UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Departament de Teoria del Senyal
i Comunicacions

LEARNING TO EXTRACT FEATURES FOR 2D - 3D MULTIMODAL REGISTRATION

PhD Dissertation

Author
Alba Pujol-Miró

A DISSERTATION PRESENTED TO
THE DEPARTMENT OF TEORIA DEL SENYAL I COMUNICACIONS
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE IN
DOCTORAT EN TEORIA DEL SENYAL I COMUNICACIONS

Advisors
Javier Ruiz-Hidalgo
Josep R. Casas

Universitat Politècnica de Catalunya
March 2020

Thesis advisors: Professors Javier Ruiz-Hidalgo and Josep R. Casas Alba Pujol-Miró

Learning to extract features for 2D - 3D multimodal registration

Abstract

The ability to capture depth information form an scene has greatly increased in the recent years. 3D sensors, traditionally high cost and low resolution sensors, are being democratized and 3D scans of indoor and outdoor scenes are becoming more and more common.

However, there is still a great data gap between the amount of captures being performed with 2D and 3D sensors. Although the 3D sensors provide more information about the scene, 2D sensors are still more accessible and widely used. This trade-off between availability and information between sensors brings us to a multimodal scenario of mixed 2D and 3D data.

This thesis explores the fundamental block of this multimodal scenario: the registration between a single 2D image and a single unorganized point cloud. An unorganized 3D point cloud is the basic representation of a 3D capture. In this representation the surveyed points are represented only by their real word coordinates and, optionally, by their colour information. This simplistic representation brings multiple challenges to the registration, since most of the state of the art works leverage the existence of metadata about the scene or prior knowledges.

Two different techniques are explored to perform the registration: a keypoint-based technique and an edge-based technique. The keypoint-based technique estimates the transformation by means of correspondences detected using Deep Learning, whilst the edge-based technique refines a transformation using a multimodal edge detection to establish anchor points to perform the estimation.

An extensive evaluation of the proposed methodologies is performed. Albeit further research is needed to achieve adequate performances, the obtained results show the potential of the usage of deep learning techniques to learn 2D and 3D similarities. The results also show the good performance of the proposed 2D-3D iterative refinement, up to the state of the art on 3D-3D registration.

Keywords: 3D, Deep learning, Multi-domain, Feature detection

UNESCO codes: 120304 - Artificial intelligence, 120903 - Data analysis

Directors: Professors Javier Ruiz-Hidalgo and Josep R. Casas Alba Pujol-Miró

Learning to extract features for 2D - 3D multimodal registration

Resum

La capacitat de captar informació de profunditat d'una escena ha augmentat molt els darrers anys. Els sensors 3D, tradicionalment d'alt cost i baixa resolució, s'estan democratitzant i escànners 3D d'escents interiors i exteriors són cada vegada més comuns.

Tot i això, encara hi ha una gran bretxa entre la quantitat de captures que s'estan realitzant amb sensors 2D i 3D. Tot i que els sensors 3D proporcionen més informació sobre l'escena, els sensors 2D encara són més accessibles i àmpliament utilitzats. Aquesta diferència entre la disponibilitat i la informació entre els sensors ens porta a un escenari multimodal de dades mixtes 2D i 3D.

Aquesta tesi explora el bloc fonamental d'aquest escenari multimodal: el registre entre una sola imatge 2D i un sol núvol de punts no organitzat. Un núvol de punts 3D no organitzat és la representació bàsica d'una captura en 3D. En aquesta representació, els punts mesurats es representen només per les seves coordenades i, opcionalment, per la informació de color. Aquesta representació simplista aporta múltiples reptes al registre, ja que la majoria dels algoritmes aprofiten l'existència de metadades sobre l'escena o coneixements previs.

Per realitzar el registre s'exploren dues tècniques diferents: una tècnica basada en punts clau i una tècnica basada en contorns. La tècnica basada en punts clau estima la transformació mitjançant correspondències detectades mitjançant Deep Learning, mentre que la tècnica basada en contorns refina una transformació mitjançant una detecció multimodal de la vora per establir punts d'ancoratge per realitzar l'estimació.

Es fa una avaluació àmplia de les metodologies proposades. Tot i que es necessita més investigació per obtenir un rendiment adequat, els resultats obtinguts mostren el potencial de l'ús de tècniques d'aprenentatge profund per aprendre similituds 2D i 3D. Els resultats també mostren l'excel · lent rendiment del perfeccionament iteratiu 2D-3D proposat, similar al dels algoritmes de registre 3D-3D.

Directores: Professors Javier Ruiz-Hidalgo and Josep R. Casas Alba Pujol-Miró

Learning to extract features for 2D - 3D multimodal registration

Resumen

La capacidad de captar información de profundidad de una escena ha aumentado mucho en los últimos años. Los sensores 3D, tradicionalmente de alto costo y baja resolución, se están democratizando y escáneres 3D de escents interiores y exteriores son cada vez más comunes.

Sin embargo, todavía hay una gran brecha entre la cantidad de capturas que se están realizando con sensores 2D y 3D. Aunque los sensores 3D proporcionan más información sobre la escena, los sensores 2D todavía son más accesibles y ampliamente utilizados. Esta diferencia entre la disponibilidad y la información entre los sensores nos lleva a un escenario multimodal de datos mixtos 2D y 3D.

Esta tesis explora el bloque fundamental de este escenario multimodal: el registro entre una sola imagen 2D y una sola nube de puntos no organizado. Una nube de puntos 3D no organizado es la representación básica de una captura en 3D. En esta representación, los puntos medidos se representan sólo por sus coordenadas y, opcionalmente, por la información de color. Esta representación simplista aporta múltiples retos en el registro, ya que la mayoría de los algoritmos aprovechan la existencia de metadatos sobre la escena o conocimientos previos.

Para realizar el registro se exploran dos técnicas diferentes: una técnica basada en puntos clave y una técnica basada en contornos. La técnica basada en puntos clave estima la transformación mediante correspondencias detectadas mediante Deep Learning, mientras que la técnica basada en contornos refina una transformación mediante una detección multimodal del borde para establecer puntos de anclaje para realizar la estimación.

Se hace una evaluación amplia de las metodologías propuestas. Aunque se necesita más investigación para obtener un rendimiento adecuado, los resultados obtenidos muestran el potencial del uso de técnicas de aprendizaje profundo para aprender similitudes 2D y 3D. Los resultados también muestran el excelente rendimiento del perfeccionamiento iterativo 2D-3D propuesto, similar al de los algoritmos de registro 3D-3D.

Contents

Listing of figures

Listing of tables

Acronyms

CNN Convolutional Neural Network. 15, 18, 31, 32, 36, 69, 72, 73, 81, 90, 93, 131, 141, 149, 150, 164

CT Computed Tomography. 10, 12

DL Deep Learning. 131

FPFH Fast Point Feature Histogram. 92, 131, 198

GNN Graph Neural Network. 141

GPU Graphic Processing Unit. 158

HED Holistically-Nested Edge Detection. 108, 109

ICP Iterative Closest Point. xvi, xviii, 19, 47, 48, 57, 58, 60–63, 70, 95, 112, 123, 124, 126, 129, 130, 132, 133, 140, 191, 194–198

ISS Intrinsic Shape Signatures. 17, 24, 25, 74–78, 80, 82, 162

LiDAR Laser Imaging Detection and Ranging. 3, 5, 6, 9–12, 15, 26

LM Levenberg–Marquardt. 42, 61, 118, 129, 185, 195

MLESAC Maximum Likelihood Estimation Sample Consensus. 43

MRI Magnetic Ressnance Imaging. 10, 12, 47, 48

PCA Principal Component Analysis. 50, 98

RANSAC RAndom SAmple Consensus. xvi, 18, 38–41, 43, 44, 61, 70, 112, 116–118, 127, 128, 131, 132, 140, 185–187, 195, 198

RMSE Root Mean Squared Error. xvi, xviii, 62, 70, 118–121, 124, 126–129, 133, 153–155, 188, 190–192, 194, 196, 198

SIFT Scale Invariant Feature Transform. 17

SLAM Simultaneous Localization And Mapping. 48

SVD Singular Value Decomposition. 40, 129

ToF Time of Flight. 1–3

TSDF Truncated Signal Distance Function. 152

Related publications

Pujol-Miró, A., Casas, J. R., & Ruiz-Hidalgo, J. (2019). Correspondence matching in unorganized 3D point clouds using Convolutional Neural Networks. *Image and Vision Computing.*
DOI 10.1016/j.imavis.2019.02.013.

Pujol-Miro, A., Ruiz-Hidalgo, J., & Casas, J. R. (2017, August). Registration of images to unorganized 3D point clouds using contour cues. In *2017 25th European Signal Processing Conference (EUSIPCO)* (pp. 81-85). IEEE.
DOI 10.23919/EUSIPCO.2017.8081173

Casanova, A., **Pujol-Miró, A.**, Ruiz-Hidalgo, J., & Casas, J. R. (2016, December). Interactive registration method for 3D data fusion. In *2016 International Conference on 3D Imaging (IC3D)* (pp. 1-8). IEEE.
DOI 10.1109/IC3D.2016.7823456

The research presented on these publications was supported by the Spanish Ministry of Science and Innovation via a doctoral grant to the author (FPU14/05256) and developed in the framework of projects TEC2013-43935-R and TEC2016-75976-R, financed by the Ministerio de Economía, Industria y Competitividad and the European Regional Development Fund (ERDF).

To Alex.

Acknowledgments

This thesis would not have been possible without the support from many people. I would like to thank my family and friends for their unconditional help and encouragement throughout all the steps of this journey.

I would like to especially thank Alex, who not only provided emotional support but also become an on-call mathematician for insightful discussions.

I would also like to thank my advisors Javier Ruiz and Josep R. Casas for their tutoring throughout this thesis, providing many useful insights and guidance to develop the research described in this document.

This thesis would not have been possible without the economical support of the Spanish ministry and the logistical support of the UPC's Image Processing group. This research was supported by the Spanish Ministry of Science and Innovation via a doctoral grant to the first author (FPU2014), and developed in the framework of projects TEC2013-43935-R and TEC2016-75976-R, financed by the Ministerio de Economía, Industria y Competitividad and the European Regional Development Fund (ERDF).

1

Introduction

COMPUTER vision and image processing techniques have traditionally been focused on single-view 2D images. There have been numerous developments that allow computers to have a better understanding of the surrounding world.

However, 2D images represent only a single projection of the underlying scene. The pinhole camera model can be used to describe the projection of the world into a 2D plane. This model, depicted in Figure 1.1, assumes no distortion effects from the camera lens.

In this model, the light rays from the scene are projected into a 2D lattice with a common centre point, named the camera centre. This transformation does not preserve sizes nor angles and does not allow differentiating between a small close object and a far large object.

This missing information of the scene, however, can be captured using 3D sensors. The first sensors were developed around the 1980s [6]. Different techniques were used to capture the distance information, such as Time of Flight (ToF), amplitude

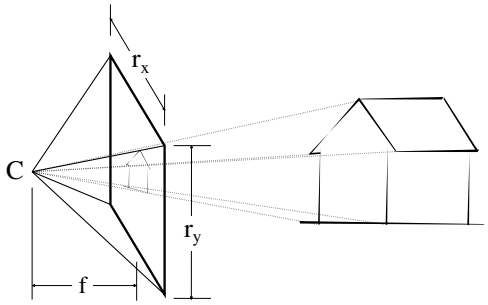


Figure 1.1: Representation of a pinhole camera. From each point in the scene a line is traced to the camera centre c and is represented into a plane with size (r_x, r_y) located into a distance f from the camera. All the dimensions are defined in pixels, since its absolute size has no relevance to the captured image.

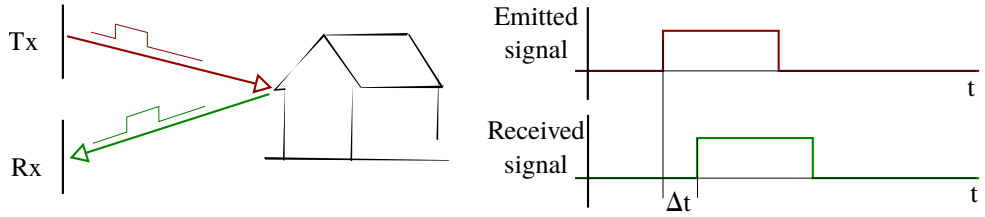


Figure 1.2: Diagram showing the operation of a Time of Flight sensor. A burst is transmitted and reflected into the measured surface. Measuring the time from the emission to the reception and knowing the wavelength speed, the distance to the surface can be known.

modulation, structured light, moire patterns or focusing techniques, among others.

One of the most relevant techniques on indoor captures were the Time of Flight (ToF) sensors, derived from radar sensors. These sensors use radar techniques by measuring the time taken by a signal to travel the distance between the camera and the surface, as shown in Figure 1.2. The values are represented in a 2D lattice with distance values for each pixel.

These first sensors had two significant drawbacks. The resolution and frame rate acquired using a ToF camera were significantly lower than the ones obtained from image cameras. Furthermore, their cost was expensive, with prices ranging from \$1.000 to \$100.000 [6]. These two factors hindered the spreading of these cameras in non-specialized settings.

The largest spreading of 3D cameras was achieved by the Kinect sensors [113] developed by Microsoft. These sensors provided an RGBD image combining both colour information –RGB– and distance of the objects to the camera –D.

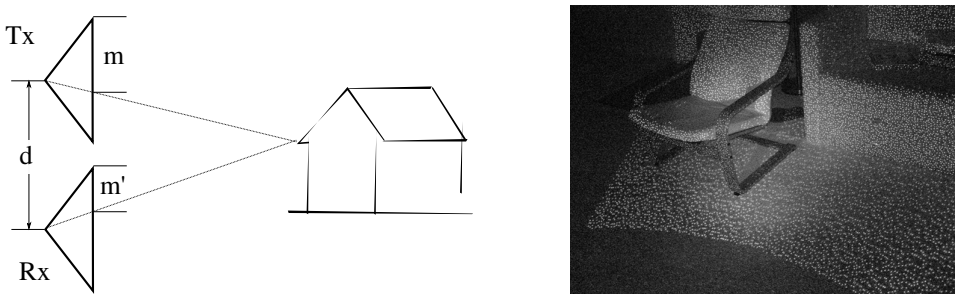


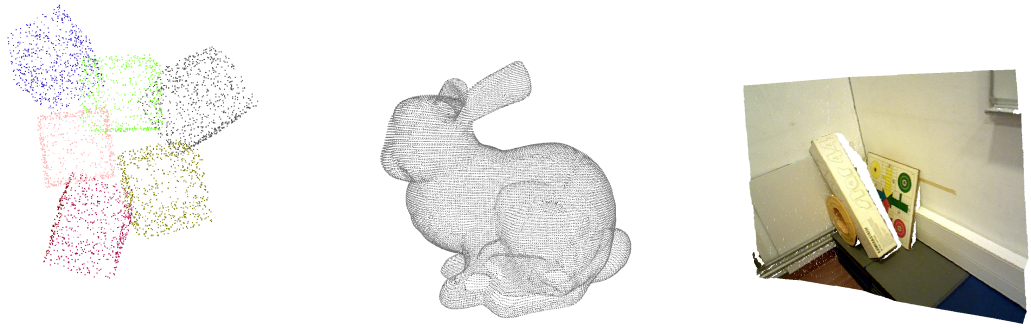
Figure 1.3: Diagram showing the capture of depth data from two cameras (left). The known distance d between cameras provides slightly different views form each camera. When a structured light pattern like the used by Kinect sensors (right) is projected into the scene each pixel of the projected pattern can be tracked into a pixel of the capturing camera. Right image extracted from bbzippo.wordpress.com

The technology used to capture the depth is based on the projection of an IR light pattern on the scene. The projected pattern is captured using an IR camera. By measuring the disparity between the projected pattern and the captured pattern, the distance to the sensor is obtained. Figure 1 shows the procedure applied.

The same housing contained an RGB camera, allowing to relate the obtained depth information to the colour information. Its low cost –compared to early ToF sensors– democratized its use in research environments [35]. Since then numerous sensors have been developed, such as Asus Xtion or Kinect One. These sensors capture RGBD images with varying degrees of image resolution, frame rate and depth resolution.

In addition to RGBD sensors, there are other methods to capture distance from the surrounding surface to the sensor. For example, Laser Imaging Detection and Ranging (LiDAR) sensors [83] use a laser to measure the distance between the sensor and the surface. In these sensors the depth is captured by moving a laser header, allowing a spherical scan of the scene. This scan can take different shapes: stripes, circles, zigzag, etc. Additionally, these measures can only be captured in the scene seen directly around the sensor.

Unlike RGBD sensors, the data captured using LiDAR sensors can not be represented into a single regular 2D image. There are different representations to visualize the data, like spherical images or line strides, but these representations do not preserve the captured strides nor scene geometry. However, there is a representa-



(a) Cubes **(b)** Bunny **(c)** Kinect

Figure 1.4: Several examples of different point clouds. (a) is a synthetic point cloud constructed by randomly sampling the surface of a 3D model, (b) is a point cloud built by combining several range images scanning the same 3D real world object, and (c) is a point cloud obtained using a Kinect sensor on an indoor scene.

tion that allows combining the data obtained from different depth sensors: the point cloud. In this format, the depth data is stored as a set of 3D euclidean coordinates (x, y, z) for each measured point. In addition to the location, information about the colour and the luminance can be stored in this same point.

There are two main point cloud types: unorganized point clouds and organized point clouds. Unorganized point clouds only contain a set of x, y, z coordinates of points where a surface has been detected. This point cloud can also contain additional readings for each point, such as luminance or colour of the point surface.

Organized point clouds contain all the data stored on an unorganized point represented into a 2D lattice, relating each point to their four closest neighbours in this lattice. This point cloud representation is equivalent to RGBD image representation. It cannot be obtained when the point cloud cannot be projected onto a single plane, such as combined point clouds from several captures.

This work is focused on the usage of unorganized point clouds since it is the most versatile form and can be obtained from the combination of several RGBD captures. Figure 1.4 shows several examples of unorganized point clouds obtained using different techniques.

The use of 3D data improves the performance of various computer vision tasks.

For example, objects that were too similar in appearance could be distinguished by using its size or its distance to the camera as a relevant factor. Better performance can be achieved by using the depth and size measurements in tasks such as image segmentation or object identification [73, 87].

Although 3D sensors are becoming increasingly widespread, there is still a high cost and quality gap between image and 3D sensors. There are several large scale image datasets with millions of images. As an example, the ImageNet dataset [19] contains more than 14 million annotated images. These datasets are used in many image recognition techniques, such as facial and handwriting recognition, object detection or aerial imaging [10, 63, 40].

In the 3D/depth domain, however, the available datasets are mainly focused on specific tasks such as aerial recognition, medical imaging or LiDAR navigation. One of the most generic datasets, the 3DMatch dataset [111], is comprised by a bundle of several datasets and contains only 47 sequences with 200k frames of unannotated captures.

Multi-domain registration can be used to improve the tasks mentioned above. The 2D images can be augmented with the depth information provided by the point cloud, and the 3D point cloud can be augmented with colour information provided by the images. Additionally, this registration can gather all multimodal information into a single domain and contain transference of interpreted scene information between domains.

A relationship between the different data sources must be established to merge multimodal data. One of the most common relationships is to find the location of the camera into the 3D scene. The camera location relates 3D points to the corresponding view in the 2D image or the pixel image to its corresponding depth. There are already several methods to combine data from different domains. However, most of these algorithms are constrained to a single sensor type or require multiple images to relate both data.

1.1 Problem Definition and Document Structure

The main goal of this thesis is to explore algorithms to get a multi-domain registration in an unconstrained case. The registration can be defined as the estimation of the extrinsic parameters of the image camera concerning the 3D point cloud. These extrinsic parameters are the rotation and translation of the regarding the point cloud origin coordinates.

As illustrated in Equation (1.1), the camera coordinates of a 3D point are described by the intrinsic camera matrix and the extrinsic camera matrix. The intrinsic camera matrix contains information relative to the camera geometry, its focal point and its projection centre. The extrinsic camera matrix contains information relative to the camera rotation and translation on the scene.

$$\underbrace{\begin{bmatrix} m \\ n \\ 1 \end{bmatrix}}_{\text{Camera coordinates}} = \underbrace{\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Intrinsic camera matrix}} \cdot \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}}_{\text{Extrinsic camera matrix}} \cdot \underbrace{\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}}_{\text{3D world coordinates}} \tag{1.1}$$

Therefore, the main goal of this thesis is to develop a method to compute this extrinsic camera matrix, also called the transformation matrix. The extrinsic matrix should be computed using only a single image and a single unorganized 3D point cloud with no prior information about the initial location, no additional images of the same scene or no known patterns located on the scene.

In the next chapter, the current developments of this problem in several fields are described. Development of 3D-2D registration using different data types -medical images, LiDAR, RGBD data- is explored, alongside with single-domain 3D-3D and 2D-2D registration. As it will be seen on the state of the art strategies, a relevant part of the registration process is the detection, extraction and processing of relevant features on the matching data. Therefore, the current state of the art methods to compute features on 2D and 3D captures are also be surveyed.

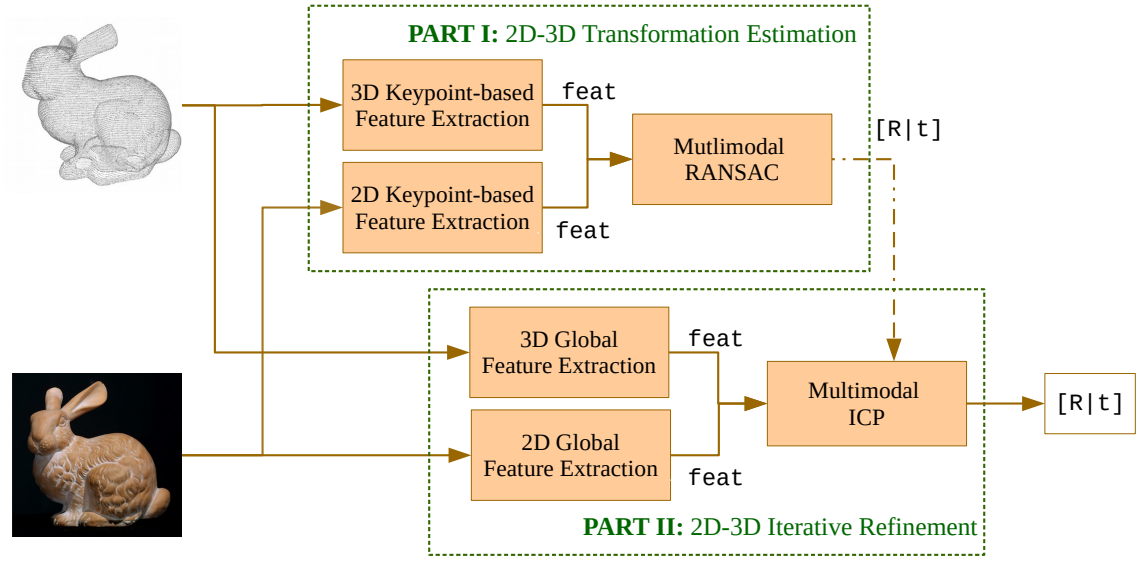


Figure 1.5: Global pipeline of the algorithm proposed in this thesis.

The proposed method is presented in two parts. The first part of this thesis focuses on the acquisition of an initial alignment between the image and the point cloud. The second part presents techniques to refine the transformation obtained in the first part. Figure 1.5 shows the overall pipeline followed in this thesis.

The method to compute the initial alignment uses deep learning feature matching to estimate the transformation between captures. Feature matching techniques have been typically used in both 3D-3D and 2D-2D registration methods. In this method, deep learning techniques are used to compute a multimodal and multi-domain descriptor to match heterogeneous data directly.

A method to perform a global registration by iteratively aligning geometrical features is presented in the refinement step. This method adapts existing 3D-3D methods to a multimodal and multi-domain setting. This method is only suitable to refine initial registrations since the initial displacement on the camera location is relevant to the algorithm performance.

A comparison between the two techniques and their symbiosis to perform an overall registration is evaluated. Individually, each method is evaluated in the quality of the detected features and the performance of the registration problem. Additionally, a brief overview of the complexity and computation cost of both algorithms is per-

formed. Their strengths and shortcomings, depending on application requirements, are also evaluated.

On the final part of this thesis, a summary of the contributions achieved in this thesis is stated. Additionally, a brief overlook to the possible improvements and open research lines is explored.

2

Background and Approach

The ability to have a common reference frame for multimodal captures is researched in many fields. Some examples of these fields are medical imaging, airborne terrain mapping, LiDAR navigation and urban scene reconstruction.

The work presented in this thesis will focus on the most generic 2D-3D registration scenario: the direct registration between a single unorganized point cloud and a single 2D image. However, due to the scarcity of methods for this generic scenario this state of the art will cover the registration of multimodal data using 2D images and several types of 3D data, focusing on three relevant fields: medical imaging, LiDAR data and RGBD images.

A classification of the methods used in the different fields is stated using several benchmarks as a baseline. The feasibility of using the various strategies to perform a 2D-3D registration is explored, and their strengths and shortcomings are evaluated.

The methods presented have a strong bias to work in a single dimension, either being 2D or 3D. This bias makes it interesting to also explore the current state of

the art on single-modal 2D and 3D registration. The summary presented in the last part of this state of the art is focused on the data used in this thesis: 2D images and unorganized 3D point clouds. An overview of the most commonly used strategies to align either two 2D images or two 3D point clouds will be stated.

2.1 Multimodal Registration

Computer vision and image processing techniques have been successfully used in several medical fields to aid in medical diagnostics and treatments. The imaging technologies that are used in the medical field have a significant variance: different resolution, wavelength, dimensionality, etc.

In many cases, there is a need to be able to relate data from different domains obtained using various techniques. There is a specific scenario when this registration is essential: in computer-aided interventions, high-detail 3D data is acquired before the intervention. During the intervention, only low-detail 2D data is available. The ability to relate the pre-operative acquisition onto the low-resolution data obtained during intervention helps physicians in performing the operation [54].

Two of the most common sources of 3D imaging, Magnetic Ressnance Imaging (MRI) and Computed Tomography (CT), obtain neither a 2D image nor a 3D point cloud. The data collected is formed by a 3D lattice of 'voxels': regularly sampled 3D pixels. This 3D voxel lattice can be seen as several 2D images stacked into a third dimension forming a 3D dense volume.

The acquisition technology, however, differs between methods: MRI uses magnetic fields to generate the data, while Computed Tomography (CT) scans are captured using X-ray measurements.

Another field with relevant developments in the 2D-3D registration field is the airborne optical imagery. In this use case the goal is to register Laser Imaging Detection and Ranging (LiDAR) data to RGB images [57]. LiDAR is a surveying technology that measures distance by illuminating a target with a laser light beam. The re-

flection from the different objects allows the measurement of the distance and the reflectivity of the object.

These sensors can either be mounted on a static pole or an aircraft. In a static pole, a mirror system and rotors move the beam to perform the pan/tilt movement, having a spherical scan centred on the scanner position. When mounted on an aircraft, the system scans the terrain tracking both the sensor movement and the aircraft movement resulting in planar scans. Infrared images are also commonly captured alongside depth images.

LiDAR sensors are also used in autonomous vehicles to gather depth information around the vehicle. Since the captured resolution is usually significantly lower than the RGB images, the captured scans are either smoothed out images or non-dense depth maps. This novel field uses the new deep learning techniques to perform the registration [26, 81].

Aside from medical imaging and LiDAR images, multimodal registration is also used on data augmentation and scene acquisition. One example is the fusion of multimodal data on urban scenes. When capturing an urban scene, a series of 2D images and 3D range scans are acquired. Registration algorithms are used to map all the captures into a common coordinate system.

By fusing data from several sources, photorealistic captures of the scene can be obtained [91, 50]. These technologies can also be applied to enhance the quality of 3D models by using 2D images of the same object [109].

In general, range image captures are obtained by discrete sampling the object surfaces from a specific location. These sensors are used for a wide variety of tasks, both in indoor and outdoor settings. Unlike the captures obtained using medical sensors, this data only represents the surfaces of the scanned object.

If the capture is performed using only a range sensor, the acquired data can be represented into a 2D depth image. However, when multiple scans are combined, the data can only be represented into a single domain by using unorganized point

clouds.

There is a wide variety of applications and methods for 2D-3D registration. There is extensive literature regarding the methods available in several fields.

In the medical imaging domain Markelj et al. [54] reviewed several techniques used to register MRI scans, CT scans and 3D models to X-ray images. The methods presented are classified based on the dimensional correspondence strategy and the 2D-3D registration features.

Mishra and Zhang [57] reviewed several methods to register airborne LiDAR data to RGB images. The methods presented provide registration between infrared intensity images, depth maps from the LiDAR scanner and RGB images from the colour cameras. Since all captures are represented as planar images, the transformation to align them is a rotation, translation and scale on this plane. The methods are classified based on the nature of the detected features.

Stamos [90] reviewed the methods used in the registration of point clouds and RGBD data to 2D colour images for urban scenes. The techniques presented are classified based on the dimension the registration is performed and the features used to perform the registration.

In the next sections, the different methods presented in the various reviews are classified on two main characteristics: the dimension in which the registration is performed and the features used in the registration.

2.1.1 Dimension-based Categorization

Multimodal registration embraces a wide range of problems and strategies used to solve them. In many cases, multimodal registration does not entail multi-domain registration. In these cases, the registration is performed in the domain in which the data was captured. This is the case of many captures performed using LiDAR sensors, which can be represented as 2D depth maps.

In the cases where the registration involves multi-dimensional captures between 2D and 3D dimensions, three main strategies can be used. Two main strategies are to either bring the 3D data into the 2D dimension or the 2D data into the 3D dimension. In the first case, the registration is carried out in the 2D domain, while in the second case, the registration is performed using single-domain techniques on the 3D domain.

However, bringing the data into another domain entails either losing information – in 3D to 2D domain adaptation – or interpolating information from other sources – in 2D to 3D domain adaptation. The last strategy used entails performing the registration between domains, without directly translating the data between them.

2D-2D Registration

The first group of methods form multi-dimensional 2D-3D registration that will be explored involve bringing the 3D data into the 2D dimension. With all captures in the 2D dimension, 2D-2D registration methods can be applied. These methods take advantage of existing 2D techniques to perform the registration. Several strategies are used to bring the 3D data into the 2D domain. These techniques involve either performing several 2D projections of the 3D data or having additional 2D information about the 3D model.

In many of the techniques used in medical imaging, the 3D data is projected into several 2D planes [27, 9, 51]. The different projections are compared in the 2D domain, and the most suitable transformation is selected. This procedure can be applied when a rough initial location is known, thus minimizing the number of possible 2D projections.

Another case where 2D-2D registration can be used is when additional 2D data is available. [109, 93, 79] use 2D-2D matching techniques to align a new 2D image into pre-existing 2D images. The pre-existing 2D images are already referenced in the 3D point cloud. Thus, the registration between 2D images indirectly becomes a registration between a 2D image and a 3D point cloud.

3D-3D Registration

Likewise, registration can be performed into the 3D dimension if the 2D data can be brought into the 3D domain. Two main strategies can be used to bring the 2D data into the 3D domain: back-projection and reconstruction.

Back-projection techniques are used in medical imaging [46, 2, 96] Virtual rays are formed by connecting 2D points, representing an object's silhouette, a curve or edges, with the X-ray source. Registration is performed in 3D by minimizing the distance between occluding contour and visual rays.

Reconstruction techniques involve having several 2D captures. If these captures are calibrated and overlapped, an approximate 3D reconstruction can be obtained. This technique is used both in medical imaging [115, 70, 55, 1] and in urban scenes [50, 92, 68, 114].

Direct 2D-3D Registration

There are few methods perform that perform a multimodal registration without performing a domain translation. The methods in this category use specific features that can be related between domains.

One domain that this procedure is used is on the registration of urban scenes. Urban scenes typically have many straight lines. The vanishing points can be estimated by computing straight lines in the 2D image,

The image and the point cloud can be related in terms of translation and rotation using the vanishing point as a reference point [91, 48, 49].

2.1.2 Feature-based Categorization

A distinctive aspect of the different multimodal registration methods is the features that are used to perform the registration. Some anchor points, common points or

distinctive characteristics are used to relate different captures.

There are two basic feature types: extrinsic features and intrinsic features. Extrinsic features are features that are manually placed on the scene.

In medical imaging, some methods use artificial markers introduced in the patients [38, 59]. Objects like stainless steel beads, hollow plastic balls filled with an aqueous solution or gold seeds are introduced and traced.

In this section, the focus will be on the intrinsic features, those naturally present in the scene. From these intrinsic features, the main focus will be on those methods that automatically detect the features without the need for human intervention.

The intrinsic features can be further classified in local or global features. Local features are those that detect salient characteristics in both data [118]. In airborne LiDAR registration these features can be corners [106], straight lines [30], roofs [43], surface patches [78], etc. In medical imaging anatomical structures present on both data are also detected [23].

Global features are those that describe globally all the capture. Some examples are mutual information methods [56, 102], frequency-based methods [106] or gradients [104].

Recently deep learning techniques have also been used to perform multimodal registration. In [26] a deep learning technique was developed to learn displacements between a LiDAR capture projected on a 2D plane and an RGB image. This method only obtains the displacement up to a small set of possible displacements. In [81] an end to end neural network that directly estimates the six degrees of freedom of the transformation between a depth image and an RGB image was developed. These methods use Convolutional Neural Networks (CNNs) to learn features from the RGB and the depth images. In the first method, the optical flow is also used to compute the features.

2.2 Same-domain Registration Methods

As it has been seen in the previous section, there are only a few methods that do not reduce the multimodal registration problem into a single-domain registration. These methods leverage the specific characteristics of their respective registration problems to obtain the transformation. The most relevant ones use vanishing points present on the structure or the ability to project the 3D data into a limited set of 2D planes.

Most methods, however, rely on translating the data into a single-domain framework. Therefore, in this section of the state of the art, single-domain registration methods are explored.

Among the different methods available in single-domain registration, two methods stand out. The first one is used in both 2D and 3D domains to compute the transformation between two images or point clouds. The second one is used in the 3D domain to refine a rough initial transformation estimation.

2.2.1 Single Domain Transformation Estimation

In the 2D domain, two images can be related by using epipolar geometry. The epipolar geometry defines the relationship between two camera views independently of the scene structure. This relationship is encapsulated into the fundamental matrix F. For any pair of matching points p in one image and q in the other image, represented in homogeneous coordinates, the equation $q^T F p = 0$ is satisfied. At least 7 points need to be matched between images to obtain the fundamental matrix.

In the 3D domain, the transformation $T = [R|t]$ defines the rotation and translation between two point clouds. For any pair of points matching points in euclidean coordinates p, q representing the same 3D location, the equation $Rq + t = p$ is satisfied.

The following procedure allows the estimation of either the fundamental matrix

F or the transformation matrix T by obtaining a set of corresponding points and computing the corresponding matrix.

Fist of all, several points –called keypoints– are selected on both data. The most relevant characteristics in a keypoint detector are the specificity, defined as the capability to differentiate the point from other keypoints on different sources, and the repeatability, defined as the ability to detect the keypoint on all the sources accurately [97]. The specificity depends on which features are observed on the correspondence matching steps, but general characteristics requirements should be fulfilled in most cases. For instance, areas with high textures or sharp contours are favoured over flat surfaces, where the location of keypoints cannot be established accurately.

In 3D point clouds, some methods use randomly selected points instead of computed keypoints [111]. The review in [97] shows that Intrinsic Shape Signatures (ISS) [116] provides the highest repeatability on most datasets. Besides, the detected points have good performance when matched using state-of-the-art 3D descriptors, as shown in [28].

Several algorithms can be used to perform the match between the selected keypoints. A descriptor is extracted from each keypoint, and then the descriptors are compared. Historically, the procedure done is to manually define a set of equations to obtain a fixed-length vector from an image patch centred on the query keypoint.

In the 2D domain, the most commonly used hand-crafted descriptor algorithm is Scale Invariant Feature Transform (SIFT) [52]. Although further variants have been developed, it is still one of the best performing descriptors [41].

In the 3D domain, one of the best performing algorithms is Rotational Projection Statistics (RoPS) [29]. RoPS projects the point neighbourhood into various rotated patches in several planes and extracts statistics about the projection. Another algorithm with good performance is Point Feature Histogram (PFH) [76]. PFH computes a histogram describing the local geometry around a query point.

Recent developments exploit Convolutional Neural Networks (CNN's) to perform matching between two 2D keypoints. Zagoruyko [110] presents a method to establish correspondences by querying if two keypoints match based on their surrounding pixels on the image. For each query keypoint, a 64x64 image patch is extracted. Then, these patches are classified using a CNN that provides a score indicating how similar the patches are.

In the 3D domain, [111] developed a technique that uses 3D CNN's to perform a match between two point clouds. This technique represents the surroundings of a 3D keypoint as a $30 \times 30 \times 30$ binary occupancy voxelization. This voxelization is used to train a 3D dense CNN that outputs a feature vector for each patch. This feature vector is compared using euclidean distance, and a RANSAC algorithm is used to find the transformation.

RAndom SAmple Consensus (RANSAC) is an iterative algorithm to find the parameters of a mathematical model given a set of measures [21]. This algorithm is robust when there are low-noise correct measures –inliers– mixed with grossly incorrect measures –outliers.

The gist of this procedure is applied in this thesis to compute the 2D-3D registration between a single image and a single point cloud. Using Convolutional Neural Networks to compute descriptors on both 2D and 3D data, initial registration is estimated by computing the transformation between matched keypoints through a RANSAC estimation.

2.2.2 3D-3D Iterative Refinement

In many scenarios, a rough initialization in the data to be aligned can be obtained. This is the case, for example, when a sequence is being aligned. Once one frame is aligned, the remaining frames can be assumed to be on the same approximate reference.

In cases where a rough initial alignment is known, a local optimization method can

be used. The most popular method in this category in the 3D domain is the Iterative Closest Point (ICP) [112, 7]. This method computes the parameters of the output transformation by iteratively computing the transformation that best aligns closest points.

This algorithm has been widely used, and his performance improved in several areas, such as finding good initial guesses [25, 53] and estimating better point features [39, 84], among others. In [69] several ICP methods are evaluated using a common metric and under several datasets. In particular, ICP variants using point-to-line and point-to-plane distances are compared and evaluated.

In this thesis, a variant of the ICP algorithm that can work with point clouds and images is developed. This algorithm is used as a refinement step to improve the results obtained using the initial RANSAC alignment.

Part I

Methodology

3

2D-3D Transformation Estimation

The first part of this thesis focuses on the estimation of the 2D-3D transformation using correspondences. Fig. 3.1 shows the pipeline to align a single image and a single point cloud using correspondence estimation. This chapter explores the different parts of this system.

First of all, in Section 3.1, a brief overview of the possible keypoint selectors for both 2D and 3D domains is stated, evaluating their suitability to be used in the proposed pipeline. The next block is the 2D/3D patch selection. In this block, explained in Section 3.2.1 the techniques used to represent both 2D and 3D neighbourhood into a dense patch are explained. Finally, the patches obtained are used to extract a modal-agnostic descriptor using deep learning techniques, described in Section 3.3.

Once the descriptors are obtained, the next step is to estimate the transformation. In the final part of this chapter, described in Section 3.4, the different considerations on the distance metric and fitness metric are stated.

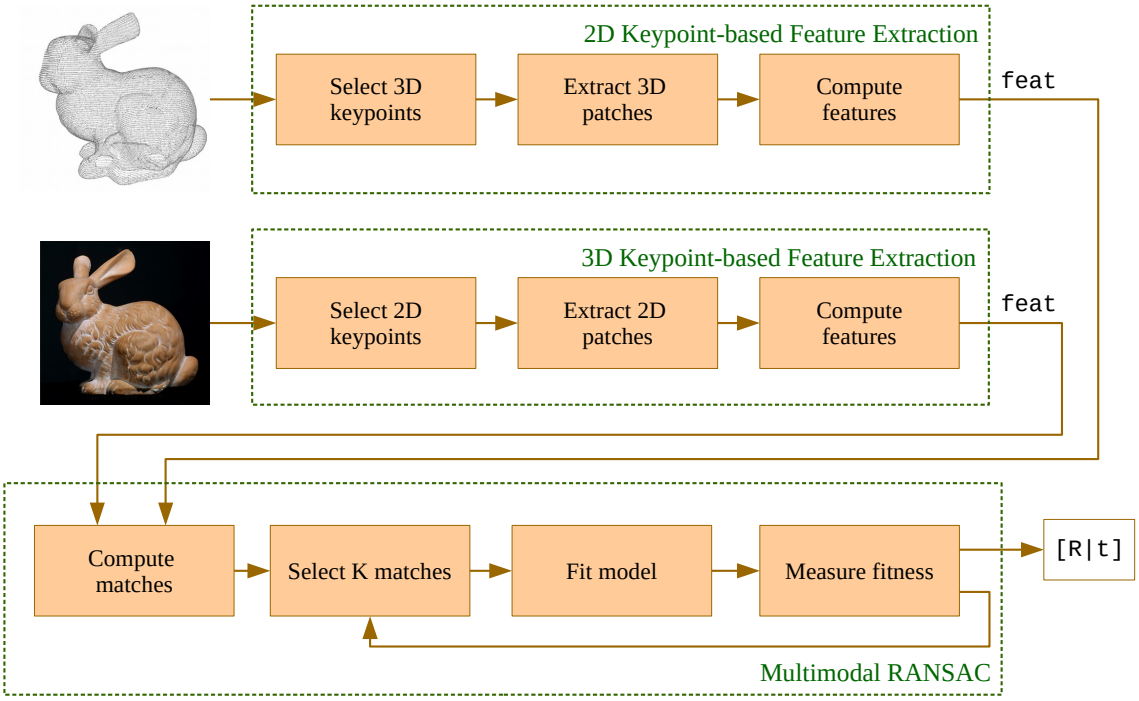


Figure 3.1: Parts of the proposed 2D-3D transformation estimation algorithm.

3.1 Keypoint Selection

There are several algorithms used to extract relevant points on 2D and 3D captures. Both in the 2D and 3D domain there are well-performing keypoints, such as Harris [33] –2D– and Intrinsic Shape Signatures (ISS) [116] –3D. However, it needs to be taken into account that most state of the art keypoint detectors are developed to be used into a single-domain scenario. Therefore, all benchmarks performed on this regard only state the performance of the keypoints on a single domain scenario.

Furthermore, the specificity of a keypoint needs to be evaluated in conjunction with the feature descriptor applied. The specificity is especially relevant in the proposed architecture since the specific characteristics of the data fed to the network will influence their performance.

Given that, there are two main alternatives to use as a keypoint detector. The first one is to use well-performing keypoint detectors in single-domain architectures and match them accurately to obtain the best performance possible.

The second one is not to use any keypoint detector and randomly select points on

both data. Having no keypoint has several advantages. First of all, by not selecting any specific feature on the different data sets, the network is fed unbiased data on the training stage. Feeding the network unbiased data allows the system to develop techniques to detect features that may not be selected using a specific keypoint detector.

Furthermore, not having a keypoint selection has a practical advantage. The fast computation time of random keypoint selection allows having faster training and testing times. This speed has direct effects on the algorithm performance by allowing to test more configurations in the same time frame.

All these factors are evaluated for the different keypoints. In Chapter 6 a comparison between state of the art keypoints –Harris (2D) and ISS (3D)– and random keypoint detectors is performed. As will be seen, the results favour the usage of random keypoint detections.

3.2 Patch Representation

Once the query points are obtained, the next step in the procedure is to translate the keypoint neighbourhood into a data type that can be fed to the network. The data acquired from 3D point clouds and the data obtained from 2D images are vastly different. Therefore, two different strategies are proposed to extract data from the 3D and 2D domains.

3.2.1 3D Patch Representation

In the 3D domain, the proposed strategy consists in obtaining a local lattice structure around the query keypoints, analogously to the image patches. The local lattice structure allows obtaining an RGBD representation of the surrounding patches that can be classified using existing 2D CNNs.

A 3D point cloud of a scene can be seen as a sampling of the 3D surfaces that are

contained in this scene. When the sampling is done from a single viewpoint –for example, a single static Kinect or LiDAR sensor–, the scene can be projected using an orthogonal projection in a 2D plane. This orthogonal projection will not have occlusions between points and will preserve all the information from the captured points. However, if the available point cloud is a combination of captures from several viewpoints, the existence of a 2D representation where the points from different viewpoints are not overlapped is not guaranteed.

When matching keypoints, however, the interest area is reduced to a small neighbourhood of the query point. In this case, only the points –and thus, the surfaces– close to the query point are relevant. In this close neighbourhood, the likelihood of not finding a projection plane which does not have overlapped points from several surfaces is minimized.

Additionally, another factor has to be taken into account. In a point cloud, the density of the points may change on the scene. For example, on a Kinect camera, the point density decreases with the distance of the sensor to the camera. When several captures are merged, there can be a disparity between the point density of different captures in the same neighbourhood. The proposed projection technique should obtain a similar result independently of the point density on the query point neighbourhood.

Taking into account these two factors the following projection strategy is presented. A small neighbourhood $\mathcal{X}_{p_j}$ is selected por each keypoint p_j in the point cloud P using a KD-tree with a fixed radius r, as shown in Eq. (3.1). Each 3D point in the neighbourhood is represented using its 3D coordinates p_m and its intensity value y_m. Analogously, the set X_{p_j} defines the set with only the 3D coordinates p_m and the set set Y_{p_j} defines the set with only the intensity values i_m

$$\begin{aligned} \mathcal{X}_{p_j} &= \{(p_m, y_m) \,|\, \|p_m - p_j\| \leq r\} \\ X_{p_j} &= \{p_m \,|\, \|p_m - p_j\| \leq r\} \\ Y_{p_j} &= \{y_m \,|\, \|p_m - p_j\| \leq r\} \end{aligned} \tag{3.1}$$

The obtained neighbourhood is centred by computing the centroid $\overline{p_j}$ using the 3D coordinates p_m and subtracting it from the query point set X_{p_j}, obtaining X'_{p_j} as shown in Eq. (3.2).

$$\overline{p_j} = \frac{1}{\#X_{p_j}} \sum_{p_m \in X_{p_j}} p_m$$
$$X'_{p_j} = X_{p_j} - \overline{p_j} = \{p'_m\} \tag{3.2}$$

Using PCA, the principal eigenvectors of this neighborhood X'_{p_j} are obtained. This eigenvectors, defined as $\nu_{p_j} = \{\nu^0_{p_j}, \nu^1_{p_j}, \nu^2_{p_j}\}$, are ordered by decreasing eigenvalue. These eigenvectors define the optimal parallel projection plane to minimize the overlapping between points. This plane is defined using its normal vector $\nu^2_{p_j}$.

