). «The limits and potentials of deep learning for robotics». *The International Journal of Robotics Research*, **37(4-5)**, pp. 405–420. doi: 10.1177/0278364918770733
- SZCZEPANSKI, SARA M and SAALMANN, YURI B (2013). «Human fronto-parietal and parieto-hippocampal pathways represent behavioral priorities in multiple spatial reference frames». *Bioarchitecture*, **3(5)**, pp. 147–152. ISSN 1949-100X. doi: 10.4161/bioa.27462
- SZUFNAROWSKI, FILIP (2013). «Stewart platform with fixed rotary actuators : a low cost design study». *Faculty of Technology, Bielefeld University, Germany*, pp. 1–11. ISSN 0094114X. doi: 10.1016/S0094-114X(99)00006-3
- TEIXEIRA, JOÃO C. and RODRIGUES, ANTONIO J. (1997). «An applied study on recursive estimation methods, neural networks and forecasting». *European Journal of Operational Research*, **101(2)**, pp. 406–417. ISSN 03772217. doi: 10.1016/S0377-2217(96)00406-7
- THIER, P.; JUNKER, M.; ARNSTEIN, D.; SMILGIN, A. and DICKE, P. W. (2015). «Microsaccade Control Signals in the Cerebellum». *Journal of Neuroscience*, **35(8)**, pp. 3403–3411. ISSN 0270-6474. doi: 10.1523/jneurosci.2458-14.2015
- TRUONG, HARLEY; ABDALLAH, SAMER; ROUGEAUX, SEBASTIEN and ZELINSKY, ALEXANDER (2000). «A Novel Mechanism for Stereo Active Vision». *Australian Conference on Robotics and Automation*
- UEXKÜLL, J VON (1926). *Theoretical biology*. Harcourt, Brace & Co.
- UMMENHOFER, BENJAMIN; ZHOU, HUIZHONG; UHRIG, JONAS; MAYER, NIKOLAUS; ILG, EDDY; DOSOVITSKIY, ALEXEY and BROX, THOMAS (2017). «Demon: Depth and motion network for learning monocular stereo». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5038–5047
- UPTON, EBEN and HALFACREE, GARETH (2014). *Raspberry Pi user guide*. John Wiley & Sons

- VAUGHAN, RICHARD and ZULUAGA, MAURICIO (2006). «Use Your Illusion: Sensorimotor Self-simulation Allows Complex Agents to Plan with Incomplete Self-knowledge». In: Stefano Nolfi; Gianluca Baldassarre; Raffaele Calabretta; John C. T. Hallam; Davide Marocco; Jean-Arcady Meyer; Orazio Miglino and Domenico Parisi (Eds.), *From Animals to Animats 9*, pp. 298–309. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-540-38615-5
- VILEE, CALUDE A. (2019). «Morphology»
<https://www.britannica.com/science/morphology-biology>
- VOJNIKOVIĆ, BOZO and TAMAJO, ETTORE (2013). «Gullstrand's optical schematic system of the eye modified by Vojnikovic & Tamajo.» *Collegium antropologicum*, **37 Suppl 1**, pp. 41–5. ISSN 0350-6134
- WAIBEL, MARKUS; BEETZ, MICHAEL; CIVERA, JAVIER; D'ANDREA, RAFFAELLO; ELFRING, JOS; GALVEZ-LOPEZ, DORIAN; HÄUSSERMANN, KAI; JANSSEN, ROB; MONTIEL, JMM; PERZYLO, ALEXANDER et al. (2011). «Roboearth». *IEEE Robotics & Automation Magazine*, **18(2)**, pp. 69–82
- WAN, JIAFU; TANG, SHENGLONG; YAN, HEHUA; LI, DI; WANG, SHIYONG and VASILAKOS, ATHANASIOS V (2016). «Cloud robotics: Current status and open issues». *IEEE Access*, **4**, pp. 2797–2807
- WANG, Y M; LI, Y and ZHENG, J B (2010). «A camera calibration technique based on OpenCV». In: *The 3rd International Conference on Information Sciences and Interaction Sciences*, pp. 403–406. doi: 10.1109/ICICIS.2010.5534797
- WAPLER, MATTHIAS; URBAN, VOLKER; WEISENER, THOMAS; STALLKAMP, JAN; DÜRR, MARK and HILLER, ANDREA (2003). «A Stewart platform for precision surgery». *Transactions of the Institute of Measurement & Control*, **25(4)**, pp. 329–334. ISSN 01423312. doi: 10.1191/0142331203tm092oa
- WHITLEY, DARRELL and SUTTON, ANDREW M. (2012). *Handbook of Natural Computing*. chapter Genetic Algorithms — A Survey of Models and Methods, pp. 637–671. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-540-92910-9. doi: 10.1007/978-3-540-92910-9_21
- WIXTED, J.T.; PHELPS, E.A. and DAVACHI, L. (2018). *Stevens' Handbook of Experimental Psychology and Cognitive Neuroscience, Learning and Memory*. Stevens' Handbook of Experimental Psychology and Cognitive Neuroscience. Wiley. ISBN 9781119170037

- WOLPERT, D M; MIAL, R C and KAWATO, M (1998). «Internal models in the cerebellum.» *Trends in cognitive sciences*, **2(9)**, pp. 338–47. ISSN 1364-6613
- WU, YUXIN and HE, KAIMING (2018). «Group normalization». In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 3–19
- XU, DAN; WANG, WEI; TANG, HAO; LIU, HONG; SEBE, NICU and RICCI, ELISA (2018). «Structured attention guided convolutional neural fields for monocular depth estimation». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3917–3925
- YARBUS, ALFRED L (2013). *Eye movements and vision*. Springer
- YE, JIANMING (1998). «On measuring and correcting the effects of data mining and model selection». *Journal of the American Statistical Association*, **93(441)**, pp. 120–131. ISSN 1537274X. doi: 10.1080/01621459.1998.10474094
- YU, YANG (2018). «Towards Sample Efficient Reinforcement Learning». In: *Twenty-Seventh International Joint Conference on Artificial Intelligence*, pp. 5739–5743. IJCAI
- ZEILER, M. D. (2012). «ADADELTA: an adaptive learning rate method». *arXiv preprint arXiv:1212.5701*
- ZHOU, TINGHUI; BROWN, MATTHEW; SNAVELY, NOAH and LOWE, DAVID G (2017). «Unsupervised learning of depth and ego-motion from video». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1851–1858
- ZHOU WANG; BOVIK, A. C.; SHEIKH, H. R. and SIMONCELLI, E. P. (2004). «Image quality assessment: from error visibility to structural similarity». *IEEE Transactions on Image Processing*, **13(4)**, pp. 600–612. ISSN 1941-0042. doi: 10.1109/TIP.2003.819861
- ZHU, MICHAEL H. and GUPTA, SUYOG (2018). «To prune, or not to prune: Exploring the efficacy of pruning for model compression». *6th International Conference on Learning Representations, ICLR 2018 - Workshop Track Proceedings*