This parallel projection provides a transformation that is invariant to the camera viewpoint and the point density of the area. In most cases, the PCA projection gives a strong plane vector, but the rotation within the plane is not well-defined. The orientation of the patches is normalized to a common direction to obtain rotation invariant patches.

The intensity values of the points contained in the neighbourhood are used to define a common orientation between all the patches. The direction of maximum intensity η_{p_j} is computed using the centered coordinates c and their corresponding intensity values Y_{p_j}, as shown in Eq. (3.3).

$$\eta_{p_j} = \sum_{(p'_m, y_m) \in X'_m, Y_m} p'_m \cdot y_m \tag{3.3}$$

This information allows the definition of a new base $\gamma_{p_j} = \{\gamma^0_{p_j}, \gamma^1_{p_j}, \gamma^2_{p_j}\}$. This base has the same third vector as the original base, giving $\gamma^2_{p_j} = \nu^2_{p_j}$, but the vectors in the plane are defined based on the direction of maximum intensity η_{p_j}, as shown in Eq. (3.4). The vector $\gamma^0_{p_j}$ is computed as the parallel projection of η_{p_j} in the plane defined by the normal vector $\nu^2_{p_j}$. Finally, $\gamma^1_{p_j}$ is defined as an orthogonal vector to

$\gamma^0_{p_j}$ and $\gamma^2_{p_j}$ to build the base.

$$\begin{aligned} \gamma^0_{p_j} &= \eta_{p_j} - \langle \eta_{p_j}, \nu^2_{p_j} \rangle \cdot \nu^2_{p_j} \\ \gamma^1_{p_j} &= \gamma^2_{p_j} \times \gamma^0_{p_j} \\ \gamma^2_{p_j} &= \nu^2_{p_j} \end{aligned} \tag{3.4}$$

The points X'_{p_j} are transformed using the base γ_{p_j}, obtaining $X''_{p_j} = \{p''_m\}$. This projection provides a representation invariant to the camera location and with a common orientation on all the query neighbourhoods.

The next step is to obtain an RGBD image from these oriented points. This image is obtained by performing an orthogonal projection on the transformed point set into a regular $N \times N$ lattice. This lattice is defined within the range of the neighbourhood radius r, from $-r$ to r on the normalized x and y axis.

Each 3D point is projected into a cell, assigning their RGB value to the cell, to obtain the RGB information. If more than one point is assigned to the same cell, the values of the points assigned to the same cell are averaged.

For the projection of the depth channel, we follow a similar procedure. The depth value d_m is assigned to the depth channel projection for each point in X''_{p_j} . This value is computed as the distance between the point and the projection plane (defined by the principal vector $\gamma^2_{p_j}$): $d_m = \langle p''_m, \gamma^2_{p_j} \rangle$. This distance is also normalized using the neighbourhood radius r. If more than one point is assigned to the same cluster on the lattice, the final value is the average of the distance of the assigned points.

To summarize, the acquisition of an RGBD patch for each query point of an unorganized 3-D point cloud can be made following these steps, shown in Fig. 3.2.

1. For each given query point, obtain a subset of the point cloud given by the points which are within distance r of the query point (Fig. 3.2.a)
2. Compute the PCA decomposition ν_{p_j} of the point subset (Fig. 3.2.b).
3. Compute the direction of maximum intensity η_{p_j} to make the transformation invariant to rotation (Fig. 3.2.c).

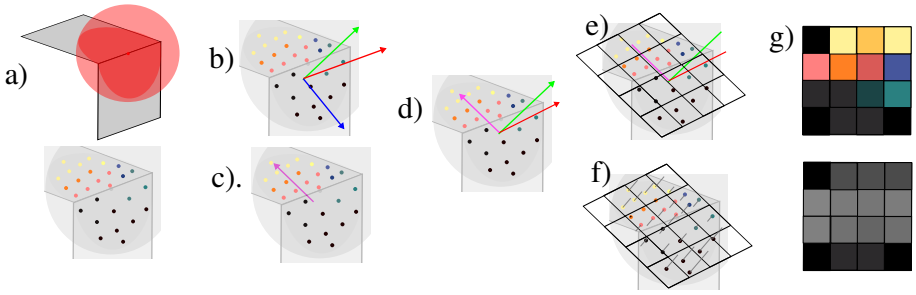


Figure 3.2: Schematic of the computation of an RGBD image from a 3D point cloud keypoint: a) Underlying structure and sampled points, b) PCA decomposition, c) Direction of maximum intensity, d) Rotation of the PCA vectors to match the direction of maximum intensity, e) Projection lattice (4x4), f) Projection of the points to the lattice, g) Obtained RGB (top) and depth (bottom) images

4. Rotate the base ν_{p_j} to have the projection of $\nu^2_{p_j}$ as the main plane vector (Fig. 3.2.d), defining a new base γ_{p_j}.

5. Define a lattice in the projection plane of size $N \times N$. This lattice should be scaled to the range of the projected points (Fig. 3.2.e).

6. Project the points into the lattice. For each cluster, if more than one point is assigned, their RGB values are averaged. The depth value is defined as the distance of each point to the projection plane. In the same manner, if more than one point is assigned, their depth values are averaged (Fig. 3.2.f).

7. The final result is an RGBD image which is independent to the point density of the underlying surface and has a defined orientation (Fig. 3.2.g).

This procedure has two basic metaparameters: the neighbourhood radius on the 3D point cloud, named r, and the size of the projection lattice, referenced as $N \times N$. Both metaparameters need to be tuned to obtain a discriminable representation of the point neighbourhood.

The neighbourhood radius affects the amount of information about the neighbourhood that will be gathered. A small radius allows for a faster computation by collecting fewer points but loses information about the surroundings. However, a large enough radius will contain information about the surroundings that is no longer representative of the point neighbourhood.

The size of the projection is directly correlated with the point density in the neighbourhood radius. If a small projection size is used the resulting RGBD image loses detail present on the point cloud. However, if the projection size is too large, the

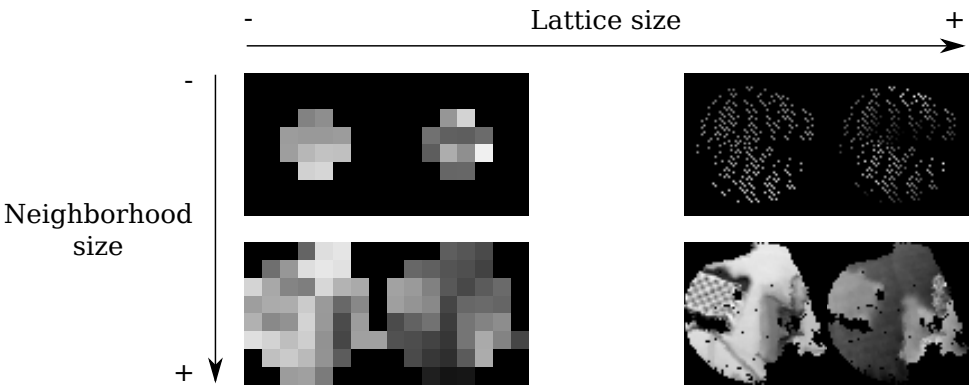


Figure 3.3: Illustrative representation of the effects of patch size and neighborhood query radius. In each case, the left image corresponds to the intensity information, whereas the right image corresponds to the depth information.

neighbourhood points are not able to fill all the lattice. When the lattice is sparsely filled, stripes and artefacts not representative of the surroundings are left on the resulting image.

To better illustrate the effects explained above, Fig. 3.3 provides four illustrative examples of the obtained patch for various neighbourhood and lattice sizes. As it can be seen, the three differences mentioned above are present. First of all, each patch has associated depth information (right) for each RGB information (left). As it can be seen, increasing the lattice size without gathering more points (top right) produces undesirable artefacts on the image, produced by blank clusters inside the main area. It can also be observed that lowering the patch size produces a blurred image due to several points being averaged in the same cluster (bottom left). Finally, it can also be observed that the patch has a roundish shape, produced by the query of the points using a ball distance.

3.2.2 2D Patch Representation

In the 2D domain, the image patch that represents the neighbourhood of a keypoint q_j is straightforwardly acquired by cropping the image in a $N' \times N'$ patch centred in the keypoint.

Unlike in the 3D scenario, the cropped patch size does not define a world size in meters but instead defines a projection distance in pixels. Therefore, two cropped

patches with the same N' can represent vastly different sizes in the real world.

Analogously to the 3D patch computation, this $N' \times N'$ patch is then scaled to an output size $N \times N$ to be fed on the network. This scaling has two main effects depending on if it is a downscale ($N' > N$) or an upscale ($N' < N$).

On the one hand, the downscale ($N' > N$) searches to mimic the blurring effect produced on the 3D patch when a large number of 3D points are projected in a small 2D patch. On the other hand, the upscale ($N' < N$) allows to feed a small patch into a large network.

This representation of a 2D neighbourhood presents another difference concerning the representation of the 3D neighbourhood: the lack of rotation normalization. In the 3D domain, this patch normalization is obtained by computing the direction of maximum intensity.

A similar approach is tried on the 2D patch, but the tests performed in this regard show that this method does not provide any significant improvement.

3.3 Network Architectures

By using the compressed RGBD representation of the query neighbourhood, existing Convolutional Neural Networks (CNN's) architectures in the correspondence matching field can be directly applied to 3D point clouds. Classical architectures use the network to learn if two images are or not correspondences. The network is fed the two images to learn this correspondence, outputting a likelihood score. The typical goal is to have high scores for matching pairs and low scores to non-matching pairs.

There are several network structures used in 2D to solve this problem. One of the most successful approaches concatenates the 2D patches in a 3-dimensional tensor. A sequential CNN is trained to provide a similarity score between patch pairs [110].

This structure is successfully tested with the proposed patches architecture in a single-modal 3D scenario. Three different structures –using only intensity, only

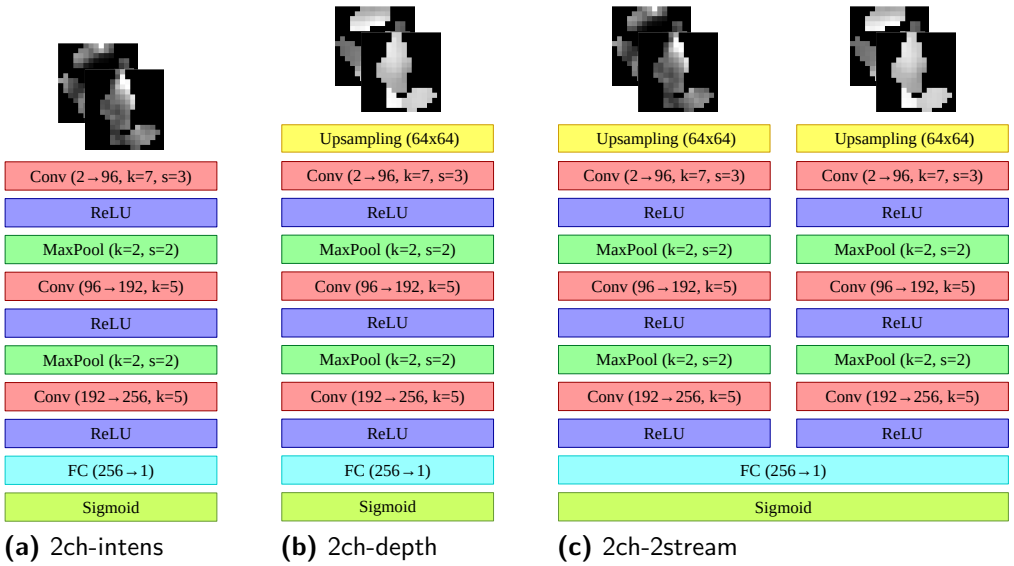


Figure 3.4: End to end registration networks. a) U(64) - C(2->96,k=7,s=3) - ReLU - P(k=2,s=2) - C(96->192, k=5, s=1) - ReLU - P(k=2,s=2) - C(192->256, k=3, s=1) - ReLU - F(256->1) - Sigmoid, b) same structure as a) but input with depth images, c) Two branches U(64) - C(2->96,k=7,s=3) - ReLU - P(k=2,s=2) - C(96->192, k=5, s=1) - ReLU - P(k=2,s=2) - C(192->256, k=3, s=1) - ReLU, joined with F(512->1) - Sigmoid. Notation : U(N) is the initial up-sampling to size $N \times N$ C($i- > n$, k=k, s=s) is a convolutional layer with i input channels, n filters of spatial size $k \times k$ applied with stride s, P(k=k, s=s) is a max-pooling layer of size $k \times k$ applied with stride s, and F($i- > n$) denotes a fully connected linear layer with i input units and n output units.

depth and combining intensity and depth values– have been tested in the work presented in [71]. In Fig. 3.4 the tested network structures are shown.

Despite their good performance, these structures, also called end-to-end structures, have a major drawback. M points are selected on each capture to perform the registration between two captures. Each point in P needs to be compared against each point in Q to match the two set of points P and Q from each capture. M^2 passes have to be performed in the network, one for each pair, to compare them using the network structure presented above, also called end-to-end structure.

On traditional hand-crafted descriptor techniques, the approach used is to obtain a fixed-length vector –descriptor– for each pair element. The descriptors for each element CNNs can also be used to obtain a descriptor for element, limiting the passes on the network to one pass for each patch, or M passes per capture. The obtained descriptors can be efficiently compared using sorting algorithms such as KD-tree that perform a fast neighbour search using euclidean distances.

Some registration set-ups also benefit from having the ability to pre-compute all the descriptors for one capture before having the capture to be matched against. This feature is especially useful when one capture is matched against several captures, avoiding the need to re-compute all the possible matches for each capture.

Feature-based structures have one main shortcoming: they cannot combine pair data on early stages of the network. Feature-based structures are constrained to use only data from one source to compute the descriptor. Therefore, these structures typically have lower performances than end-to-end structures. However, this disadvantage, also present on hand-crafted descriptors, in most cases, does not outweigh the computational benefits of feature-based structures. In this thesis, feature-based structures will be used to extract features from the patches extracted either from 2D or 3D keypoints.

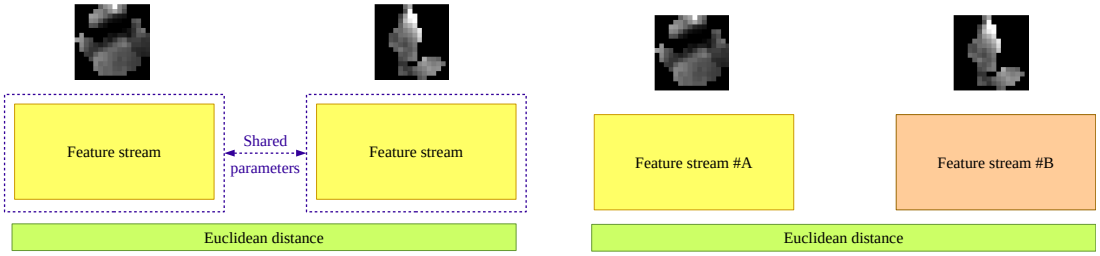
3.3.1 Training Architectures

Feature-based structures developed output a fixed-length feature vector for each given input. The vector obtained from two patches should be similar –in terms of Euclidean distance– if the two patches are a correspondence. If the patches are not representing the same scene location, the obtained vectors should be further apart.

Unlike end-to-end structures, feature-based networks do not directly output a confidence value between patches. The loss function minimized in the training process on the end-to-end structure presented is a SoftMargin loss. This loss S_L, seen in Eq. (3.5), for matching pairs ($y = 1$) penalises low scores, whilst for non-matching pairs ($y = -1$) penalises high scores. In this equation $f(x_p, x_q)$ represents the output of the network f for a given patch pair x_p, x_q.

$$S_L = max(0, 1 - y \cdot f(x_p, x_q)) \tag{3.5}$$

In a feature-based structure, the network does not directly perform the comparison between patches. The output from two different patches must be compared to train a feature-based structure. This comparison is done using the Contrastive loss [31].



(a) Siamese **(b)** Two-stream

Figure 3.5: Architectures used to train feature-based networks. a) Siamese b) Two-stream

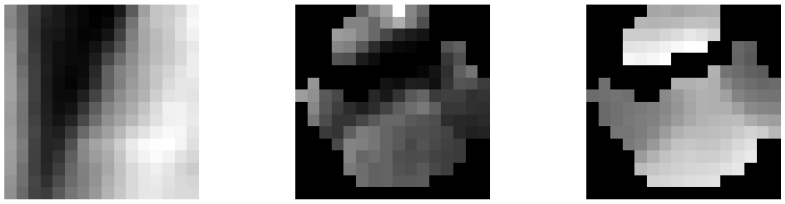
This loss C_L, also seen in Eq. (3.6), for matching pairs ($y = 1$) penalises high distances, whilst for non-matching pairs ($y = 0$) penalises low distances. This loss introduces a margin factor of m. This margin factor limits the non-matching samples to only those that are within this distance to focus the training process on ambiguous pairs. In this equation $f(x_p), f'(x_q)$ represents the output of the networks f and f' for a given patch pair x_p, x_q.

$$C_L = y \cdot |f(x_p) - f'(x_q)|^2 + (1 - y) \cdot \max\left(0, m - |f(x_p) - f'(x_q)|\right)^2 \tag{3.6}$$

Two different architectures are used in the training stage on feature-based structures: Siamese or two-stream. In a Siamese architecture, two different patches are forwarded through the same network structure to obtain two feature vectors, which are compared using Euclidean distance to obtain a distance value. This architecture is usually represented in literature as two branches that share parameters are joined using a Euclidean distance layer, as shown in Fig. 3.5. With this configuration, each branch produces the same output when presented with the same patch.

Two stream architectures are similar to siamese architectures. However, instead of using the same feature-based structure on both streams, two different structures are used on each stream. This structures may have the same layers, but the internal layer parameters are trained separately. Figure 3.5 shows a visual representation of two-stream architectures.

Siamese architectures are useful when training same-domain architectures, while two-stream architectures are used in 2D-3D registration. Technically, two-stream architectures can also be used to train same-domain architectures. However, train-



(a) 2D intensity **(b)** 3D intensity **(c)** 3D depth

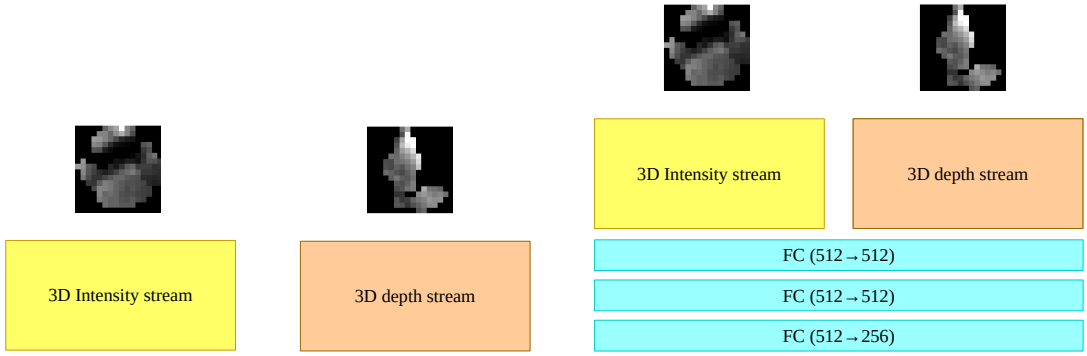
Figure 3.6: Patches obtained from registration data. b.c) Patch obtained from a keypoint using the 3D point cloud: intensity (b) and depth (c) values. a) Patch obtained from the 2D image using the same keypoint location.

ing two different feature extractors on a same-domain architecture has been shown to have lower performances than using a siamese architecture [110]. Therefore, a siamese architecture is used in this thesis when working with same domain data, and a two-stream architecture is used on multimodal scenarios.

There are other architectures used in state of the art methods to train correspondence matching. Triplet loss is a loss function that compares an anchor sample to a match and a non-match [82]. This loss is used on siamese architectures with three branches. Since the main focus of this thesis is the multimodal 2D-3D matching using two-stream architectures, this loss function has not been tested.

3.3.2 Single-Channel vs Multi-Channel Architectures

In the 2D domain, the patches obtained are image croppings with a defined size. When working with 3D data, the patch representation obtained above allows representing the surroundings of a keypoint into an RGBD lattice. However, even using this representation, the intensity patch obtained from a point cloud differs from the intensity image obtained when cropping the surroundings of the same keypoint into a 2D image. Fig. 3.6 shows an example of the patches representing a keypoint. Fig. 3.6b and Fig. 3.6c are obtained by selecting an arbitrary point on an unorganized point cloud. Fig. 3.6a is obtained by cropping an arbitrary neighbourhood on the same keypoint in the original 2D image. The difference in appearance and dimensions between the different domains suggests that the best approach is to define different network parameters for each patch type.



(a) Intensity **(b)** Depth **(c)** Intensity + depth

Figure 3.7: 3D patch structures a)Using only intensity b) Using only depth c) Combining both inputs

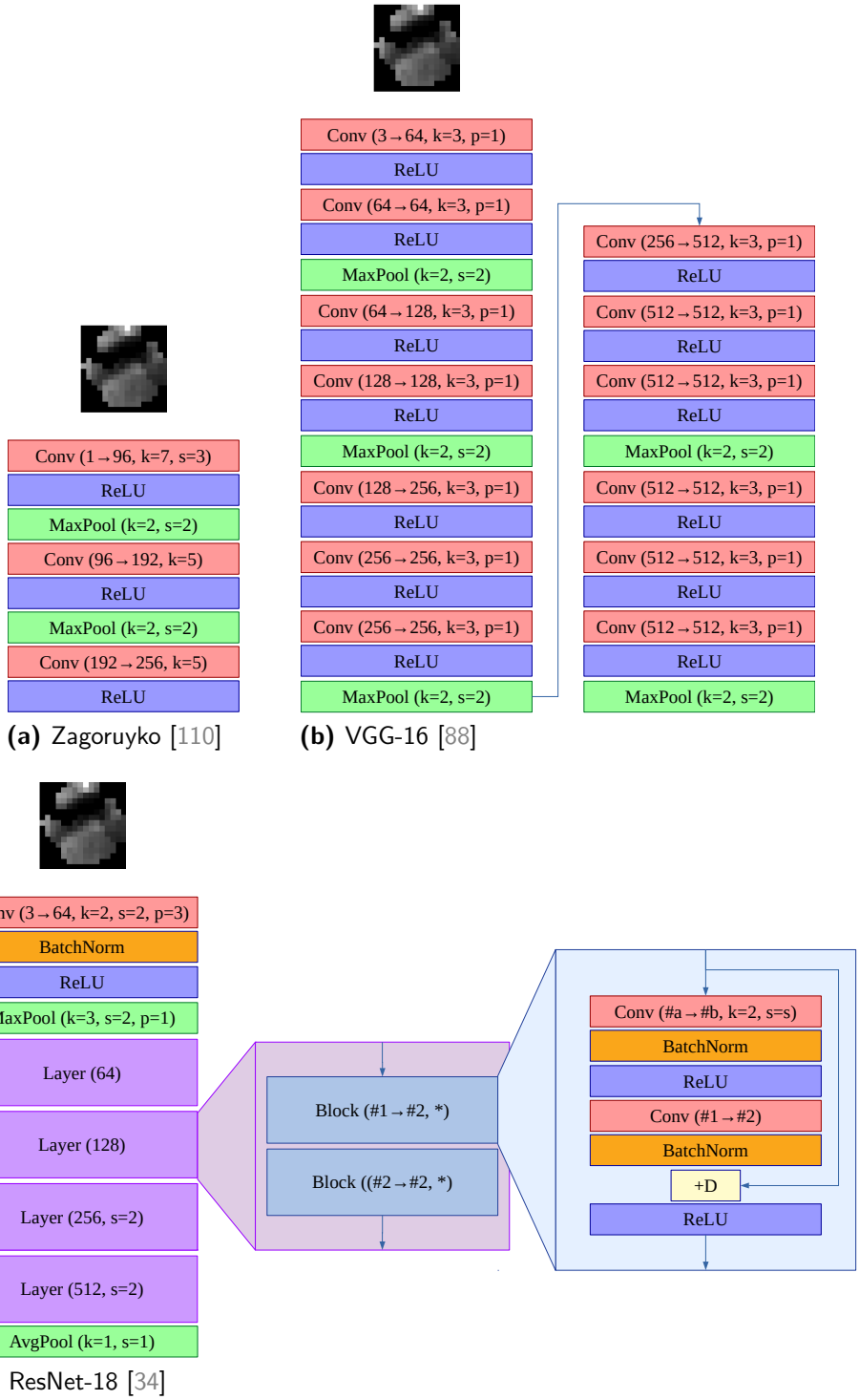
In the 2D domain, there is only a single channel available to perform the registration. Therefore the intensity patch is used to extract the feature vector. In the 3D domain, however, there are two different channels: intensity and depth. In general, when working with two-channel patches on CNNs, there are two alternatives: use only one channel as input –either intensity or depth, as seen in Fig. 3.7a and Fig. 3.7b – or to combine both channels to extract the features.

This combination can be done in early stages by feeding a two-channel patch on the network, or in late stages by concatenating the result of two network streams. [4] suggests that mixing intensity and depth channels in the same input gets worse results than providing the different channels as input on separate network streams. Therefore, a late-stage structure with two streams –without sharing parameters– is proposed, as seen in Fig. 3.7c.

Therefore, the building of the network structure is reduced to finding a suitable network structure to extract a feature vector for each one of the possible data types: 2D intensity patches, 3D intensity patches and 3D depth patches.

3.3.3 Single-channel Feature Structures

In the 2D domain, single-domain registration techniques using CNNs have already been used to extract features from 2D patches. Zagoruyko [110] also presents a network structure to extract features from 2D patches, represented in Fig. 3.8a. This



(a) Zagoruyko [110] **(b)** VGG-16 [88]

(c) ResNet-18 [34]

Figure 3.8: State of the art network structures for single-channel inputs explored in this thesis. Notation Conv($i- > n$, k=k, s=s, p=p) is a convolutional layer with i input channels, n filters of spatial size $k \times k$ applied with stride s and padding p, P(k=k, s=s, p=p) is a max-pooling layer of size $k \times k$ applied with stride s and padding p, and +D an skip connection.

network uses the same structure used in the end-to-end registration architecture to extract a feature from each patch.

This structure can be defined as a 'shallow network' due to the relatively low amount of layers. Having smaller networks has several advantages in terms of computation time and memory usage. However, a network too shallow may not allow capturing enough discriminative features to perform the correspondence matching reliably. In this thesis, the usage of deeper networks for this task is also explored. Two different structures are evaluated: the VGG structure [88] and the ResNet [34] structure.

The VGG structure [88] provides a deeper layer structure that can allow capturing finer details. The specific structure used in this thesis is the VGG-16, shown in Fig. 3.8b. As it can be seen, this specific structure contains five blocks of Convolutional - ReLu layers followed by a MaxPool layer. A pre-trained version id fine-tuned with the training data to perform the correspondence detection.

The ResNet [34] structure is the most complex network explored in this thesis. The distinctive feature of this network is the usage of skip connections to avoid vanishing gradients. The specific structure used in this this thesis is the ResNet-18, shown in Fig. 3.8c. Since both the dataset size and the patches used in both 2D and 3D domains are relatively small, deeper networks such as Resnet-50 or Resneet-101 are not relevant.

In Chapter 6 more details about the train and test settings used to obtain the different networks will be stated. Additionally, the different structures and architectures will be tested and evaluated in different settings.

3.4 Multimodal RANSAC

The procedure described in the previous sections allows computing a descriptor for a given keypoint. In this section, those descriptors are used to estimate the transformation between an image and a point cloud. This section focuses on the usage of RANSAC to estimate the transformation $T = [R|t]$ that aligns the source

points P form a 3D point cloud to the target points Q from a 2D image.

To apply this procedure M keypoints from the point cloud and M keypoints from the image are selected. These keypoints are denoted as $P = \{p_i \in \Re^3, i \in 1, \dots M\}$ for the point cloud keypoints and $Q = \{q_i \in \Re^2, i \in 1 \dots M\}$ for the image keypoints. For each one of these keypoints the corresponding descriptors s_{p_i} and s_{q_i} are computed. If the approach is successful, those descriptors with the less euclidean distance between them, denoted $d_e(s_{p_i}, s_{q_i})$, will represent corresponding points.

Using these descriptors, a set of corresponding keypoints $\mathcal{M}$ can be established. With this correspondence set, the RANSAC algorithm can be applied. The steps of this algorithm can be summarized as follows:

1. From the initial correspondence set $\mathcal{M}$, randomly select N correspondences. N should be big enough to allow a complete definition of the transformation.
2. Find the transformation that minimizes the distance between paired keypoints using the N selected keypoint pairs.
3. Measure the fitness of the obtained transformation using the complete set of correspondences $\mathcal{M}$.

These steps are performed until a good transformation is found or a certain number of iterations is reached. The output parameters of this algorithm will be the transformation of the iteration with the best fitness score.

In the next section, the different steps of this RANSAC algorithm are discussed.

3.4.1 Correspondence Selection

The basic approach to select correspondences between frames given a set of descriptors is to select the points with closer descriptors in the euclidean space. This selection is done two-fold: for each keypoint p_i in P, the point q_j in Q with closest descriptor is selected. The pair (p_i, q_j) is only chosen as a valid match if there is no point in P which descriptor is closer to the descriptor in q_j than p_i.

Formally, this match collection $\mathcal{M}$ is defined in Eq. (3.7). The descriptors for p_i and q_j are represented as s_{p_i} and s_{q_j}, respectively. $d(\cdot)$ represents the euclidean distance.

$$\mathcal{M} = \left\{ (p_i, q_j) \;\middle|\; \begin{array}{l} i = \underset{t=1\ldots N}{\arg\min}\; d(s_{p_t}, s_{q_j}) \\ j = \underset{t=1\ldots N}{\arg\min}\; d(s_{p_i}, s_{q_t}) \end{array} \right\} \subsetneq P \times Q \tag{3.7}$$

In each iteration z of the RANSAC algorithm a random subset of $\mathcal{M}$, denoted $\mathcal{N}_z = \{(p, q)\} \subsetneq \mathcal{M}$ is selected to compute the transformation. This subset must contain enough points (N) to completely define the transformation between source and target given a distance metric.

3.4.2 2D-3D Transformation Estimation

Given a set of corresponding points $\mathcal{N}_z$ selected on an iteration, the next step is to estimate the transformation $T(\mathcal{N}_z) = [R(\mathcal{N}_z)|t(\mathcal{N}_z)]$ between the 2D image and the 3D point cloud using the correspondence set $\mathcal{N}(z)$.

This section will focus on the estimation of a transformation for a single iteration. For brevity, in this section the notation z of the current iteration is dropped, and $\mathcal{N}_z$ is referred as $\mathcal{N} = \{(p, q)\}$ and $T(\mathcal{N}_z) = [R(\mathcal{N}_z)|t(\mathcal{N}_z)]$ is referred as $T = [R|t]$.

In a 3D-3D registration scenario, to determine the transformation between a set of matching points $\mathcal{O} = \{(\psi, \chi)\}$ the euclidean distance between the points of the transformed source data and the target data is minimized, as shown in Eq. (3.8). In this scenario, both ψ and χ are three-dimensional vectors. This minimization can be analytically solved using Singular Value Decomposition (SVD) as shown in [89].

$$T_{\Re^3} = \underset{R,t}{\arg\min} \sum_{(\psi,\chi)\in\mathcal{O}} |R\psi + t - \chi| \tag{3.8}$$

In a 2D-3D scenario, however, the ideal transformation is the transformation that minimizes the distance between the image coordinates q (two-dimensional) and the

3D point clouds p (three-dimensional). This distance can be defined in two domains.

On the one hand, the distance can be measured in the projection plane as the 2D euclidean distance between the 2D target pixels and the transformed 3D points projected in the 2D plane, as shown in Eq. (3.9). $\pi(\cdot)$ denotes the projection of the homogeneous 3D vector to a 2D vector by dividing it for the last coordinate, as $\pi(x) = [x^x/x^z, x^y/x^z]$ for a given $x = [x^x, x^y, x^z]$.

On the other hand, the distance can be measured as the minimum distance between the line formed by the camera centre and the pixel in the camera plane and the transformed 3D point cloud, as shown in Eq. (3.10). In both equations $\underline{q}$ represents the homogeneous coordinates of q, as $\underline{q} = [q^x, q^y, q^z, 1]$ for a given $q = [q^x, q^y, q^z$. K represents the 3×3 intrinsic camera matrix.

$$T_p = \arg\min_{R,t} \sum_{(p,q)\in\mathcal{N}} \left\|\pi\left(K\left(R \cdot p + t\right)\right) - q\right\|^2 \tag{3.9}$$

$$T_e = \arg\min_{R,t} \sum_{(p,q)\in\mathcal{N}} \frac{\left\|K^{-1}\underline{q} \times \left(R \cdot p + t\right)\right\|^2}{\|K^{-1}\underline{q}\|^2} \tag{3.10}$$

Albeit both minimizations are not exactly equivalent, [74] shows that the point-to-line measure can be made equivalent to the projected euclidean distance by rescaling each error vector using a factor $\eta = \left\|K^{-1}\underline{q} - \left(R \cdot p + t\right)\right\|^2 / \upsilon$, where υ is the distance from the point p to the projection ray [75].

It has been shown that both error metrics have similar performance in most configurations [75]. Therefore, in this thesis the error metric used to estimate the RANSAC transformation is the point-to-line distance shown in Eq. (3.10).

The point-to-line distance can also be represented using Plücker coordinates [85]. Plücker coordinates offer a representation of 2D lines using six homogeneous coordinates. This representation is used in ray tracing applications to compute relationships and distances between points and lines.

A line defined by two points A and B is defined in Plücker coordinates $l = (w : m)$ as $w = (B - A)/||B - A||$ and $m = A \times w$. The distance between the line $l(w : m)$

and an arbitrary point C is defined as $||C \times w - m||$.

In Eq. (3.10) the line is defined by the origin point $(0, 0, 0)$ and the projected point $K^{-1}\underline{q}$. The 3D point is defined by $R \cdot p + t$. Eq. (3.11) demonstrates that both distance measures are equivalent.

$$l = (w : m) = \left(\frac{(B - A)}{\|B - A\|} : A \times w \right) = \left(\frac{(K^{-1}\underline{q})}{\|K^{-1}\underline{q}\|} : (0, 0, 0) \right)$$
$$d^2 = \|C \times w - m\|^2 = \left\| (R \cdot p_i + t) \times \frac{(K^{-1}\underline{q})}{\|K^{-1}\underline{q}\|} \right\|^2 = \frac{\|K^{-1}\underline{q} \times (R \cdot p + t)\|^2}{\|K^{-1}\underline{q}\|^2} \quad (3.11)$$

This distance metric is minimized using a least-squares algorithm, as shown in [13]. The chosen optimizer in this thesis is the Levenberg–Marquardt (LM) algorithm [58], typically used in 3D-3D registration problems to estimate trasnformations using point to plane distances.

To apply the LM algorithm the transformation T is decomposed into the six basic parameters $\beta = (\alpha, \beta, \gamma, t_x, t_y, t_z)$, where α, β, γ are the rotation angles on the threee main axes and t_x, t_y, t_z are the components of the translation vector t.

The result of this minimization process leads a transformation $T = [R|t]$ that represents the optimal transformation of the correspondence set $\mathcal{N}$.

These steps are performed for various $\mathcal{N}$ neighbourhoods, denoted $\mathcal{N}_z$, yielding a transformation for each subset, denoted $T(\mathcal{N}_z) = [R(\mathcal{N}_z)|t(\mathcal{N}_z)]$.

3.4.3 Fitness Score and Stopping Criteria

The transformation obtained for each $\mathcal{N}_z$ iteration, denoted $T(\mathcal{N}_z)$, is evaluated using the full correspondence set $\mathcal{M}$. The fitness score is measured using a cost function assigned to each transformation, as shown in Eq. (3.12). The transforma-

tion with less cost (and thus, highest fitness) is selected.

$$\begin{aligned} C_z &= \sum_{(p,q)\in\mathcal{M}} \rho(p, q, R(\mathcal{N}_z), t(\mathcal{N}_z)) \\ k_\Omega &= \arg\min_z C_z \\ T &= [R(\mathcal{N}_{z_\Omega})|t(\mathcal{N}_{z_\Omega})] \end{aligned} \tag{3.12}$$

The basic cost function assigns $\rho(\cdot) = 0$ to all inline pairs and $\rho(\cdot) = 1$ to all outline pairs, as shown in Eq. (3.13).

$$\rho(p, q, R, t) = \begin{cases} 0 & \frac{\|K^{-1}\underline{q}\times(R\cdot p+t)\|^2}{\|K^{-1}\underline{q}\|^2} < \delta \\ 1 & \text{otherwise} \end{cases} \tag{3.13}$$

Therefore, this cost function counts how many inliers are present in the correspondence set for the given transformation. The transformation with most inliers at the end of all the iterations is selected.

Other cost functions can be used to select the optimal transformation. Maximum Likelihood Estimation Sample Consensus (MLESAC) [99] uses the distance value as a weight, capped into the threshold value δ', as shown in Eq. (3.14).

$$\rho(p, q, R, t) = \begin{cases} \frac{\|K^{-1}\underline{q}\times(R\cdot p+t)\|^2}{\|K^{-1}\underline{q}\|^2} & \frac{\|K^{-1}\underline{q}\times(R\cdot p+t)\|^2}{\|K^{-1}\underline{q}\|^2} < \delta' \\ \delta & \text{otherwise} \end{cases} \tag{3.14}$$

In this thesis, the improved MLESAC approach will be implemented, given that this method generally offers better performance without increasing the computational cost of the algorithm.

To define the minimal amount of iterations needed of the RANSAC algorithm to reach the optimal transformation, the probability of finding the optimal transformation on a given iteration, q, is taken into account. In the basic RANSAC implementation a subset $\mathcal{N}$ of N elements is selected from the correspondence set $\mathcal{M}$. The probability of these M points to be a full set of inliers is determined by the distri-

bution of inliers and outliers in $\mathcal{M}$. With a known percentage of inliers and outliers in $\mathcal{M}$, denoted y, the probability of having a full set of inliers in each iteration is defined in Eq. (3.15), where W represents the number of pairs in $\mathcal{M}$.

$$q = \prod_{i=0}^{W-1} \frac{yW - i}{W - i} \tag{3.15}$$

It must be taken into account that if the points are selected based on the descriptor distance, the probability cannot be straightforwardly computed. In this case, more accurate statistics of the distances given to inliers and outliers must be taken into account.

In the scenarios where the probability q of having a correct transformation on a given iteration can be computed, the number of iterations h to find the transformation with an ϵ error margin can be estimated as:

$$h = \left\lceil \frac{\log \epsilon}{\log(1 - q)} \right\rceil \tag{3.16}$$

Since this thesis applies the standard correspondence selection algorithm, the needed number of iterations to find an appropriate transformation can be estimated using the previous equations. In Chapter 8, the appropriate values for this application are explored.

3.5 Summary

The algorithm proposed in this chapter allows obtaining a registration between a 3D point cloud and a 2D image. The proposed algorithm uses deep learning techniques to estimate multimodal features. A deep neural network architecture has been proposed to estimate a set of features from 3D and 2D keypoints. The 2D-3D transformation is estimated using the RANSAC algorithm by minimizing the point-to-line distance between correspondences. The performance of this algorithm is evaluated in Chapter 6.

In the next chapter, a refinement algorithm for the same 2D-3D registration scenario is proposed.

4

2D-3D Iterative Refinement

The second part of this thesis is focused on the development of a refinement algorithm for 2D-3D registration. This algorithm can be applied as a refinement step after initial registration, such as the registration provided with the algorithm proposed in the previous chapter. It can also be applied as a standalone method to correct small displacements such as camera movements in sequences.

The proposed algorithm is an iterative registration algorithm based on the foundations of the Iterative Closest Point (ICP) method. ICP is an algorithm used to align two unorganized point clouds. To perform the alignment, only the point coordinates in which a surface has been detected are used. The algorithm aims to minimize the distance between neighbouring points, thus obtaining the transformation that provides the best alignment of the sampled surfaces represented in the point cloud.

This algorithm, however, cannot be directly applied in dense point clouds, in which a regular lattice is sampled, and a signal is measured on each point. This is the case, for example, of the volumetric image obtained from an MRI scan. The brain surface is detected on the data to be registered to apply the ICP algorithm to an

MRI scan [17, 36], reducing the problem to a surface registration problem.

2D images can also be defined as a sampling in a regular lattice with a value measured on each point. Therefore, traditional ICP techniques can not directly be applied. Since in the 3D domain surfaces are detected to perform the registration, an equivalent detection should be made into the 2D images.

In three-dimensional space, surfaces can be defined as the transition between two volumes. This transition is directly detected when using range sensors. When working with volumetric images, changes in the measured values are marked as surfaces. In a two-dimensional space surfaces define the entirety of the captured plane, and thus cannot be used to perform the ICP registration.

In the 2D domain, the equivalent measure to the 3D surfaces, defined as a transition between 3D volumes, would be to detect 2D edges or transitions between 2D surfaces. Therefore, the edges can be detected as changes in the measured values in the 2D surface. This procedure is used in Simultaneous Localization And Mapping (SLAM) techniques to obtain correspondences between 2D images [98].

Unlike the SLAM scenario, in which all the edge detection is performed in the 2D domain, in the scenario presented in this thesis, the detection must be performed in different modalities. In this chapter, a multimodal edge detection algorithm is proposed. This algorithm has two variants depending on the dimension of the input data. In the 3D domain, both changes in the surface slope and the intensity values are taken into account. In the 2D domain only changes to the intensity value are taken into account.

Once the edges in the different dimensions are obtained, the next step is to perform the ICP registration process. The considerations on the distance metric selections in the different parts of the algorithm will also be explained.

The complete pipeline of the proposed procedure is shown in Fig. 4.1.

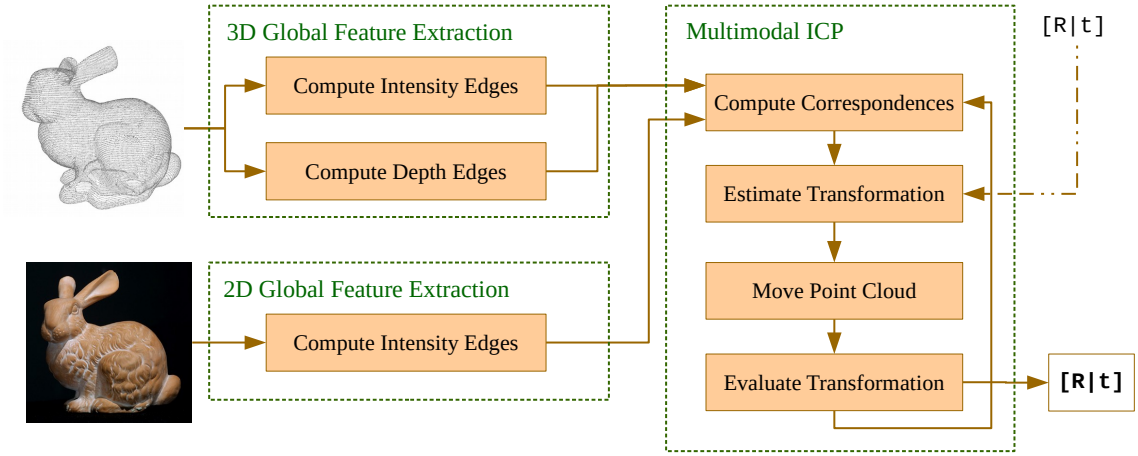


Figure 4.1: Parts of the proposed refinement algorithm.

4.1 3D Edge Detection

The contours that can be detected in a 2D image are based on changes in the intensity values captured by the camera. These changes can be caused by different sources, such as colour variations, surface reflections and object occlusions.

First of all, the objects present in the scene can have different colours in different parts of the object. These changes can be measured by detecting variations on the intensity values between neighbouring 3D points.

An object with a solid colour can also present intensity variations in the 2D image when light is reflected from different angles. In the 3D domain, these changes can be detected as variations in the surface slopes.

Finally, another source for image contours is object occlusions. In this case, the transition between two non-neighbouring objects causes an intensity change in the 2D image.

The edge detection algorithm presented in this thesis aims to detect changes in intensity values caused by light reflections and slope changes. Unlike occlusions, both of these contours are independent of the viewpoint and can be computed without knowing the transformation between the image and the point cloud.

The proposed algorithm uses a state of the art geometric edge detector and enhances

it with an intensity edge detector. The algorithm is constructed leveraging the multiscale procedure of the geometric edge detector to obtain a combined geometric and intensity edge detector.

Both the geometric edge detector and the proposed intensity edge detector algorithm yield a score for each point. The geometric score and the intensity score are combined into a single score that is thresholded to detect edge points.

4.1.1 Geometric Edge Detection

The state of the art algorithm proposed by Pauly [66] is used to detect geometric contours in the 3D point cloud. This algorithm uses a PCA-based approach to select the points with high curvature on their neighbourhood.

The algorithm presented in [66] follows the following procedure to detect contour changes on each point p_i of a point cloud P:

1. Select the k closest points to the point p_i using a KD-tree approach, denoted $\mathcal{X}_{p_i}(k) = \{p_m\} \subsetneq P$.
2. Compute the mean vector $\overline{\mathcal{X}_{p_i}(k)}$ and the covariance matrix $C(\mathcal{X}_{p_i}(k))$, defined as:

$$\overline{\mathcal{X}_{p_i}(k)} = \frac{1}{k} \sum_{p_m \in \mathcal{X}_{p_i}(k)} p_m \tag{4.1}$$

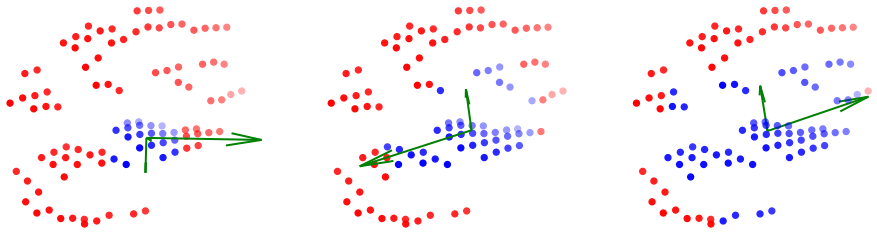
$$C(\mathcal{X}_{p_i}(k)) = \frac{1}{k} \begin{bmatrix} p_1 - \overline{\mathcal{X}_{p_i}(k)} \\ p_2 - \overline{\mathcal{X}_{p_i}(k)} \\ \cdots\cdots \\ p_k - \overline{\mathcal{X}_{p_i}(k)} \end{bmatrix}^T \cdot \begin{bmatrix} p_1 - \overline{\mathcal{X}_{p_i}(k)} \\ p_2 - \overline{\mathcal{X}_{p_i}(k)} \\ \cdots\cdots \\ p_k - \overline{\mathcal{X}_{p_i}(k)} \end{bmatrix}, p_m \in \mathcal{X}_{p_i}(k) \tag{4.2}$$

3. The surface variation is computed as:

$$\sigma^g(\mathcal{X}_{p_i}(k)) = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2} \tag{4.3}$$

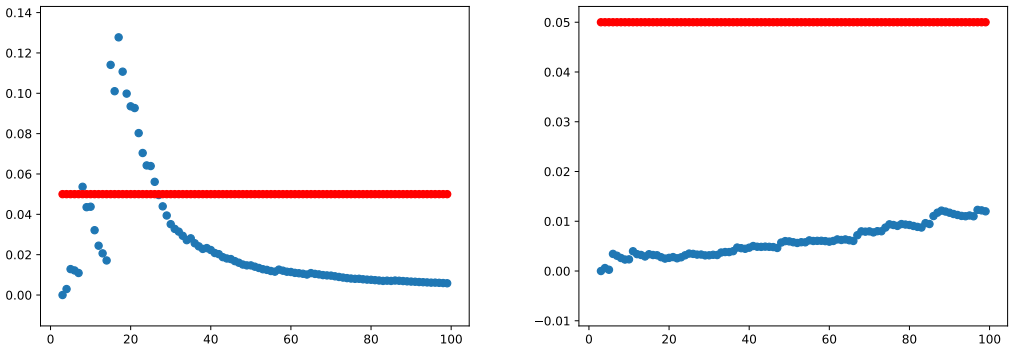
   where λ_j are the eigenvalues of $C(\mathcal{X}_{p_i}(k))$ in increasing order $\lambda_0 \leq \lambda_1 \leq \lambda_2$.

To illustrate this procedure, Fig. 4.2 shows the eigenvectors, which length is scaled to the corresponding eigenvalue, for three different k values.



(a) k=20 **(b)** k=40 **(c)** k=60

Figure 4.2: Representation of the eigenvectors (green arrows) for three different neigbourhood sizes (in blue).



(a) Edge surface **(b)** Flat surface

Figure 4.3: Behaviour of the surface variation $w_{g,i}$ for different k neighbourhoods (blue). Two examples are shown: for a edge surface and for an flat surface. The threshold $\sigma_{max} = 0.05$ is also shown as a red dotted line.

The previous steps are repeated for different k values, between $K^g_{\min}$ and $K^g_{\max}$. For each $\mathcal{X}_{p_i}(k)$ neighbourhood a $\sigma(\mathcal{X}_{p_i}(k))$ surface variation can be computed. To compute the feature weight for the point p_i the different $\sigma_{p_i}(k)$ are thresholded:

$$w^g_{p_i} = \frac{1}{K^g_{\max} - K^g_{\min}} \sum_{k=K^g_{\min}}^{K^g_{\max}} \omega\left(\sigma^g(\mathcal{X}_{p_i}(k))\right) \qquad \omega(\sigma^g) = \begin{cases} 1 & \sigma^g > \sigma^g_{\max} \\ 0 & \sigma^g \leq \sigma^g_{\max} \end{cases} \tag{4.4}$$

In Fig. 4.3 two examples of the behaviour of the surface variation $\sigma(\mathcal{X}_{p_i}(k))$ for different k neighbourhoods are shown. The first example represents the surface variation obtained on a flat surface and the second example shows the behaviour on a strong contour.

4.1.2 Intensity Detection

The proposed intensity edge detection algorithm leverages the multiscale approach of the geometric edge detector algorithm to compute intensity-based contours.

In each step of the geometric detection algorithm, an increasing k neighbourhood is selected. This neighbourhood can also be used to detect intensity features on each $\mathcal{X}_{p_i}(k)$ neighbourhood.

The intensity features are defined as abrupt intensity changes in a dominant direction in a small neighbourhood. Therefore, the intensity feature detection algorithm we propose can be applied to point clouds which have an intensity value assigned to each point. This algorithm is based on the computation of the geometrical center $\overline{\mathcal{X}_{p_i}(k)}$ and the center of mass $M_{\mathcal{X}_{p_i}(k)}$ for each point p_i and each neighbourhood k.

The geometrical centre is computed as the mean of all keypoints shown in Eq. (4.1), as performed in the geometric edge detection algorithm. Therefore, this value is only computed once in each step and used in both intensity and depth edge detection.

In physics, the center of mass is the point where the weighted position vectors relative to this point sum to zero. In this work, the center of mass is analogously defined as the mean location weighted by the intensity value, as defined in Eq. (4.5). In this equation y_m represents the intensity value for each point p_m.

$$M_{\mathcal{X}_{p_i}(k)} = \frac{1}{k} \sum_{p_m \in \mathcal{X}_{p_i}(k)} y_m \cdot p_m \tag{4.5}$$

Fig. 4.4 shows the location of the geometrical center $\overline{\mathcal{X}_{p_i}(k)}$ (red cross) and the center of mass $M_{\mathcal{X}_{p_i}(k)}$ (blue star) in several cases. There is a shift between the geometrical centre and the mass centre when a centred sharp contour is present(left). However, when the contour is skewed, the behaviour of the mass centre changes depending on the predominant luminance level (middle left, middle right).

It should also be noted that the presented algorithm does not have a strong response

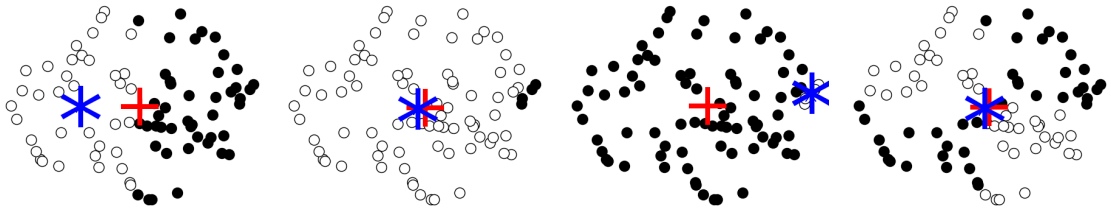


Figure 4.4: Test cases for the intensity feature detection algorithm. Black and white points: luminance values in the analysis neighborhood. Red cross: geometrical center. Blue star: center of mass. Left: Centered contour. Middle-left: High luminance patch with side contour. Middle-right: Black patch with side contour. Right: Saddle point in luminance.

on saddle points (right). However, in the use case presented in this thesis, this fact has no significant impact on the result.

To mitigate the different behaviour on skewed contours an additional centre of mass is computed. This center of mass $M'_{\mathcal{X}_{p_i}(k)}$ inverts the intensities to balance the response for different intensity patterns, as shown in Eq. (4.6). The variance score for each k neighbourhood $\sigma^g(\mathcal{X}_{p_i}(k))$ is computed as the minimum distance between the geometrical centre and the centre of masses, as shown in Eq. (4.7).

$$M'_{\mathcal{X}_{p_i}(k)} = \frac{1}{k} \sum_{p_m \in \mathcal{X}_{p_i}(k)} (1 - y_m) \cdot p_m \tag{4.6}$$

$$\sigma^y(\mathcal{X}_{p_i}(k)) = min\left(\|\overline{\mathcal{X}_{p_i}(k)} - M_{\mathcal{X}_{p_i}(k)}\|^2, \|\overline{\mathcal{X}_{p_i}(k)} - M'_{\mathcal{X}_{p_i}(k)}\|^2 \right) \tag{4.7}$$

This dual computation allows to have a symmetrical response on the contours and select only those points that have an abrupt intensity change on their close neighbourhood. This method also avoids thick contours when increasing the analysis neighbourhood.

This variance score is computed in increasing k levels, similarly to Pauly algorithm [66]. The intensity edge score $w^y_{p_i}$ is incremented each time that $\sigma^y(\mathcal{X}_{p_i}(k))$ is higher than a certain threshold $\sigma^y_{\max}$. However, the absolute distance between the two centers also depends on the search radius. $\sigma^y(\mathcal{X}_{p_i}(k))$ is normalized with the maximum distance of the h neighbors to the query point $\nu = d_{\max}$. Therefore, the

intensity edge score is computed as shown in Eq. (4.8).

$$w_{p_i}^y = \frac{1}{K_{\max}^y - K_{\min}^y} \sum_{k=K_{\min}^y}^{K_{\max}^y} \omega\left(\sigma^y(\mathcal{X}_{p_i}(k))\right) \qquad \omega(\sigma^y) = \begin{cases} 1 & \sigma^y/\nu > \sigma_{\max}^y \\ 0 & \sigma^y/\nu \leq \sigma_{\max}^y \end{cases} \tag{4.8}$$

4.1.3 Score Combination and Further Optimizations

The presented multiscale analysis allows computing both geometric and intensity contours in a single pass. The geometric contours are computed using Pauly's multiscale feature detection [66], and the intensity contours are computed using the proposed feature detection method.

Intensity and gradient scores are computed for each point p_i in the cloud, obtaining a $w_{p_i}^g$ score for the gradient detection and a $w_{p_i}^y$ score for the intensity detection. These two scores are combined as shown in Eq. (4.9).

$$w_{p_i} = max\left(w_{p_i}^g, w_{p_i}^y\right) \tag{4.9}$$

The original article used a minimum spanning tree followed by a smoothing step to obtain the 3D contours. However, the application proposed in this thesis requires having neither smooth not thin contours. To reduce the computational cost of the algorithm simple thresholding is performed to select the contour points with $w_{p_i} > th$ as edge points.

An additional optimization is proposed to reduce the computational speed of the algorithm. A typical point cloud contains large flat areas, both in terms of contour and depth. Therefore, an algorithm is proposed to avoid computing the edge score w_{p_i} on all points.

The proposed algorithm is a watershed-like algorithm in which a random subset of the original point clouds $\mathcal{S} \subsetneq P$ are selected as seed points. For each one of this seed points p_s, the score w_{p_s} is computed. If $w_{p_s} > th * \alpha$, with $\alpha \leq 1$, all points on a small neighbourhood $k = 5$ are added on the subset $\mathcal{S}$. The algorithm finishes

when no points are left in S without computing their score.

This algorithm has been shown to reduce the computational cost significantly, especially in point clouds with low textured areas.

4.2 2D Edge Detection

The 2D edge detector proposed on this thesis aims to find intensity contours on 2D images that are related to geometric and intensity changes in the 3D domain. To obtain coherent detections between both domains, the 3D intensity edge detection algorithm is adapted to 2D images.

The algorithm presented in this thesis has two main modifications from the version of this algorithm introduced in [72]: the algorithm used to select the k-neighbourhood and the computation of the geometrical centre and centre of mass in a 2D image. In this section, the steps to compute the edge score w_{q_i} for a 2D pixel q_i in an image Q are explained.

Analogously to the k-neighbourhood in the 3D domain, in each step of the algorithm a $(2k+1)\times(2k+1)$ patch, centred in the query point q_i in the image, is selected. This patch is selected with radius K and, if the patch size exceeds the image boundaries, a mirror of the image is used in the cropping. The pixels included in this cropping are defined as $\mathcal{X}_{q_i}(k)$.

Unlike the 3D neighbourhood, this 2D patch is a regular lattice with well-defined distances between pixels. Therefore, in a 2D patch, the geometrical centre $\overline{\mathcal{X}_{q_i}(k)}$ can be defined as the centre point of the patch. This point is defined as the coordinate origin, with $\overline{\mathcal{X}_{q_i}(k)} = (0, 0)$. Likewise, the coordinates of each pixel q_i respect to the patch centre are defined within the range $[-k, k]$.

Using this coordinate origin, the center of masses $M_{\mathcal{X}_{q_i}(k)}$ is defined using in the

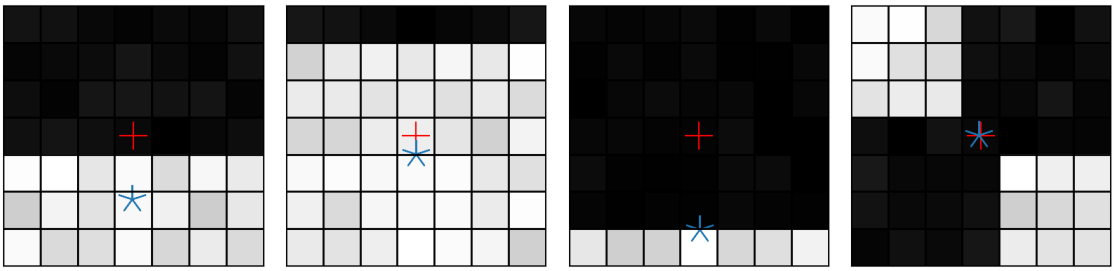


Figure 4.5: Test cases for the intensity feature detection algorithm in a 2D patch with K=3. Red cross: geometrical center. Blue star: center of mass. Left: Centered contour. Middle-left: High luminance patch with side contour. Middle-right: Black patch with side contour. Right: Saddle point in luminance.

same procedure applied in the 3D domain, denoted in Eq. (4.10).

$$M_{\mathcal{X}_{q_i}(k)} = \frac{1}{(2k+1)^2} \sum_{q_m \in \mathcal{X}_{q_i}(k)} y_m \cdot q_m \tag{4.10}$$

As shown in Fig. 4.5, the weighted mean of the 2D patch using the intensity values has the same behaviour as the weighted mean of the close neighbourhood of a 3D point. If the patch contains a skewed contour the response is non-symmetrical. To solve this issue the same symmetrical center of masses $M'_{\mathcal{X}_{q_i}(k)}$ is computed in the same procedure applied in the 3D domain, denoted in Eq. (4.11).

$$M'_{\mathcal{X}_{q_i}(k)} = \frac{1}{(2k+1)^2} \sum_{q_m \in \mathcal{X}_{q_i}(k)} (1 - q_m) \cdot q_m \tag{4.11}$$

The variation score $\sigma^y(\mathcal{X}_{p_i}(k))$ is computed using the same procedure defined in Eq. (4.7). however, since in the 2D patch the geometrical center $\overline{\mathcal{X}_{q_i}(k)} = (0, 0)$, this equation can be simplified as shown in Eq. (4.12)

$$\sigma^y(\mathcal{X}_{q_i}(k)) = min\left(\|M_{\mathcal{X}_{q_i}(k)}\|^2, \|M'_{\mathcal{X}_{q_i}(k)}\|^2\right) \tag{4.12}$$

In the 2D images, the edge score for each image point on a $k \times k$ patch , w_{q_i}, is directly defined as $w^y_{q_i}$, since no edge contours are computed. This edge score for each point w_{q_i} is computed iterating over different k values, as shown in Eq. (4.13).

In this case, however, the normalizing factor ν is defined as the patch radius k.

$$w_{q_i} = w^y_{q_i} = \sum_{k=K_{min}}^{K_{max}} \omega(\sigma^y(\mathcal{X}_{q_i}(k))) \qquad \omega(\sigma) = \begin{cases} 1 & \sigma/\nu > \sigma_{\max} \\ 0 & \sigma/\nu \leq \sigma_{\max} \end{cases} \tag{4.13}$$

It must be noted that there are some dissimilarities between the 2D and 3D intensity edge detection algorithms regarding the argument parameters.

In the 3D intensity edge detection algorithm, the parameter k represents the number of points in the neighbourhood. In the 2D algorithm, however, the parameter k represents the patch radius, with each patch containing $(2k+1)^2$ points.

Moreover, the normalizing factor ν is computed differently on both algorithms. Albeit the computations offer similar results, the optimal threshold value $\sigma_{\max}$ may differ between algorithms.

Therefore, the different parameters of the algorithms should be independently selected for the 2D and 3D algorithms.

4.3 Iterative Transformation Estimation

The algorithms described above allow detecting edges on both 2D and 3D domains. The algorithms are not tuned to obtain a precise line detection, but instead to obtain the same detections on both domains. Furthermore, while having similar performances, the algorithm with an overall lower number of detections will be preferred, to speed up the matching process.

Therefore, the result of these algorithms will be a set $P = \{p_i \in \Re^3, i \in 1 \dots M\}$ with the detected edge points on the 3D point cloud and a set $Q = \{q_j \in \Re^2, j \in 1 \dots N\}$ with the detected pixels on the 2D image. To find the transformation $T = [R|T]$ that aligns both sets the ICP algorithm is used.

Using an established state of the art algorithm for the registration step has one main advantage: most performed improvements on the original algorithm, as well

as possible future developments, can be adapted to the proposed method.

The original Iterative Closest Point (ICP) is an iterative algorithm used to find the optimal transformation between two 3D point clouds by minimizing the distance between neighbouring 3D points. A high-level summary of the ICP procedure is as follows:

1. Establish correspondences between point clouds using the Euclidean distance between points and selecting the nearest neighbour.

2. Compute the rigid (rotation-translation) transformation that minimizes the Euclidean distance between points.

3. Iterate until convergence.

The modifications introduced in each step of the algorithm to adapt it to the 2D-3D registration process are described in the next sections. This modified algorithm is referred to as EdgeICP in future sections.

4.3.1 Correspondence Selection

The first step in each iteration is to match each point in the image to their closest corresponding point in the 3D point cloud, obtaining a correspondence set $\mathcal{M}_z$ for each z iteration.

The image points Q, –denoted target points– are fixed during the iterations. The points P of the point cloud –denoted source points–are transformed in each z iteration with the transformation estimation on the previous iteration, giving $P(T_{z-1})$. To describe the initial point set $P(T_{-1})$ the transformation obtained in the initial registration is used.

This section will focus on the estimation of the correspondences $\mathcal{M}_z$ between the points Q and the points $P(T_{z-1})$. For simplicity, the z notation is dropped in this section, and $\mathcal{M}_z$ is referred to as $\mathcal{M}$ and $P(T_{z-1})$ is referred to as P.

If both sets were in the 3D domain, the distance between points would be measured as the euclidean distance, and a KD-tree could be constructed to search efficiently for closest points.

In the presented 2D-3D scenario, however, the distance is defined as the point to line distance between each 2D point q_j and each 3D point p_i, as shown in Eq. (4.14), where $\underline{q_j}$ represents the point q_j in homogeneous coordinates, as $\underline{q_j} = [q_j^x, q_j^y, 1]$ for a given $q_j = [q_j^x, q_j^y]$.

$$d_p(p_i, q_j) = \frac{\|p_i \times K^{-1}\underline{q_j}\|}{\|\underline{q_j}\|} \qquad \forall i \in 1 \dots M, j \in 1 \dots N \tag{4.14}$$

This distance measure, however, does not fulfil the requirements to be a metric distance and thus a KD-tree representation cannot be used to search for neighbouring points.

There are two main alternatives to find neighbouring points in the 2D-3D domain: use a full search between all possible point pairs, or use an equivalent metric distance in a KD-tree.

A full search between all possible points implies measuring the distance between all $M \times N$ possible pairs. Since M and N are significantly large in most cases, this strategy is computationally expensive.

The euclidean distance in the projected camera plane, explained in Chapter 3, is another measure that can be used to establish matching correspondences. This distance, shown in Eq. (4.15), represents the distance between a 2D point and the projected 3D point into the camera plane. In this equation, K is the camera matrix and $\pi(\cdot)$ denotes the projected of a 3D vector into a 2D space by dividing by the third coordinate.

$$d_e(p_i, q_j) = \pi(Kp_i) - q_i \qquad \forall i \in 1 \dots M, j \in 1 \dots N \tag{4.15}$$

By representing all 3D points into their 2D projections $\tilde{p}_i = \pi(Kp_i)$, this distance fulfills all the requirements to be considered a metric distance and can be used

to build a KD-tree. Using this KD-tree each point q_i is matched with a point p_j, obtaining a set of matches denoted $\tilde{\mathcal{M}}$, formally defined in Eq. (4.16).

$$\tilde{\mathcal{M}} = \left\{ (p_i, q_j) \;\middle|\; \begin{array}{l} i = \underset{t=1\ldots M}{\arg\min}\; d_e(p_i, q_j) \\ j \in 1 \ldots N \end{array} \right\} \subsetneq P \times Q \tag{4.16}$$

However, for each element $(p, q) \in \tilde{\mathcal{M}}$, p this set does not define the closest point to q in terms of the point-to-line distance d_p.

However, there is a relationship between the distances d_e and d_p. In the point clouds where there are not great disparities between depth values on the close neighbourhood, the points with a low d_e also have a low d_p.

The approach followed in this thesis is to select the r closest points to q, measured in the d_e distance, giving the set $\mathcal{R}_p$. Since the subset $\mathcal{R}_p$ is significantly smaller than P, the distance $d_p(p_k, q_j)$ can be computed for each point p_k in the subset $\mathcal{R}_p$.

This computation gives the new match set $\mathcal{M}$, with pairs of multimodal points locally minimizing the point to line distance. This set is defined formally in Eq. (4.17).

$$\mathcal{M} = \left\{ (p_k, q_k) \;\middle|\; \begin{array}{l} p_k = \underset{p_k in \mathcal{R}_p}{\arg\min}\; d_p(p_k, q_j) \\ j \in 1 \ldots N \end{array} \right\} \tag{4.17}$$

Although the points in this set are not guaranteed to have a minimal d_p distance, the tests performed show that this correspondence errors do not hinder the performance of the overall registration.

A correspondence set $\mathcal{M}$ is defined in each iteration z of the EdgeICP algorithm, being denoted $\mathcal{M}_z$ the correspondence set $\mathcal{M}$ obtained in the zth iteration.

4.3.2 Transformation Estimation

Once the set $\mathcal{M}_z$ has been obtained, the next step is to estimate the transformation $T(\mathcal{M}_z)$ that minimizes the sum of the errors between each pair of points. Using the same abbreviation applied in the previous section, the notation z of the current iteration is dropped, referring to $\mathcal{M}_z$ as $\mathcal{M}$ and to $T(\mathcal{M}_z)$ as T.

In Section 3.4.2, a similar scenario is proposed, where a small subset of the correspondence set is selected to estimate the transformation. The chosen strategy to estimate the transformation on each RANSAC iteration has been to minimize the point to line distance using the Levenberg–Marquardt (LM) algorithm.

LM is a non-linear least-squares minimization algorithm used to solve curve-fitting problems. In this case, the goal is to find the six six basic parameters of the transformation T, defined as $\beta = (\alpha, \beta, \gamma, t_x, t_y, t_z)$.

In each iteration l of the LM algorithm the parameters β_l are estimated using the previous parameters β_{l-1} and the vector of differences δ_l, as $\beta_l = \beta_{l-1} + \delta_l$. Eq. (4.18) is solved to find the vector of differences δ_l. In this equation e_l represents the vector of residuals and J_l represents the Jacobian matrix computed in each step l.

$$(J_l^T J_l + \lambda I)\delta_l = J_l^T e_l \tag{4.18}$$

There are two key differences between the scenario presented in the transformation estimation on each RANSAC iteration and the transformation estimation on each EdgeICP iteration: the number of correspondences and the initial transformation.

While in the RANSAC estimation only enough points to determine the transformation are used, typically a number lower than 10, on the EdgeICP estimation all the edge points are used in the computation. Although the edge detections algorithms are tuned to prefer a low point count, the number of detections will still be over the thousands.

Having more points to compute the transformation increases the computation com-

plexity of the algorithm. Therefore, strategies to minimize the computational cost of each iteration must be applied.

LM-ICP, presented in [22], proposes to compute one single iteration of the LM minimization in each ICP step. Unlike RANSAC, in the ICP pipeline the previous transformation is already a good estimation of the next transformation, and therefore using a single step will yield a suitable transformation.

Using this method a transformation $T(\mathcal{M}_z)$ is found in each z iteration, with the transformation being defined by $\beta_{T(\mathcal{M}_z)} = \beta_{T(\mathcal{M}_{z-1})} + \delta_z$.

4.3.3 Transformation Evaluation

The obtained transformation is evaluated in each step of the EdgeICP algorithm. This evaluation is used to stop the iterations when the algorithm is considered to have reached the optimal transformation. A fitness score is computed in each step z of the eDgeICP algorithm. The algorithm is considered to have reached the optimal transformation when the increment of this score is under a threshold ϵ, as $fitness(z) - fitness(z-1) < \epsilon$.

To measure the fitness of the transformation the Root Mean Squared Error (RMSE) is used. The RMSE in an iteration z is defined as the sum of squared error for all matching points in the iteration, as shown in Eq. (4.19). In this equation $\mathcal{M}_z$ defines the set of matching points in the iteration z.

$$RMSE(z) \sum_{p,q \in \mathcal{M}_z} d_p(p,q) \tag{4.19}$$

Since the RMSE is minimal for the optimal transformation, the stopping criteria is achieved when $RMSE(z-1) - RMSE(z) < \epsilon$. The criteria used to select the threshold ϵ is discussed in Chapter 8

4.4 Summary

The procedure described in this chapter allows refining an initial estimation of a transformation between a single image and a point cloud. The proposed algorithm uses a multimodal edge detection procedure to obtain coherent detections on both 2D and 3D domains. These edge detections are used in a modified version of the ICP algorithm to estimate the transformation between a 2D image and a 3D point cloud.

When combined with the proposed initial registration described in Chapter 3, this algorithm allows the estimation of the transformation without any additional information on the scene.

In the next part, the different steps of the complete pipeline presented in Chapters 3 and 4 are evaluated.

Part II

Experiments

5

Experimental Setup

An algorithm to obtain a multimodal 2D-3D registration has been proposed in the previous chapters. This algorithm should be able to obtain the transformation between an image and a point cloud without needing any additional information.

The algorithm proposed in this thesis has been divided into two main blocks. In the first block, a set of correspondences are estimated using deep learning techniques. These correspondences are used to obtain an initial estimation of the transformation.

In the second block, a refinement technique is proposed. This refinement technique uses the activations of a multi-modal edge detection algorithm to refine the initial transformation iteratively.

This part presents a comprehensive evaluation of the different steps of each block needed to obtain the registration. This part is organised into four chapters.

First of all, an overview of the experimental setup is stated. The tools, databases and

evaluation metrics used in the different steps are explained. The detailed information about the experimental setup can be found in Appendix A.

The different steps of each main block are evaluated in the next chapters. Chapter 6 is devoted to assessing the influence of each step on the accuracy of the correspondence matching algorithm. A benchmark of this algorithm against state of the art methods is shown in Section 6.5.

The impact of the different parameters of the 2D-3D edge detection algorithm is stated in Chapter 7. A benchmark of this algorithm against state of the art methods is provided in Section 7.6.

Finally, the overall registration pipeline is tested and compared against state of the art algorithms. Both the initial estimation algorithm and the refinement algorithm are evaluated in Chapter 8 to obtain the transformation. The performance of each part, alongside the performance of the complete pipeline, is evaluated. Finally, a benchmark against state of the art methods is provided in Section 8.3.

The different aspects of the procedure followed in carrying out the experiments described in this part are explained in this chapter. This explanation is divided into three main blocks: the datasets used in each step of the process, the evaluation metrics used in the proposed evaluations and the software and hardware in which the tests are carried out.

5.1 Data sources and datasets

Data from several data sources have been used to develop and evaluate the different algorithms presented in this thesis. This data is used on the different stages of the development to evaluate the performance of the proposed algorithms.

The data sources used in the development of this thesis are:

- Manual RGBD captures performed in the smart room. Three sequences with 50 frames per sequence are captured using a handheld Kinect camera with no

location information. These captures are used as visual tests on early stages of the edge detection algorithm.

- Stanford bunny. This largely used capture is an unorganised 3D scan of a bunny statue. In addition to the 3D scan, an RGB photography of the same statue is also available, though no registration data between the image and the 3D scan is available. This capture is used in this thesis as a visual test on the iterative refinement registration process.
- CoRBS dataset [103]. This dataset contains several registered RGBD sequences of four different scenes. In this thesis, 300 frames from two different sequences are used on initial testing stages of the correspondence matching algorithm.
- TUM RGB-D SLAM Dataset [95]. This dataset contains several short sequences of registered RGBD frames in various scenarios. Nine of these sequences, each one containing between 200 and 2000 frames, are used on initial developments of the training stage of the correspondence matching algorithm.
- 3DMatch dataset [111]. This data source contains a compilation of several registered RGBD datasets. These sequences make the largest dataset used in the training and testing stage, containing 47 different sequences with lengths between 600 and 15000 frames.

In Appendix A.1 detailed information about the different data sources can be found.

From these data sources, several datasets have been built in the different stages of the development of this thesis. The results presented in this thesis, however, are focused on two primary datasets created to evaluate the different parts of the algorithms: a dataset of matching keypoints and a dataset of overlapping frame pairs.

Similarly to [111], the matching keypoints dataset is built by choosing random keypoints on two frames of the same sequence and labelling them as match or non-match. A balanced set of matches and non-matches is selected. This dataset is used to evaluate the CNN correspondence matching algorithm presented in Chapter 3. More details about this dataset can be found in Appendix A.1.2.

The overlapping frame pairs dataset is built by selecting random frame pairs with at least a 30% overlapping between them. This frame pairs are used both to evaluate

the edge detection algorithm presented in Chapter 4 and the modified RANSAC and ICP registration procedures. More details about this dataset can be found in Appendix A.1.3.

5.2 Metrics

The output of the algorithm presented in this thesis is a 4×4 transformation matrix representing the rotation and translation that aligns the point cloud to the 2D image. Several metrics are used in the literature to evaluate this transformation, such as the translation distance error or the difference between rotation angles.

To evaluate this transformation in a 3D-3D registration, the metric developed in [15], also used in [111], can be used. This metric is defined as the RMSE between ground-truth 3D correspondences $\mathcal{K}_{ij}^{*} = (p*, q*)$, defined as shown in Eq. (5.1). In this equation $(p*, q*)$ represents a pair of 3D points, T_{ji} represents the evaluated transformation and τ is the distance threshold.

$$E_{RMSE}^{2} = \frac{1}{\mathcal{K}_{ij}^{*}} \sum_{(p^*, q^*) \in \mathcal{K}_{ij}^{*}} \|p^* - T_{ji} q^*\|^2 \leq \tau^2 \tag{5.1}$$

Although this thesis presents a 2D-3D registration scenario, this metric can be used since the ground truth 3D-3D information is available in the datasets used. The parameter values used on the computation of this metric can be found in Appendix A.2.1.

However, this evaluation metric only takes into account the Euclidean distance between the original points and transformed points. In a 2D-3D scenario, in addition to valuing this distance, other measures are of interest. Some of these metrics are the projection error on the 2D plane and the reprojection error in the 3D space. To evaluate these errors, two new evaluation metrics, based on the distance measures minimised in the registration process, are proposed.

The first error measures the error in the projection plane as the 2D euclidean distance

between the 2D target pixels and the transformed 3D points projected in the 2D plane. This error metric is shown in Eq. (5.2), where $d_e(p, q)$ represents this distance measure between a projected 3D point p into the 2D camera plane and a 2D point q.

$$E^2_{RMSE,e} = \frac{1}{\mathcal{K}_{ij}} \sum_{(p,q)\in\mathcal{K}_{ij}} d_e(p, q) \leq \tau_e^2 \tag{5.2}$$

The second error measures the minimum distance between the line formed by the camera centre and the pixel in the camera plane and the transformed 3D point cloud. This error metric is shown in Eq. (5.3), where $d_p(p, q)$ represents this distance measure between a 3D point p and the line 3D line defined by the 2D point q.

$$E^2_{RMSE,p} = \frac{1}{\mathcal{K}_{ij}} \sum_{(p,q)\in\mathcal{K}_{ij}} d_p(p, q) \leq \tau_p^2 \tag{5.3}$$

More details about these metrics and their computation can also be found in Appendix A.2.1.

In addition to the global registration metric, each part of the registration algorithm is also independently evaluated.

The main contribution of the first part of this thesis is the multi-modal 2D-3D correspondence matching algorithm. A balanced set of matching/non-matching points is created on the testing dataset split to evaluate this algorithm. The accuracy of the algorithm is measured as the false positive rate at 95% recall ($FPR@95$). The details on this metric can be found in Appendix A.2.2.

The main contribution in the second part of this thesis is the multi-modal edge detection. In this part, the evaluation differs from traditional edge detection algorithms, in which the main goal is to obtain a set of thin lines that define contours in the scene.

In the scenario presented in this thesis, a good edge detection algorithm must have high repeatability –correct detections vs all detections– while having an overall low amount of detections. The proposed measure favours both having a similar amount

of keypoints on both data and having an overall low amount of detections. This measure is defined in Eq. (5.4). In this equation, r represents the repeatability of the detections and d the ratio of detections in each frame. The details and properties of this measure are explained in Appendix A.2.3.

$$S(\lambda) = (1 - \lambda)r + \lambda(1 - d) \tag{5.4}$$

In this equation, the parameter λ can be tuned to define the relevance of the fraction of correct detections among the overall number of detections. It has been observed that with $\lambda = 0.5$ a good balance between the number of detections and overall performance is obtained.

5.3 Hardware and Software

The main codebase of this thesis has been written in Python [101]. The different CNNs presented in this thesis are coded using PyTorch [65]. Open3D [117] has been used to work with point clouds and OpenCV [11] to work with 2D images. The earliest parts of the codebase are written in C++ [94] using PCL [77] and OpenCV [11].

Regarding the execution platform, the training, parameter tuning and testing are mainly performed in the Image Processing Group server cluster. The profiling tests are performed in an Nvidia Jetson TX2 development kit [61].

More details about the hardware and software used can be found in Appendix A.3.

6

Correspondence Detection Evaluation

This chapter is devoted to the evaluation of the correspondence detector algorithm. This algorithm, presented in Chapter 3 as the main block of the proposed 2D-3D registration algorithm, aims to obtain a set of correspondences between two multimodal data sources. The necessary pipeline to get a correspondence, as shown in Fig. 6.1, can be summarised in three fundamental steps: detect a keypoint, compute a patch around the keypoint and obtain a feature vector from the patch using a CNN. In the final stage, the vectors from each stream are compared using euclidean distance. It should be noted that albeit performing a similar task, each stream has vastly different input and intermediate data, and therefore the internal blocks differ between streams.

This chapter analyses the characteristics of the three main blocks of the correspondence detector: the keypoint detector, the patch generator and the network structures. The effect of the different parameters that can be used on each step is explored.

In the final part of this chapter, a benchmark between the proposed correspondence

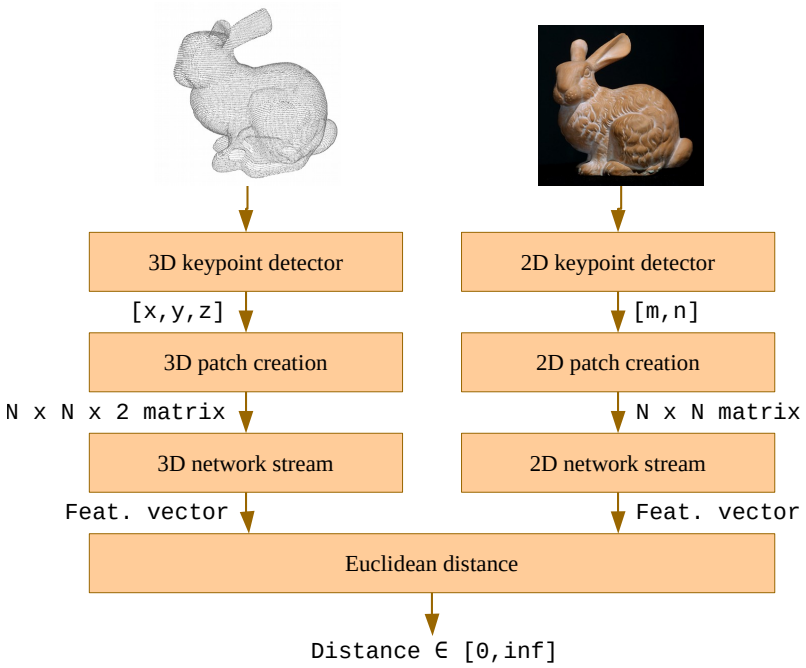


Figure 6.1: Algorithm followed to find the correspondence score from a pair candidate.

detector and the state of the art correspondence detectors is evaluated.

6.1 Keypoint Detector

Section 3.1 presents two main alternatives regarding the keypoint detector process: either use a state of the art keypoint detector –ISS in the 3D domain and Harris in the 2D domain– or select random keypoints on the data to be registered.

Using a random keypoint selector has one main advantage: the execution time. Table 6.2 shows the execution time to compute the keypoints for a single frame using random detections and state of the art keypoint detectors in both 2D and 3D domain. The exact parameters of this experiment can be found in Appendix B.1.1.

As can be seen, using random detections present a clear advantage in terms of execution time. This advantage is especially relevant in the 3D domain, where the evaluated state of the art keypoint detector, ISS, is extremely slow compared to a random selection.

The extremely high execution time of the ISS keypoint detector –and comparatively,

Method	Keypoint detector	Execution time (s)	Relative execution time
Random	2D	1.54E−04	1 ×
Harris	2D	1.54E−02	100 ×
Random	3D	5.24E−04	1 ×
ISS	3D	1.67E+02	319196 ×

Table 6.1: Execution time of different keypoint detector methods to compute the keypoints for a single frame. The relative time between methods (ratio) is also shown.

the low execution time of the random keypoint detector– is the main reason to use a random keypoint detector throughout the different results presented on this thesis. However, it is also interesting to evaluate the performance loss in terms of repeatability and specificity when using a random keypoint detector versus when using a state of the art detector.

The repeatability defines the capacity to detect the same locations in two different data sets. 1000 frame pairs with at least 30% overlapping are selected from the TUM dataset to perform this evaluation. For each frame in the pair, the three proposed keypoint detectors are computed. The exact parameters of this experiment can be found in Appendix B.1.2.

The fast keypoints are computed with different parameters to mitigate the influence of the number of keypoints in the repeatability. The ISS keypoint detector is only computed on a single set of parameters due to its high execution time. The obtained results can be seen in Fig. 6.2.

The specificity defines the capacity to extract similar features from the detected keypoints. Therefore, this measure is also influenced by the keypoint extractor used. The specificity of the proposed keypoint detectors is tested in a single-domain correspondence matching using three different feature extractors. These extractors are built using three siamese network architectures with three different patch inputs: 3D intensity patches, 3D depth patches and 2D image patches. Additionally, a two-stream architecture trained with a combination of 2D image patches and 3D intensity patches is also shown. These networks are trained using a balanced set of matches and non-matches obtained using the different keypoint detectors. The obtained

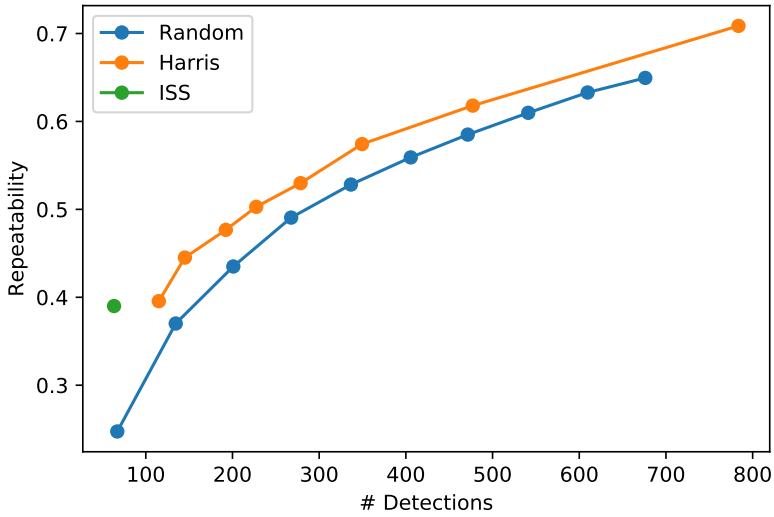


Figure 6.2: Graphic showing the repeatability of several keypoint detectors depending on the number of detections. The repetability of the ISS detector is only computed for a single number of detections due to its high execution time.

Keypoint detector	2D	3D(intensity)	3D(depth)	2D/ 3D(intensity)
Random	**0.694**	**0.473**	**0.701**	0.775
Harris	0.827	0.829	0.744	**0.764**
ISS	0.843	0.668	0.805	0.870

Table 6.2: Specificity (in FPR95) when using several keypoint detectors to extract three different features (lower is better).

results are presented in Table 6.2. The exact parameters of this experiment can be found in Appendix B.1.3.

As can be seen, using a random detector implies having a lower repeatability than both Harris and ISS detectors. The lower repeatability of the random detector, however, can be compensated by detecting a large number of detections, as performed in [111]. Moreover, it also has to be taken into account that in a 2D-3D correspondence matching scenario, both ISS and Harris detectors would be combined. A small test regarding this combination can be seen in Appendix B.1.2, where the combination Harris-ISS with the default parameters yielded only a 0.22 repeatability score.

Regarding the specificity of the different methods, it can be seen that, surprisingly, the method with the best specificity is the random keypoint selector. This performance is superior on the single-modal tests –2D, 3D(intensity), 3D(depth)–, while

on the multimodal analysis –2D / 3D(intensity)– the results when using the Harris keypoint detector are similar. This performance is likely to be caused by not constraining the network input to a specific keypoint type, and allowing the network to extract features from various keypoint types.

It should be noted that in work produced during this thesis presented in [71], where the ISS keypoint detector was used, the specificity yielded in both tests was higher. Although the same data sources were used in both sets, in [71] the sequences were limited to a length of 500 frames, thus not having substantial differences in viewpoints in the scene.

Therefore, the lower execution time of the random keypoint detection and its specificity makes this keypoint detector the adequate one to be applied in this pipeline.

6.2 Patch Generation

The next step in the 2D-3D correspondence pipeline is the extraction of a neighbourhood of a 2D-3D patch into a dense lattice. This procedure is performed with two different algorithms depending on whether the input data belongs to a 2D image or a 3D point cloud.

In this section, the different parameters of both algorithms are examined. First of all, the parameters of the 3D patch generator are explored. This generator has two main parameters, the neighbourhood radius and the projection lattice size. The different options for these parameters are evaluated using a siamese network in three different scenarios: using only the intensity channel, using only the depth channels and combining the intensity and depth channels.

The next evaluation is done in the 2D patch generator. Different cropping sizes and projection sizes are explored for this generator using a siamese network.

Finally, the performance of both generators in a multimodal scenario is evaluated using a two-stream architecture.

6.2.1 3D Patch Parameters

This section explores the different parameters regarding the 3D patch generation. The different parameters are trained using a siamese network to perform this analysis.

The patches obtained using the algorithm defined in Section 3.2.1 contain an intensity channel and a depth channel. The patches are built using the cloud neighbourhood of the keypoints using a radius r and projecting them to a $N \times N$ lattice.

The effect of the neighbourhood radius and the output size parameters is evaluated in this section in three different settings: using only the intensity channel, using only the depth channel and using both channels.

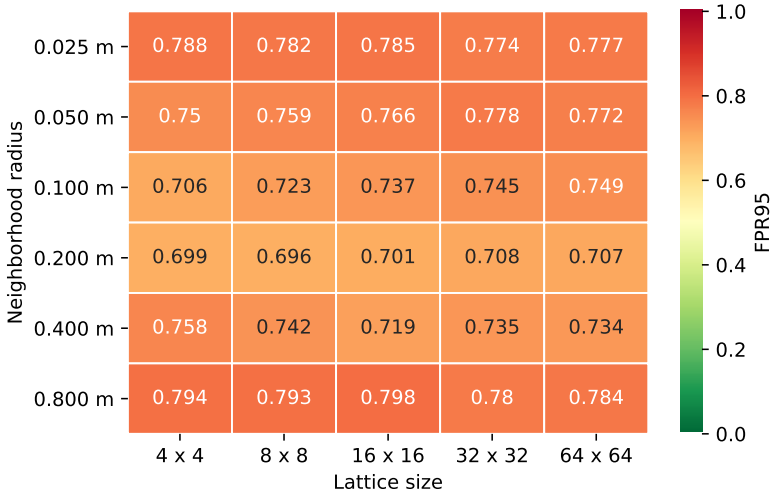
The results obtained in these tests can be seen in Fig. 6.3. The details of this experiment can be found in Appendix B.2.1.

On the first view, it can be seen that the best results are obtained when using only the intensity channel. The performance decreases significantly when only using the depth channel, and the combination of the intensity and depth channel also does not improve the results obtained when using only the intensity channel. This difference shows that the depth channel present in this database does not contain useful information that can be extracted using this network structure.

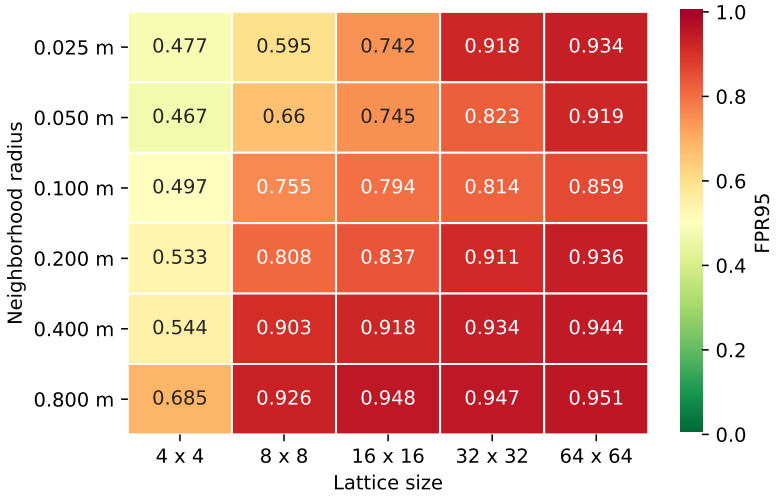
However, earlier experiments were performed using a small database using only 500 frames and the ISS keypoint detector. The results obtained using this database were presented in [71]. This results, measured using the area under the ROC curve, obtained an AUC on the validation split of 0.978 using the intensity channel and 0.948 using the depth channel. The performances decreased to 0.96 and 0.84 on the test split. Moreover, in these tests, the combination of the intensity and depth channels provide a slight performance increase.

There are three critical differences between the results presented in Fig. 6.3 and the results presented in [71]: the network structure, the database size and the keypoint

(a) Using only the intensity channel



(b) Using only the depth channel



(c) Compining intensity and depth channels

Figure 6.3: FPR95 values for different radius and ouptut sizes using the different input channels.

detector.

The network structure used in [71] is a two-channel architecture, unlike the siamese architecture presented in this thesis. In [110] both structures were used to perform the registration, and a slight decrease in performance was observed when using the siamese structure. However, these factors should not be a leading factor in the differences observed between the two experiments.

The database used in both structures is vastly different: while the database used in [71] was built using only 500 frames, the database used in the experiments presented in this thesis is built using several sequences of thousands of frames. Therefore, the main difference in the keypoint matching datasets, apart from its size, is the disparity between frames. This disparity between viewpoints can lead to inconsistencies between the data captured from two sensors that can affect the results. These inconsistencies, however, should have a similar influence on both the intensity and depth channels. Therefore, although the difference in global performance between both tests can be attributed to this fact, this is not likely to be the cause of the low performance of the depth channel.

The keypoint detector used in [71] is the ISS keypoint detector, while the database used in this results section is built using the random keypoint detector. The differences in specificity between both keypoint detectors have already been analysed in Section 6.1. In the obtained results, shown in Table 6.2, it can be seen that the difference on performance between the intensity and depth performance is bigger on the random keypoint detector than the ISS keypoint detector. This difference can be caused by the inherent lower resolution of the depth sensors, which provide large flat areas with sharp contours. Therefore, this can cause the depth patch generator to be more affected by selecting keypoints on large flat areas than the intensity channel.

Therefore, it can be concluded that the depth patch obtained during the patch generation contains less valuable information to perform the registration and that in general scenarios, the intensity channel outperforms the depth channel.

Two parameters can be analysed on the results obtained using only the intensity channel: the selected neighbourhood radius on the 3D point cloud and the projection lattice size.

Regarding the neighbourhood radius, it can be seen that neighbourhoods between $0.05m$ and $0.2m$ are preferred. This parameter is constrained between a trade-off for generalisation – analysing only a local neighbourhood– and having meaningful information about the neighbourhood. It is relevant to know that the average distance between two points in the neighbourhood in this dataset is $0.006m$. Therefore, not enough points are contained in neighbourhoods with a radius lower than $0.05m$. The generalisation effect is lost in neighbourhoods with a radius higher than $0.2m$.

The lattice size affects how the points in the neighbourhood are represented in the 2D image. The neighbourhoods selected in this analysis are being constrained between a minimal meaningful size –4×4– and the input size of the Zagoruyko [110] network used in these tests. As it can be seen the patches are preferred to be as smaller as possible –4×4– independently of the selected neighbourhood.

The high performance of small lattice sizes on small neighbourhoods can be attributed to the low number of points in the neighbourhood. As explained in Section 3.2.1, a selection of a large lattice size on small neighbourhoods can produce non-relevant artefacts on the patch generation.

However, this effect does not apply to large neighbourhoods. In this case, another variable should be taken into account: the CNN used to train the descriptors. Having high performances on small patches can be attributed to the need of the network to have smoothed information about the patch neighbourhood to avoid falling in local minima. However, it should also be noted that the differences in the performance obtained between small patches and medium patches is small, often restricted to the third decimal, and thus may not have a meaningful effect in a real-world application.

In addition to the correspondence matching performance, the execution time with the different parameters can be evaluated. The details of this experiment can be found in Appendix B.2.2.

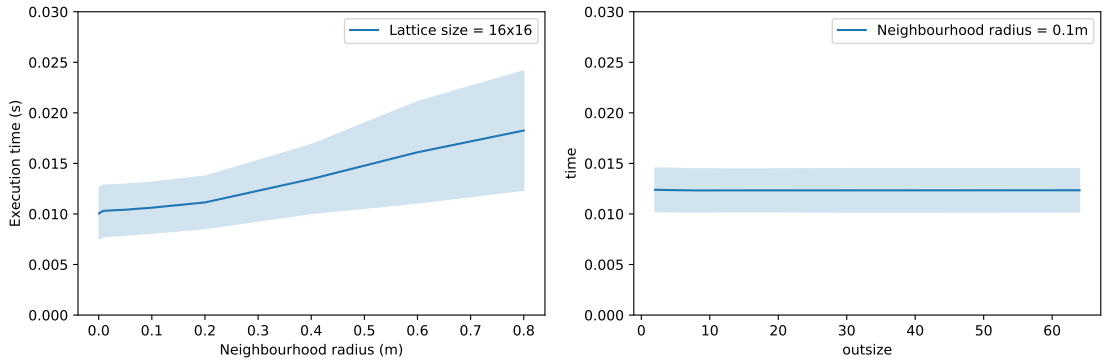


Figure 6.4: Execution time for for different neighbourhood radius (left) and output sizes(right). The mean value is shown with a solid line and the greyed out area shows the variance of the measure.

Fig. 6.4 shows the execution time for different radiuses and different outsizes. It can be seen that the radius has a strong influence on the computation time, while the lattice size does not affect the computation time. It can also be seen that the variance on the computation time increases with the neighbourhood size. These results suggest that the computation time depends on the number of points that are present in a specific neighbourhood.

Whereas on the keypoint detector the execution time was a relevant factor to discard the ISS keypoint detector, in this case, there are no substantial differences in terms of the execution time between the different parameter values. Therefore, the execution time is not considered a relevant parameter to select the optimal combination unless there is a time-critical application.

6.2.2 2D Patch Parameters

The patch extraction procedure in the 2D domain is a more straightforward procedure than the 3D patch extraction. The 2D image already provides a regular lattice to define the neighbouring points. Therefore, the patch for a certain keypoint is simply extracted by cropping a $N' \times N'$ neighbourhood in the image centred in the keypoint. If the neighbourhood is not fully contained in the image, the resulting 2D patch is zero-filled on its boundaries.

An additional step is added to mimic the 3D procedure. The obtained $N' \times N'$ patch

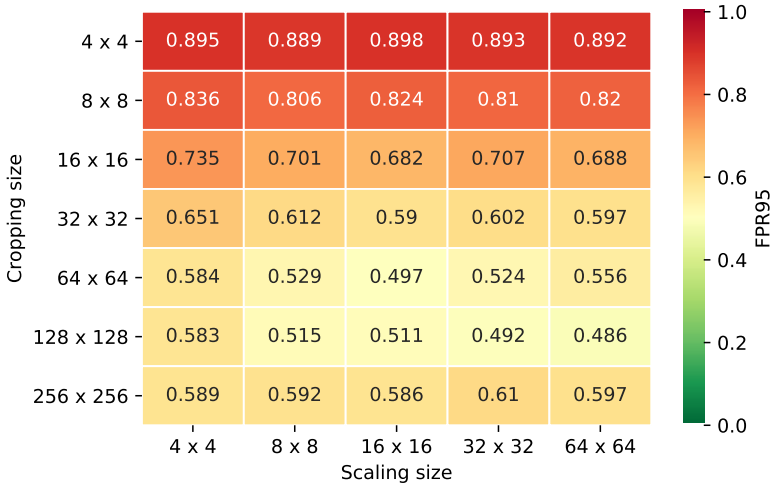


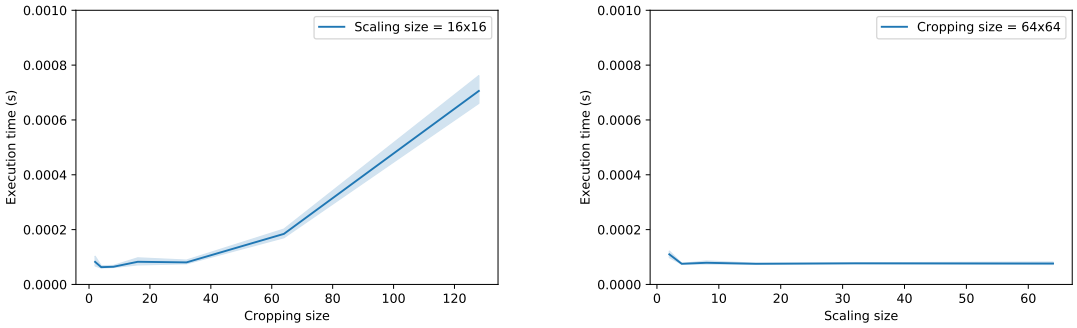
Figure 6.5: FPR95 on different cropping patch sizes / ouptut sizes on the 2D patch generation.

is scaled to a $N \times N$ size, akin to the lattice size in the 3D patch extraction. If $N > N'$ this scaling is an upsampling, while if $N < N'$ this scaling is a downsampling. Therefore, the 2D patch extraction has two relevant parameters: the $N' \times N'$ cropping size and the $N \times N$ scaling size.

A 2D-2D correspondence matching is performed using a siamese architecture to evaluate the influence of these parameters in the registration pipeline. Figure 6.5 shows the results obtained training patches extracted from different $N' \times N'$ neighbourhood sizes (between 4×4 and 256×256) scaled to a $N \times N$ patch (between 4×4 to 64×64, the input patch size of the network used in this test). The details of this experiment can be found in Appendix B.2.3.

As it can be seen, the best results are when cropping a neighbourhood of either 64×64 or 128×128. Unlike in the 3D domain, this measure does not represent a real-world size. The equivalent real-world size of this patch is affected by the distance from the scene to the camera.

Regarding the optimal scaling size, it can be seen that values from 8×8 to 64×64 offer good performances. In particular, the best performance is obtained when a 128×128 neighbourhood is scaled to a 64×64 size. In this case, the cropping size has a more significant influence than the scaling size. This behaviour shows that the network is not very sensitive to the generated patch size, unlike the 3D



(a) Cropping size **(b)** Output size

Figure 6.6: Evaluation of the influence on the computation time of the cropping size and output size parameters of the 2D patch generation

scenario. Furthermore, it should be noted that the influence of the scaling factor is also mitigated since the scaling does not introduce any artefacts on the generated patch.

Aside from the performance, the different 2D patch generation parameters are also evaluated in terms of their execution time. The details of the performed experiment can be found in Appendix B.2.4.

In Fig. 6.6 the execution time for different cropping sizes/output sizes are stated. The solid line represents the mean value on 100 different executions, whereas the faded area represents the variation of the measure. As can be seen, there is a correlation between the cropping size and the execution time, whereas the output size does not influence the computation time.

6.3 Networks

The network structure used in the previous tests is the Zagoruyko [110] structure, used in its original article to establish matches between 2D patches. The previous tests have shown that this structure can also be successfully used to establish matches between 3D patches.

This structure, however, has a shallow layer structure that may not offer the best possible performance on the correspondence matching scenario. In this section,

Network structure	FPR95	Cropping size	Scaling size
Zagoruyko [110]	0.486	128×128	64×64
VGG [88]	0.555	64×64	64×64
ResNet [34]	**0.392**	128×128	64×64

Table 6.3: Best performing parameters for each network structure on the 2D-2D registration pipeline.

Network structure	FPR95	Neighbourhood radius	Lattice size
Zagoruyko [110]	0.398	0.05m	4×4
VGG [88]	0.406	0.4m	32×32
ResNet [34]	**0.341**	0.2m	8×8

Table 6.4: Best performing parameters for each network structure on the 3D-3D registration pipeline.

the performance of two general-purpose state of the art network structures on this correspondence matching scenario is also tested.

The selected structures, as stated in Section 3.3, are VGG [88] and ResNet [34]. The feature extractor layer of each one of these networks is used on each branch of the correspondence matching structure. These network structures, unlike the Zagoruyko [110] structure, are not trained from scratch. A pre-trained version of both networks is fine-tuned to the correspondence matching structure.

Another difference between the Zagoruyko [110] structure and the VGG [88] and ResNet [34] structure is the input patch size. Whilst the Zagoruyko [110] network has a 64×64 input patch size, both the VGG [88] and ResNet [34] structures have a 224×224 input patch size. This size difference has a practical effect on the execution time, as discussed at the end of this section.

Due to this high execution time, the Hyperband [47] optimisation algorithm is used to find the best performing parameters for each network structure and each domain. This algorithm measures the performance on different time steps and distributes the available resources among the best performing tests. This algorithm is selected due to its high generalisation capabilities on many scenarios [47]. In addition to using the Hyperband algorithm, a two-step testing process is used with a small training split. The additional details on these tests can be found in Appendix B.3.

Table 6.3 shows the best performing combinations on the three network structures on the 2D-2D registration, and Table 6.4 shows the best performing combinations on the three network structures on the 3D-3D registration. As can be seen, the best performance is obtained when using the ResNet [34] structure. Its performance is slightly superior to the performance of the Zagoruyko [110] structure. Regarding the VGG [88] structure, it can be observed that the performance obtained when using this structure is slightly inferior in both domains.

The optimal parameters for each structure and each domain can also be compared. It can be seen that in the 2D domain, both ResNet [34] and Zagoruyko [110] have the same optimal parameters. The optimal patch parameters when using the VGG [88] structure are slightly different, but this difference can be attributed to slight variations on the performance.

However, in the 3D domain, the optimal parameters differ between the network structures used, both in the neighbourhood radius and the output size. This difference can be caused by the ability of deeper networks to learn more complex patterns on larger neighbourhoods. This behaviour is not observed in the 2D domain, where the shallow network can already learn the intricate patterns present on the larger 2D patches.

Another evaluation metric that must be taken into account is the execution time of each structure, presented in Fig. 6.7. The execution time is measured on inference when forwarding a single batch through the network with different batch sizes. In this figure, the solid line represents the average value, whereas the translucid area represents the variance across different executions. More details about this experiment can be found in Appendix B.3.1.

These results show that there are significant differences between the network structures, with the Zagoruyko [110] network being significantly faster and both VGG [88] and ResNet [34] being significantly slower. These results are coherent with the complexity of the networks. It can also be observed that the computation time is directly proportional to the batch size.

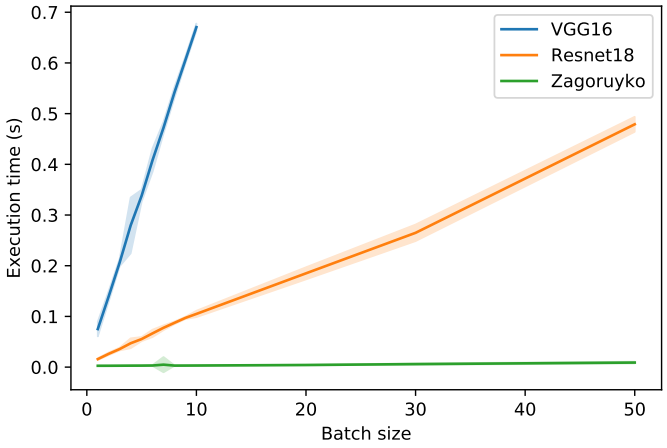


Figure 6.7: Execution time (in seconds) of the evaluated network structures for different batch size. The mean value is marked in solid color, whereas the variance is shown as a translucid area.

It can also be observed that the VGG [88] structure is the structure with the slowest computation time. This network structure also does not obtain a better performance than the ResNet [34] structure. Therefore, this network structure is discarded.

The two remaining structures present a trade-off between performance and computation time. The Zagoruyko [110] structure presents a slightly worse performance but a faster computation time than the ResNet [34] structure. Therefore, in the remaining tests, both the Zagoruyko [110] and the ResNet [34] structures are considered.

6.4 Multimodal analysis

The previous sections have focused on the single-modal correspondence matching in the 2D domain and the 3D domain. The tests have proven that the proposed structures can learn discriminative parameters in a single domain scenario. However, having a discriminative method in the single-modal scenario does no guarantee having a multi-domain descriptor. Therefore, the previous tests are repeated in a multimodal scenario.

The tests presented in this section aim to find the optimal set of parameters to perform a multimodal 2D-3D correspondence matching using the Zagoruyko [110] and the ResNet [34] structures. Both structures are considered due to the trade-off presented between algorithm performance and computational time.

	Network	2D Patches		3D Patches	
FPR95		Crop	Scaling	Radius	P. size
0.646	Zagoruyko [110]	128×128	64×64	0.05m	4×4
0.593	ResNet [34]	128×128	64×64	0.2m	16×16

Table 6.5: Top combinations on the multimodal correspondence matching scenario.

The Hyperband algorithm is used to perform an efficient search on four different axes: the image cropping size, the image output size, the cloud neighbourhood radius and the cloud output size. Only the best performing combinations on the 2D and 3D single modal combinations are considered to reduce the search space for the different parameters. The best performing method for each network can be seen in Table 6.5. Additional details of this experiment can be found in Appendix B.4.

When comparing the performances to the single-modal experiments, it can be seen that the FPR95 obtained in both combinations is significantly larger than the performances obtained in the single-modal tests. This decrease in performance can be caused by several factors: the training dataset size, the cloud projection plane, the patch cropping size and the network structure.

The problem that these network structures are trying to solve is a significantly more difficult problem than the single-modal correspondence matching. However, in both cases, the training dataset size is 200.000 pairs. This dataset may not be big enough to train this problem.

The difficulty of this problem is also related to the different data represented in the cloud and the images. Knowing the real-world depth information when registering 3D-3D patches has two main advantages.

First of all, the orthogonal projection plane in the 3D point cloud, computed using the PCA decomposition, should be similar in the two 3D clouds. However, this PCA projection is independent on the viewing plane of the 2D camera and therefore may affect the descriptor accuracy.

The 3D point cloud also has another main advantage: the real-world scale. When selecting a neighbourhood in the 3D point cloud, the radius is selected in real-world

Network	Original	Synthetic cropping	Synthetic projection plane
Zagoruyko [110]	0.646	0.716 (+0.070)	0.629 (−0.017)
ResNet [34]	0.593	0.553 (−0.010)	0.439 (−0.154)

Table 6.6: Errors obtained when applying synthetic modifications on the patch generation.

units, common between frames. Therefore, a 0.2m radius contains the same physical locations in both data. However, in 2D images, the real-world scale is lost in the projection, and an arbitrary cropping size is selected.

A synthetic test is performed to measure the effect of these factors on the correspondence matching process. Two artificial computations are performed. The projection plane of the 3D cloud neighbourhood is modified to be the projection plane of the 2D image to be registered, and the cropping size on the 2D image is defined as the size of the projected 3D neighbourhood into the 2D image.

The performance on the 2D-3D correspondence matching is compared before and after applying those synthetic modifications. Table 6.6 shows the FPR95 measurements when applying the proposed synthetic modifications with the two network structures. Two main conclusions can be extracted from these results. On the one hand, the performance of the system increases when using the camera plane as the projection plane, especially when using the ResNet [34] structure.

On the other hand, the results are not conclusive when artificially selecting the patch size in the 2D image as the projected neighbourhood size. The error rate slightly decreases when using the ResNet [34] structure. However, the error rate slightly increases when using the Zagoruyko [110] structure. This slight increase may be caused by the small neighbourhood used with this network (0.05m). Albeit being the optimal size for this combination, the resulting patch size may not be representative enough.

6.5 Benchmark

The influence of the different parameters on the performance of the algorithm has been evaluated in the previous section. This evaluation has given a set of parameters to perform single domain 2D-2D, single domain 3D-3D and multi-domain 2D-3D correspondence matching with a shared architecture.

The different experiments have shown that a CNN architecture can be used to learn matching and non-matching patches. Specifically, the ResNet-18 [34] structure offers the best error performance, while the Zagoruyko [110] structure offers the best computational time performance.

Regarding the patch representation, it has been found that both single modal and multimodal combinations have the peak performance with the same set of parameters. In 2D images, a 128×128 patch is projected into a 32×32 lattice, while in 3D point clouds a $0.2m$ neighbourhood is projected into a 16×16 lattice.

This section compares the obtained results with state of the art methods. The structures used are referred to as 'Fast' or 'Deep' to simplify the nomenclature. The 'Fast' network uses the Zagoruyko [110] structure, while the 'Deep' network uses the ResNet-18 [34] structure.

The results presented in the previous section, however, were given on the validation split of the dataset. To be able to compare this algorithm with state of the art, the results on the test split are computed. A bigger training dataset is used, containing 500.000 matching / non-matching pairs, to compute these results. The details of this experiment can be found in Appendix B.5.

Table 6.7 shows the comparison between validation and test results in the three explored modalities: single domain 2D-2D, single domain 3D-3D and multi-domain 2D-3D correspondence matching. As can be seen, the performance of all the algorithm has a sharp decrease in the test split, reducing the disparities between the 'Fast' and the 'Deep' architectures.

Domain	Architecture	Validation error	Test error
Single modal 2D-2D	Fast	0.486	0.683
	Deep	**0.392**	**0.547**
Single modal 3D-3D	Fast	0.398	**0.502**
	Deep	**0.341**	0.564
Multi modal 2D-3D	Fast	0.646	0.739
	Deep	**0.593**	**0.692**

Table 6.7: Comparison between the validation and test errors on the best performing methods for each modality.

Initially, this difference can be attributed to overfitting in the training process. However, analysing the learning rate curves on different epochs suggest otherwise. Figure 6.8 shows the evolution of the validation and test error during the training epochs using both network structures. As can be seen, both validation and test present similar curves and show the overfitting spot in which the error starts to increase. However, the difference between the errors in validation and test is present from the first epoch.

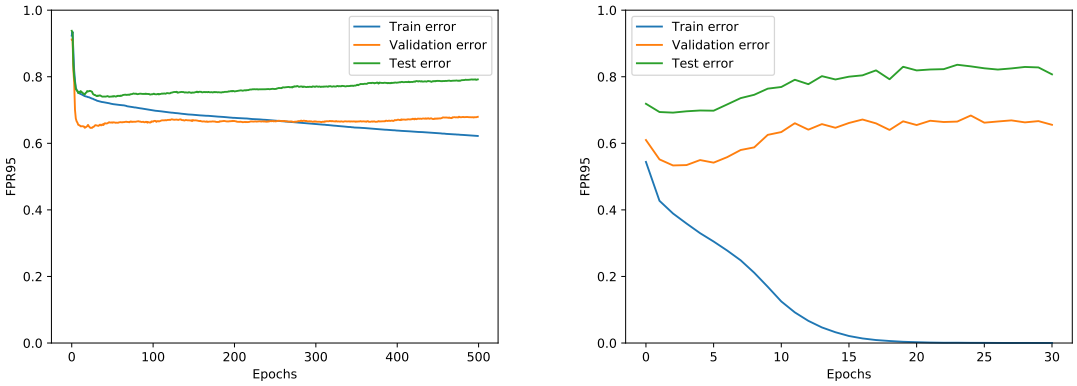
This difference can also be attributed to factors outside the training process. As explained in Appendix A.1, the dataset used to build the training database used in these tests had data from five different sources initially. However, two of these sources are discarded since no RGB data is available.

The subsequent division in train/validation/test performed in this thesis has given a train and validation split that contains data from three different sources but a test split that contains data from only one source. This imbalance may be a leading cause of the difference between the train and test splits.

This evaluation on the test split, however, allows providing a comparison with the state of the art algorithms on single modal 2D and 3D correspondence matching.

For single modal 2D correspondence matching, the algorithms selected to perform the evaluation is the algorithm presented in [110]. These results are obtained using a different database than the 3DMatch database used in this thesis.

For single modal 3D correspondence matching, the algorithms selected to perform



(a) 'Fast' pipeline **(b)** 'Deep' pipeline

Figure 6.8: Evolution of the validation and test errors on the 'Fast' and 'Deep' pipelines on the multimodal 2D-3D registration

.

	2D-2D	3D-3D	Multimodal 2D-3D
DeepCompare [110]	**0.136**[1]	-	-
3DMatch [111]	-	**0.353**[2]	-
FPFH [76]	-	0.613[2]	-
Spin Images [39]	-	0.837[2]	-
Fast	0.683	0.502	0.739
Deep	0.547	0.564	**0.692**

[1] Extracted from [110]. The database used in this test is not the same database used in the remaining tests and it is extracted from the Photo Tourism dataset [12].
[2] Extracted from [111].

Table 6.8: Comparison between different methods on 2D, 3D and multimodal 2D-3D correspondence matching. The FPR95 values for the different methods are shown.

the evaluations are the algorithms presented in the benchmark in [111]. These algorithms include the Fast Point Feature Histogram (FPFH) proposed in [76] and the Spin Images proposed in [39]. The results presented are the values stated in [111].

Table 6.8 presents the benchmark results in the different modalities. Regarding the 2D domain, it can be seen that the results obtained are significantly lower than the results obtained in [110].

It should be noted, however, that in the 2D domain both the 'Fast' method and the method in [110] use an identical structure trained with different databases. However,

the performance obtained in the 'Fast' test is significantly lower.

The only significant difference between these tests is the database used to evaluate the algorithm. In [110], a database with 64×64 patches extracted from the Photo Tourism Dataset is used. This dataset contains only three sequences with a total of 2340 images. Moreover, the patches present in this database have been previously normalised both in rotation and scale [105].

The database used in this thesis, however, has diverse data sources, 3 times more sequences and 100 times more frames. Moreover, there is no normalisation step on the patch generation. Since the structure of [110] and the structure in 'Fast' are virtually the same, the difference in performance can be attributed to those factors.

It can be seen that the proposed methods offer lower performance in the 3D domain than the method proposed in [111]. However, both proposed methods outperform all non-CNN methods. It should also be taken into account the disparity observed between the train and test scenarios (Table 6.7) when comparing the different methods. Further research on techniques to decrease this disparity can also lead to a performance increase to levels similar to the best-performing state of the art methods.

It is also worth noticing that the proposed algorithms obtain similar performances on both 2D and 3D domains. This similar performance shows that the proposed pipeline to represent a 3D neighbourhood into an intensity patch is a valid strategy to obtain correspondences between point clouds.

The proposed methods, moreover, offer one main advantage concerning the state of the art methods: the ability to obtain correspondences between domains. As discussed before, the performance in this multimodal scenario is significantly lower than the performance on single-domain scenarios.

However, it should be taken into account that the evaluation dataset used in this thesis has proven to have high data variability. The results obtained when using smaller datasets, as seen in [71], are significantly better. This fact, coupled with

the fact that the error rate obtained is inferior to some of state of the art hand-crafted descriptors, show the potential of the proposed algorithm in more controlled datasets.

7

Edge Detection Evaluation

The second part of this thesis is focused on the registration of a 3D point cloud into a 2D image using an iterative refinement algorithm. The proposed pipeline leverages a modification of the ICP registration algorithm to align edges from 3D point clouds and 2D images.

This chapter focuses on the analysis of the proposed 2D-3D edge detection algorithm. The proposed algorithm describes a method to extract intensity contours from 2D images and geometrical and intensity contours from 3D point clouds. The detections obtained from a 2D image and a 3D point cloud are compared to evaluate this algorithm, as shown in Fig. 7.1. The evaluation is done using the repeatability measure explained in Section 5.2.

A multiscale approach is used to detect the different contour types on different modalities. On a broader view, the proposed algorithm for each contour type and modality described in Chapter 4 can be summarised as follows. For each query point, a measure σ is computed for a set of neighbourhoods with size $K_{\min}$ to $K_{\max}$. A threshold on this measure $\sigma_{\max}$ is established. The score w for the query

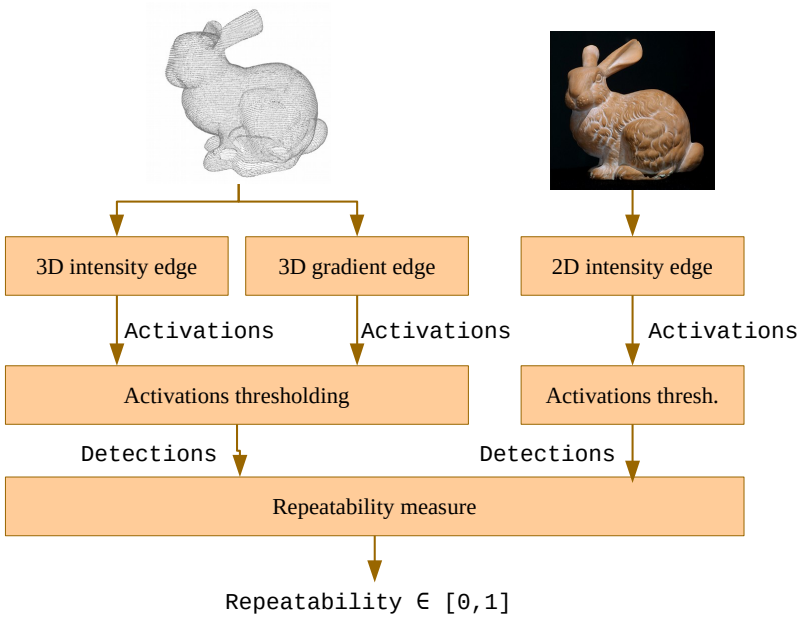


Figure 7.1: Evaluation pipeline for the multiscale edge detection method proposed in this thesis.

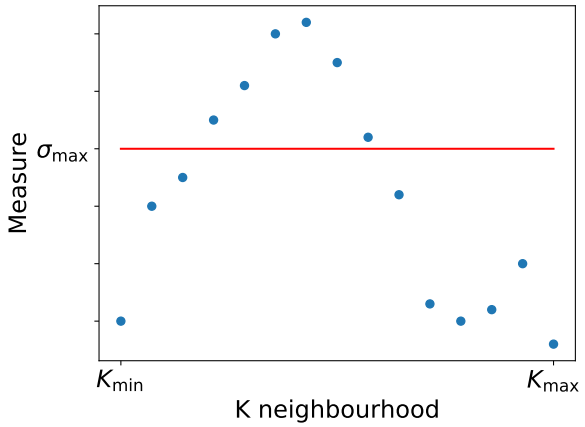


Figure 7.2: Visual representation of the influence of the parameters $K_{\min}$, $K_{\max}$ and $\sigma_{\max}$ on the point score w.

point is defined as the percentage of σ that have exceeded the threshold $\sigma_{\max}$. A final threshold th is applied to w to establish the edge points. In Fig. 7.2 a visual representation of the different parameters can be found.

Although there are variations on the meaning of the different parameters between the different modalities and edge types, some common analysis can be performed on the different parameters. $K_{\min}$ and $K_{\max}$ define the analysis area of the algorithm. The optimal values of these parameters show the neighbourhood sizes in which the detections are relevant. In general, low values are discarded due to noise in the data, and high values are discarded due to not having enough generalisation capabilities.

$\sigma_{\max}$ and th define the thresholds applied to the σ parameter. In general, there are two different options to provide repeatable detections with an overall low amount of detections: have a low $\sigma_{\max}$ and a high th and have a high $\sigma_{\max}$ and a low th. In the first case, the algorithm is sensitive to smooth curvatures but only selects those that are consistent in most neighbourhoods. In the second case, only strong variations are detected, but the point is selected even if they are only present on a few neighbourhoods.

However, the two remaining combinations can also occur. A low $\sigma_{\max}$ and th denotes that the system has a very low sensitivity and all detected contours should be taken into account. Likewise, a high $\sigma_{\max}$ and th denotes that the system has too high sensitivity and that only the strongest contours must be taken into account.

Sections 7.1 to 7.3 examine the influence of these parameters on the performance of the proposed edge detections: 3D geometrical detection, 3D intensity detection and 2D intensity detection.

This evaluation is performed using frame pairs extracted from the 3DMatch dataset, as explained in Appendix A.1.3, and evaluated used the quality measure described in Section 5.2. This measure takes into account both the repeatability and the number of detections of each method. As discussed in Appendix A.2.3, a balanced ratio between the repeatability and the number of correspondences is desired. However, it should be noted that different λ values –and thus, different parameter values– may be desired for different applications.

This analysis is carried with different values of the edge detection parameters $K_{\min}$, $K_{\max}$, $\sigma_{\max}$ and th. The parameters $\sigma_{\max}$ and th are intrinsically bounded by their characteristics, but the parameters $K_{\min}$ and $K_{\max}$ are unbounded. The range for these parameters is defined using the knowledge gathered during this thesis on their reasonable values.

The performance of the algorithm on different domains is also explored. Section 7.4 discusses the best joint combinations for all the parameters in the scope of 3D-3D, 2D-2D and 2D-3D edge detection. This evaluation is performed using the quality

measure explained in Section 5.2.

Finally, the computational performance of this algorithm is also explored. The effects on the heuristic speedup optimisation proposed in Section 4.1.1 on the 3D point cloud are explored in Section 7.5.

7.1 3D Geometrical Edge Detection

The 3D geometrical contour detection used in this thesis is the implementation of the multiscale geometrical edge detection developed by Pauly [66]. This algorithm computes a variance score σ^g based on the eigenvalues of the PCA decomposition of the points in the neighbourhood. This value measures the non-coplanarity of these points using the third eigenvalue.

The maximum variance $\sigma^g_{\max}$ defines the threshold on the variance score computed for each neighbourhood. Since the score σ^g_k for each neighbourhood k is defined based on the PCA eigenvalues as $\sigma^g_k = \lambda_3/(\lambda_1 + \lambda_2 + \lambda_3)$, where λ_3 is the smallest eigenvalue, this value is bound between 0 and 1/3.

Several random combinations of the different parameter values are evaluated. The quality measure described in Section 5.2 is computed using a small subset of frame pairs for each combination. More details of this experiment can be found in Appendix B.6.2

Fig. 7.3 shows a pair plot between the different parameters. The values are clustered in two groups depending on their quality score using the mean value as the threshold. This representation visualises the correlation between each parameter in a set of 2D plots. The diagonal elements contain the distribution of the different parameters, clustered based on the quality measure. This analysis allows for evaluating the overall preferred values for each parameter, as well as correlations between several parameters.

Observing the values $K^g_{\min}$ and $K^g_{\max}$ it can be seen that there is a wide range of

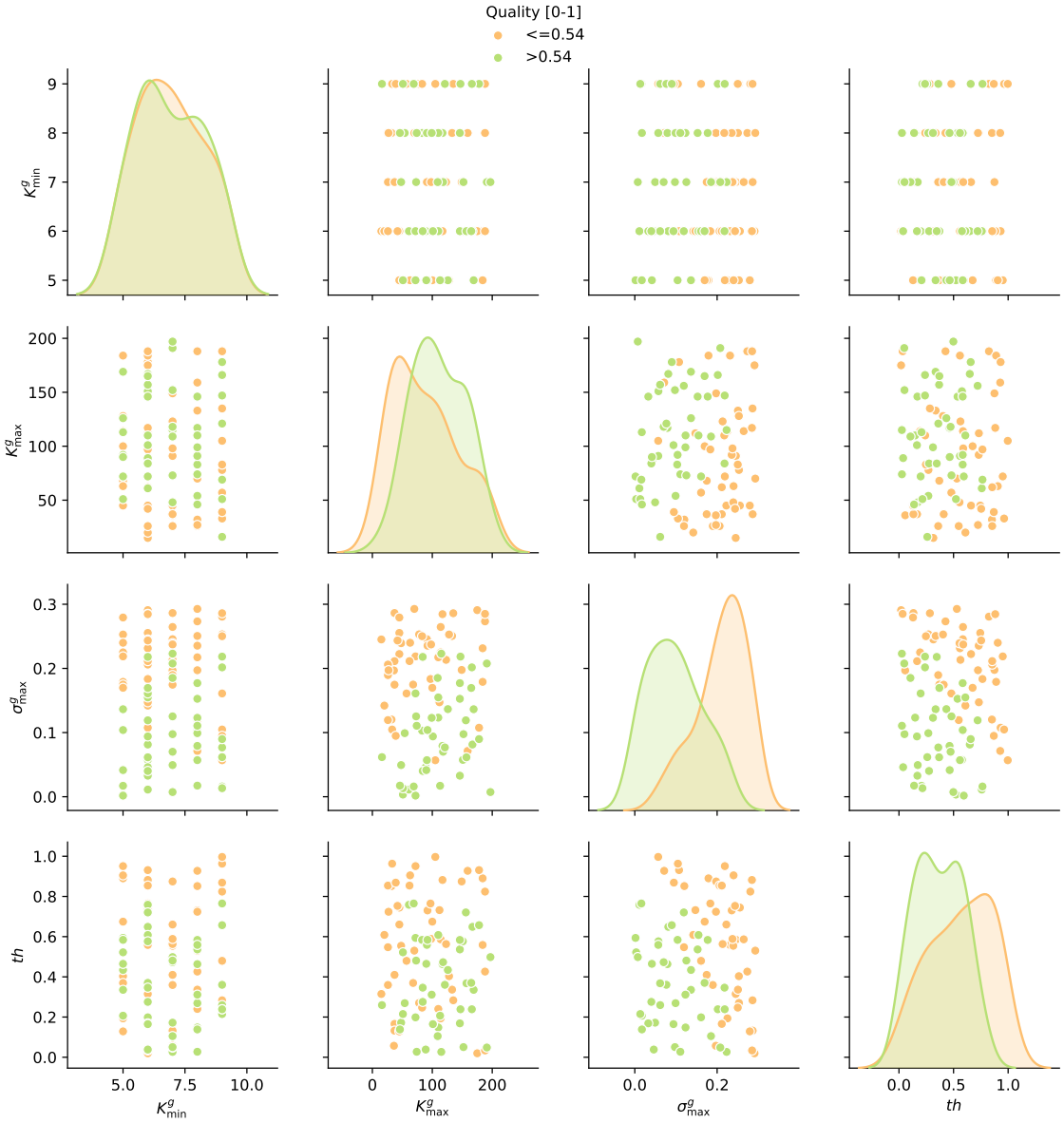


Figure 7.3: Pair plot showing the performance of several parameters on the 3D geometrical edge detection algorithm. The tests are clustered based on their quality score. The diagonal plots contain the distribution for each score.

neighbourhoods with meaningful data. The similar performance for the different $K^g_{\min}$ values shows that the small neighbourhoods, however, do not improve the performance.

Regarding the $\sigma^g_{\max}$ and th parameters, it can be seen that the best performances are obtained with small values for both parameters, showing the low sensitivity of the measure.

7.2 3D Intensity Edge Detection

The proposed 3D intensity edge detection method detects edge points by computing the weighted average of a local neighbourhood and detecting non-symmetries in a specific neighbourhood. The parameters analysed for this algorithm are $K^y_{\min}, K^y_{\max}$, $\sigma^y_{\max}$ and th^y. Similarly to the previous section, a random search between several values of these parameters is performed. The details of this experiment can be found in Appendix B.6.1.

Fig. 7.4 shows a pair plot between the different parameters. The values are clustered in two groups depending on their quality score using the mean value as the threshold. The diagonal elements contain the distribution of the different parameters. The first conclusion that can be directly extracted is that the standalone 3D intensity edge detection obtains, in general, higher quality than the 3D geometrical edge detection. Although the numeric of each sample is not shown in the graphic, it can be seen that the mean value used as the threshold is higher in the intensity edge detection.

It can also be seen that there is also not a clear optimal value for the $K^y_{\min}$ parameter, obtaining similar performances with all the explored values with a slight preference for the small tested values. Like in the 3D geometrical edge detection explored in the previous section, this behaviour shows that variations on small neighbourhoods are also not significantly relevant to the global result. It can also be seen that the larger values are preferred for the $K^y_{\max}$, showing that large areas are needed to detect relevant edges.

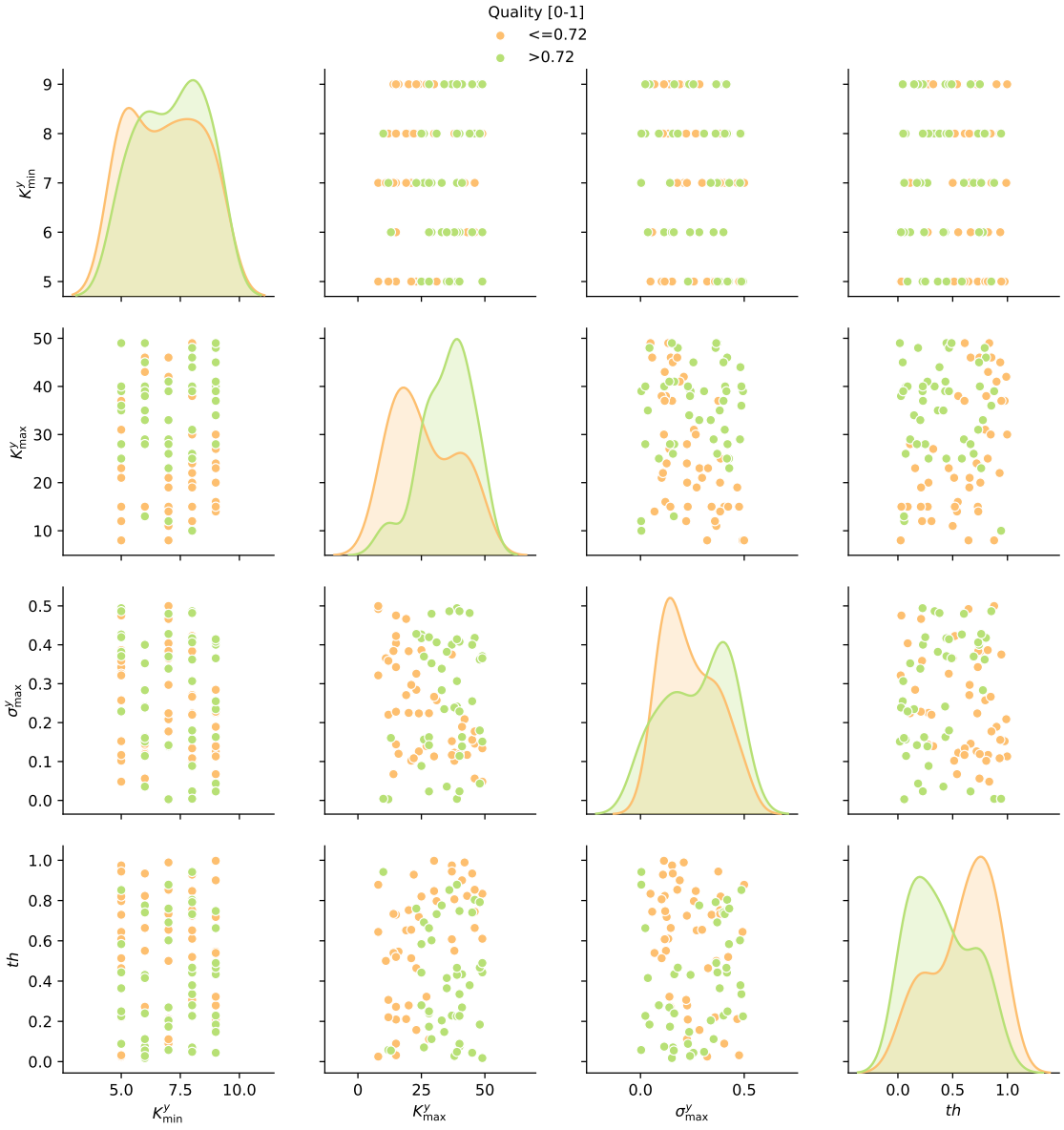


Figure 7.4: Pair plot showing the performance of several parameters on the 3D intensity edge detection algorithm. The tests are clustered based on their quality score. The diagonal plots contain the distributions for each score.

Regarding the parameters $\sigma^y_{\max}$ and th^y, it can be seen that to obtain high-quality detections slightly large values for the $\sigma^y_{\max}$ parameter and small values for the th^y parameter are preferred. This behaviour shows that only strong edges that are consistent in several neighbourhoods are relevant in edge detection.

7.3 2D Edge detection

The third edge detection procedure proposed in this thesis is the 2D edge detection. This procedure presents a new method to select the neighbourhood points, albeit using the same variation score computation on each neighbourhood.

This method leverages the inherent lattice structure of the image to select increasing K neighbourhoods by cropping a $2K + 1 \times 2K + 1$ patch surrounding each point being analysed. Since in this case the neighbourhood size increases in a quadratic fashion –a patch in a K neighbourhood contains $(2K + 1)^2$ pixels– the parameter analysis differs significantly.

A random search between the $K_{\min}$, $K_{\max}$, $\sigma_{\max}$ and th parameters is performed to provide this analysis. The details of this experiment can be found in Appendix B.7.

Fig. 7.5 shows the pair plot obtained when performed this analysis. As in the previous tests, this pair plot contains the correlation between pairs of parameters. The diagonal plots contain the distribution for each parameter. As can be seen, in general, the performance of the 2D intensity edge detection is similar to the performance of the 3D intensity edge detection, obtaining similar mean values.

Like in the previous tests, the explored values for the parameter $K_{\min}$ offer similar performances, asserting the non-relevance of the small neighbourhoods on the result.

Observing the parameter $K_{\max}$ it can be seen that, in general, the best performances are obtained with the large tested neighbourhoods. To analyse this parameter, however, it should be taken into account that with the proposed algorithms a $K = 10$ neighbourhood radius in the 2D image contains the same points than a $K = 400$

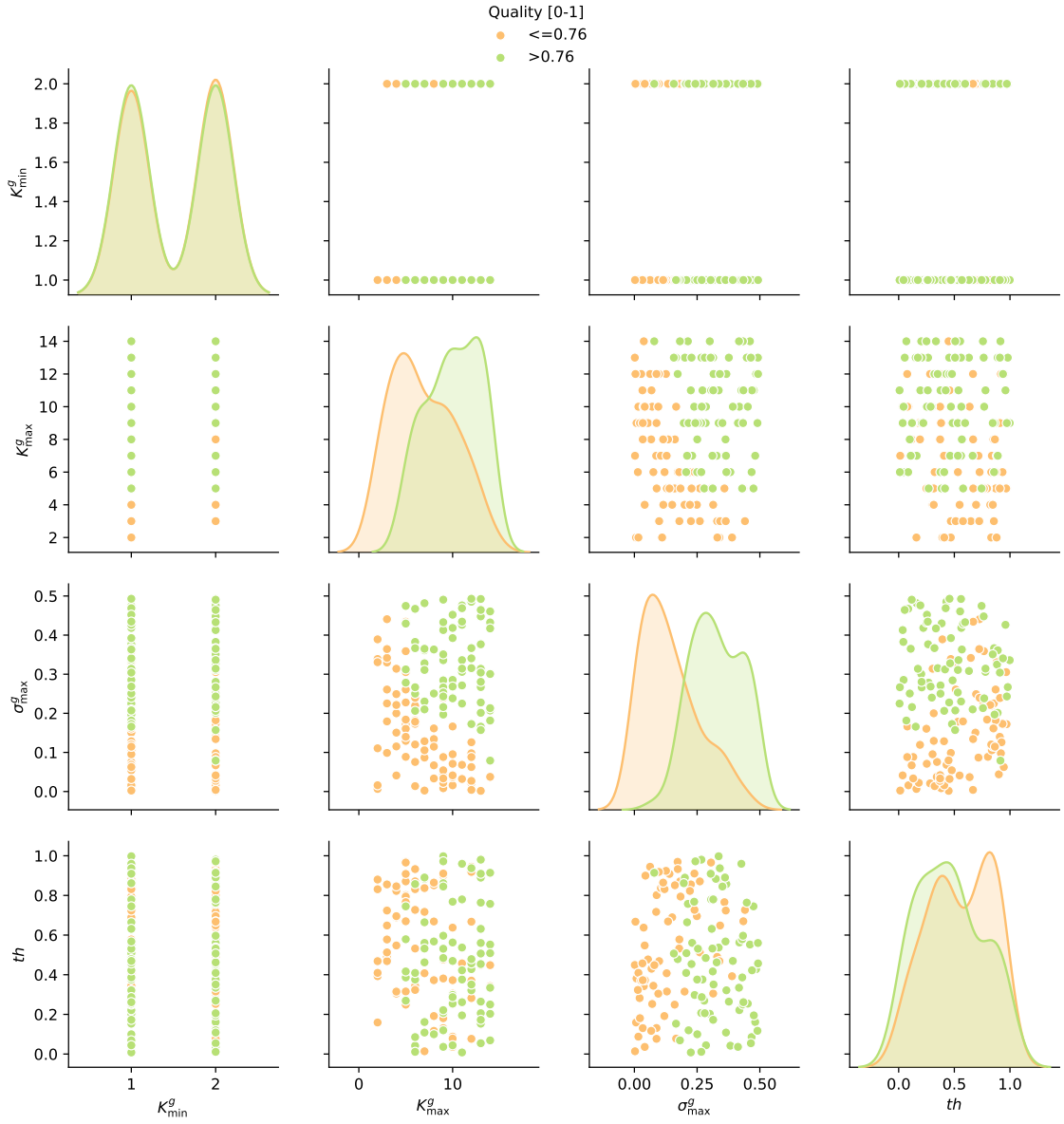


Figure 7.5: Pair plot showing the performance of several parameters on the 2D intensity edge detection algorithm. The tests are clustered based on their quality score. The diagonal plots contain the distributions for each score.

neighbourhood radius in the 3D point cloud. Therefore, these results show that large neighbourhoods provide better performance to detect consistent edges.

Regarding the $\sigma_{\max}$ and th parameter a similar behaviour to the 3D intensity edge detection can be observed, preferring large $\sigma_{\max}$ and small th values. This behaviour is expected since the σ parameter is computed using a similar algorithm in the 2D and 3D domains.

7.4 Global parameter optimization

The analysis performed on the previous sections has allowed establishing correlations and restrictions between the different parameters of the proposed algorithm on edge detection. This information is leveraged in the experiments explained in this section to constrain the search space for each parameter to detect the best performing parameter combinations efficiently.

In the 3D-3D edge detection, the parameters for both the 3D geometrical edge detection and the 3D intensity edge detection are jointly optimised. This experiment, the details of which can be found in Appendix B.6.3, performs a random search between several parameter combinations. However, to test several combinations on a short amount of time, a hyperparameter optimisation algorithm, typically used to test several parameters on deep learning algorithms, is used.

The selected algorithm for this test is the same algorithm used in the first part of this thesis to train hyperparameters: the Hyperband [47] optimisation algorithm. This algorithm measures the performance on different time steps and distributes the available resources among the best performing tests. This algorithm is selected due to its high generalisation capabilities on many scenarios [47].

In each time step of the testing process, only a small subset of the validation patch matching dataset is used. Only those tests with high-quality measures are carried out on the full dataset. The usage of this method, therefore, yields inaccurate results on low performing tests but allows to estimate the best performing parameters in

	Quality	$K_{\min}$	$K^g_{\max}$	$\sigma^g_{\max}$	$K^y_{\max}$	$\sigma^y_{\max}$	th
#1	0.743	6	103	0.148	18	0.010	0.279
#2	0.741	5	177	0.102	24	0.006	0.922
#3	0.740	8	66	0.120	29	0.035	0.213
#4	0.739	7	133	0.072	29	0.042	0.158
#5	0.739	7	106	0.174	27	0.054	0.115

Table 7.1: Table containing the 5 best performing combinations on the 3D edge detection algorithm, which combines the 3D geometrical edge detection and 3D intensity edge detection algorithms.

	Quality	$K_{\min}$	$K_{\max}$	$\sigma_{\max}$	th
#1	0.829	2	13	0.448	0.764
#2	0.818	2	13	0.492	0.560
#3	0.815	1	14	0.461	0.552
#4	0.810	1	13	0.304	0.909
#5	0.806	1	12	0.493	0.457

Table 7.2: Table containing the 5 best performing combinations on the 2D edge detection algorithm.

less computational time. The results of this experiment can be seen in Table 7.1.

A similar technique is performed for the 2D-2D edge detection evaluation. In this case, however, since this edge detection only involves the 2D intensity edge detection algorithm, the usage of the prior constraints and the hyperparameter optimisation algorithm do not provide significant improvements in the performance.

The results of this experiment, the details of which can be found in Appendix B.7, can be seen in Table 7.2.

The evaluation proposed 2D-3D edge detection algorithm involves all three edge detection algorithms proposed in this thesis. The search space to find the best combination in this scenario is, therefore, larger than the search spaces in both the 3D-3D and 2D-2D evaluations.

In this case, it is especially relevant both to constrain the parameters with the observed behaviours on previous sections and to use the hyperparameter optimisation algorithm described before. The results of the experiment performed leveraging these two factors, explained in detail in Appendix B.8, can be seen in Table 7.3.

		Cloud edge detection						Image edge detection			
	Quality	$K_{\min}$	$K^g_{\max}$	$\sigma^g_{\max}$	$K^y_{\max}$	$\sigma^y_{\max}$	th	$K_{\min}$	$K_{\max}$	$\sigma_{\max}$	th
#1	0.813	7	96	0.029	30	0.456	0.857	2	9	0.285	0.661
#2	0.803	7	86	0.111	20	0.296	0.972	1	2	0.428	0.832
#3	0.784	6	79	0.034	41	0.250	0.552	2	6	0.022	0.777
#4	0.773	9	147	0.116	40	0.454	0.991	1	13	0.353	0.793
#5	0.772	5	50	0.097	45	0.207	0.761	2	3	0.253	0.555

Table 7.3: Table containing the 5 best performing combinations on the multimodal 2D-3D edge detection algorithm. This detection combines the 3D geometrical edge detection, 3D intensity edge detection and 2D intensity edge detection algorithms.

The best performing parameter combination obtained in this test is used both in the benchmark presented in Section 7.6 and the final transformation refinement presented in Chapter 8.

7.5 Random subsampling seeding speedup

The procedure to compute the edge points for a 3D point cloud described in this thesis involves performing an expensive computation on all the points on the cloud. This computation requires an average of 36 seconds per frame on the Jetson TX2 board described in Appendix A.3, which can be too computationally expensive for certain applications.

A heuristic approach is proposed in Section 4.1.1 to speed up the algorithm. The proposed algorithm selects a small subset of the point cloud as seeding points for edge points. For each of this seed point, the quality score is computed. If the score is higher than a certain threshold, the points surrounding this point are also computed and selected as seeding points. This approach allows avoiding to compute the score for each point in the cloud, thus reducing the computational time.

However, the likelihood of not detecting a portion of the edges increases when a small subset or a large threshold factor is selected. Therefore, there is a trade-off between the computational time and the algorithm performance.

The effects of this approach are evaluated in this section. The algorithm is evaluated in two metrics to perform this evaluation: the percentage of wrongly computed

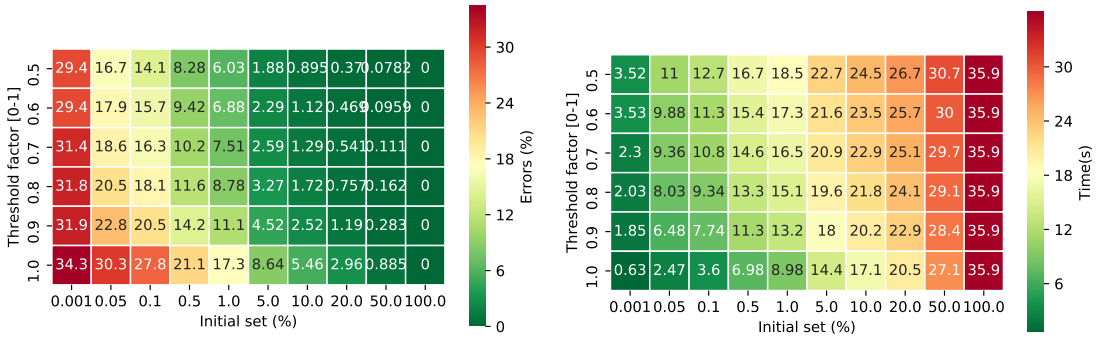


Figure 7.6: Errors (%, left) and computational time (seconds, right) for several parameter combinations.

points and the computational time. The tuned parameters in these tests are the percentage of random points selected as seeding points and the threshold to select seeding points, described as a factor of the final threshold of the algorithm.

The results of this test, the details of which can be found in Appendix B.9, are laid out in Fig. 7.6. As can be seen, there is a clear inverse correlation between the computational time and the percentage of wrongly computed points.

Both parameters can be tuned to take into account the computational needs and performance needs of this algorithm using these results. Since the goal of this thesis is to evaluate the potential of the algorithm in terms of performance, a conservative value with 1% of errors is selected. Looking at the figure, it can be seen that, for those values with an error between 1% and 1.2%, the less computational time is obtained when selecting a 20% of the points as seeding points and using as a threshold $0.9th$, where th is the final threshold of the edge detection algorithm.

The same method described above can be applied to 2D images. However, due to the inherent lattice structure of the 2D image, the procedure has a much faster computation time in 2D images. A test performed with similar characteristics as the presented test in 2D images has found that the average computation time on a single image is around 2.97s. Therefore, the random subsampling speedup is not applied to 2D images.

Method	2D-2D domain	3D-3D domain	2D-3D domain
Canny [14]	0.584	0.608[2]	0.513[2]
HED [108][1]	**0.778**	**0.744**[3]	0.636[3]
Pauly [66]	-	0.637	-
Proposed	0.747	0.679	**0.691**

[1] The implementation in [60] has been used in this test.
[2] The 3D detection is done by detecting Canny contours on depth images.
[3] The 3D detection is done by detecting HED contours on depth images.

Table 7.4: Benchmark on edge detection against state of the art methods. The quality measure is used as evaluation metric.

7.6 Benchmark

In the previous sections, the parameters of the proposed methods to perform a multi-domain edge detection have been evaluated. In this section, the best performing methods on each domain –single modal 2D-2D, single-modal 3D-3D and multi-modal 2D-3D edge detection– are compared against the existing state of the art algorithms.

The canonical algorithm to perform edge detection in 2D images is the Canny [14] edge detector. This algorithm detects edge contours present in a 2D image using variations in the intensity value. The Canny edge detection algorithm can also be used on depth images [20]. In this case, the Canny algorithm is applied to the disparity map image to detect sharp changes in depth values.

Recently, new developments in deep learning have provided state of the art methods to detect semantic contours. The most relevant of these methods is the Holistically-Nested Edge Detection (HED), a deep learning method to detect semantic contours on 2D images. This method is also tested to both intensity and depth images.

However, none of these methods is able to detect contours in unorganised point clouds. In this regard, the state of the art method used in this benchmarking is the Pauly edge detection [66]. This algorithm is used in the proposed method alongside the proposed intensity edge detector.

Table 7.4 presents a comparison between the state of the art method and the proposed edge detector. This evaluation is also performed using the quality measure

explained in Section 5.2 for each method using the full test split of the database. The details on the parameters of this evaluation can be found in Appendix B.10.

As it can be seen, the HED method outperforms all the non-deep learning methods on both 2D-2D and 3D-3D contour detections, showing that the semantic contour detection provides a reliable source of repeatable contours. However, when focusing on the non-deep learning algorithms it can be seen that the proposed method obtains the best performance on both 2D-3D and 3D-3D registrations.

The contours provided by the proposed 2D image edge detection have shown to provide significant higher quality, comparable to the deep learning approach and clearly outperforming the Canny edge detection. In terms of the cloud edge detection, it can be seen that the addition of intensity contours to the edge detection provides a performance increase regarding the state of the art Pauly edge detection.

In the 2D-3D domain, however, the proposed algorithm obtains the best performance among all the evaluated methods. The quality of the 2D-3D detection is higher than both the HED and the Canny edge detection. This result is remarkable since both the HED and Canny edge detection use depth images instead of unorganised point clouds to perform the detection.

This benchmark shows that the proposed method obtains excellent performances on the proposed quality measure, especially on the 2D-3D edge detection evaluation. Moreover, the proposed method provides the ability to detect repeatable contours between 2D images and unorganised 3D point clouds. This ability is used as explained in Chapter 4 to refine the transformation estimation between a 2D image and a 3D point cloud.

8

Transformation Estimation Evaluation

Chapters 6 and 7 have evaluated the keypoint correspondence detector proposed in Chapter 3 and the multimodal edge detection proposed in Chapter 4) These two algorithms are used in the pipeline proposed in this thesis to estimate the transformation between a single image and a single point cloud using the pipeline shown in Fig. 8.1.

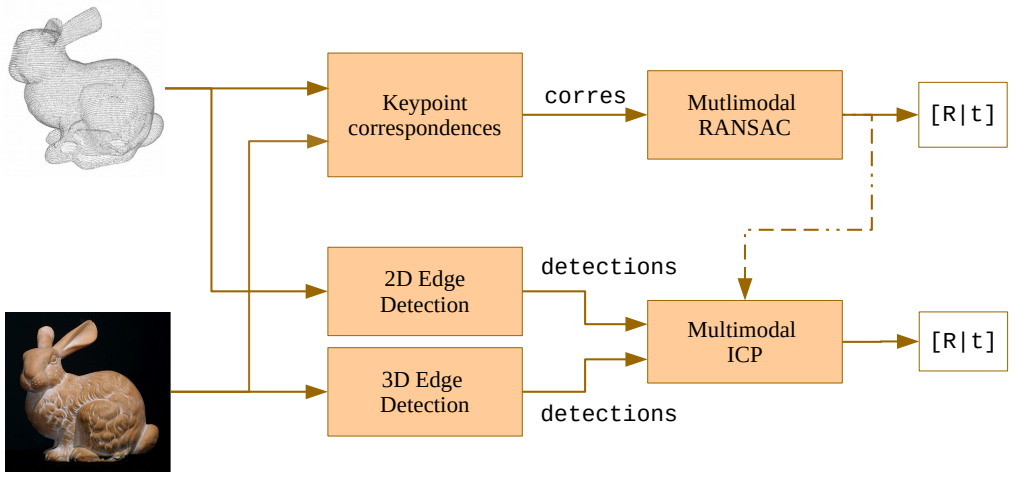


Figure 8.1: 2D-3D registration pipeline.

The following sections evaluate the performance of the keypoint correspondence detector and the multimodal edge detection to estimate the transformation between two frames in both single and multimodal scenarios. First of all, the performance of the initial estimation using the RANSAC algorithm is evaluated in Section 8.1. This evaluation is performed in both the 3D-3D registration scenario and 2D-3D registration scenario, focusing on the different steps of the algorithm.

Section 8.2 evaluates the performance of the iterative refinement applied to the edges obtained by the multimodal edge detection. This evaluation is also performed in both the 3D-3D registration scenario and 2D-3D registration scenario, comparing its performance against the unmodified ICP algorithm.

Finally, Section 8.3 provides a comprehensive evaluation of the full pipeline, comparing it against the state of the art methods and proposing a final discussion on this evaluation chapter.

8.1 Initial transformation estimation

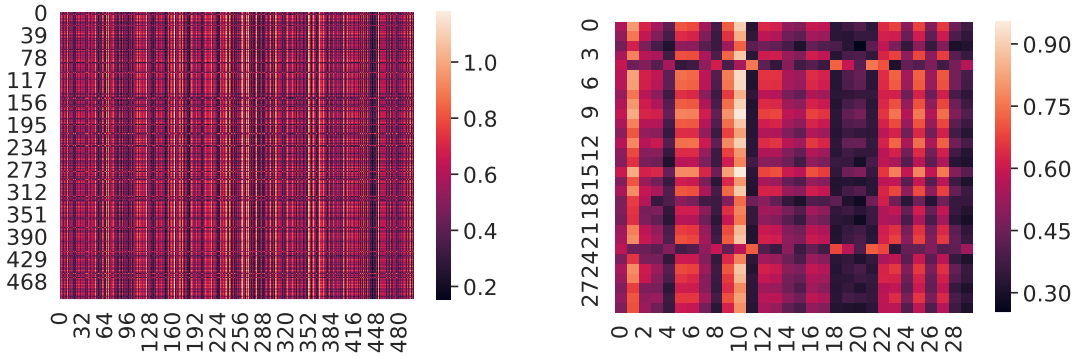
This section evaluates the different aspects of the initial transformation estimation obtained when using the correspondence detector explained in Chapter 3 and evaluated in Chapter 6. This evaluation is performed using a set of correspondence frames obtained from the validation split of the 3DMatch dataset, as explained in Appendix A.1.3.

Two different architectures are proposed in Chapter 6 for the correspondence detector: the 'Fast' architecture, which uses a faster but simpler network, and the 'Deep' architecture, which uses a slower but deeper network. Table 8.1 summarises the FPR95 values obtained for each network and each modality in the validation split.

These two structures are used in the evaluation presented in this section. First of all, the performance of the correspondence detection to estimate a set of correspondences between two frames is evaluated. The next section evaluates the performance of the RANSAC estimator. These evaluations allow framing the results obtained in the

Architecture	Single modal 3D-3D	Multi modal 2D-3D
Fast	0.398	0.646
Deep	0.341	0.593

Table 8.1: FPR95 error measurements obtained when using the different architectures on the validation split on the correspondence matching dataset.



(a) Heatmap **(b)** Zoom in

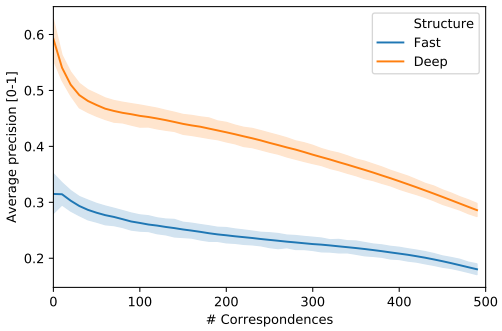
Figure 8.2: Visual representation of the distance matrix between each keypoint combination. This distance matrix is obtained when using a random frame pair of the validation split with the 'Fast' architecture in the 3D-3D registration pipeline.

single-modal and multimodal transformation estimation presented in the final part of this section.

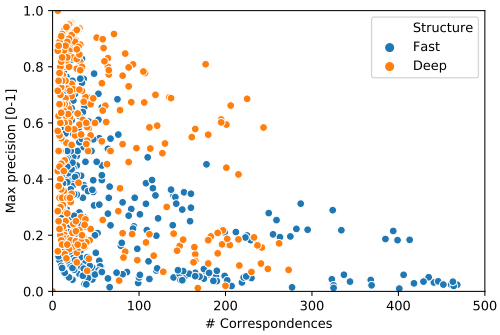
8.1.1 Correspondence estimation

The initial transformation is obtained by estimating the transformation between a set of keypoints obtained using the correspondence detector. These correspondences are obtained by selecting a random keypoint set on both data to be matched and computing the distance between them in the descriptor-defined space. This selection results in a matrix D where each element contains the distance between each keypoint pair p_i, q_j in the descriptor-defined space. A visual representation of this matrix can be seen in Fig. 8.2.

The keypoint combinations with less euclidean distance are selected, as explained in Section 3.4.1. These keypoint pairs are used as the correspondence set to perform the registration. As it can be seen, when N keypoints are selected in both sets, at most N of the $N \times N$ possible combinations can be selected without repeating



(a) Average amount of correct detections for each correspondence selection



(b) Scatter plot of the maxuimum correct amount of detections and the amout of correspondences selected in each case

Figure 8.3: Visualizations of the correspondence precision for the different architectures trained in the 3D-3D registration scenario

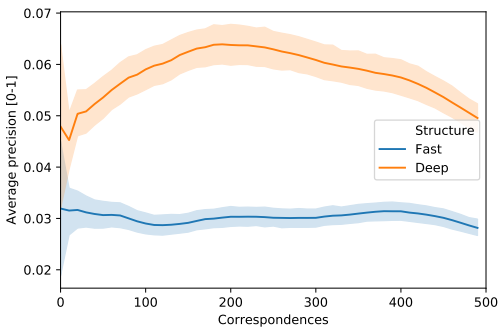
keypoints. The results presented in this section select a random set of $N = 500$ points. Therefore, at most, 500 of the 250000 possible combinations can be selected.

This scenario is significantly different from the training scenario of the keypoint detector. The database used to train the keypoint detector contained a balanced amount of matches and non-matches, whereas on the application to select keypoint correspondences only a small subset of the possible pairs is a true correspondence.

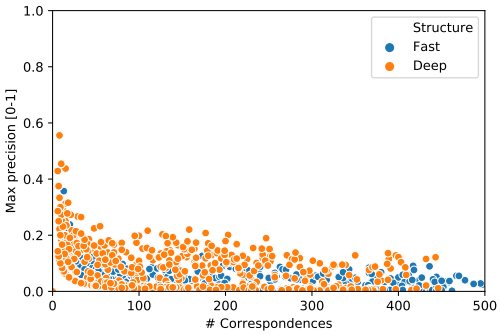
Therefore, the error measure used in the previous sections to evaluate the keypoint detectors does not evaluate the performance of the correspondence detector in this scenario. The proposed evaluation measures the precision obtained when k keypoint pairs –with $k \leq N$– are selected. Different k values are tested for each image pair in the database. More details about this test can be found in Appendix B.12.1.

Fig. 8.3a shows the obtained average precision for each k correspondence length. As can be seen, better precision values are obtained when using the 'Deep' architecture. Its curve shows that, on average, half the correspondences selected in the first k correspondences are true correspondences. This figure is slightly lower with the 'Fast' architecture, where only 1/3 of the correspondences are correct correspondences.

It is also interesting to see the distribution of the optimal correspondence set size for each frame pair. Figure 8.3b shows the optimal amount of correspondences and the maximum precision achieved on each frame pair. As can be seen, the optimal



(a) Average amount of correct detections for each correspondence selection



(b) Scatter plot of the maxuimum correct amount of detections and the amout of correspondences selected in each case

Figure 8.4: Visualizations of the correspondence precision for the different architectures trained in the multimodal 2D-3D registration scenario

correspondence length is below 50 correspondences on most frame pairs, both when using the 'Fast' architecture and the 'Deep' architecture.

The previous analysis, however, was focused on the 3D-3D domain, where the performances of the proposed architectures are on the state of the art levels on 3D-3D registration. The performance obtained in the 2D-3D domain, however, is significantly lower.

Fig. 8.4 shows the same evaluation on the correspondence precision performed on the multimodal 2D-3D scenario. As it can be seen in Fig. 8.4a, this increase in the error rate directly translates to a sharp decrease in the average precision for the different k correspondence lengths, with the 'Deep' architecture obtaining the best performance.

Observing Fig. 8.4b it can also be seen that, in this scenario, the optimal size for the correspondence set on the different frame pair is scattered in a bigger range, extending to up to 200 correspondences. It is also relevant to see the decrease in precision for the correspondence sets with between 300 and 500 correspondences.

Another factor needs to be taken into account to analyse this decrease: the overlapping between frames. Since the frames being registered do not have a 100% overlap, there are some randomly selected keypoints in non-overlapping areas. These keypoints will not have a correspondence in the other frame, therefore limiting the

maximum amount of keypoints that can be matched.

8.1.2 RANSAC performance

The previous results have allowed evaluating the performance of the correspondence detector in the registration pipeline. These results have shown that while the 3D-3D registration offers good performance on the validation set, the 2D-3D correspondence matching algorithm has a low precision in the scenario presented.

Therefore, the next step on the analysis of the transformation estimation is the evaluation of the capability of the RANSAC algorithm to estimate the transformation with false correspondences present on the correspondence set.

Iterations

As explained in Section 3.4, the RANSAC algorithm is an iterative method used to estimate a set of parameters using a set of measurements. The most relevant parameter of this algorithm is the number of iterations performed.

A random subset of the correspondence set is selected on each iteration to estimate the transformation. After a certain number of iterations, the estimated transformation that obtains the highest amount of inliers is selected.

Therefore, the probability that all the correspondences selected in an iteration are correct in the subset can be computed when the ratio of true/false correspondences is known. As explained in Section 3.4, this information also allows computing the needed amount of iterations to have an iteration where all the selected correspondences are correct within a confidence value.

It also has to be taken into account, however, that performing more iterations also increases the computation time. Appendix B.12 describes the test performed to measure this computation time.

		Confidence values		
		0.8	0.9	0.99
Correspondence ratio	0.03	2.21E+09	3.16E+09	6.32E+09
	0.1	1.61E+06	2.30E+06	4.61E+06
	0.2	25147	35977	71953
	0.3	2207	3157	6315
	0.4	392	561	1122
	0.5	102	146	292
	0.6	34	48	96
	0.9	2	3	6

Table 8.2: Estimated amount of needed iterations for different correspondence ratios (rows) and confidence values (columns).

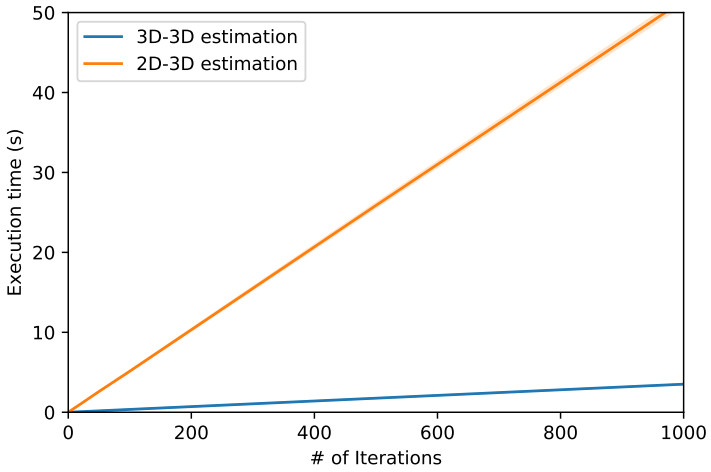
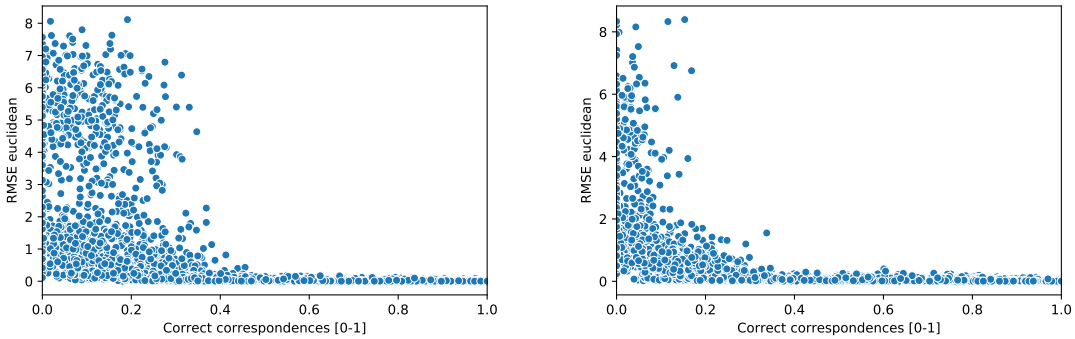


Figure 8.5: Execution time (in seconds) of the RANSAC pipeline for different number of iterations.

Table 8.2 shows the estimated amount of iterations for different correspondence ratios and confidence values, and Fig. 8.5 shows the computation time of the 3D-3D and 2D-3D RANSAC algorithms for different computation times. As it can be seen, the needed number of iterations –and therefore, the computation time– exponentially increases when the correspondence ratio decreases.

This trade-off between computation time and performance is taken into account jointly with the results obtained in the previous section to define the number of iterations. It is estimated that between 100 and 1000 iterations a reasonable amount of correct transformations can be retrieved. The results presented in this thesis are obtained with 500 iterations of the RANSAC algorithm on both the 3D-3D and 2D-3D algorithms. According to the results in Fig. 8.5, with this number of iterations



(a) 3D-3D registration **(b)** 2D-3D registration

Figure 8.6: RMSE on the euclidean point distance for different ratios of correct/incorrect correspondences.

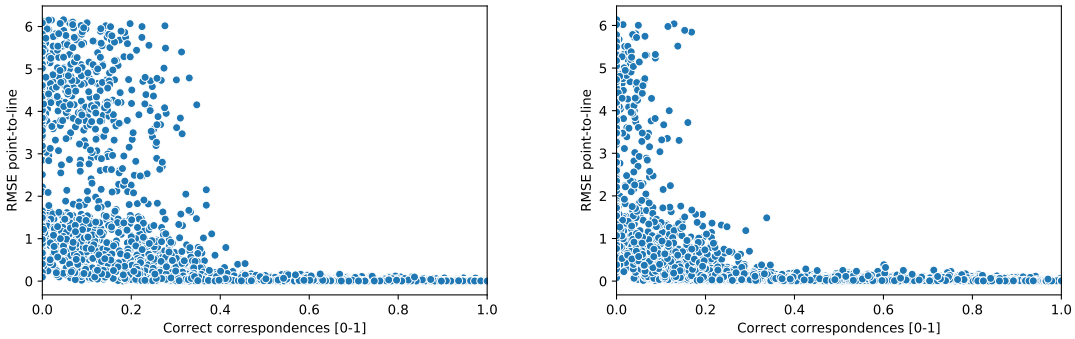
the 3D-3D registration is performed in 1.8s and the 2D-3D registration is performed in 19.6s. This time difference is mainly caused by the sub-optimal implementation of the LM algorithm on the 2D-3D transformation estimation.

Robustness to Outliers

The most common method to estimate a 3D-3D transformation is to compute the transformation that minimises the 3D Euclidean distance between correspondence points. This thesis has proposed a RANSAC estimation of the 2D-3D transformation by minimising the point to line distance between correspondences.

The goal of the following experiment is to compare the performance of the proposed estimation to the state of the art 3D-3D registration minimising euclidean point distances. This analysis is performed by artificially selecting different ratios of true and false correspondences and performing the registration. The details of this experiment can be found in Appendix B.12.2.

Figure 8.6 shows the Root Mean Squared Error (RMSE) of the euclidean 3D distance between points in the two different transformation estimation processes. Figure 8.6a shows the RMSE when estimating the transformation using the euclidean 3D distance between two 3D points used in the 3D-3D registration pipeline. Figure 8.6b shows the RMSE when estimating the transformation using the point-to-line distance between a 3D point cloud and the line defined by a 2D image pixel, as ex-



(a) 3D-3D registration **(b)** 2D-3D registration

Figure 8.7: RMSE on the point-to-line distance for different ratios of correct/incorrect correspondences.

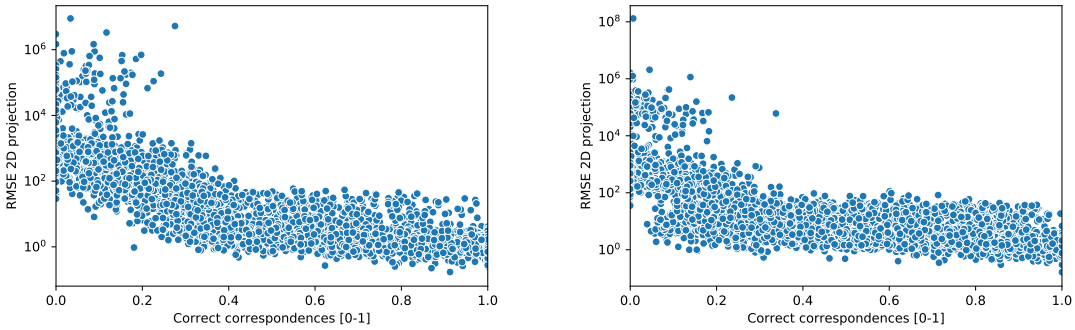
plained in Section 3.4 and used in the 2D-3D registration pipeline.

As can be seen, both estimations start to provide consistently correct estimations when at least a 40% of the correspondences are correct correspondences. These results align with the confidence values obtained in Table 8.2.

It can also be observed that the 2D-3D transformation estimation obtains, on average, lower RMSE values when using a low correspondence ratio. However, the tests performed regarding the accuracy of both methods have shown that this lower RMSE does not translate to higher accuracy of the transformation estimation and only affects the incorrectly estimated transformations.

It is also interesting to measure the RMSE of the other evaluation metrics proposed in Section 3.4: the 3D point-to-line distance and the 2D projection distance in the camera plane. Figure 8.7 shows the RMSE measured using the 3D point-to-line distance between a 3D point cloud and the line defined by a 2D image pixel. Figure 8.7 shows the RMSE measured using the 2D euclidean distance between the projection of the 3D point cloud into the camera plane and the 2D image pixel. Both error metrics are explained in more detail in Appendix A.2.1.

It can be seen that the RMSE on the 3D point to line distance shown in Fig. 8.7 follows a similar pattern that the RMSE on the 3D euclidean distance, both in the 3D-3D registration (Fig. 8.7a) and 2D-3D registration (Fig. 8.7b). This shows that both measures have similar properties, endorsing the decision to use the 3D



(a) 3D-3D registration **(b)** 2D-3D registration

Figure 8.8: RMSE on the projected 2D distance for different ratios of correct/incorrect correspondences. Note that this graphic is plotted using the logarithmic scale.

point-to-line distance as the minimization measure.

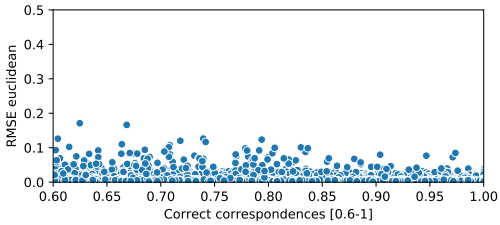
However, observing the RMSE on the projection on the 2D camera plane, it can be seen that this error measure is highly unstable, showing errors up to 10^6 pixels. This instability is likely to be caused by the non-linearity present when points are behind the camera plane. In a correct registration estimation, this factor should not affect – no points should lie behind the camera plane –, but in incorrect estimations, some points can lie behind the camera plane and cause spikes in the error measures. This non-stability is also discouraging the usage of this distance measure to estimate the transformation.

These results also allow to evaluate the quality of the estimated 2D-3D transformation when a correct estimation is established. Figure 8.9 shows a cropped version of the graphics presented in Figs. 8.6 to 8.8, focusing on the area where the correctly estimated transformations are computed.

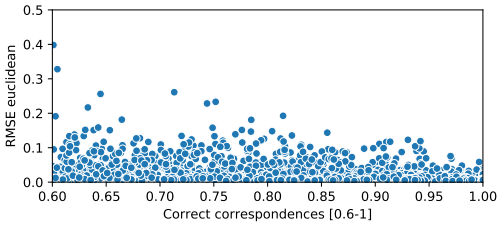
Observing the RMSE on the 3D euclidean distances (Figs. 8.9a and 8.9b) it can be seen that, on average, the 3D-3D estimation obtains slightly lower RMSE values than the 2D-3D estimation. This shows that the 2D-3D distance measure has a higher imprecision to estimate the transformation.

These error measures can also be used to define a threshold in which a transformation estimation is deemed correct. As explained in Appendix A.2.1, the state of the art evaluation considers a correct transformation those with an RMSE below 0.2m.

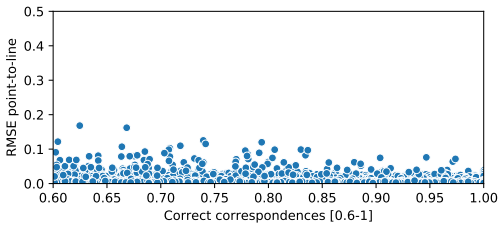
Observing Figs. 8.9a and 8.9b it can be seen that on both the 3D-3D and 2D-3D transformation estimations the RMSE values are below this threshold, and therefore the same threshold is used.



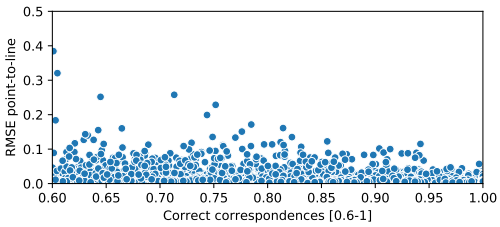
(a) 3D-3D registration RMSE 3D eculidean



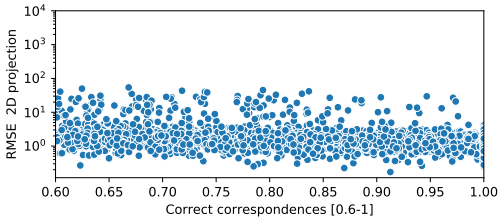
(b) 2D-3D registration - RMSE 3D eculidean



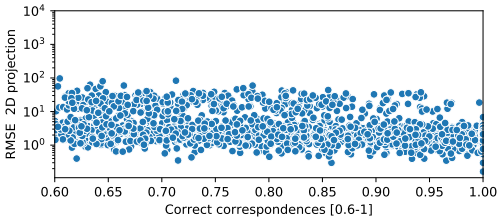
(c) 3D-3D registration - RMSE 3D point-to-line



(d) 2D-3D registration - RMSE 3D point-to-line



(e) 3D-3D registration - RMSE 2D projection



(f) 2D-3D registration - RMSE 2D projection

Figure 8.9: Zoom in on the correctly performing samples on Figs. 8.6 to 8.8.

The RMSE on the 3D point-to-line distances (Figs. 8.9c and 8.9d) obtains similar results. The absolute error is slightly smaller, since for any pair of points the point-to-line distance is always smaller or equal than the point-to-point distance of a point contained inside the line.

Regarding the RMSE on the 2D projection distances on the camera plane (Figs. 8.9e and 8.9f) it can be seen that the average error ranges between 10 and 100 pixels on both estimations. The different behaviour between the 3D point-to-line measure and the 2D Euclidean distance may cause the transformation not to be representative enough on short camera distances where the 2D Euclidean distance sharply increases. This high error must be taken into account when applying this algorithm

to applications such as 3D point cloud colouring. In these cases, it may be advisable to add a refinement step to minimise this error.

8.1.3 Registration results

The previous sections have analysed the performance of the correspondence detector and the ability of the transformation estimation to correctly estimate the transformation when incorrect correspondences are present in the correspondence set.

It has been found that both the 3D-3D and the 2D-3D transformation estimation algorithms can estimate the transformation with a considerable ratio of incorrect correspondences. The 3D-3D transformation obtains a correct estimation with only a 40% of correct correspondences, and the 2D-3D transformation obtains a correct estimation with only a 50% of correct correspondences.

The performance of the proposed correspondence detectors has also been evaluated. The 'Fast' 3D-3D detector obtains, on average, 30% of correct correspondences, and the 'Deep' 3D-3D detector obtains an average of 50% of correct correspondences on the validation set. However, the 'Fast' 2D-3D detector only obtains on average a 3% of correct correspondences, and the Deep' 3D-3D detector obtains an average of 5% of correct correspondences.

These results show that while the 3D-3D registration is likely to obtain a good performance, the 2D-3D registration is not likely to be able to obtain a correct registration. These hypotheses are confirmed by the results obtained when evaluating the full pipeline, shown in Table 8.3. The details of this experiment can be found in Appendix B.12.3.

As can be seen, the 3D-3D registration pipeline obtains a correct registration on a high number of tests. The 'Fast' pipeline obtains a 64% of correct registrations, whereas the 'Deep' pipeline obtains an 83% of correct registrations. These results align with the average number of correspondences obtained with each method.

However, the 2D-3D registration is not able to correctly estimate the transformation,

Domain	Method	Recall @ 0.2RMSE	RMSE @ Top10%
3D-3D	Fast	0.642	**0.040**
	Deep	**0.829**	0.063
2D-3D	Fast	-	**0.489**
	Deep	-	0.598

Table 8.3: Evaluation of the 3D-3D registration and 2D-3D registration on the validation split using several metrics.

neither using the 'Fast' nor the 'Deep' pipeline. Therefore, the error rate of these methods is proven to be too high to obtain a correct registration.

8.2 Transformation refinement

This section evaluates the performance of the transformation refinement performed using the edge detection algorithm described in Section 4.3. This method leverages the ICP registration algorithm to refine an initial transformation for both 3D-3D and 2D-3D registrations.

To summarise, the core of the ICP algorithm can be described as follows:

1. Pair each point from the source frame to the closest point in the target frame.
2. Estimate the transformation that minimises the distance between pairs of points.
3. Evaluate the error of the transformation.
4. Repeat steps 1 to 3 until the error between two iterations does not change above a certain threshold.

The original ICP is applied using the full point cloud and using the 3D euclidean point to point distance to estimate the transformation and measure the error. As explained in Section 4.3, this thesis proposes two fundamental modifications to apply this algorithm int the 2D-3D registration algorithm.

The first modification is to use only a relevant subset of the point cloud to perform the registration. This modification allows transforming the image into a non-dense representation suitable to perform the registration.

The second modification is to use the 3D point to line distance between the 3D point and the line defined by the projection of the pixel. As has been seen in the previous section, this distance allows estimating the transformation in the 2D-3D registration scenario accurately.

This section evaluates the performance of the ICP algorithm when these modifications are applied, both in the 3D-3D and 2D-3D registration pipelines. This evaluation also assesses the performance of the edge detector evaluated in Chapter 7 to estimate the transformation and the robustness of the proposed transformation estimation against noise on the detections and the initial transformation.

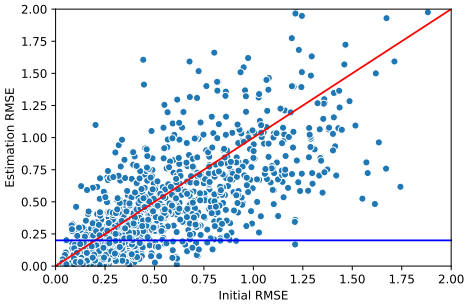
8.2.1 Effect of the initial transformation

The first part of this evaluation is focused on the robustness of the proposed EdgeICP pipeline against errors on the initial transformation. Three different algorithms are compared in this evaluation: the original ICP algorithm, the 3D-3D EdgeICP using edge detection and the 2D-3D EdgeICP using both edge detection and 2D-3D transformation estimation.

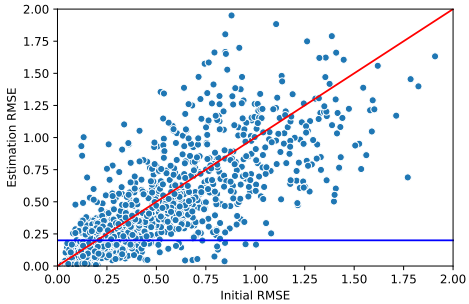
The edge detection used in this test is artificially modified to have 100% repeatability, allowing only to evaluate the influence of the initial transformation. More details about this test can be found in Appendix B.13.1.

Figure 8.10 shows the RMSE measured in the 3D point-to-point distance of the initial transformation –x-axis– and the estimated transformation –y-axis for the three different methods. Two guiding lines are also added in these graphics. The red line denotes the points in which the initial RMSE equals the final RMSE. Therefore, all points below this line are experiencing an improvement in the estimated transformation.

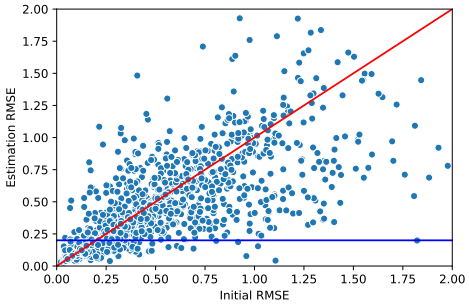
The blue line marks all points whose estimated transformation has a RMSE of 0.2m, the threshold value used in the previous sections to define a correct transformation. All the points below this line are the points in which the algorithm estimated a



(a) 3D-3D ICP [112, 7] without edge detection



(b) 3D-3D EdgeICP



(c) 2D-3D EdgeICP

Figure 8.10: Evaluation of the capability of the different methods to estimate the transformation when errors are present on the initial alignment.

Method	Improvement ratio	Recall @ 0.2RMSE
ICP [112, 7]	0.512	**0.427**
3D-3D EdgeICP	0.561	0.380
2D-3D EdgeICP	**0.617**	0.339

Table 8.4: Performance comparison between the proposed algorithms. 'Improvement ratio' is the percentage of samples [0-1] in which the RMSE of the estimated transformation is lower than the RMSE of the initial transformation. Recall@0.2RMSE is the percentage of samples [0-1] in which the estimated transformation has an RMSE below 0.2m.

correct transformation.

It can be seen that all methods offer similar performance, being only able to handle small corrections in the transformation estimation. It can also be seen, however, that neither using the edge detection algorithm nor estimating the transformation using the 2D-3D distance provide any significant change on the algorithm performance in this evaluation.

To perceive the subtle differences between the different methods, in Table 8.4 numeric evaluation of the different algorithms is presented using the thresholds defined in the graphics. The first column represents the ratio of experiments in which the transformation provided an improvement regarding the initial RMSE, while the second column represents the ratio of experiments with a RMSE lower than 0.2m.

Comparing the state of the art ICP with the EdgeICP it can be seen that the first obtains a better performance in terms of the recall measure, while the latter obtains a better performance in terms of the improvement ratio. This result shows that using only the edges to perform the registration allows avoiding falling in local minima that avoid improving the registration. However, this lack of points affects the quality of the registration, obtaining a lower ratio of correctly estimated transformations.

The same trade-off can be observed when comparing the EdgeICP method on the 3D-3D registration and the 2D-3D registration pipelines. The algorithm obtains a performance in terms of the recall measure in the 3D-3D registration, while the 2D-3D registration obtains a higher improvement ratio. In this case, however, these differences are caused by the different characteristics of the 3D-3D and 2D-3D transformation estimations.

Domain	Quality	Repeatability	Detection ratio
3D-3D	0.743	0.979	0.492
2D-3D	**0.858**	**0.857**	**0.140**

Table 8.5: Performance of the proposed edge detectors on the validation split.

These values are coherent with the evaluation performed on the RANSAC estimation. This evaluation showed that the 2D-3D transformation estimation provides overall lower RMSE on incorrect estimations but has a similar overall imprecision on the estimation.

The same behaviour can be observed in these results. The higher improvement ratio of the 2D-3D transformation estimation can be attributed to the lower RMSE of the estimation transformation. Likewise, the slightly higher Recall of the 3D-3D transformation estimation can be attributed to the lower imprecision of the 3D-3D distance metric.

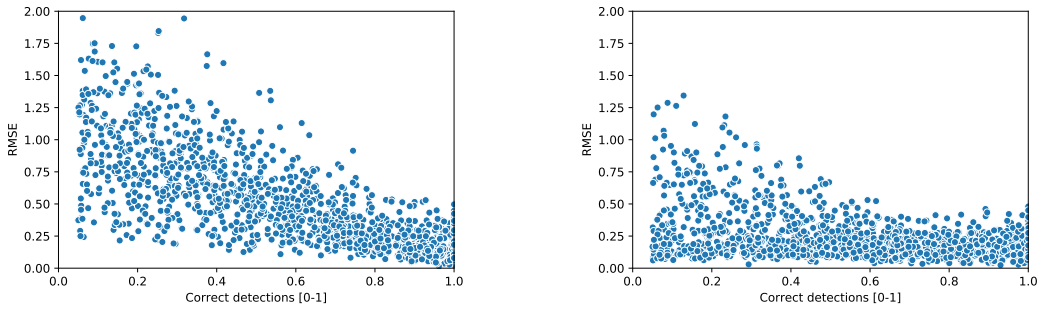
8.2.2 Edge detection evaluation

The next part on this evaluation of the transformation refinement assesses the effects of the edge detection method quality on the registration performance. As explained in Section 5.2, two different metrics are combined to assess the quality of the edge detector: the repeatability of the algorithm and the overall number of detections.

Table 8.5 shows the performance of the proposed edge detectors on the validation split measured using the three metrics. As can be seen, the detector with the best quality also has higher repeatability and the lower detection ratio.

The influence of the repeatability of the edge detection into the registration pipeline is assessed by artificially generating detections with different repeatabilities and evaluating the RMSE on each detection. The details of this experiment can be found in Appendix B.13.2.

Figure 8.11 presents the results of this experiment in the 3D-3D and 2D-3D registration pipelines. As can be seen, the RMSE error steadily increases when the



(a) 3D-3D EdgeICP **(b)** 2D-3D EdgeICP

Figure 8.11: Evaluation of the RMSE (in the euclidean point-to-point distance) for different ratios of correct detections, on both the 3D-3D and 2D-3D registrations.

Edge detection sensivity	3D-3D registration	2D-3D registration
Low	**0.65**	0.54
Medium	0.66	**0.57**
High	**0.65**	0.54

Table 8.6: Evaluation of the performance (in terms of the Recall@0.2mRMSE) for different edge detection sensitivities.

ratio of correct detections decrease. It can also be seen that the 2D-3D estimation provides an overall lower error when a low ratio of correct detections is present. These results are also coherent with the RANSAC analysis, where the 2D-3D transformation estimation also obtained lower RMSE values on the wrongly estimated transformations.

The edge detection algorithm also has another relevant parameter: the sensitivity. The sensitivity of the edge detection influences the overall number of detections obtained by the edge detection algorithm. The last test on this section evaluates the performance of three artificial edge detections with perfect repeatability and three different sensitivity levels on the 3D-3D and 2D-3D transformation refinement pipelines. The details of this experiment can be found in Appendix B.13.3.

Table 8.6 shows the Recall@0.2RMSE for the three edge detections on the 3D-3D and 2D-3D pipelines. As can be seen, the 3D-3D obtains similar performances with all the sensitivity levels, with slightly better performance on the 'low' and 'high' sensitivity levels.

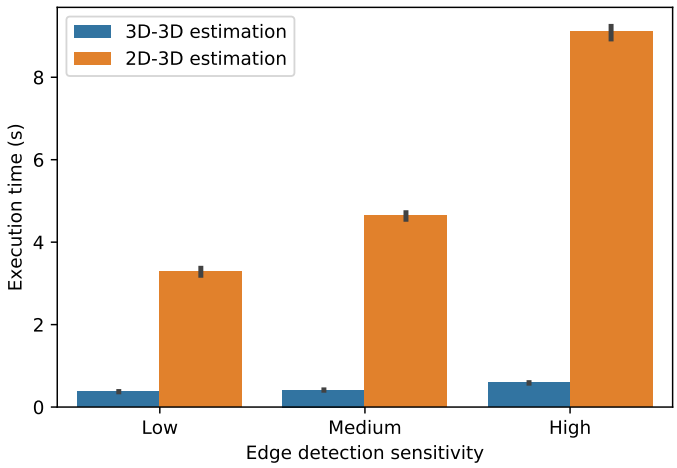


Figure 8.12: Execution time (in seconds) of the ICP pipeline for different edge sensitivities.

However, the 2D-3D registration obtains a slightly better performance with the 'medium' sensitivity level. It is theorised that the extreme sensitivity levels may amplify the 2D-3D transformation imprecision and produce higher RMSE levels, and therefore lower recall measures.

The edge sensitivity may also have another side effect: the variation of the computation time. The transformation estimation in each step of the ICP is performed using the full point cloud. Therefore, since an edge detector with high sensitivity retrieves more points than an edge detection with low sensitivity, this transformation is more computationally expensive.

Figure 8.12 shows the average and standard deviation of the computation time for different sensitivity levels. The details of this experiment can be found in Appendix B.14.

As can be seen, a higher sensitivity also entails a longer execution time. It can also be observed the difference in execution time between the 3D-3D and 2D-3D estimations. As explained in Section 8.1.2, the 3D-3D estimation is solved using the SVD method, whereas the 2D-3D estimation is solved using LM. However, it should be noted that the execution time of the 2D-3D estimation is significantly sped up by only performing a single LM iteration on each ICP iteration, as explained in Section 4.3.2.

Domain	Method	Recall @ 0.2RMSE	RMSE @ Top10%
-	Initial transformation	0.687	0.049
3D-3D	EdgeICP	**0.765**	**0.008**
2D-3D	EdgeICP	0.720	0.038

(a) Results obtained without modifying the initial alignment.

Domain	Method	Recall @ 0.2RMSE	RMSE @ Top10%
-	Random transformation	0.534	0.084
3D-3D	Random + EdgeICP	0.655	**0.008**
2D-3D	Random + EdgeICP	**0.700**	0.053

(b) Results obtained modifying the initial alignment.

Table 8.7: Evaluation of the ICP and EdgeICP on the validation split.

8.2.3 Registration results

The previous tests have evaluated the different aspects of the EdgeICP algorithm. The obtained results have shown that the performance of the proposed algorithm is up to the state of the art ICP algorithm, provided that the edge detection has adequate repeatability and sensitivity.

This section evaluates the performance of the full transformation refinement pipeline on the validation dataset. As explained in Section 5.1, the frame registration dataset is built by selecting frame pairs with at least a 30% overlap. However, due to the nature of the 3DMatch sequences, the displacement between frames is low.

Therefore, two different evaluations are proposed, shown in Table 8.7. The first evaluation, shown in Table 8.7a, is performed using the dataset as-is. The second evaluation, shown in Table 8.7b, is performed by adding random initial additional misalignment. The details of this experiment can be found in Appendix B.14.1.

As can be seen, both the 3D-3D and 2D-3D EdgeICP are able to increase the initial Recall on both experiments. The performance of both algorithms is similar, with the 3D-3D obtaining the best performance on the original dataset and the 2D-3D obtaining the best performance when additional misalignment is added.

As it has been seen in Section 8.2.2, the 2D-3D detector has higher repeatability

and lower detection ratio that the 3D-3D detector. Although the 3D-3D is able to obtain a slightly higher performance on the original dataset, the performance of the algorithm drops when additional misalignment is added.

The 2D-3D algorithm, however, is able to maintain similar performances despite the additional misalignment, showing the potential of this registration algorithm.

8.3 Benchmark

The experiments shown in Sections 8.1 and 8.2 have evaluated several aspects of the proposed transformation estimation and iterative refinement, in both the 3D-3D and 2D-3D domains. As no previous state of the art has provided an extensive evaluation on the 2D-3D domain, this section evaluates the performance of these algorithms against the state of the art algorithms in 3D-3D registration.

The first state of the art algorithm reviewed in this section is the 3DMatch[111] algorithm. This algorithm estimates the transformation between two 3D point clouds using RANSAC andDeep Learning. The similarities between the voxelized surroundings of each 3D keypoint are trained using a Convolutional Neural Network (CNN). The obtained correspondences are used to estimate the transformation between the 3D point clouds using the RANSAC algorithm.

The evaluation of this method presented in this section is extracted from the original article. Although the same sequences are used in this thesis evaluation, the registration dataset is built differently. The differences between the two datasets are detailed in Appendix A.1.3.

The 3DMatch[111] article also evaluates two state of the art methods: Fast Point Feature Histogram (FPFH) and Spin Images. These methods compute several characteristics of the surroundings of a keypoint to extract a hand-crafted descriptor. The evaluation of this methods is extracted from [111], and therefore the registration dataset from [111] is used.

Method	Recall @ 0.2RMSE	RMSE @ Top10%
3DMatch [111][1]	0.668	-
FPFH [76][1]	0.442	-
Spin Images [39][1]	0.518	-
ICP[112, 8][2]	0.549	0.019
Fast	0.402	0.073
Deep	0.678	0.038
EdgeICP	0.572	0.015
Fast + EdgeICP	0.544	0.017
Deep + EdgeICP	**0.747**	**0.013**

[1] Results extracted from [111]. These results were obtained using a database combining blocks of 50 frames (see Appendix A.1.3).
[2] Results computed with the same database used in this thesis.

Table 8.8: Evaluation of the 3D-3D registration on the test split using different metrics.

The last benchmarked state of the art algorithm is the Iterative Closest Point (ICP) registration algorithm. This algorithm refines an initial transformation by iterative aligning the neighbouring points. The evaluation of this algorithm is computed in the database used in this thesis.

These algorithms are compared to the algorithms proposed in this thesis. The initial transformation estimation using deep learning features is tested using two different configurations. The 'Fast' configuration uses a simple but faster structure, whereas the 'Deep' configuration uses a deeper but slower structure.

The iterative refinement method is also evaluated. The best parameter combinations found in Chapter 7 for the 3D-3D and 2D-3D estimations are used in this evaluation.

The full pipeline proposed in this thesis is also evaluated. In this evaluation, both the 'Fast' and 'Deep' estimations are refined using the EdgeICP algorithm.

The results of these evaluations in the 3D-3D domain, explained in detail in Appendix B.15, can be seen in Table 8.8. As can be seen, the best performing algorithm us the 'Deep' registration with the EdgeICP refinement. These results show that the EdgeICP algorithm allows for improving the wrongly estimated transformations estimated using RANSAC.

Regarding the standalone deep learning pipeline, it can be seen that the best perfor-

Method	Recall @ 0.2RMSE	RMSE @ Top10%
Fast	0	1.068
Deep	0	1.070
EdgeICP	**0.501**	**0.069**
Fast+EdgeICP	0	0.571
Deep + EdgeICP	0	0.523

Table 8.9: Evaluation of the 2D-3D registration on the test split using several metrics.

mance, also surpassing the state of the art algorithms, is obtained using the 'Deep' pipeline. It must be noted, however, that the database used in the state of the art tests is a different database, and the lower disparity of the database used in this thesis provide an unfair comparison between the different methods.

The proposed methods are also evaluated in the 2D-3D domain. These results, which can be seen in Table 8.9, confirm the results obtained in Section 8.1.3. The results obtained both sections have shown that neither the 'Fast' nor the 'Deep' pipeline can obtain correct 2D-3D registrations.

Regarding the 2D-3D EdgeICP refinement, however, it can be seen that the algorithm obtains a similar performance than the 3D-3D EdgeICP algorithm. Moreover, it is also able to reduce the RMSE on the incorrectly estimated transformations when using either the 'Fast' or the 'Deep' estimations.

The results presented in this section show that the algorithms proposed in this thesis are domain agnostic and can provide transformations between multimodal correspondences. Moreover, in the 3D-3D domain the algorithms proposed in this thesis can obtain a transformation with performances comparable to the state of the art algorithms. However, the performance on the 2D-3D domain is not good enough to estimate an initial correct transformation.

9

Conclusions

BEING able to seamlessly interpret captures obtained using different 2D and 3D sensors is an essential landmark on computer vision and scene understanding. Nowadays, there is a vast amount of data captured every day. Most of this data is presented in the form of 2D images, but depth sensors are steadily being more used in many fields.

This thesis aims to explore this field by developing a fundamental problem: the direct registration between 2D images and 3D point clouds. The solution to this problem will allow to seamlessly integrate data between different domains and modalities into a common reference frame.

The 2D-3D registration problem is currently being applied in many fields, from medical imaging to terrain surveying. Each one of these fields has data acquired in different formats, and several algorithms are available to obtain the registration between them seamlessly.

Chapter 2 presents an overview of the state of the art algorithms applied in these

fields. The different algorithms are classified using two parameters: the domain in which the registration is performed and the features used in the registration.

The exploration of the registration domain has shown that most methods perform the registration in a single domain, either by translating the 3D data into the 2D domain or by combining several 2D images into a point cloud. The methods that use a direct 2D-3D registration mainly leverage specific features on the scenes, like vanishing points on urban scenes.

The feature-based categorization has shown that there are a wide variety of features used by the state of the art methods, from global features like Mutual Information to local features like corners, straight lines or roofs. Deep learning techniques are also used to learn displacement between depth and RGB images.

The work presented in this thesis aims to estimate the transformation between a 2D image and a 3D point cloud without any prior information on the scene. Two complementary approaches are explored, having into account the explored methods in the state of the art exploration: a keypoint-based approach and a geometrical approach.

The keypoint-based approach leverages the deep learning architectures on 2D-2D and 3D-3D registration to obtain a 2D-3D registration. A novel representation of an unorganized 3D neighbourhood into a regular 2D lattice is proposed. This representation allows using 2D-2D registration techniques in both the 3D-3D and 2D-3D domains.

The geometrical approach develops a low-level feature detection to establish anchor points for a refinement pipeline. A multimodal edge detection designed to have high cross-domain repeatability is developed. The proposed pipeline uses the edge detections to estimate the transformation using the pinhole camera model geometry.

9.1 Contributions

The work developed in the explored approaches has provided several relevant contributions. The keypoint-based approach, explained in Chapter 3 and evaluated in Chapters 6 and 8, has produced the following contributions:

- A representation of an unorganized 3D point cloud neighbourhood into an RGBD image with minimal information loss. This representation is successfully used to train a deep learning network to detect correspondences between 3D keypoints.
- The exploration of different network architectures and network structures to provide a correspondence matching between domains with minimal structural changes on the network architecture.
- The exploration of optimization techniques to define the transformation between a set of multimodal keypoints. The proposed 2D-3D optimization technique offers performances on the same levels as 3D-3D registration techniques without using depth information.

The edge-based approach has been explained in Chapter 4 and evaluated in Chapters 7 and 8. The relevant contributions on this part of the thesis are the following:

- A multimodal edge detection algorithm This detector obtains coherent edges between 2D images and 3D point clouds combining intensity and geometrical information. The performance of this algorithm is up the state of the art algorithms on 2D-2D and 3D-3D domains and exceeds the state of the art algorithms on the 2D-3D domain.
- An iterative refinement technique that obtains the transformation between a 3D point cloud and a 2D image. This technique offers performances surpassing the state of the art algorithms in the 3D-3D domain. It also obtains similar performances on the 2D-3D registration despite not having depth information on the image.

These contributions have produced two relevant publications. [71], published in 2019, details the proposed representation of the 3D point cloud into a 2D patch and its usage on the 3D-3D correspondence estimator. [72], published in 2017, details the multimodal edge detection and 2D-3D registration.

9.2 Shortcomings and future work

The work produced in this thesis has obtained relevant developments in both the field of single modal and multimodal registrations. The algorithm proposed in Chapter 3 has shown that the 3D domain can be locally represented in 2D patches. It has also shown that existing deep learning structures can be successfully trained to detect correspondences using these patches.

This thesis has also proposed a multimodal edge detection algorithm with performances exceeding the state of the art algorithms. This edge detection is also successfully used to estimate 3D-3D and 2D-3D transformations.

In addition to these methods, this thesis also presents an in-depth study of the usage of 3D-3D transformation estimation techniques –RANSAC and ICP– to perform 2D-3D registration, obtaining similar performances to the 3D-3D registration algorithms.

However, there has been one main shortcoming on the proposed algorithms: the performance of the correspondence detector in the 2D-3D domain. This thesis proposed a domain-agnostic registration method pipeline using the same structures to detect both single modal and multimodal correspondences. These structures have shown to detect correspondences on both 2D-2D and 3D-3D domains correctly. However, the performance on the 2D-3D domain is not good enough to correctly estimate the transformation between two frames.

The evaluation shown in Chapter 6 has shown that the 2D-3D correspondence matching algorithm obtained a significantly higher error rate than the 3D-3D correspondence matching algorithm. This performance has been proven to be low enough not to obtain a proper amount of correspondences when performing the registration.

These results give a wide exploration field in the 2D-3D registration domain. The first exploration path to be taken into account is the improvement of the proposed 2D-3D correspondence estimation algorithm.

The results obtained on single modal 2D-2D and 3D-3D domains have shown that the proposed patch representation provides enough discriminative features in their respective domains. Therefore, the lack of performance of the 2D-3D registration can have two main factors: the inability of the proposed network pipeline to estimate to detect cross-domain discriminative features or the lack of those cross-domain discriminative features in the proposed patches.

The supposition that the deep learning pipeline causes the low performance of this algorithm leads to the exploration of different structures and hyperparameters set to perform the registration. Albeit an extensive exploration of several network structures and training hyperparameters has already been performed, a more extensive test with a wide range of network structures and optimization algorithms may lead to a performance increase.

The low performance of the algorithm can also be caused by a lack of cross-domain features on the proposed patch representation. The experiments performed in this thesis have shown that the projection plane in which the 3D neighbourhood is represented has a significant influence on the performance. Therefore, techniques that provide a very coarse initial alignment could be used to define the projection plane.

In this regard, the usage of Supervoxels[64] and Ultrametric Contour Maps[3] to establish an initial alignment was explored in the early stages of this thesis. However, the development of these methods was abandoned to focus on the more promising deep learning pipeline. The obtained results on this thesis hint that this field may be worth exploring.

Different representations of the 3D domain can also be explored. It is of especial relevance the Graph Neural Networks (GNNs) structure[**?**], able to detect features directly on the unorganized point cloud structure. A hybrid pipeline between the GNN architecture and traditional CNNs is an ambitious proposal that may lead to successful results.

One of the most relevant developments that would benefit all approaches, however, is the usage of large-scale databases to train the different structures. The database

used in this thesis is relatively large, but when compared to the large-scale image databases, it can be seen that it has a narrow variety of scenes and sensors. The capture of a large-scale point cloud database would also be an interesting achievement to improve on the detection of global features on the point clouds.

A

Experimental setup

This thesis presents a set of experiments to evaluate the different aspects of the proposed algorithms. This Appendix explains the specific environment on which this experiments are carried out.

The content of this appendix is divided into three main sections: the datasets used in the evaluation process, the measures used on each part and their characteristics, and the hardware and software specifications used on the different experiments.

A.1 Datasets

This section details the different aspects of the data used in this thesis to develop and test the proposed algorithms.

The first part of this section details the different data sources used in this thesis and their characteristics. The second part of this section details the specific databases built from these data sources, used in the tests presented in this thesis.

Figure A.1: Examples of the captured data on the Image Processing Group smart room

A.1.1 Data sources

This thesis has used data from five different sources during the development of this thesis: manual captures, Stanford Bunny, CoRBS, TUM RGB-D SLAM Dataset and 3DMatch dataset. In this section, the characteristics of each one are detailed.

Manual RGBD captures

These captures are performed in the Image Processing Group smart room. A handheld KinectV1 is used to capture several sequences with textured data. These captures have no location ground truth information and therefore, are only used for qualitative evaluations.

The captured sequences are:

- A global sequence of the smart room.
- A sequence with a still person standing in the middle of the room.
- A sequence with a three-faced cube with a checkerboard located in one of its faces.
- Multiple sequences in which a corner of the room where a board game and textured objects were present is captured.

Some examples of the captured data can be seen in Fig. A.1.

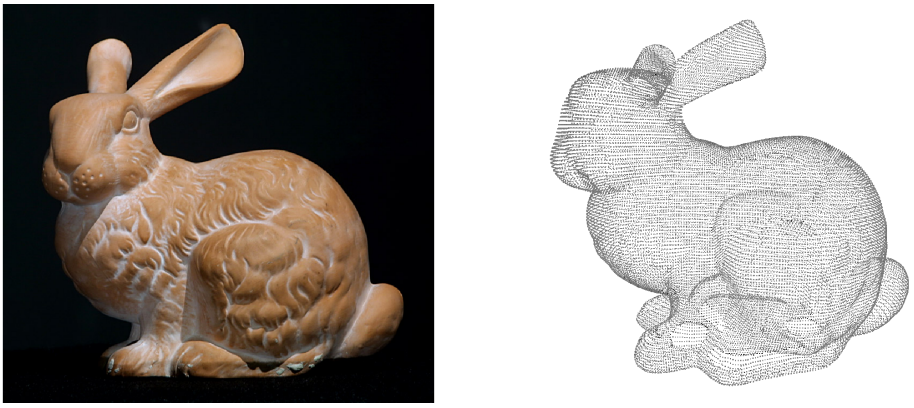


Figure A.2: Views of the image and 3D point cloud of the Stanford Bunny

Stanford bunny

The Stanford Bunny is a largely used 3D scan of a ceramic bunny perfomed by the Stanford University Computer Graphics Laboratory. The scan was performed using a Cyberware 3030 MS taking 10 captures with a total of 3632272 points.

The version used in this thesis is the reconstruction version using the zipper method, which contains 35947 points. The points are regularly sampled.

In addition to the 3D scan, an RGB image of the Stanford Bunny is also available. This image, taken 10 years later, has some minor differences to the scan. The 3D point cloud and the 2D image are used to test the quality of the 2D-3D registration process qualitatively. However, since there is no ground truth transformation, the quality of this registration is not numerically evaluated. The RGB image and a rendering of the 3D capture can be seen in Fig. A.2.

CoRBS: Comprehensive RGB-D Benchmark for SLAM using Kinect v2

This dataset provides several sequences of RGBD images captured using a Kinect V2. In addition to the RGBD information, the ground truth camera location for each frame is also available. The captures are performed on four different scenes: `human`, `desk`, `electrical cabinet` and `racing car`. Several sequences are captured on each sequence.

```
avg_period = np.average(d_tstamps[101:201]-d_tstamps[100:200])
rgb_idx = []
depth_idx = []
location_idx = []
for i, tstamp in enumerate(d_tstamps):
    min_rgb = np.argmin(np.abs(rgb_tstamps-tstamp))
    min_location = np.argmin(np.abs(loc_tstamps-tstamp))
    if np.abs(rgb_tstamps[min_rgb]-tstamp) < avg_period and \
            np.abs(loc_tstamps[min_location]-tstamp) < avg_period:
        rgb_idx.append(min_rgb)
        depth_idx.append (i)
        location_idx.append(min_location)
```

Listing 1: Algorithm used to establish matches between timestamps.

The data given by the authors consists of three lists of timestamped data: one for the colour images, one for the depth and one for the camera locations. Before working with this data, the three sources need to be aligned. To perform this aligned a procedure similar to the one proposed in the TUM dataset is used.

The code used to perform the alignment can be seen in Listing 1. Using this algorithm each depth frame is matched with their closest RGB and location.

Two captures have been used during the development of this thesis: the capture `D1` of the `desk` scene and the capture `E1` of the `electrical cabinet` scene. These sequences have been used in the initial testing stages of the 2D-3D correspondence matching algorithm to verify that the trained parameters are universal between different scenes and sensors. Specifically, the first 100 frames of each sequence have been used in this test.

Some visual captures of the two sequences used can be seen in Fig. A.3.

TUM RGB-D SLAM Dataset

This dataset contains several sequences of a large number of scenes. Like the CoRBS dataset, the data of this dataset is also represented by three sequences of timestamped data. To register the three sources –RGB, depth and location– the same

Figure A.3: RGB images of the the `desk` sequence(left) and the `electrical cabinet` sequence (right) of the CoRBS dataset.

approach is used.

A subset of this sequences have been used in several stages of the development of the algorithms presented in this thesis. This subset has been selected having into account the texture of the presented scenes and discarding the calibration scenes and the scenes with dynamic objects. This subset contains frames from the following sequences:

- Train split: `f1/plant` (235 frames) , `f2/flowerboquet` (2070 frames), `f1/desk` (595 frames), `f1/room` (1360 frames)
- Validation split: `f2/dishes` (2105 frames), `f3/cabinet` (218 frames), `f3/teddy` (1440 frames)
- Test spit: `f1/teddy` (718 frames), `f2/metallicsphere` (1356 frames)

This selected subset gives a total of 4260 training frames, 3763 validation frames and 2074 test frames.

It should be noted, however, that the initial split used in the work presented in [71] contained only the `f1/desk` sequence split between the train and test data, and the first 100 frames of the `f1/teddy` sequence as test data.

The data is given in the same format as the CoRBS dataset: three lists of timestamped data: one for the colour images, one for the depth and one for the camera locations. The same procedure explained in Section A.1.1 is used to obtain an aligned list of frames.

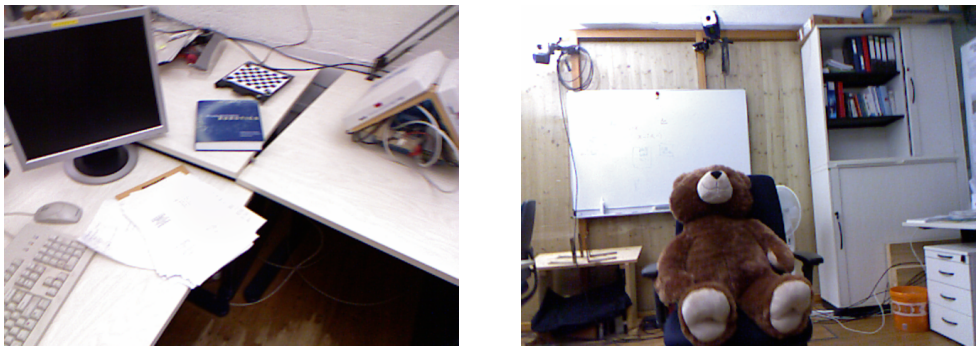


Figure A.4: RGB images of the the `f1/desk` sequence(left) and the `f1/teddy` sequence (right) of the TUM RGB-D SLAM dataset.

Some visual captures of the data presented in this dataset can be seen in Fig. A.4.

3DMatch Database

The dataset using in the main part of the results presented in this thesis is the dataset proposed in 3DMatch [111]. This dataset contains sequences from the SUN3D database [107, 32], Microsoft 7 scenes database [86], Stanford RGB-D Scenes Dataset v2 [44], BundleFusion [18], and Analysis by Synthesis [100]. the data provided by 3DMatch [111] has already aligned the RGB, depth and location information.

Two sources are discarded in this thesis. The Analysis by Synthesis sequences contain only depth information, and the colour and depth images of the Microsoft 7 scenes sequences are not aligned.

The same train/test split used in 3DMatch is used in all the train/tests presented in this thesis. However, four sequences have been removed from the train split an used as a validation split: `sun3d-mit_w20_athena-sc_athena_oct_29_2012_scan1_erika`, `rgbd-scenes-v2-scene_13`, `rgbd-scenes-v2-scene_14` and `bundlefusion-office3`.

Some visual captures of the data presented in this dataset can be seen in Fig. A.5.

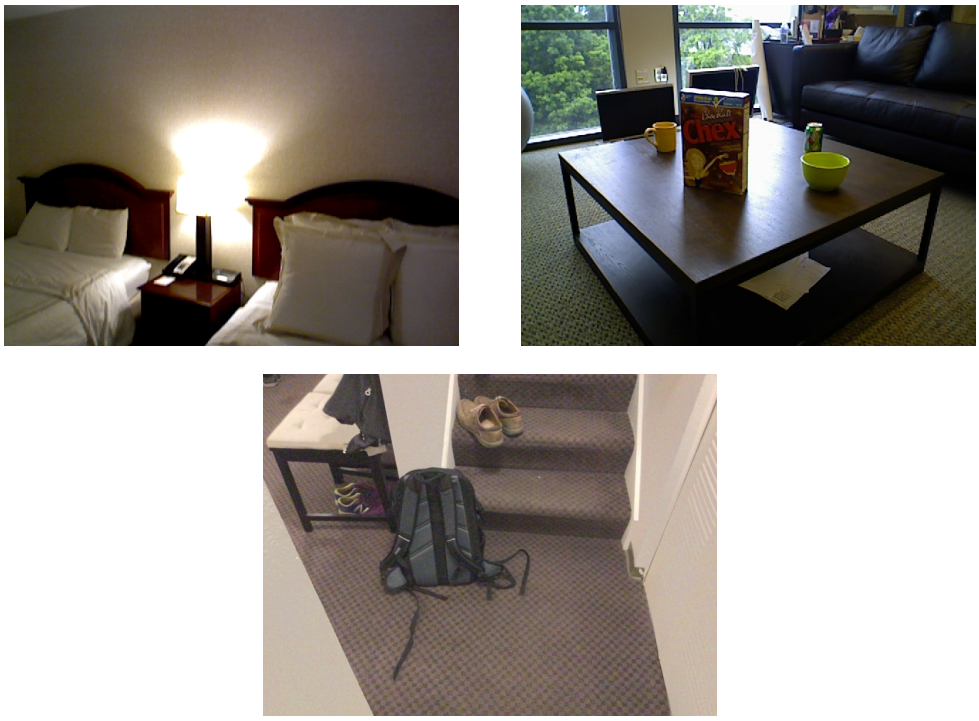


Figure A.5: Examples of the RGB frames present in the SUN3D dataset, Stanford RGB-D Scenes dataset and BundleFusion

A.1.2 Correspondence matching dataset

A 2D-3D correspondence matching algorithm is developed in the first part of this thesis. A correspondence dataset is built using pre-registered frames to develop and test this algorithm.

In order to train the CNN, a balanced set of matches and non-matches is extracted from the registered frames in a sequence. These sequences must fulfil two requisites: have the colour image calibrated to the depth image and have the ground truth transformation for each frame. This will allow representing the frames in a sequence as a set of coloured point clouds with a common reference frame.

A random point p is selected from a random frame in the sequence to build the matching set. Another random frame form the same sequence is gathered. If there is a point q in this frame that is no further than a certain threshold from p, the pair p, q is selected as a matching pair.

The simplest approach to select the non-matching pairs is to grab two random

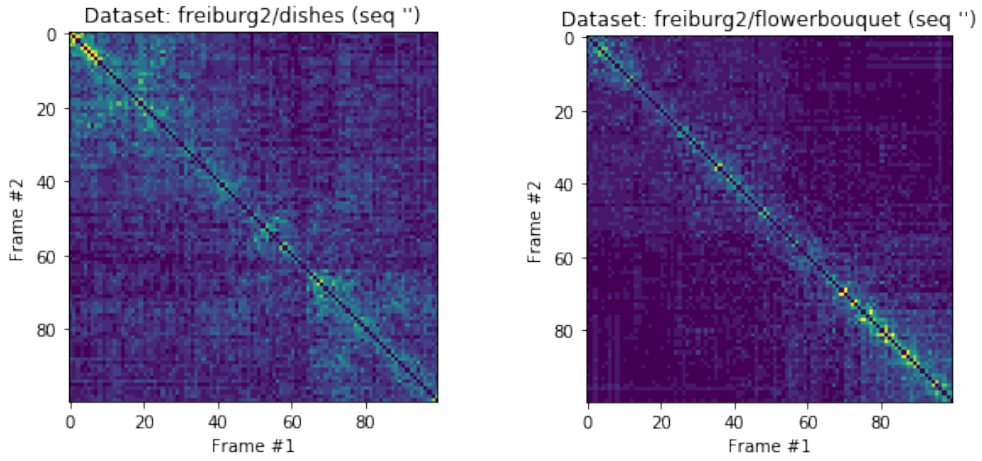


Figure A.6: Percentage of overlapping points between each frame in the first 100 frames of the `f2/dishes` and `f2/flowerbouquet` sequences.

points p', q' from two random frames –and check that they are further apart than the defined threshold. However, this approach can present certain issues depending on certain characteristics of the sequences.

The sequences presented in both the TUM and 3DMatch data sources are captured by moving a handheld sensor around a scene. This provides a non-random overlapping between the different frames. As an example, Fig. A.6 shows the percentage of overlapping points between the first 100 frames of the `f2/dishes` and the `f2/flowerbouquet` sequences.

As can be seen, each frame only has a significative overlapping between its neighbouring frames. Therefore, the matching points set will be strongly biased to have the same distribution –only containing matches from neighbouring frames.

However, the simplest approach to select non-matching frames does not have this distribution, and it instead contains non-matching pairs from all possible frames. It has been observed that this bias can offset the learning process of the CNN. Some of the early tests performed showed that the network learned to distinguish between overlapping and non-overlapping frames instead of matching and non-matching points.

The approach followed in this thesis to solve this issue is to select a non-matching pair p', q' for each frame pair a matching pair p, q has been acquired. This approach deletes the effect of the overlapping bias in the training dataset.

This approach can even be further constrained to select the same point p for both the matching pair p, q and the non-matching pair p, q'. This approach is similar to the approach followed when using a triplet loss, where each point p is paired with a matching point q and a non-matching point q', getting a triplet p, q, q'. It should be noted that this approach is not immune to the distance frame bias if q and q' are selected from different frames.

The results presented in this thesis use two different databases, one extracted from the TUM sequences and one using the 3DMatch sequences.

The TUM dataset is used in two scenarios: the keypoint selection evaluation and time performance benchmarks. The small size of the sequences has allowed computing costly keypoint detectors in a reasonable time frame easily. Since the database size and content are not relevant to the time performance benchmarking this database is also used to perform these tests.

The correspondence matching dataset extracted from the TUM sequences is divided into three splits: train, validation and test; as discussed in Section A.1.1 of this Appendix. The train split contains 50.000 matching points and 50.000 non-matching points. The validation and test split both contain 5.000 matching points and 5.000 non-matching points.

The 3DMatch dataset is used in the remaining tests presented in this thesis. The authors of the 3DMatch paper had already divided the dataset into a train and test split. They also provide a pre-made test set of 5.000 matches and 5.000 non-matches to perform the evaluations. However, among these pairs, there are pairs from the Microsoft 7 scenes dataset. As discussed in Section A.1.1 of this Appendix, the RGB and depth images provided in this dataset are not aligned. Therefore, the pairs extracted from this dataset are discarded, going from a total of 10.000 testing points to 8000 testing points.

The remaining train sequences for this dataset are split into train and validation splits, as described in Section A.1.1. A correspondence dataset with 100.000 matching pairs and 100.000 non-matching pairs on the train split, and 10.000 matching

pairs and 10.000 non-matching pairs on the validation split is built. This dataset is used for most of the presented tests in this thesis. For the final tests, however, an extended train split version with 250.000 matching pairs and 250.000 non-matching pairs is built.

A.1.3 Registration dataset

A frame pair dataset is built to evaluate both the edge detection algorithm presented in the second part of this thesis and the final registration.

The state of the art method in 3D-3D registration used as a baseline in this thesis [111] uses a 3D cloud correspondence ground truth by combining frames from the original sequences. The clouds used in this database are built by combining blocks of 50 frames using the Truncated Signal Distance Function (TSDF). The TSDF is a scalable RGBD integration algorithm that allows combining data from several RGBD frames into a single point cloud with a constant sampling distance.

The given test dataset contains around 60 TSDF clouds for each sequence, therefore spanning the first 300 frames of the sequence. The pair matching database is establishing by selecting the cloud pairs that fulfil two conditions: have at least a 30% overlap between both clouds and be at least two clouds apart (=100 original frames).

The work presented in this thesis aims to perform registration between a 2D image and a 3D point cloud. Therefore, whist the same technique could be applied to the 3D point clouds involved in the registration; there is not a similar technique that allows the combination of several 2D images.

Not having the ability to combine several 2D images, therefore, does not allow to use the same database used in the state of the art work [111]. The parameters of the state of the art database, however, are evaluated in order to evaluate the proposed algorithms using a database with a similar level of difficulty. The evaluated measures are the rotation angle along the main axis, the magnitude of the translation vector,

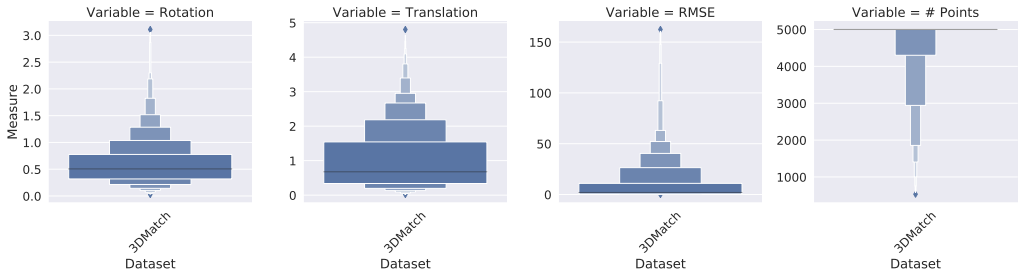


Figure A.7: Evaluation of the different parameters of the 3DMatch [111] registration dataset: rotation, translation, RMSE and # of points.

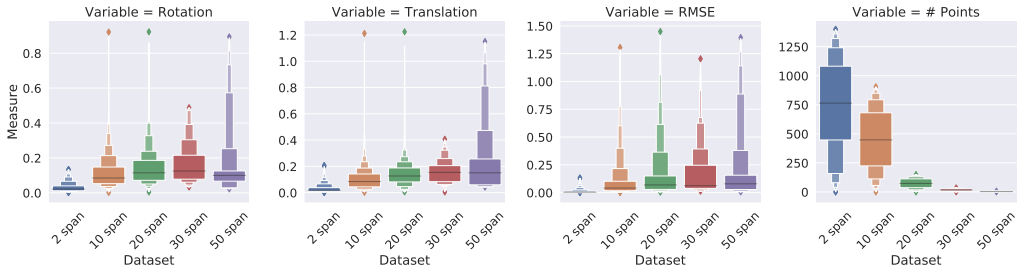


Figure A.8: Evaluation of the different parameters on the built databases with 2, 5, 10, 20 and 50 frame skip: rotation, translation, RMSE and # of points.

the RMSE and the # of matching points between frames. The obtained results can be seen in Fig. A.7.

The alternative used to build a database is to use the original frames as the registration frames. Albeit the combined clouds can be used in the 3D part of the 2D-3D registration, this is discarded to avoid having an uneven number of points between the 2D and 3D data that can skew the overlapping measurement.

Different databases are built that force different minimum frame spans to provide different levels of difficulty: 2 frames, 5 frames, 10 frames, 20 frames and 50 frames. A database with at least a 100 frame skip between pairs was tried to be built, but no frames were found to have at least a 30% overlap with this span.

These databases are also evaluated using the same measures: the rotation angle along the main axis, the magnitude of the translation vector, the RMSE and the # of matching points between frames. The obtained results can be seen inFig. A.8.

Comparing the graphics in Fig. A.7 andFig. A.8 it can be seen that none of the proposed datasets has similar charactersitics to the 3DMatch dataset in all the

measured parameters.

First of all, it can be seen that both the rotation, translation and RMSE are higher in the 3DMatch dataset than in any of the proposed datasets. However, there is a greater difference in the number of overlapping points: whilst in the 3DMatch dataset, the average of overlapping points between two frames is around 5000 points, in all the proposed datasets there are less than 1000 points on average in the overlapping area. This value further decreases when the minimum span between frames increases, having an average span of fewer than 250 points with spans of 20 frames and further.

Having into account that there is not a great difference between the different frame spans in terms of rotation, translation and RMSE a minimum frame span of 10 frames is selected for the results presented in this thesis.

A.2 Evaluation metrics

In this section, the different evaluation metrics used in this thesis to evaluate the different parts of the proposed algorithm are detailed.

A.2.1 Transformation estimation

The chosen metric to evaluate the estimated transformation is the metric developed in [15], also used in [111]. This metric measures the error of a transformation T_{ji} as the location error between ground-truth correspondences in terms of the Root Mean Squared Error (RMSE), as defined in Eq. (A.1). The recall metric is computed using this condition to define a correct transformation*.

$$E^2_{RMSE} = \frac{1}{\mathcal{K}^*_{ij}} \sum_{(p^*,q^*)\in\mathcal{K}^*_{ij}} \|p^* - T_{ji}q^*\|^2 \leq \tau^2 \tag{A.1}$$

*Extracted from http://redwood-data.org/indoor/registration.html

To define the correspondences $\mathcal{K}_{ij}^{*} = \{(p^{*}, q^{*})\}$,[15] considers that two points that are close than $7.5cm$ are a corresponding point, and that two points that are further apart than $7.5cm$ are a non-match. The threshold of the error metric E_{RMSE}^{2} in which a transformation is considered correct is $\tau = 0.2m$.

The distance threshold to consider a pair (p^{*}, q^{*}) a match or a non-match should be adequately tuned depending on the characteristics of the data being evaluated. The data used in in [15] is uniformly sampled with $5cm$ voxels. Therefore, in this case, the allowed error factor is $1.5\times$ the voxel size. However, the overall size of the scene and sensor limitations must also be taken into account when defining the allowed error factor.

In the final registration process, the same data source used in [111] is used. However, as explained on Section A.1.3, there is a core limitation on the 2D-3D registration: two 2D frames cannot be seamlessly combined into a 3D frame, and therefore the same procedure cannot be applied. Nevertheless, the evaluations performed in Section 8.1 show that a the same threshold of $0.2m$ on the RMSE value provides an accurate representation of the transformation estimation on the 3D-3D domain.

However, this evaluation metric only takes into account the Euclidean distance between the original points and transformed points. In a 2D-3D scenario, in addition to valuing this distance, other metrics may be of interest, such as the projection error on the 2D plane or the reprojection error in the 3D space. To evaluate these errors, two new evaluation metrics, based on the distance measures minimized in the registration process, are proposed.

The first new proposed error metric measures the reprojection error in the camera plane as the 2D euclidean distance between the 2D target pixels and the transformed 3D points projected in the 2D plane. This error metric is shown in Eq. (A.2), where $d_e(p, q)$ represents this distance measure between a 3D point p and a 2D point q.

$$E_{RMSE,e}^{2} = \frac{1}{\mathcal{K}_{ij}} \sum_{(p,q)\in\mathcal{K}_{ij}} d_e(p, q) \leq \tau_e^{2} \tag{A.2}$$

The second proposed error metric measures the minimum distance between the line formed by the camera centre and the pixel in the camera plane and the transformed 3D point cloud. This error metric is shown in Eq. (A.3), where $d_p(p, q)$ represents this distance measure between a 3D point p and a 2D point q.

$$E^2_{RMSE,p} = \frac{1}{\mathcal{K}_{ij}} \sum_{(p,q)\in\mathcal{K}_{ij}} d_p(p, q) \leq \tau_p^2 \tag{A.3}$$

A.2.2 Correspondence matching

The main contribution of the first part of this thesis is the multimodal 2D-3D correspondence matching algorithm. A balanced set of matching/non-matching points is created on the testing dataset split to evaluate this algorithm. The accuracy of the algorithm is measured as the false positive rate at 95% recall ($FPR@95$). In some early tests, the area under the ROC curve (AUC_{ROC}) was also used. However, the $FPR@95$ metric is preferred to compare against state of the art methods [110, 111].

A.2.3 Edge detection

The main contribution in the second part of this thesis is the multimodal edge detection. In this part, the evaluation differs from traditional edge detection algorithms, in which the main goal is to obtain a set of thin lines that define contours in the scene.

In the algorithm proposed in this thesis, the accuracy on the line detection is not as important as to have the same detections on both sets to be registered. Therefore, the evaluation metric differs from the traditional edge detection algorithms, in which an edge ground truth is established.

To define a detection as a correct detection between two frames P and Q the frame Q is selected as a target. For each point q_j in Q, a small subset of all the points in P

is selected. This subset is selected with radius $< 7.5cm$, denoted $\{p_i\}$, to maintain coherence with the transformation estimation error.

If q_j is an activation and any point in $\{p_i\}$ is also an activation, the point is marked as a true positive (TP). If all points in $\{p_i\}$ are non-activations, the point is marked as a false negative (FN). Likewise, if q_j is not an activation and any point in $\{p_i\}$ is an activation, the point is marked as a false positive (FP). Finally, if both q_j and all points in $\{p_i\}$ are non-activations, the point is marked as true negative (TN). All points q_j that do not have any neighbouring points in P are discarded in the evaluation process.

In the scenario presented in this thesis a good edge detection algorithm will have a high repeatability –correct detections vs. all detections– whilst having a overall low amount of detections. This evaluation scenario is similar to the repeatability evaluation in keypoint detection algorithms. To evaluate the repeatability in keypoint detectors the ratio of correctly detected point vs. the overall amount of detections is computed [97, 80] as shown in Eq. (A.4).

$$r = \frac{TP}{TP + \min(FN, FP)} \tag{A.4}$$

However, this evaluation metric is not relevant in extreme cases [24]. If one capture has only a single detection, matching this detection agains a detection on the other set yields a non-relevant $r = 1$. Moreover, marking all the points as keypoints also yields $r = 1$.

In keypoint detection algorithms, these extreme cases are solved using a non-maxima suppression, obtaining a regular and distributed set of keypoints [116]. However, in the edge detection algorithm applied in this thesis, this technique is not applicable.

Therefore, the quality measure used favors both having a similar amount of keypoints on both data and having an overall low amount of detections. This quality measure is defined in Eq. (A.5) as the ratio between the repeatability r and the ratio of detections d, defined as shown in Eq. (A.6).

$$S(\lambda) = (1 - \lambda)r + \lambda\frac{1}{d} \tag{A.5}$$

$$r = \frac{TP}{TP + \min(FN, FP)} \qquad d = \frac{TP + \max(FN, FP)}{TP + FP + TN + FN} \tag{A.6}$$

The parameter λ allows to tune the relevance of the accuracy of the detections and the overall number of detections. The optimization performed in Chapter 7 is done using a conservative value of $\lambda = 0.5$. This parameter, however, can be tuned epending on the needs of the algorithm.

A.3 Hardware and software

The different algorithms proposed in this thesis are mainly programmed using python [101][†]. The main libraries used are `numpy` [62], `scikit` [67] & `scikit-learn` [67] –for numerical operations–, `open3D` [117] –to work with 3D point clouds–, `opencv` [11] –to work with images–, `pytorch` [65] –to train the deep learning models– and `numba` [45] & `cython` [5] – to speed up the computation of critical paths. In addition to this libraries, other helper tools are used to visualize data –`jupyter` [42], `matploltib` [37]–, for storage –`h5py` [16], `pyyaml`, `tad4bj`– and to orchestrate trainings –`ray tune`–. Older versions of some algorithms are programed in C++ [94] using `opencv` [11] for images and `PCL` [77] for point clouds.

Regarding the execution environment, the executions are mainly performed in the Image Processing Group server cluster. This cluster contains several heterogeneous machines with both Pascal and Turing GPU's that have been changing during the development of this thesis.

The profiling tests presented on this thesis, however, are performed in an Nvidia

[†]The main code of this thesis is available at https://github.com/imatge-upc/multimodal-registration

Jetson TX2 development kit [61]. This environment is selected due to its standard configuration and ability to replicate the tests consistently.

The main technical specifications from this board are (extracted from [61]) :

GPU 256-core NVIDIA Pascal™ GPU architecture with 256 NVIDIA CUDA cores

CPU Dual-Core NVIDIA Denver 2 64-Bit CPU
Quad-Core ARM® Cortex®-A57 MPCore

Memory 8GB 128-bit LPDDR4 Memory
1866 MHx - 59.7 GB/s

Storage 32GB eMMC 5.1

Power 7.5W / 15W

B

Experiments

This appendix contains a comprehensive list of all the relevant experiments performed and explained in this thesis. A significant portion of these experiments are referenced in Part II. However, this appendix focuses on the description of the exact procedure, raw data and environment in which the different experiments are performed.

The next sections detail the different experiments performed. These experiments are sorted following the same order used in Chapters 6 to 8.

B.1 Keypoint analysis

The different keypoint detectors are analysed from three different viewpoints: their computational performance, their repeatability and their specificity.

B.1.1 Computational performance

Four different keypoint detector procedures are evaluated:

Random(2D) Detect 500 random keypoints in a image.

Harris[33](2D) Detect Harris keypoints in a image with the following parameters: neigbourhood size:2, sobel aperture:3, K:0.05, threshold: 0.02*max().

Random(3D) Detect 500 random keypoints in a point cloud.

ISS[116](3D) Detect ISS keypoints in a image with the following parameters: salient radius: 0.036, non maxima supression: 0.192, normal radius: 0.024, border radius: 0.006, gamma21: 0.975, gamma32: 0.975, min. neighbours: 3.

The details on the meaning of the different parameters on the keypoint detectors can be found in their respective sources.

The code for both random and Harris detector is in Python using OpenCV, while the code for the ISS detector is done in C++ using PCL. The analysis is done by computing each keypoint detector on 50 different captures and averaging the execution time.

This test is performed in an Nvidia Jetson TX2 development kit [61], which specifications can be found in AppendixA.3.

The results obtained are shown in the following table. This table is shown in Chapter 6 as *Table 6.1: Execution time of different keypoint detector methods to compute the keypoints for a single frame. The relative time between methods (ratio) is also shown.*

Method	Keypoint detector	Execution time (s)	Relative execution time
Random	2D	1.54E−04	1 ×
Harris	2D	1.54E−02	100 ×
Random	3D	5.24E−04	1 ×
ISS	3D	1.67E+02	319196 ×

B.1.2 Repeatabilty

Two different analyses are performed to compute the repeatability of the different keypoints. First of all, a computation using the same parameters specified above for each keypoint detector is performed. A set of 100 frame pairs having at least a 30% overlap is selected from the TUM dataset. The keypoints are computed for each frame in the pair using the three detectors being analysed (Harris, ISS, Random).

The repeatability is measured as the # of matched keypoints vs the minimum amount of detections on both frames. The points that are outside the overlapping area are discarded. The following table shows the repeatability results when using each possible combination of keypoint detector on each possible frame.

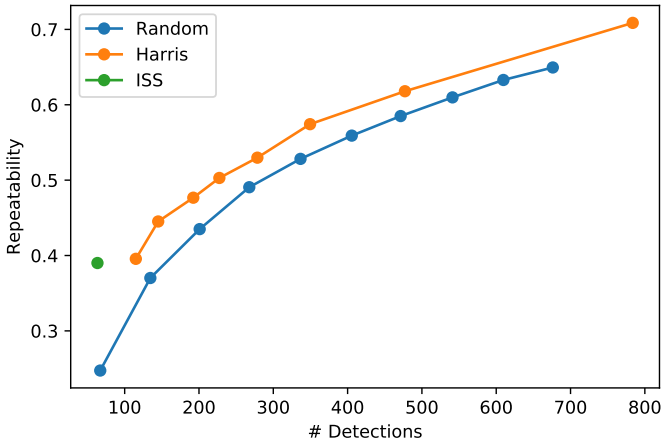
	Harris	ISS	Random
Harris	0.71	0.17	0.19
ISS	0.17	0.31	0.43
Random	0.22	0.44	0.47

However, the absolute number of keypoints detected in each case can affect the repeatability. In the results shown in the previous table, the average of detections for each keypoint detector was ISS 81, Harris 582 and Random 374.

It should be noted, in the random procedure, that the cause for having less than 500 keypoints is that the keypoints are selected on the 2D domain, and those points that do not have a valid depth value are discarded.

The repeatability is also computed using different thresholds –using the same keypoint detector on both frame pairs to counteract this effect. Those results are also obtained using 100 frame pairs. Only a single threshold is evaluated in the ISS keypoint detector due to its large computation time.

The results obtained are shown in the following figure. This figure is shown in Chapter 6 as *Fig. 6.2: Graphic showing the repeatability of several keypoint detectors depending on the number of detections. The repetability of the ISS detector is only computed for a single number of detections due to its high execution time.*



B.1.3 Specificity

The specificity of the different keypoint detectors is measured by training three different CNNs to discern if two keypoints are a match or a non-match.

The keypoint pairs databases are build using the different keypoint detectors to perform this test. Each database contains 100.000 matching keypoints on the training split. For each matching pair, an additional non-matching pair is computed form the same frames, given a total of 200.000 pairs. Likewise, a test dataset with 10.000 matching pairs and 10.000 non-matching pairs is built using the same approach with sequences not used on the training split. The dataset used in this experiment is the TUM dataset.

The three tested CNNs use the state of the art layer structure Zagoruyko[110]. This structure, referred in Chapter 3, has the following layers: U(64) - C(1->96,k=7,s=3) - ReLU - P(k=2,s=2) - C(96->192, k=5, s=1) - ReLU - P(k=2,s=2) - C(192->256, k=3, s=1) - ReLU.

Three different patch selectors are used form the three different data domains explored in this thesis: 2D patches, 3D patches (intensity channel) and 3D patches (depth channel). The parameters used in each case are the following:

3d-intens 16×16 patch with radius 0.1, intensity channel only

3d-depth 16×16 patch with radius 0.1, depth channel only

2d 64×64 cropping patch scaled to a 64×64 patch.

The following training parameters are used to train the network: batch size 256, learning rate 0.001, weight decay 0.005, epochs 200, loss function 'ContrastiveLoss' and optimiser 'adagrad'.

The obtained results measured in FPR95 can be seen in the following table. The best performing keypoint detector for each data type is highlighted. This table is shown in Chapter 6 as *Table 6.2: Specificity (in FPR95) when using several keypoint detectors to extract three different features (lower is better).*

Keypoint detector	2D	3D(intensity)	3D(depth)	2D/ 3D(intensity)
Random	**0.694**	**0.473**	**0.701**	0.775
Harris	0.827	0.829	0.744	**0.764**
ISS	0.843	0.668	0.805	0.870

B.2 Patch building

The different parameters of the 2D and 3D patch generators are evaluated in this experiments block.

The first tests are devoted to the evaluation of the 2D patch detection algorithm. The performance using different cropping sizes and output sizes is explored, as well as the proposed modifications. The execution time for different parameter values is also computed.

The second tests are devoted to the evaluation of the 3D patch detection algorithm. The performance using different neighbourhood radiuses and output sizes is explored in three different scenarios: using only the intensity channel, using only the depth channel and using both channels. The execution time in different cases is also computed.

Finally, a test combining 2D and 3D patch detections is performed. Due to its low performance, an additional synthetic test with ground truth generations is also performed.

B.2.1 3D patch radius/size

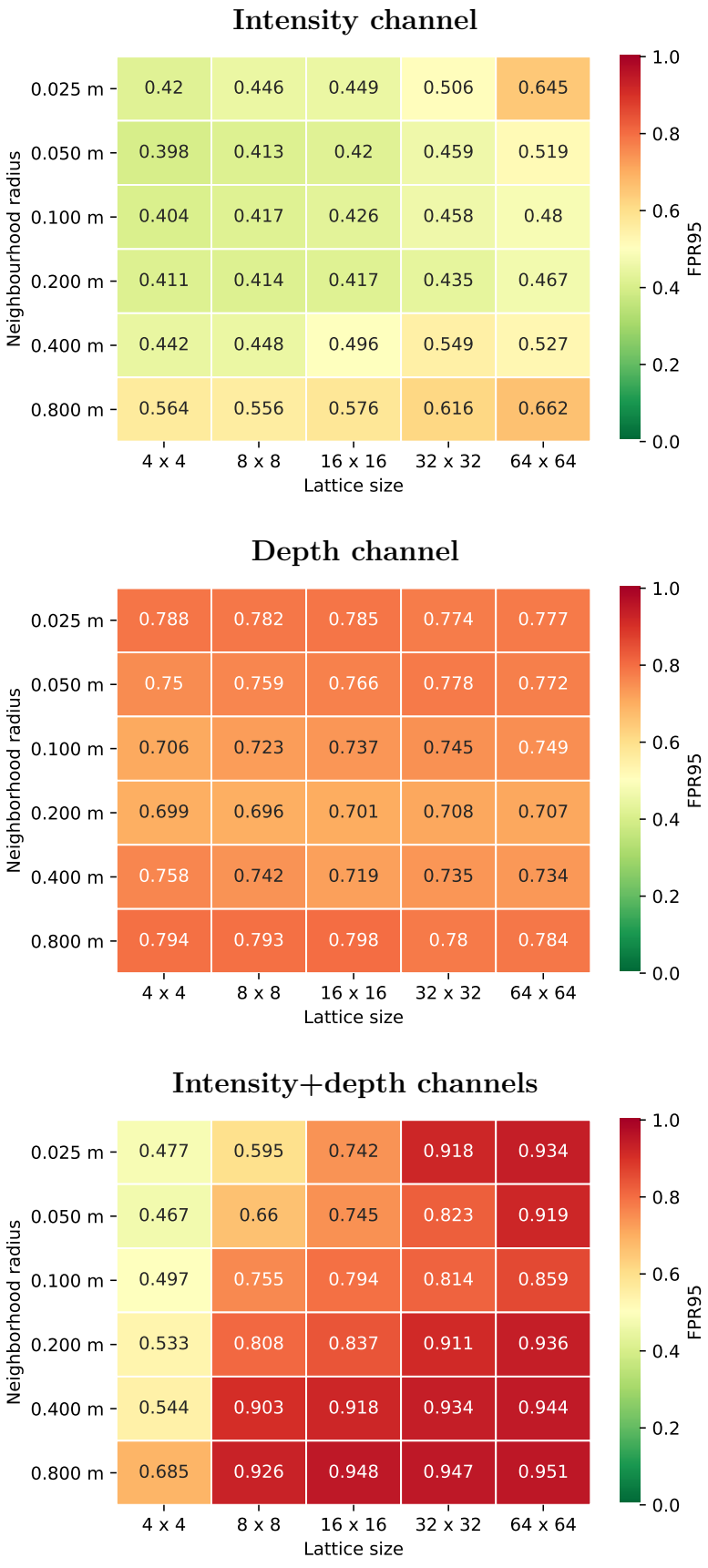
Likewise to the 2D domain, the proposed algorithm in the 3D domain allows tuning the radius and outsize to the optimal value. A grid on different radius and outsize values is tested to find the optimal radius/outsize combination. This experiment is also performed using random keypoint detections on the 3DMatch dataset, building a dataset with 200000 train pairs and 20000 validation pairs, using the same procedure described before.

Unlike the 2D domain, the patches obtained in the 3D domain have two different channels, one with the intensity values and one with the depth values. In this experiment, three different tests are performed: one for the intensity image, one for the depth image and one using both channels.

In the intensity and depth only tests, a siamese network is trained for each combination using the Zagoruyko[110] structure. This network structure, however, is not suitable to work with multi-channel images. The network structure proposed in Chapter 3 is used to provide results using both the intensity and depth channel.

The different parameters used in the training process are batch size 256, learning rate 0.001, weight decay 0.005, epochs 100, loss function 'ContrastiveLoss' and optimiser 'adagrad'. These parameters have been optimised using the Hyperband [47] algorithm.

The following figure shows the FPR95 value obtained in the different tests. This figure is shown in Chapter 6 as *Fig. 6.3: FPR95 values for different radius and ouptut sizes using the different input channels.*


Intensity channel
Neighbourhood radius
0.025 m
0.050 m
0.100 m
0.200 m
0.400 m
0.800 m
0.42 0.446 0.449 0.506 0.645
0.398 0.413 0.42 0.459 0.519
0.404 0.417 0.426 0.458 0.48
0.411 0.414 0.417 0.435 0.467
0.442 0.448 0.496 0.549 0.527
0.564 0.556 0.576 0.616 0.662
4 x 4
8 x 8
16 x 16
32 x 32
64 x 64
Lattice size
FPR95
1.0
0.8
0.6
0.4
0.2
0.0
Depth channel
Neighborhood radius
0.788 0.782 0.785 0.774 0.777
0.75 0.759 0.766 0.778 0.772
0.706 0.723 0.737 0.745 0.749
0.699 0.696 0.701 0.708 0.707
0.758 0.742 0.719 0.735 0.734
0.794 0.793 0.798 0.78 0.784
Intensity+depth channels
Neighborhood radius
0.477 0.595 0.742 0.918 0.934
0.467 0.66 0.745 0.823 0.919
0.497 0.755 0.794 0.814 0.859
0.533 0.808 0.837 0.911 0.936
0.544 0.903 0.918 0.934 0.944
0.685 0.926 0.948 0.947 0.951

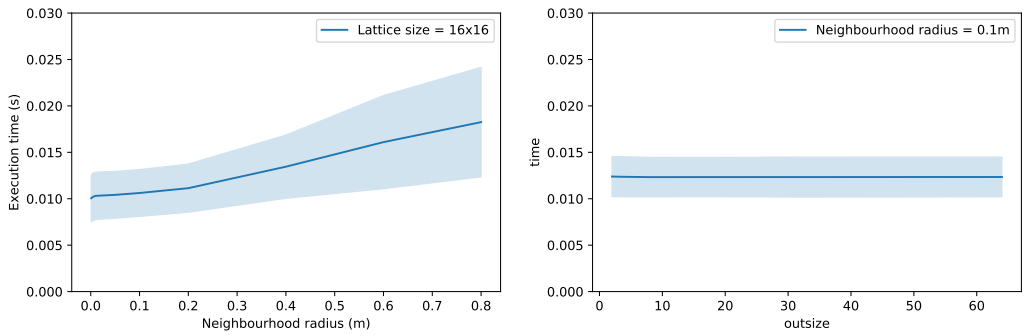
B.2.2 3D PATCH EXECUTION TIME

The execution time for the different parameters is computed. A set of 100 keypoints is randomly selected from frames of the 3DMatch dataset to perform this test. For each keypoint the specified patch is computed 10 times.

The first test evaluates the influence of the neighbourhood radius on the execution time. The cropping sizes from 0.001m to 0.8m are evaluated in this experiment. The next test evaluates the influence of the output patch size on the execution time. The output patch sizes from 2×2 to 64×64 are evaluated in this experiment. These tests are performed in both the intensity and depth channels.

This test is performed in an Nvidia Jetson TX2 development kit [61], which specifications can be found in AppendixA.3.

The results of both tests can be seen in the following figure. This figure is shown in Chapter 6 as *Fig. 6.4: Execution time for for different neighbourhood radius (left) and output sizes(right). The mean value is shown with a solid line and the greyed out area shows the variance of the measure.*



B.2.3 2D PATCH RADIUS/SIZE

The first experiment analyses the optimal patch size in both the cropping size and the output size. This experiment is performed using random keypoint detections on the 3DMatch dataset, building a balanced dataset with 200000 train pairs and 20000 validation pairs, using the same procedure described before.

A patch is built for each keypoint, and a siamese network is trained using the Zagoruyko[110] structure. The different parameters used in the training process are batch size 256, learning rate 0.001, weight decay 0.005, epochs 100, loss function 'ContrastiveLoss' and optimiser 'adagrad'. These parameters have been optimised using the Hyperband [47] algorithm.

The obtained results can be seen in the following figure. This figure is shown in Chapter 6 as *Fig. 6.5: FPR95 on different cropping patch sizes / ouptut sizes on the 2D patch generation.*

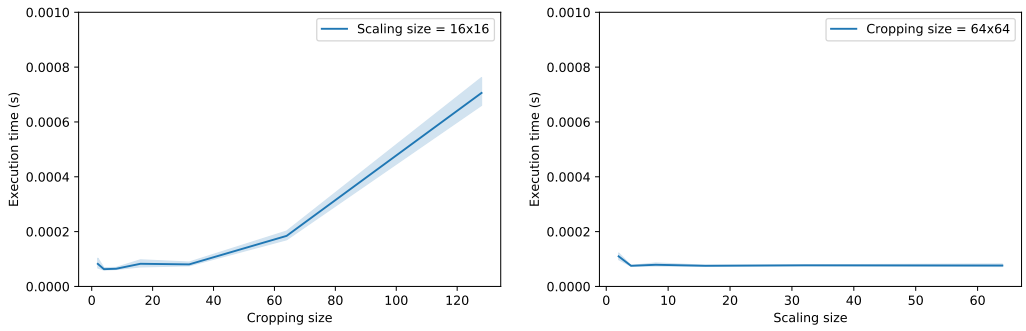
B.2.4 2D PATCH EXECUTION TIME

The execution time for the different parameters is computed. A set of 100 keypoints is randomly selected from frames of the 3DMatch dataset to perform this test. For each keypoint the specified patch is computed 10 times.

The first test evaluates the influence of the cropping patch size on the execution time. The cropping sizes from 2×2 to 256×256 are tested in this experiment. The next test evaluates the influence of the output patch size on the execution time. The output patch sizes from 2×2 to 64×64 are tested in this experiment.

This test is performed in an Nvidia Jetson TX2 development kit [61], which specifications can be found in AppendixA.3.

The results of both tests can be seen in the following figure. This figure is shown in Chapter 6 as *Fig. 6.6: Evaluation of the influence on the computation time of the cropping size and output size parameters of the 2D patch generation*



B.3 Network structures

The experiments describe in this section aim to compare the performance of different network structures when performing the 2D-3D correspondence matching. The tested network structures are Zagoruyko[110], VGG[88] and ResNet [34].

Although these experiments aim to find the performance of these network structures in the 2D-2D and 3D-3D registration pipelines, the effect of the patch generation parameters needs to be taken into account. In the previous tests, the optimal parameters for the Zagoruyko[110] network have been found. However, there is no guarantee that these parameters are also the optimal parameters for the VGG[88] and ResNet [34] networks.

Since the patch parameter analysis is not relevant in this section, the Hyperband algorithm is used to find the best patch parameter combination for each network. In addition to the hyperband algorithm, an additional two-step algorithm is used. First of all, the Hyperband algorithm is used to find the 5 best performing algorithms under a small correspondence matching dataset with 50000 training patches and 5000 validation patches from the 3DMatch dataset (see Appendix A.1.2). These 5 best-performing algorithms are then re-trained using the full dataset of 200000 training patches and 20000 validation patches.

The different parameters used in the training process are batch size 12, learning rate 0.001, weight decay 0.005, epochs 20, loss function 'ContrastiveLoss' and optimiser 'adagrad'. The batch size is reduced due to the memory available on the training GPUs (11G). The number of epochs is also reduced due to the faster convergence of these algorithms. The remaining parameters have been optimised using the Hyperband [47] algorithm.

The best performing combinations for each network structure on the 2D-2D registration scenario ca be seen in the following table. This table in shown in Chapter 6 as *Table 6.3: Best performing parameters for each network structure on the 2D-2D registration pipeline.*

Network structure	FPR95	Cropping size	Scaling size
Zagoruyko [110]	0.486	128×128	64×64
VGG [88]	0.555	64×64	64×64
ResNet [34]	**0.392**	128×128	64×64

The best performing combinations for each network structure on the 3D-3D registration scenario can be seen in the following table. This table in shown in Chapter 6 as *Table 6.4: Best performing parameters for each network structure on the 3D-3D registration pipeline.*

Network structure	FPR95	Neighbourhood radius	Lattice size
Zagoruyko [110]	0.398	0.05m	4×4
VGG [88]	0.406	0.4m	32×32
ResNet [34]	**0.341**	0.2m	8×8

B.3.1 Network execution time

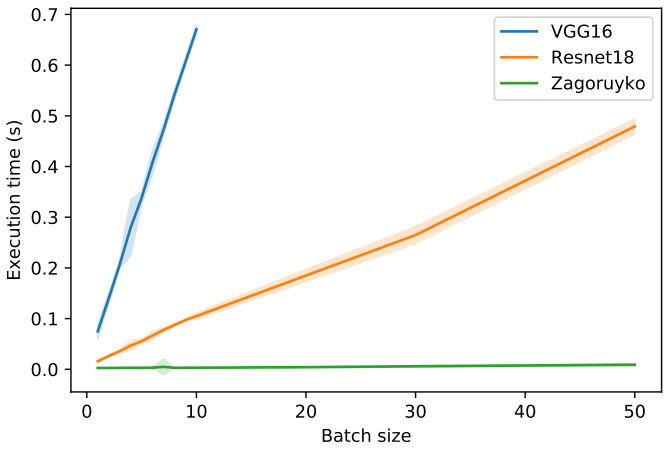
The execution time of the different network structures is tested on the validation stage. This test evaluates the execution time needed to forward a batch of patches through the feature extraction part of the network. The tested structures in this

experiment are the structures mentioned in Chapter 3: Zagoruyko[110], VGG[88] and ResNet [34].

Several batches with different batch sizes are created to perform this test. Each batch contains real 16×16 patches from the TUM dataset created from the point clouds using a $0.1m$ radius. The tested batch sizes are 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 20, 30 and 50. Each different experiment is repeated 25 times, and the time measured is averaged between 5 executions.

This test is performed in an Nvidia Jetson TX2 development kit [61], which specifications can be found in AppendixA.3.

The obtained results can be seen in the following figure. This figure is shown in Chapter 6 as *Fig. 6.7: Execution time (in seconds) of the evaluated network structures for different batch size. The mean value is marked in solid color, whereas the variance is shown as a translucid area.*



B.4 Multimodal 2D-3D correspondence matching

This test explores the optimal parameter set to perform the multimodal correspondence matching between image patches and cloud patches. In previous tests, the optimal parameters for both 2D-2D and 3D-3D registration have been found. These parameters are used as a baseline to find the optimal parameters to perform the 2D-3D registration.

Even when using only the best performing parameters on previous tests, the search space is too extensive to use a full grid search. Therefore, the same approach used in previous tests is applied. First of all, the Hyperband algorithm is used to find the 5 best performing algorithms under a small correspondence matching dataset with 50000 training patches and 5000 validation patches from the 3DMatch dataset (see Appendix A.1.2). These 5 best-performing algorithms are then re-trained using the full dataset of 200000 training patches and 20000 validation patches.

The trained network is a two-stream network in which both streams are trained independently. The different parameters used in the training process differ between structures: the Zagoruyko[110] is trained with batch size 256, learning rate 0.001, weight decay 0.005, epochs 200, loss function 'ContrastiveLoss' and optimiser 'adagrad', whilst the the ResNet [34] network is trained with batch size 8, learning rate 0.001, weight decay 0.005, epochs 50, loss function 'ContrastiveLoss' and optimiser 'adagrad'. These parameters have also been optimised using the Hyperband [47] algorithm.

The top combination for each network structure can be seen in the following table. This table is shown in Chapter 6 as *Table 6.5: Top combinations on the multimodal correspondence matching scenario.*

	Network	2D Patches		3D Patches	
FPR95		Crop	Scaling	Radius	P. size
0.646	Zagoruyko [110]	128×128	64×64	0.05m	4×4
0.593	ResNet [34]	128×128	64×64	0.2m	16×16

B.4.1 Synthetic Test on Multimodal Correspondence Matching

A synthetic test is performed to assess the influence of the different absolute size and projection plane on the correspondence matching performance. This test selects the three best configurations on the multimodal correspondence matching scenario and performs two modifications on the input data.

In the first modification, the image cropping size is artificially selected to match the 3D neighbourhood size on the real world. The second modification selects the 3D projection plane as the corresponding 2D image camera plane.

The obtained results can be seen in the following table. This table is shown in Chapter 6 as *Table 6.6: Errors obtained when applying synthetic modifications on the patch generation.*

Network	Original	Synthetic cropping	Synthetic projection plane
Zagoruyko [110]	0.646	0.716 (+0.070)	0.629 (−0.017)
ResNet [34]	0.593	0.553 (−0.010)	0.439 (−0.154)

B.5 Correspondence matching benchmarking

The final test on the first part of this thesis provides a benchmarking between the proposed algorithm and state of the art methods in 2D and 3D single-modal correspondence matching.

First of all, the proposed methods on 2D-2D, 3D-3D and 2D-3D correspondence matching are evaluated on the test split. The results shown in the following table reflect the test error obtained when performing the same tests using the best combinations:

Single modal 2D-2D Fast 128×128 cropping scaled to a 32×32 patch.

Single modal 3D-3D $0.2m$ neighbourhood radius projected in a 16×16 patch.

Multi modal 2D-3D 128×128 cropping scaled to a 32×32 patch (2D) / $0.2m$ neighbourhood radius projected in a 16×16 patch (3D).

This table is shown in Chapter 6 as *Table 6.7: Comparison between the validation and test errors on the best performing methods for each modality.*

Domain	Architecture	Validation error	Test error
Single modal 2D-2D	Fast	0.486	0.683
	Deep	**0.392**	**0.547**
Single modal 3D-3D	Fast	0.398	**0.502**
	Deep	**0.341**	0.564
Multi modal 2D-3D	Fast	0.646	0.739
	Deep	**0.593**	**0.692**

These results are compared with state of the art algorihtms in correspondence matching. This comparison can be seen in the following table.This table is shown in Chapter 6 as *Table 6.8: Comparison between different methods on 2D, 3D and multimodal 2D-3D correspondence matching. The FPR95 values for the different methods are shown.*

	2D-2D	3D-3D	Multimodal 2D-3D
DeepCompare [110]	**0.136**[1]	-	-
3DMatch [111]	-	**0.353**[2]	-
FPFH [76]	-	0.613[2]	-
Spin Images [39]	-	0.837[2]	-
Fast	0.683	0.502	0.739
Deep	0.547	0.564	**0.692**

[1] Extracted from [110]. The database used in this test is not the same database used in the remaining tests and it is extracted from the Photo Tourism dataset [12].

[2] Extracted from [111].

B.6 3D Edge Detection Parameters

The experiments presented in this section evaluate the different aspects of the proposed 3D edge detection algorithm.

B.6.1 3D Intensity Edge detection

This test evaluates the performance of the 3D intensity edge detection algorithm. This test is performed using a small subset with 200 matching frames of the frame matching dataset (see Appendix A.1.3 for details about this dataset).

The different parameters of the algorithm are tested within the valid ranges of these parameters:

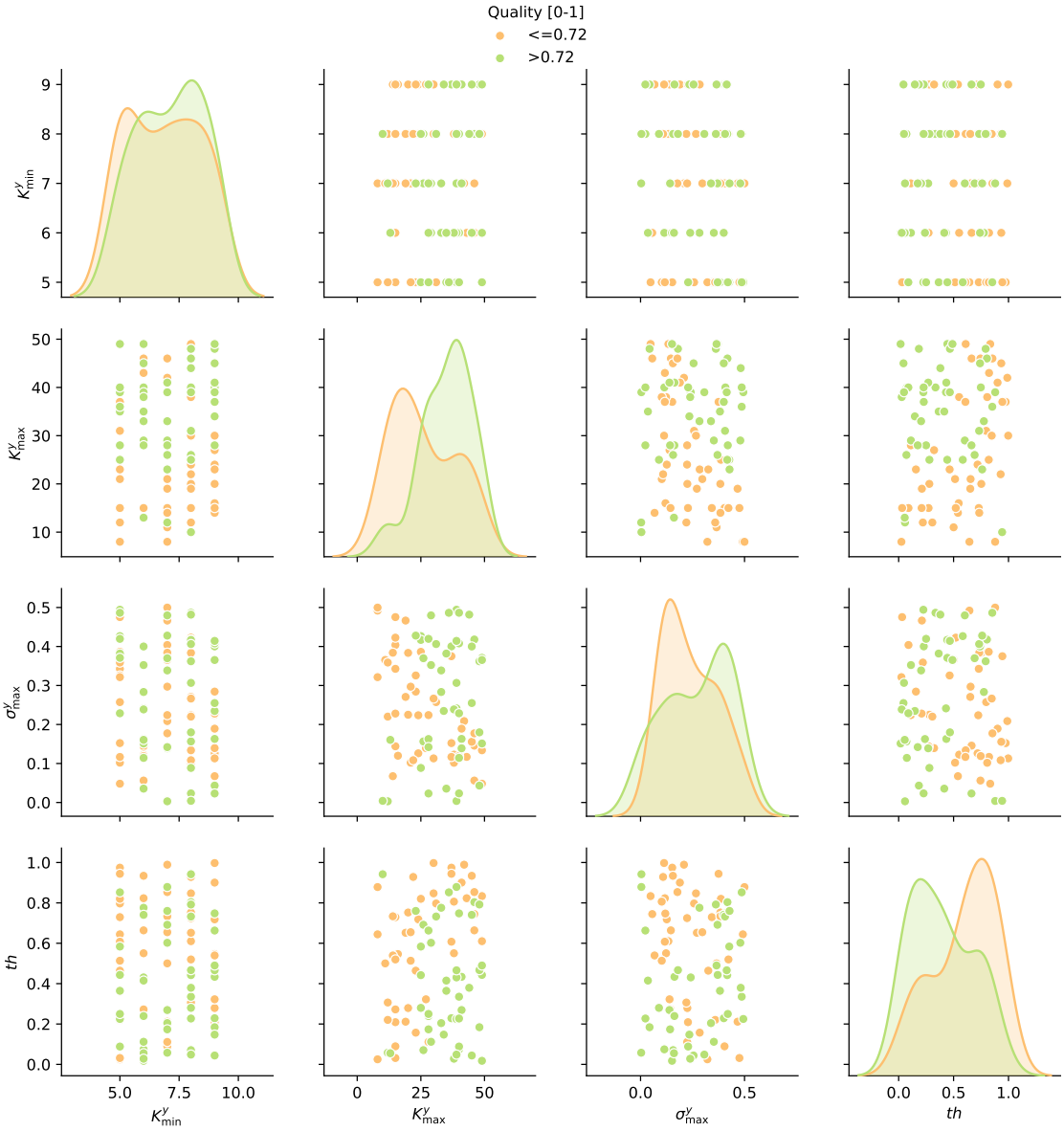
$K^y_{\min}$ Between 5 and 10, integer values.

$K^y_{\max}$ Between $K^y_{\min}$ and 50, integer values.

$\sigma^y_{\max}$ Between 0 and 0.5.

th^y Between 0 and 1.

The results obtained in this test can be seen in the following figure. This figure is shown in Chapter 7 as *Fig. 7.4: Pair plot showing the performance of several parameters on the 3D intensity edge detection algorithm. The tests are clustered based on their quality score. The diagonal plots contain the distributions for each score.*



B.6.2 3D Depth Edge detection

This test evaluates the performance of the 3D depth edge detection algorithm. This test is performed using a small subset with 200 matching frames of the frame matching dataset (see Appendix A.1.3 for details about this dataset).

The different parameters of the algorithm are tested within the valid ranges of these parameters:

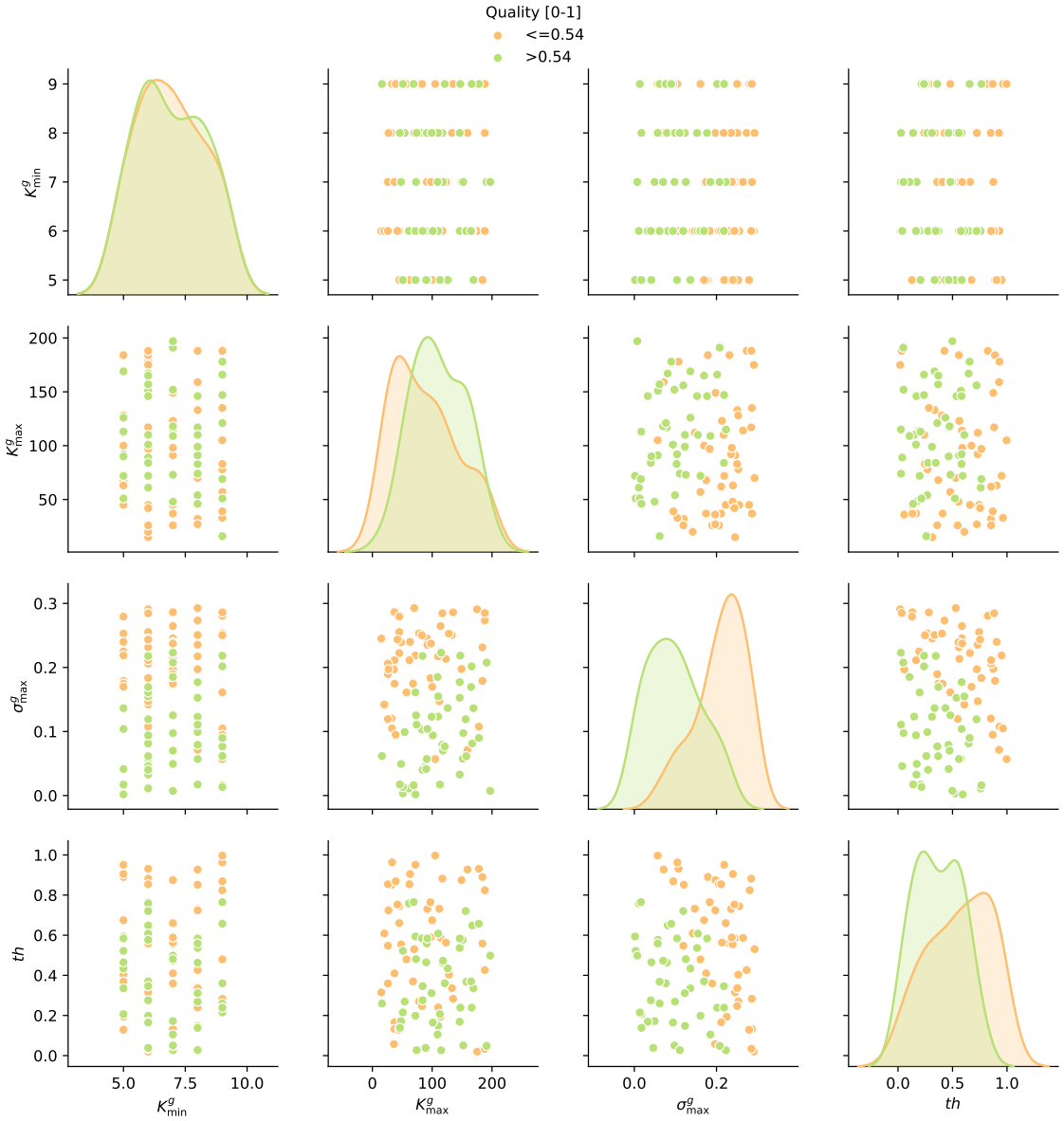
K^{g}_{min} Between 5 and 10, integer values.

$K^g_{\max}$ Between $K^g_{\min}$ and 200, integer values.

$\sigma^g_{\max}$ Between 0 and 0.3.

th^g Between 0 and 1.

The results obtained in this test can be seen in the following figure. This figure is shown in Chapter 7 as *Fig. 7.3: Pair plot showing the performance of several parameters on the 3D geometrical edge detection algorithm. The tests are clustered based on their quality score. The diagonal plots contain the distribution for each score.*



B.6.3 Combined 3D Intensity / Depth Edge detection

This test evaluates the combined performance of the 3D gradient edge detection and the 3D intensity edge detection algorithms. This test is performed using a small subset with 200 matching frames of the frame matching dataset (see Appendix A.1.3 for details about this dataset).

The different parameters of the algorithm are tested within the valid ranges of these parameters:

$K_{\min}$ Between 5 and 10, integer values.

$K^g_{\max}$ Between $K_{\min}$ and 200, integer values.

$\sigma^g_{\max}$ Between 0 and 0.3.

$K^y_{\max}$ Between $K_{\min}$ and 30, integer values.

$\sigma^y_{\max}$ Between 0 and 0.5.

th Between 0 and 1.

The results obtained in this test can be seen in the following figure. This table is shown in Chapter 7 as *Table 7.1: Table containing the 5 best performing combinations on the 3D edge detection algorithm, which combines the 3D geometrical edge detection and 3D intensity edge detection algorithms.*

	Quality	$K_{\min}$	$K^g_{\max}$	$\sigma^g_{\max}$	$K^y_{\max}$	$\sigma^y_{\max}$	th
#1	0.743	6	103	0.148	18	0.010	0.279
#2	0.741	5	177	0.102	24	0.006	0.922
#3	0.740	8	66	0.120	29	0.035	0.213
#4	0.739	7	133	0.072	29	0.042	0.158
#5	0.739	7	106	0.174	27	0.054	0.115

B.7 2D Intensity Edge detection

This test evaluates the performance of the 2D intensity edge detection algorithm. This test is performed using a small subset with 200 matching frames of the frame matching dataset (see Appendix A.1.3 for details about this dataset).

The different parameters of the algorithm are tested within the valid ranges of these parameters:

$K_{\min}$ Between 1 and 3, integer values.

$K_{\max}$ Between $K_{\min}$ and 15, integer values.

$\sigma_{\max}$ Between 0 and 0.5.

th Between 0 and 1.

The results obtained in this test can be seen in the following figure. This figure is shown in Chapter 7 as *Fig. 7.5: Pair plot showing the performance of several parameters on the 2D intensity edge detection algorithm. The tests are clustered based on their quality score. The diagonal plots contain the distributions for each score.*

A further test is done with the same search space to find the best performing parameters. This test is performed with the full dataset described in Appendix A.1.3 but using a hyperparameter optimisation algorithm.

The hyperparameter optimisation algorithm used to perform this test is the same algorithm used in the first part of this thesis to train hyperparameters: the Hyperband [47] optimisation algorithm. This algorithm measures the performance on different time steps and distributes the available resources among the best performing tests. This algorithm is applied with 20 iterations. IN each iteration 20 random frame pairs are selected for the evaluation, giving a total of 400 evaluated frames on

the best performing algorithm.

The results of this test can be seen in the following table. This table is shown in Chapter 7 as *Table 7.2: Table containing the 5 best performing combinations on the 2D edge detection algorithm.*

	Quality	$K_{\min}$	$K_{\max}$	$\sigma_{\max}$	th
#1	0.829	2	13	0.448	0.764
#2	0.818	2	13	0.492	0.560
#3	0.815	1	14	0.461	0.552
#4	0.810	1	13	0.304	0.909
#5	0.806	1	12	0.493	0.457

B.8 Multimodal 2D-3D edge detection

This test aims to find the best performing parameter set on the 2D-3D edge detection algorithm. Given the large search space for the different parameters, the search is constrained to the best performing parameters observed on the previous tests. This test is performed using a small subset with 200 matching frames of the frame matching dataset (see Appendix A.1.3 for details about this dataset). The range for each parameter is:

- Cloud edge detection:

  $K_{\min}$ Between 5 and 10, integer values.

  $K^g_{\max}$ Between $K_{\min}$ and 200, integer values.

  $\sigma^g_{\max}$ Between 0 and 0.1.

  $K^y_{\max}$ Between $K_{\min}$ and 50, integer values.

  $\sigma^y_{\max}$ Between 0 and 0.25.

  th Between 0 and 1.

- Image edge detection:

  $K_{\min}$ Between 1 and 3, integer values.

  $K_{\max}$ Between $K_{\min}$ and 10, integer values.

$\sigma_{\max}$ Between 0 and 0.5.

th Between 0.5 and 1.

The search is performed using the Hyperband algorithm by drawing 100 random samples on the defined search space. The 5 best performing combinations are seen in the following table. This table is shown in Chapter 7 as *Table 7.3: Table containing the 5 best performing combinations on the multimodal 2D-3D edge detection algorithm. This detection combines the 3D geometrical edge detection, 3D intensity edge detection and 2D intensity edge detection algorithms.*

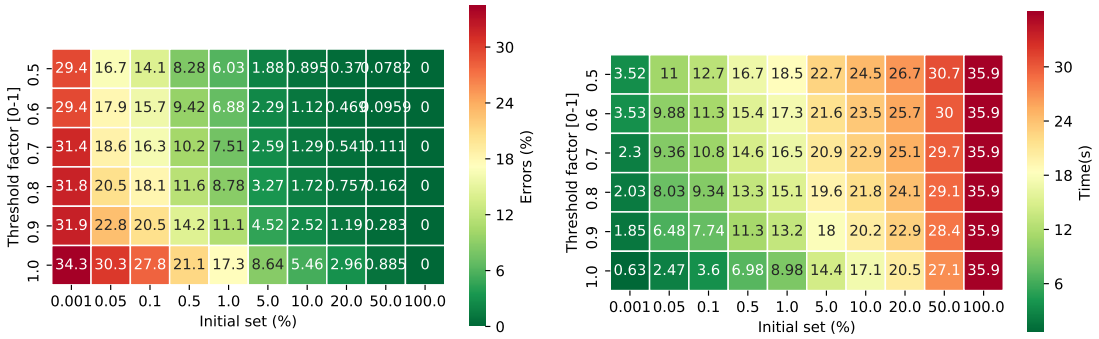
		Cloud edge detection						Image edge detection			
	Quality	$K_{\min}$	$K^g_{\max}$	$\sigma^g_{\max}$	$K^y_{\max}$	$\sigma^y_{\max}$	th	$K_{\min}$	$K_{\max}$	$\sigma_{\max}$	th
#1	0.813	7	96	0.029	30	0.456	0.857	2	9	0.285	0.661
#2	0.803	7	86	0.111	20	0.296	0.972	1	2	0.428	0.832
#3	0.784	6	79	0.034	41	0.250	0.552	2	6	0.022	0.777
#4	0.773	9	147	0.116	40	0.454	0.991	1	13	0.353	0.793
#5	0.772	5	50	0.097	45	0.207	0.761	2	3	0.253	0.555

B.9 3D Depth Edge Detection Subsampling

This test evaluates the effects of the proposed random subsampling speed up to increase the computational speed of the algorithm on 3D point clouds. This test is performed using a random draw of 10 frames on the training dataset. The full computation without applying the speedup is used as a baseline for each frame, both in terms of computation time and correct detections ground truth. Then, different values for the parameters of the speedup algorithm are tested. Each value combination is tested both in their accuracy and computational time.

This test is performed in an Nvidia Jetson TX2 development kit [61], which specifications can be found in AppendixA.3.

The results of this experiment can be seen in the following figure. This figure is shown in Chapter 7 as *Fig. 7.6: Errors (%, left) and computational time (seconds, right) for several parameter combinations.*



B.10 Edge detection benchmarking

This test presents the benchmark of the proposed method against state of the art methods. The algorithms used in this benchmarking are the Canny edge detection algorithm and the Pauly multiscale edge detector.

The parameter values used in each method to perform the evaluation is the following:

Canny Gaussian blur 5x5, th1: 100, th2:200. These values are found as the default values for the OpenCV implementation of the algorithm used in this test.

Pauly $K_{\min}$: 5, $K_{\max}$: 103, $\sigma_{\max}$: 0.148, th: 0.279. This parameters are the optimal parameters found in the 3D gradient edge detection parameter optimization.

The proposed method is evaluated with the following parameter values:

Cloud edge detection $K_{\min}$:5, $K^g_{\max}$: 103, $\sigma^g_{\max}$:0.148, $K^y_{\max}$: 18, $\sigma^y_{\max}$: 0.010, th: 0.279.

Image edge detection $K_{\min}$:1, $K_{\max}$:7, $\sigma_{\max}$:0.0024, th:0.0013.

These values are the optimal values found in the 2D-3D parameter optimization.

The results of this benchmark are presented in the following table. This table is shown in Chapter 7 as *Table 7.4: Benchmark on edge detection against state of the art methods. The quality measure is used as evaluation metric.*

Method	2D-2D domain	3D-3D domain	2D-3D domain
Canny [14]	0.584	0.608[2]	0.513[2]
HED [108][1]	**0.778**	**0.744**[3]	0.636[3]
Pauly [66]	-	0.637	-
Proposed	0.747	0.679	**0.691**

[1] The implementation in [60] has been used in this test.

[2] The 3D detection is done by detecting Canny contours on depth images.

[3] The 3D detection is done by detecting HED contours on depth images.

B.11 Transformation estimation

The following experiments evaluate the performance of the transformation estimation proposed in Chapter 3.

B.12 RANSAC execution time

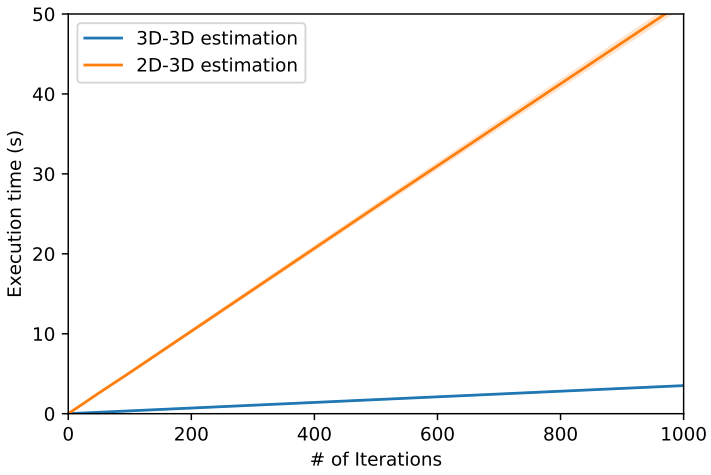
The first test performed in this evaluation is to compute the difference, in terms of execution time, between the 3D-3D and 2D-3D transformation estimation.

This test is carried out using the validation split of the dataset specified in Appendix A.1.3. This dataset contains pairs of frames with at least a 30% overlap between them and a separation of at least 10 frames. From each one of this frame pairs, 50 ground truth matches are extracted. These matches are registered using the 3D-3D and 2D-3D RANSAC algorithms with a different number of iterations.

The 3D-3D transformation estimation is computed using the direct estimation using SVD, whereas the 2D-3D estimation is computed using the Levenberg–Marquardt minimisation. Both of these computations are coded using Python and Numpy. However, the costly parts of the LM algorithm –the residual and jacobian computations– are sped up using Numba.

This test is performed in an Nvidia Jetson TX2 development kit [61], which specifications can be found in AppendixA.3.

This figure is shown in Chapter 7 as *Fig. 8.5: Execution time (in seconds) of the RANSAC pipeline for different number of iterations.*



B.12.1 Correspondence detector

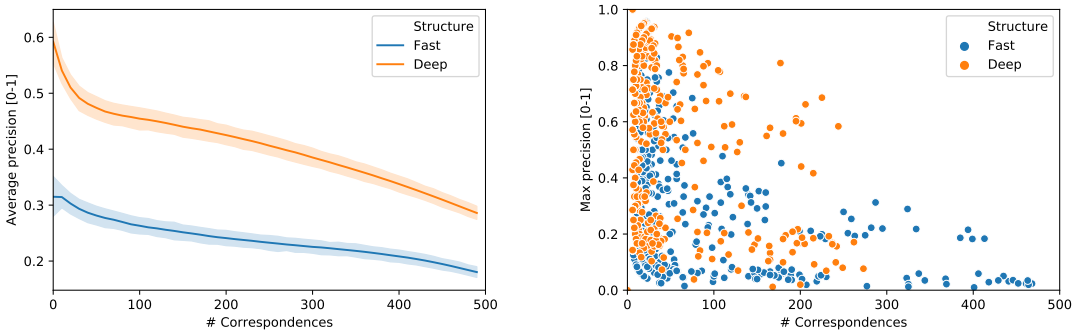
This evaluation measures the ratio of correct/icorrect correspondences generated by each of the correspondence generators proposed in this thesis: the 'Fast' and 'Deep' architectures, both in the 3D-3D transformation estimation and the 2D-3D transformation estimation.

This evaluation is performed by measuring the ratio of correct/incorrect correspondences generated on each of the frame pairs on the validation set (see Appendix A.1.3 for details on the dataset used) with different lengths of the correspondence set.

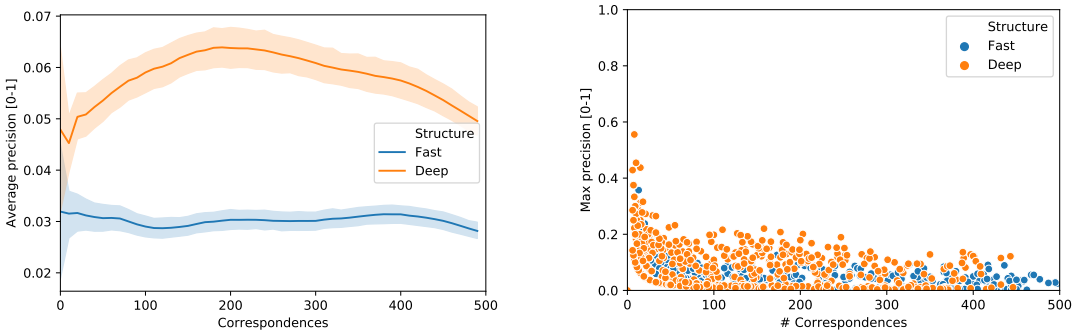
This test is carried out using the validation split of the dataset specified in Appendix A.1.3. 500 keypoints are randomly selected on each frame, and the descriptor distance between them is computed. Each correspondence set is gathered using the top-k correspondences (without repeating keypoints between correspondences), from $k = 0$ to $k = 500$.

The next figure shows the correspondence ratio for each evaluated correspondence set on the 689 frame pairs present on the validation split on the 3D-3D registration. In the left graphic, the solid line represents the average value, whereas the faded

area represents the standard deviation. The right graph shows the maximum correspondence ratio achieved on each frame and the length of the correspondence set in which this maximum is achieved. This figure is shown in Chapter 7 as *Fig. 8.3: Visualizations of the correspondence precision for the different architectures trained in the 3D-3D registration scenario*



The next figure shows the same evaluation with the 2D-3D registration. This figure is shown in Chapter 7 as *Fig. 8.4: Visualizations of the correspondence precision for the different architectures trained in the multimodal 2D-3D registration scenario*



B.12.2 RANSAC PERFORMANCE

The main goal of this experiment is to evaluate the ability of the RANSAC algorithm to estimate the transformation when errors are present in the correspondence dataset.

Two different algorithms are evaluated:

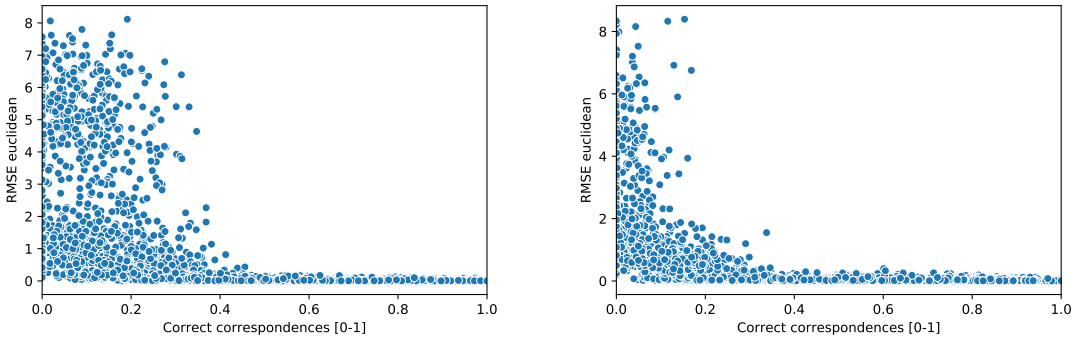
- 3D-3D transformation estimation using euclidean point to point distances
- 2D-3D transformation estimation using 3D point to line distances

This test is carried out using the validation split of the dataset specified in Appendix A.1.3. 500 keypoints are extracted in each frame. The ground truth correspondences between frame pairs are established by selecting the 500 keypoint pairs with less 3D euclidean distance without repeating keypoints. The remaining combinations are labelled as non-correspondences.

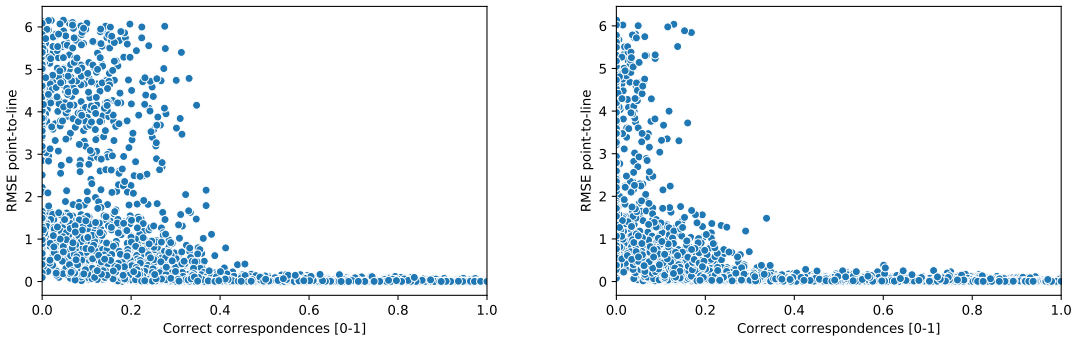
Ten different ratios of correspondences and non-correspondences are tested on each frame. These ratios are randomly selected in equally spaced intervals in order to sample a wide range of correspondence ratios. The sampling intervals are defined as $[i \cdot 0.1, i \cdot 0.1 + 0.1]$, with $i \in [0, 9]$.

The transformation is estimated using 500 iterations and selecting 6 keypoint pairs on each iteration to perform the transformation.

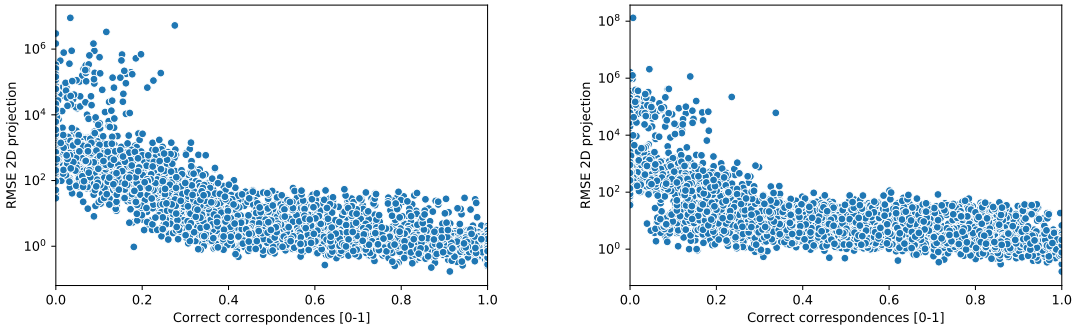
The estimated transformations are then evaluated using three different measures. The first evaluation measures the RMSE using 3D euclidean point to point distances. This result is shown in the following figure. This figure is shown in Chapter 7 as *Fig. 8.6: RMSE on the euclidean point distance for different ratios of correct/incorrect correspondences.*



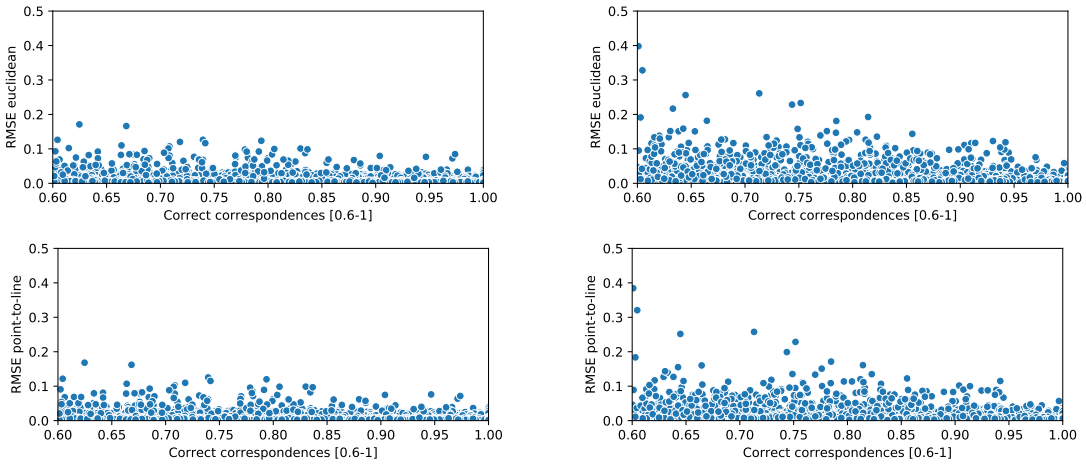
The next measure is the 3D point to line distance between the 3D point and the line defined by the projection of the 2D pixel. This result is shown in the following figure. This figure is shown in Chapter 7 as *Fig. 8.7: RMSE on the point-to-line distance for different ratios of correct/incorrect correspondences.*

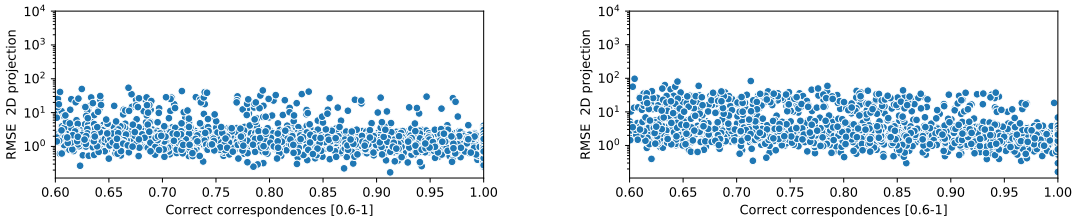


The last measure uses the 2D euclidean distance in the camera plane (in pixel units) between the pixel and the 2D projection of the 3D point cloud. This result is shown in the following figure. This figure is shown in Chapter 7 as *Fig. 8.8: RMSE on the projected 2D distance for different ratios of correct/incorrect correspondences. Note that this graphic is plotted using the logarithmic scale.*



The next figures show the same result as the previous figures focusing on the range of correctly estimated transformations. This figure is shown in Chapter 7 as *Fig. 8.9: Zoom in on the correctly performing samples on Figs. 8.6 to 8.8.*





B.12.3 Registration results

This section evaluates the performance of the different correspondence detectors evaluated in Chapter 6 to estimate the 3D-3D and 2D-3D transformations.

The methods evaluated in this section are the 'Fast' algorithm and the 'Deep' algorithm. Each one of these algorithms has been trained to perform 3D-3D and 2D-3D correspondence estimations.

This test is carried out using the validation split of the dataset specified in Appendix A.1.3. 500 keypoints are extracted in each frame, and 50 correspondences are established using the specified correspondence detectors.

The transformations are estimated using 500 iterations and selecting 6 keypoint pairs on each iteration to perform the transformation.

These transformations are evaluated using two different measures. The first measure is the Recall @ 0.2 RMSE. This measure is defined by the Recall of the transformations using as a threshold 0.2m on the RMSE measured using the point-to-point distance.

The second measure is the RMSE @ Top10%. This measure computes the average RMSE measured using the point-to-point distance on the best 10% performing algorithms.

These results can be found in the following table. This table is shown in Chapter 7 as *Table 8.3: Evaluation of the 3D-3D registration and 2D-3D registration on the validation split using several metrics.*

Domain	Method	Recall @ 0.2RMSE	RMSE @ Top10%
3D-3D	Fast	0.642	**0.040**
	Deep	**0.829**	0.063
2D-3D	Fast	-	**0.489**
	Deep	-	0.598

B.13 Transformation refinement

This next section evaluates the performance of the ICP refinement using edge detection, denoted EdgeICP. These experiments are tailored to evaluate the effects of the proposed modifications for the performance of the standard ICP algorithm.

B.13.1 Initial transformation error

The first experiment in this block measures the capability of the proposed EdgeICP algorithm to estimate the transformation giving an initial transformation error.

The dataset used in this test, the validation split of the dataset specified in Appendix A.1.3, has a relatively low initial transformation displacement. The initial registration is artificially increased by adding a random transformation, generated by adding a random initial rotation and translation.

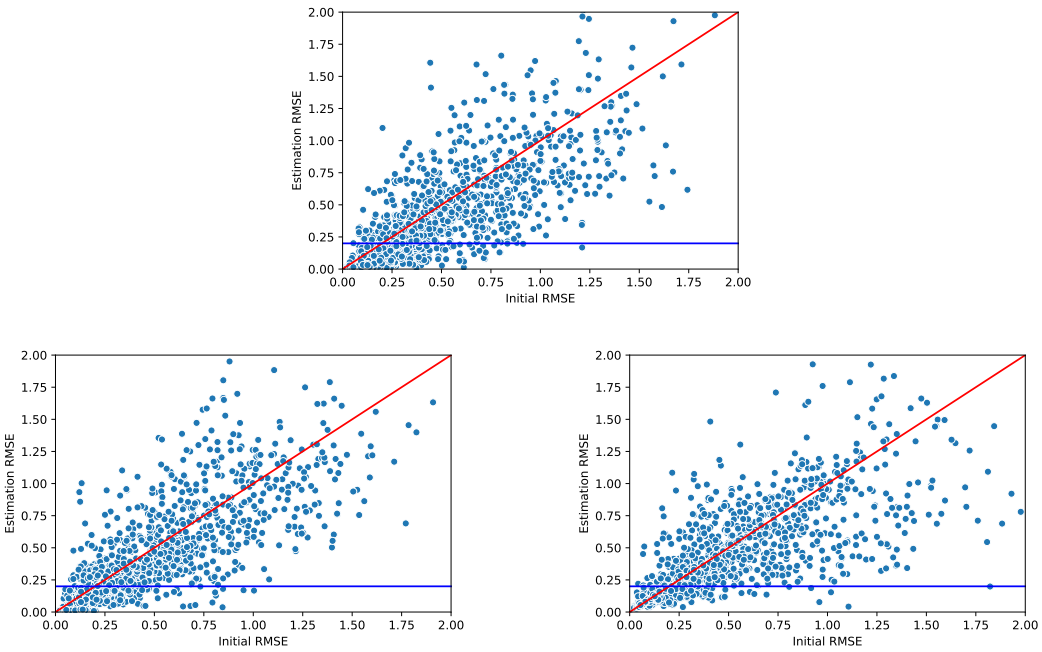
The performance of three different algorithms is compared: The standard ICP algorithm, the EdgeICP algorithm applied to 3D-3D registration and the EdgeICP algorithm applied to 2D-3D registration.

The results obtained in these tests are shown in the following figures. Each point in this figure represents a registration, where the X-axis shows the error of the initial transformation (measured in the RMSE in the euclidean 3D point to point distance) and Y-axis shows the error of the estimated transformation (also measured in the RMSE in the euclidean 3D point to point distance).

Two guides are also added in these figures. The red line denotes the points in which

the initial RMSE equals the final RMSE, and the blue line marks all points which estimated transformation has a RMSE of 0.2m.

This figure is shown in Chapter 7 as *Fig. 8.10: Evaluation of the capability of the different methods to estimate the transformation when errors are present on the initial alignment.*



The results obtained are also shown numerically to aid in the interpretation. The following table numerically shows the values defined by the two guides. The first column shows the ratio of performed transformations in which the estimated transformation has a lower RMSE than the initial transformation, defined by the red line. The second column shows the ratio of transformations in which the RMSE is below 0.2m. This table is shown in Chapter 7 as *Table 8.4: Performance comparison between the proposed algorithms. 'Improvement ratio' is the percentage of samples [0-1] in which the RMSE of the estimated transformation is lower than the RMSE of the initial transformation. Recall@0.2RMSE is the percentage of samples [0-1] in which the estimated transformation has an RMSE below 0.2m.*

Method	Improvement ratio	Recall @ 0.2RMSE
ICP [112, 7]	0.512	**0.427**
3D-3D EdgeICP	0.561	0.380
2D-3D EdgeICP	**0.617**	0.339

B.13.2 Edge detection repeatability

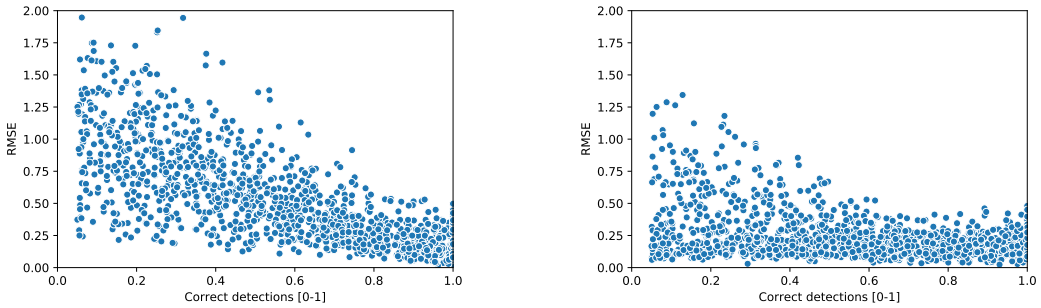
This experiment measures the influence of the edge detection repeatability into the transformation estimation. The repeatability of the edge detection is measured as the ratio of correctly detected edges over the minimum of detected edges in the frames being registered.

The different edge detection repeatabilities are simulated by artificially adding errors to ground truth detections. The ground truth detections are obtained by applying a detector to one frame and selecting the same detections in the other frame.

The detector used in this experiment is the image edge detector. The parameters used in this test are K_{min}] 1, K_{max} 9, σ_{max} 0.302 and th 0.794.

The detections obtained with the image edge detection in the source frame are translated to the target frame. As in Appendix B.12.2, ten different readabilities are tested on each frame. These ratios are randomly selected in equally spaced intervals to sample a wide range of correspondence ratios. The sampling intervals are defined as $[i \cdot 0.1, i \cdot 0.1 + 0.1]$, with $i \in [0, 9]$.

The results of this test can be seen in the following figure. This figure is shown in Chapter 7 as *Fig. 8.11: Evaluation of the RMSE (in the euclidean point-to-point distance) for different ratios of correct detections, on both the 3D-3D and 2D-3D registrations.*



B.13.3 Edge detection sensitivity

This test aims to measure the performance of the EdgeICP algorithm for different levels of sensitivity of the edge detection algorithm.

The sensitivity of the edge detection algorithm determines the overall number of edge points detected. Three different edge sensitivities for the image edge detection are tested. The parameters for the different edge detection sensitivities are the following:

Low $K_{\min}$ 1, $K_{\max}$ 9, $\sigma_{\max}$ 0.5 and th 0.8

Medium $K_{\min}$ 1, $K_{\max}$ 9, $\sigma_{\max}$ 0.302 and th 0.794

High $K_{\min}$ 1, $K_{\max}$ 9, $\sigma_{\max}$ 0.2 and th 0.5

The three edge detections are used both in the 3D-3D and 2D-3D transformation estimations. The Recall @ 0.2 RMSE (in terms of the 3D euclidean point to point distance) is measured for each combination of edge detection and transformation estimation.

These results can be seen in the following table. This table is shown in Chapter 7 as *Table 8.6: Evaluation of the performance (in terms of the Recall@0.2mRMSE) for different edge detection sensitivities.*

Edge detection sensivity	3D-3D registration	2D-3D registration
Low	**0.65**	0.54
Medium	0.66	**0.57**
High	**0.65**	0.54

B.14 EdgeICP execution time

This test evaluates the computation time of the the 3D-3D and 2D-3D transformation refinements using EdgeICP.

This test is carried out using the validation split of the dataset specified in Appendix A.1.3. This dataset contains pairs of frames with at least a 30% overlap between them and a separation of at least 10 frames. From each one of this frame pairs, 50 ground truth matches are extracted. These matches are registered using the 3D-3D and 2D-3D RANSAC algorithms with different edge detection sensitivities.

The parameters for the different edge detection sensitivities are the following:

Low $K_{\min}$ 1, $K_{\max}$ 9, $\sigma_{\max}$ 0.5 and th 0.8

Medium $K_{\min}$ 1, $K_{\max}$ 9, $\sigma_{\max}$ 0.302 and th 0.794

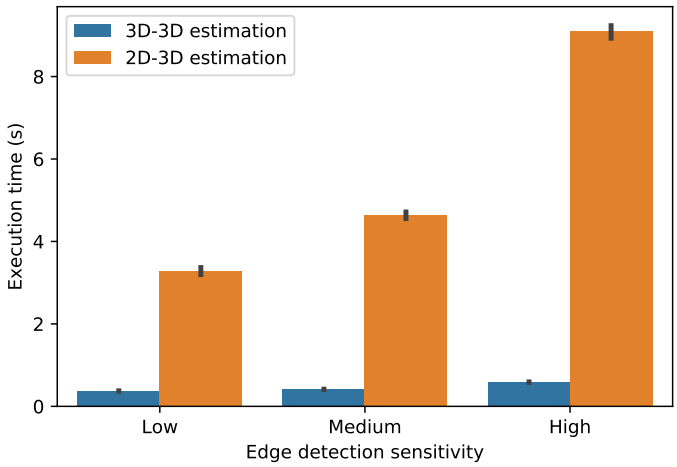
High $K_{\min}$ 1, $K_{\max}$ 9, $\sigma_{\max}$ 0.2 and th 0.5

Like the RANSAC estimation, the 3D-3D transformation estimation is computed using the direct estimation using SVD, whereas the 2D-3D estimation is computed using the Levenberg–Marquardt minimisation. Both of these computations are coded using Python and Numpy. However, the costly parts of the LM algorithm –the residual and jacobian computations– are sped up using Numba.

This test is performed in an Nvidia Jetson TX2 development kit [61], which specifications can be found in AppendixA.3.

The results of this experiment can be seen in the following figure. This figure is shown in Chapter 7 as *Fig. 8.12: Execution time (in seconds) of the ICP pipeline*

for different edge sensitivities.



B.14.1 Registration results

This test measures the performance of the EdgeICP registration pipeline on the validation dataset with the detections obtained using the edge detector proposed in Chapter 4 and evaluated in Chapter 7.

Two different methods are evaluated: the 3D-3D EdgeICP and the 2D-3D EdgeICP. These methods are evaluated using two different measures. The first measure is the Recall @ 0.2 RMSE. This measure is defined by the Recall of the transformations using as a threshold 0.2m on the RMSE measured using the point-to-point distance.

The second measure is the RMSE @ Top10%. This measure computes the average RMSE measured using the point-to-point distance on the best 10% performing algorithms.

The results obtained can be seen in the following table. In addition to the evaluated methods, the initial error of the evaluation dataset is also measured. This table is shown in Chapter 7 as *Table 8.7: Evaluation of the ICP and EdgeICP on the validation split.*

Domain	Method	Recall @ 0.2RMSE	RMSE @ Top10%
-	Initial transformation	0.687	0.049
3D-3D	EdgeICP	**0.765**	**0.008**
2D-3D	EdgeICP	0.720	0.038

As can be seen in the previous table, the initial dataset presents minimal misalignment. A random initial transformation is added to increase the initial misalignment. This initial transformation is built by randomly adding rotation and translation to the initial transformation. This additional rotation and translation are independently selected between the range $[0.05, 0.05]$ both for the three rotation degrees (in radians) and the three translation axes.

The results obtained with this added initial misalignment can be seen in the following table. This table is also shown in Chapter 7 as *Table 8.7: Evaluation of the ICP and EdgeICP on the validation split.*

Domain	Method	Recall @ 0.2RMSE	RMSE @ Top10%
-	Random transformation	0.534	0.084
3D-3D	Random + EdgeICP	0.655	**0.008**
2D-3D	Random + EdgeICP	**0.700**	0.053

B.15 Transformation estimation benchmark

The final test on this evaluation presents a benchmark between the proposed methods and several state of the art methods in the 3D-3D domain. Unlike the previous tests, the test split of the database is used in these measurements.

The state of the art methods used in this evaluation are the following:

3DMatch This algorithm is evaluated using the original dataset presented in [111]. The dataset used in this thesis is built from the same data sources but does not combine several frames (see Appendix A.1.3).

FPFH The results shown for this algorithm are also extracted from [111]. Therefore, the dataset used is the same as '3DMatch'.

Spin Images The results shown for this algorithm are also extracted from [111]. Therefore, the dataset used is the same as '3DMatch'.

ICP The results shown for this algorithm are computed in this thesis, and the database used is the database used in this thesis. This algorithm is sensitive to the initial transformation (in this dataset, the initial transformation has a 0.406 Recall @ 0.2RMSE and a 0.103 RMSE @ Top10%).

These methods are compared to the methods proposed in this thesis. The methods are the following:

Fast This algorithm uses the RANSAC approach with the 'Fast' correspondence detector.

Deep This algorithm uses the RANSAC approach with the 'Deep' correspondence detector.

EdgeICP This algorithm is sensitive to the initial transformation.

Fast + EdgeICP This algorithm uses the RANSAC approach with the 'Fast' correspondence detector combined with the EdgeICP refinement.

Deep + EdgeICP This algorithm uses the RANSAC approach with the 'Deep' correspondence detector combined with the EdgeICP refinement.

These methods are evaluated using two different measures. The first measure is the Recall @ 0.2 RMSE. This measure is defined by the Recall of the transformations using as a threshold 0.2m on the RMSE measured using the point-to-point distance.

The second measure is the RMSE @ Top10%. This measure computes the average RMSE measured using the point-to-point distance on the best 10% performing algorithms.

The results obtained in the 3D-3D domain can be seen in the following table. This table is shown in Chapter 7 as *Table 8.8: Evaluation of the 3D-3D registration on the test split using different metrics.*

Method	Recall @ 0.2RMSE	RMSE @ Top10%
3DMatch [111][1]	0.668	-
FPFH [76][1]	0.442	-
Spin Images [39][1]	0.518	-
ICP[112, 8][2]	0.549	0.019
Fast	0.402	0.073
Deep	0.678	0.038
EdgeICP	0.572	0.015
Fast + EdgeICP	0.544	0.017
Deep + EdgeICP	**0.747**	**0.013**

[1] Results extracted from [111]. These results were obtained using a database combining blocks of 50 frames (see Appendix A.1.3).

[2] Results computed with the same database used in this thesis.

The results obtained in the 2D-3D domain can be seen in the following table. This table is shown in Chapter 7 as *Table 8.9: Evaluation of the 2D-3D registration on the test split using several metrics.*

Method	Recall @ 0.2RMSE	RMSE @ Top10%
Fast	0	1.068
Deep	0	1.070
EdgeICP	**0.501**	**0.069**
Fast+EdgeICP	0	0.571
Deep + EdgeICP	0	0.523

References

[1] Abayowa, B. O., Yilmaz, A., & Hardie, R. C. (2015). Automatic registration of optical aerial imagery to a LiDAR point cloud for generation of city models. *ISPRS Journal of Photogrammetry and Remote Sensing*, 106, 68–81.

[2] Aouadi, S. & Sarry, L. (2008). Accurate and precise 2d–3d registration based on x-ray intensity. *Computer vision and image understanding*, 110(1), 134–151.

[3] Arbelaez, P. (2006). Boundary extraction in natural images using ultrametric contour maps. In *2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06)* (pp. 182–182).: IEEE.

[4] Audebert, N., Le Saux, B., & Lefèvrey, S. (2017). Fusion of heterogeneous data in convolutional networks for urban semantic labeling. In *Urban Remote Sensing Event (JURSE), 2017 Joint* (pp. 1–4).: IEEE.

[5] Behnel, S., Bradshaw, R., Citro, C., Dalcin, L., Seljebotn, D. S., & Smith, K. (2011). Cython: The best of both worlds. *Computing in Science & Engineering*, 13(2), 31–39.

[6] Besl, P. J. (1988). Active, optical range imaging sensors. *Machine vision and applications*, 1(2), 127–152.

[7] Besl, P. J. & McKay, H. D. (1992). A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2), 239–256.

[8] Besl, P. J. & McKay, N. D. (1992). Method for registration of 3-d shapes. In *Robotics-DL tentative* (pp. 586–606).: International Society for Optics and Photonics.

[9] Birkfellner, W., Wirth, J., Burgstaller, W., Baumann, B., Staedele, H., Hammer, B., Gellrich, N. C., Jacob, A. L., Regazzoni, P., & Messmer, P. (2003). A faster method for 3d/2d medical image registration—a simulation study. *Physics in medicine and biology*, 48(16), 2665.

[10] Boykov, Y. & Funka-Lea, G. (2006). Graph cuts and efficient nd image segmentation. *International journal of computer vision*, 70(2), 109–131.

[11] Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

[12] Brown, M., Hua, G., & Winder, S. (2010). Discriminative learning of local image descriptors. *IEEE transactions on pattern analysis and machine intelligence*, 33(1), 43–57.

[13] Brox, T., Rosenhahn, B., Gall, J., & Cremers, D. (2009). Combined region and motion-based 3d tracking of rigid and articulated objects. *IEEE transactions on pattern analysis and machine intelligence*, 32(3), 402–415.

[14] Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6), 679–698.

[15] Choi, S., Zhou, Q.-Y., & Koltun, V. (2015). Robust reconstruction of indoor scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[16] Collette, A. (2013). *Python and HDF5*. O'Reilly.

[17] Cutter, J. R., Styles, I. B., Leonardis, A., & Dehghani, H. (2016). Image-based registration for a neurosurgical robot: comparison using iterative closest point and coherent point drift algorithms. *Procedia Computer Science*, 90, 28–34.

[18] Dai, A., Nießner, M., Zollhöfer, M., Izadi, S., & Theobalt, C. (2017). Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics (ToG)*, 36(3), 24.

[19] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition* (pp. 248–255).: Ieee.

[20] Eitz, M., Richter, R., Boubekeur, T., Hildebrand, K., & Alexa, M. (2012). Sketch-based shape retrieval. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 31(4), 31:1–31:10.

[21] Fischler, M. A. & Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6), 381–395.

[22] Fitzgibbon, A. W. (2003). Robust registration of 2d and 3d point sets. *Image and Vision Computing*, 21(13), 1145–1153.

[23] Florin, C., Williams, J., Khamene, A., & Paragios, N. (2005). Registration of 3d angiographic and x-ray images using sequential monte carlo sampling. In *CVBIA* (pp. 427–436).: Springer.

[24] Gauglitz, S., Höllerer, T., & Turk, M. (2011). Evaluation of interest point detectors and feature descriptors for visual tracking. *International journal of computer vision*, 94(3), 335.

[25] Gelfand, N., Mitra, N. J., Guibas, L. J., & Pottmann, H. (2005). Robust global registration. In *Symposium on geometry processing*, number 3 (pp.5).

[26] Giering, M., Venugopalan, V., & Reddy, K. (2015). Multi-modal sensor registration for vehicle perception via deep neural networks. In *2015 IEEE High Performance Extreme Computing Conference (HPEC)* (pp. 1–6).: IEEE.

[27] Groher, M., Jakobs, T. F., Padoy, N., & Navab, N. (2007). Planning and intraoperative visualization of liver catheterizations: new cta protocol and 2d-3d registration method. *Academic radiology*, 14(11), 1325–1340.

[28] Guo, Y., Bennamoun, M., Sohel, F., Lu, M., Wan, J., & Kwok, N. M. (2016). A comprehensive performance evaluation of 3d local feature descriptors. *International Journal of Computer Vision*, 116(1), 66–89.

[29] Guo, Y., Sohel, F., Bennamoun, M., Lu, M., & Wan, J. (2013). Rotational projection statistics for 3d local surface description and object recognition. *International journal of computer vision*, 105(1), 63–86.

[30] Habib, A., Ghanma, M., Morgan, M., & Mitishita, E. (2004). Integration of laser and photogrammetric data for calibration purposes. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 35, 170.

[31] Hadsell, R., Chopra, S., & LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2 (pp. 1735–1742).: IEEE.

[32] Halber, M. & Funkhouser, T. (2017). Fine-to-coarse global registration of rgb-d scans.

[33] Harris, C. G., Stephens, M., et al. (1988). A combined corner and edge detector. In *Alvey vision conference*, volume 15 (pp. 10–5244).: Citeseer.

[34] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

[35] Hitomi, E. E., Silva, J. V., & Ruppert, G. C. (2015). 3d scanning using rgbd imaging devices: A survey. In *Developments in Medical Image Processing and Computational Vision* (pp. 379–395). Springer.

[36] Hufnagel, H., Pennec, X., Ehrhardt, J., Ayache, N., & Handels, H. (2008). Generation of a statistical shape model with probabilistic point correspondences and the expectation maximization-iterative closest point algorithm. *International journal of computer assisted radiology and surgery*, 2(5), 265–273.

[37] Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3), 90–95.

[38] Jin, J.-Y., Ryu, S., Faber, K., Mikkelsen, T., Chen, Q., Li, S., & Movsas, B. (2006). 2d/3d image fusion for accurate target localization and evaluation of a mask based stereotactic system in fractionated stereotactic radiotherapy of cranial lesions. *Medical physics*, 33(12), 4557–4566.

[39] Johnson, A. E. & Hebert, M. (1999). Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on pattern analysis and machine intelligence*, 21(5), 433–449.

[40] Karpathy, A. & Fei-Fei, L. (2015). Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3128–3137).

[41] Khan, N. Y., McCane, B., & Wyvill, G. (2011). Sift and surf performance evaluation against various image deformations on benchmark dataset. In *2011 International Conference on Digital Image Computing: Techniques and Applications* (pp. 501–506).: IEEE.

[42] Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., & Willing, C. (2016). Jupyter notebooks – a publishing format for reproducible computational workflows. In F. Loizides & B. Schmidt (Eds.), *Positioning and Power in Academic Publishing: Players, Agents and Agendas* (pp. 87 – 90).: IOS Press.

[43] Kwak, T.-S., Kim, Y.-I., Yu, K.-Y., & Lee, B.-K. (2006). Registration of aerial imagery and aerial lidar data using centroids of plane roof surfaces as control information. *KSCE journal of Civil Engineering*, 10(5), 365–370.

[44] Lai, K., Bo, L., & Fox, D. (2014). Unsupervised feature learning for 3d scene labeling. In *2014 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 3050–3057).: IEEE.

[45] Lam, S. K., Pitrou, A., & Seibert, S. (2015). Numba: A llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC* (pp.7).: ACM.

[46] Lavallée, S. & Szeliski, R. (1995). Recovering the position and orientation of free-form objects from image contours using 3d distance maps. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 17(4), 378–390.

[47] Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., & Talwalkar, A. (2016). Hyperband: A novel bandit-based approach to hyperparameter optimization. *arXiv preprint arXiv:1603.06560*.

[48] Liu, L. & Stamos, I. (2005). Automatic 3d to 2d registration for the photorealistic rendering of urban scenes. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2 (pp. 137–143).: IEEE.

[49] Liu, L. & Stamos, I. (2007). A systematic approach for 2d-image to 3d-range registration in urban environments. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on* (pp. 1–8).: IEEE.

[50] Liu, L., Stamos, I., Yu, G., Wolberg, G., & Zokai, S. (2006). Multiview geometry for texture mapping 2d images onto 3d range data. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2 (pp. 2293–2300).: IEEE.

[51] Livyatan, H., Yaniv, Z., & Joskowicz, L. (2003). Gradient-based 2-d/3-d rigid registration of fluoroscopic x-ray to ct. *Medical Imaging, IEEE Transactions on*, 22(11), 1395–1406.

[52] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2), 91–110.

[53] Makadia, A., Patterson, A., & Daniilidis, K. (2006). Fully automatic registration of 3d point clouds. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1 (pp. 1297–1304).: IEEE.

[54] Markelj, P., Tomaževič, D., Likar, B., & Pernuš, F. (2012). A review of 3d/2d registration methods for image-guided interventions. *Medical Image Analysis*, 16(3), 642–661.

[55] Markelj, P., Tomaževič, D., Pernuš, F., & Likar, B. (2008). Robust gradient-based 3-d/2-d registration of ct and mr to x-ray images. *Medical Imaging, IEEE Transactions on*, 27(12), 1704–1714.

[56] Mastin, A., Kepner, J., & Fisher, J. (2009). Automatic registration of LIDAR and optical images of urban scenes. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on* (pp. 2639–2646).: IEEE.

[57] Mishra, R. & Zhang, Y. (2012). A review of optical imagery and airborne lidar data registration methods. *The Open Remote Sensing Journal*, 5(1), 54–63.

[58] Moré, J. J. (1978). The levenberg-marquardt algorithm: implementation and theory. In *Numerical analysis* (pp. 105–116).: Springer.

[59] Mu, Z., Fu, D., & Kuduvalli, G. (2008). A probabilistic framework based on hidden markov model for fiducial identification in image-guided radiation treatments. *Medical Imaging, IEEE Transactions on*, 27(9), 1288–1300.

[60] Niklaus, S. (2018). A reimplementation of HED using PyTorch. https://github.com/sniklaus/pytorch-hed.

[61] NVIDIA Corporation (2014). Jetson tx2 series module data sheet. http://developer.nvidia.com/embedded/dlc/jetson-tx2-series-modules-data-sheet.

[62] Oliphant, T. E. (2006). *A guide to NumPy*, volume 1. Trelgol Publishing USA.

[63] Pal, N. R. & Pal, S. K. (1993). A review on image segmentation techniques. *Pattern recognition*, 26(9), 1277–1294.

[64] Papon, J., Abramov, A., Schoeler, M., & Worgotter, F. (2013). Voxel cloud connectivity segmentation-supervoxels for point clouds. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2027–2034).

[65] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., & Lerer, A. (2017). Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*.

[66] Pauly, M., Keiser, R., & Gross, M. (2003). Multi-scale Feature Extraction on Point-Sampled Surfaces. *Computer Graphics Forum*, 22(3), 281–289.

[67] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct), 2825–2830.

[68] Pintus, R. & Gobbetti, E. (2015). A Fast and Robust Framework for Semiautomatic and Automatic Registration of Photographs to 3d Geometry. *Journal on Computing and Cultural Heritage*, 7(4), 1–23.

[69] Pomerleau, F., Colas, F., Siegwart, R., & Magnenat, S. (2013). Comparing icp variants on real-world data sets. *Autonomous Robots*, 34(3), 133–148.

[70] Prümmer, M., Hornegger, J., Pfister, M., & Dörfler, A. (2006). Multi-modal 2d-3d non-rigid registration. In *Medical Imaging* (pp. 61440X–61440X).: International Society for Optics and Photonics.

[71] Pujol-Miro, A., Casas, J. R., & Ruiz-Hidalgo, J. (2019). Correspondence matching in unorganized 3d point clouds using convolutional neural networks. *Image and Vision Computing*, 83, 51–60.

[72] Pujol-Miro, A., Ruiz-Hidalgo, J., & Casas, J. R. (2017). Registration of images to unorganized 3d point clouds using contour cues. In *2017 25th European Signal Processing Conference (EUSIPCO)* (pp. 81–85).: IEEE.

[73] Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 652–660).

[74] Rosenhahn, B. (2003). *Pose estimation revisited*. PhD thesis, Christian-Albrechts Universität Kiel.

[75] Rosenhahn, B., Brox, T., & Weickert, J. (2007). Three-dimensional shape knowledge for joint image segmentation and pose tracking. *International Journal of Computer Vision*, 73(3), 243–262.

[76] Rusu, R. B., Blodow, N., & Beetz, M. (2009). Fast point feature histograms (fpfh) for 3d registration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on* (pp. 3212–3217).: IEEE.

[77] Rusu, R. B. & Cousins, S. (2011). 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)* Shanghai, China.

[78] Schenk, T. & Csathó, B. (2002). Fusion of lidar data and aerial imagery for a more complete surface description. *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences*, 34(3/A), 310–317.

[79] Schindler, G., Krishnamurthy, P., Lublinerman, R., Liu, Y., & Dellaert, F. (2008). Detecting and matching repeated patterns for automatic geo-tagging in urban environments. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on* (pp. 1–7).: IEEE.

[80] Schmid, C., Mohr, R., & Bauckhage, C. (2000). Evaluation of interest point detectors. *International Journal of computer vision*, 37(2), 151–172.

[81] Schneider, N., Piewak, F., Stiller, C., & Franke, U. (2017). Regnet: Multimodal sensor registration using deep neural networks. In *2017 IEEE intelligent vehicles symposium (IV)* (pp. 1803–1810).: IEEE.

[82] Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 815–823).

[83] Schwarz, B. (2010). Lidar: Mapping the world in 3d. *Nature Photonics*, 4(7), 429.

[84] Sharp, G. C., Lee, S. W., & Wehe, D. K. (2002). Icp registration using invariant features. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(1), 90–102.

[85] Shevlin, F. (1998). Analysis of orientation problems using plucker lines. In *Proceedings. Fourteenth International Conference on Pattern Recognition (Cat. No. 98EX170)*, volume 1 (pp. 685–689).: IEEE.

[86] Shotton, J., Glocker, B., Zach, C., Izadi, S., Criminisi, A., & Fitzgibbon, A. (2013). Scene coordinate regression forests for camera relocalization in rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2930–2937).

[87] Silberman, N., Hoiem, D., Kohli, P., & Fergus, R. (2012). Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision* (pp. 746–760).: Springer.

[88] Simonyan, K. & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[89] Sorkine, O. (2009). Least-squares rigid motion using svd. *Technical notes*, 120(3), 52.

[90] Stamos, I. (2010). Automated registration of 3d-range with 2d-color images: an overview. In *2010 44th Annual Conference on Information Sciences and Systems (CISS)* (pp. 1–6).

[91] Stamos, I. & Alien, P. (2001). Automatic registration of 2-d with 3-d imagery in urban environments. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2 (pp. 731–736).: IEEE.

[92] Stamos, I., Liu, L., Chen, C., Wolberg, G., Yu, G., & Zokai, S. (2008). Integrating automated range registration with multiview geometry for the photorealistic modeling of large-scale scenes. *International Journal of Computer Vision*, 78(2-3), 237–260.

[93] Stewart, C. V., Tsai, C.-L., & Roysam, B. (2003). The dual-bootstrap iterative closest point algorithm with application to retinal image registration. *Medical Imaging, IEEE Transactions on*, 22(11), 1379–1394.

[94] Stroustrup, B. (2000). *The C++ programming language*. Pearson Education India.

[95] Sturm, J., Engelhard, N., Endres, F., Burgard, W., & Cremers, D. (2012). A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*.

[96] Tomaževič, D., Likar, B., Slivnik, T., & Pernuš, F. (2003). 3-d/2-d registration of ct and mr to x-ray images. *Medical Imaging, IEEE Transactions on*, 22(11), 1407–1416.

[97] Tombari, F., Salti, S., & Di Stefano, L. (2013). Performance evaluation of 3d keypoint detectors. *International Journal of Computer Vision*, 102(1-3), 198–220.

[98] Tomono, M. (2009). Robust 3d slam with a stereo camera based on an edge-point icp algorithm. In *2009 IEEE International Conference on Robotics and Automation* (pp. 4306–4311).: IEEE.

[99] Torr, P. H. & Zisserman, A. (2000). Mlesac: A new robust estimator with application to estimating image geometry. *Computer vision and image understanding*, 78(1), 138–156.

[100] Valentin, J., Dai, A., Nießner, M., Kohli, P., Torr, P., Izadi, S., & Keskin, C. (2016). Learning to navigate the energy landscape. In *2016 Fourth International Conference on 3D Vision (3DV)* (pp. 323–332).: IEEE.

[101] Van Rossum, G. & Drake Jr, F. L. (1995). *Python tutorial*. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands.

[102] Viola, P. & Wells III, W. M. (1997). Alignment by maximization of mutual information. *International journal of computer vision*, 24(2), 137–154.

[103] Wasenmuller, O., Meyer, M., & Stricker, D. (2016). CoRBS: Comprehensive rgb-d benchmark for slam using kinect v2. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, volume . (pp.˜).: IEEE.

[104] Wein, W., Röper, B., & Navab, N. (2005). 2d/3d registration based on volume gradients. In *Medical Imaging* (pp. 144–150).: International Society for Optics and Photonics.

[105] Winder, S. A. & Brown, M. (2007). Learning local image descriptors. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on* (pp. 1–8).: IEEE.

[106] Wong, A. & Orchard, J. (2008). Efficient fft-accelerated approach to invariant optical–lidar registration. *Geoscience and Remote Sensing, IEEE Transactions on*, 46(11), 3917–3925.

[107] Xiao, J., Owens, A., & Torralba, A. (2013). Sun3d: A database of big spaces reconstructed using sfm and object labels. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 1625–1632).

[108] Xie, S. & Tu, Z. (2015). Holistically-nested edge detection. In *IEEE International Conference on Computer Vision*.

[109] Yang, G., Becker, J., & Stewart, C. V. (2007). Estimating the location of a camera with respect to a 3d model. In *3-D Digital Imaging and Modeling, 2007. 3DIM'07. Sixth International Conference on* (pp. 159–166).: IEEE.

[110] Zagoruyko, S. & Komodakis, N. (2015). Learning to compare image patches via convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4353–4361).

[111] Zeng, A., Song, S., Nießner, M., Fisher, M., Xiao, J., & Funkhouser, T. (2017). 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1802–1811).

[112] Zhang, Z. (1992). Iterative point matching for registration of free-form curves.

[113] Zhang, Z. (2012). Microsoft kinect sensor and its effect. *IEEE multimedia*, 19(2), 4–10.

[114] Zhao, W., Nister, D., & Hsu, S. (2005). Alignment of continuous video onto 3d point clouds. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(8), 1305–1318.

[115] Zheng, G., Ballester, M. Á., Styner, M., & Nolte, L.-P. (2006). Reconstruction of patient-specific 3d bone surface from 2d calibrated fluoroscopic images and point distribution model. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2006* (pp. 25–32). Springer.

[116] Zhong, Y. (2009). Intrinsic shape signatures: A shape descriptor for 3d object recognition. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops* (pp. 689–696).: IEEE.

[117] Zhou, Q.-Y., Park, J., & Koltun, V. (2018). Open3D: A modern library for 3D data processing. *arXiv:1801.09847*.

[118] Zitova, B. & Flusser, J. (2003). Image registration methods: a survey. *Image and vision computing*, 21(11), 977–1000.