

Synthetic view of retail spaces using camera and RFID sensors on a robot

Kamruddin Md. Nur

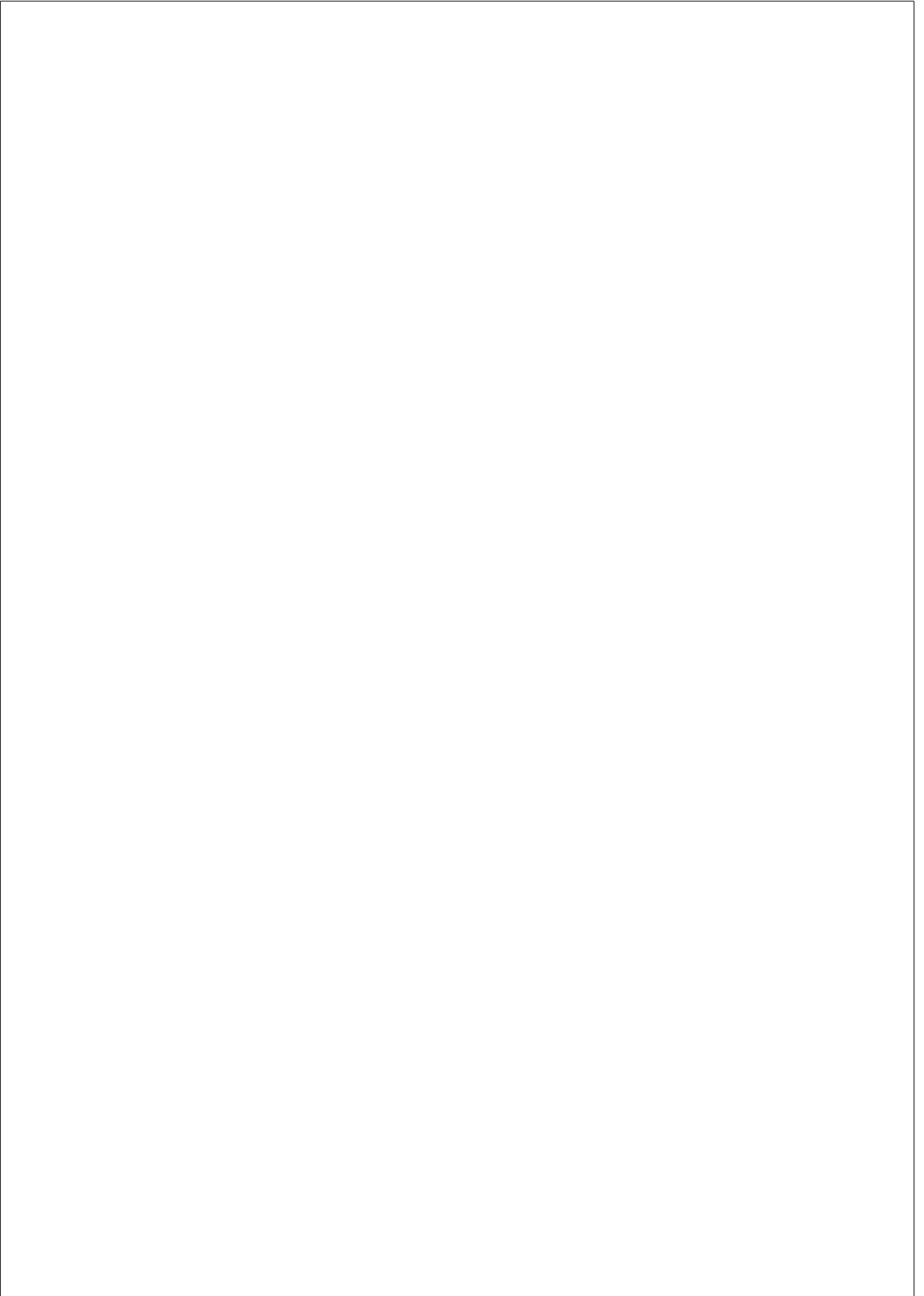
TESI DOCTORAL UPF / DOCTORAL THESIS

Thesis Advisor
Dr. Rafael Pous Andrés

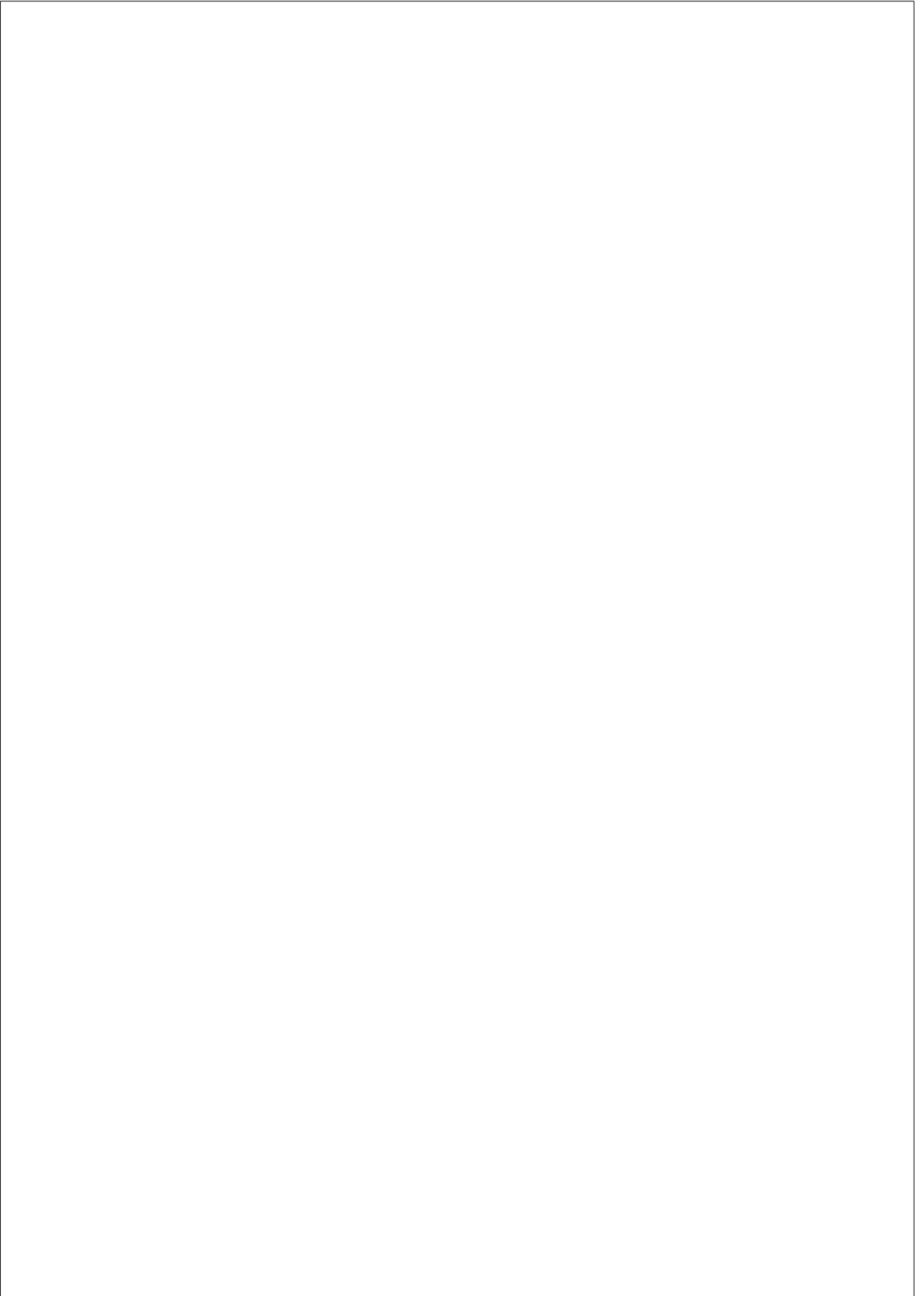
Department of Information and Communication Technologies,
Universitat Pompeu Fabra, Barcelona

November, 2015





To my parents, my wife, and my little kid Ehan.



Acknowledgments

Firstly, I would like to express sincere gratitude to my PhD advisor Dr. Rafael Pous for his patience, motivation and continuous support at UbiCA Lab, DTIC, UPF.

I would like to take this opportunity to thank DTIC faculty members, staff, and the DTIC secretariat, for their cordial support whenever I needed.

I would like to thank my parents and my family members, especially my uncle Dr. Waresul Karim for his profound faith and continuous support throughout my career and studies. I would like to thank the almighty Allah for blessing me with a good health and strength to complete this thesis.

I would like to thank specially Keonn Technologies, whose product Advan-Robot was used for this research and would like to acknowledge the cooperation of Roberto Verino, the fashion firm in whose Barcelona store the experiment of chapter 3 were conducted.

Special thanks to Dr. Ramir De Porrata-Doria, Dr. Anna Carreras, and Marc Morenza-Cinos for their helpful cooperation during the project work.

Along the way at UPF, I have made friends and found intelligent and interesting people who am truly thankful to have met.



Abstract

In this thesis, two approaches of information presentation on indoor view have been presented using Radio-Frequency Identification (RFID) and camera sensors on a robot. The goal is to capture images of the indoor environment and to create a 3D view so that users can view, navigate, and locate a product on the view. RFID is an ‘Auto-ID’ system that can identify a tagged object from a remote distance without a direct line-of-sight. An RFID system can be configured to acquire approximate locations of an RFID tagged objects. In the first approach, a Google Street View-like indoor view creation and RFID-obtained product information projection have been presented. Also, in the second approach, we explore Simultaneous Localization and Mapping (SLAM), RGB-D Red, Blue, Green, and Depth Mapping, and the RFID-obtained product information projection on a 3D point cloud map. In the first approach, we have explored how a ‘Store view’ system can be made of stitching images into panoramas and inter-linking panoramas into a Street View-like store view. The study also explores how dynamic web programming can be used to fetch and project RFID-obtained item information on the view. SLAM is a method to localize a robot while mapping the environment. With an RFID system installed on a mobile robot, all items within the environment can be identified and can be located when the robot position is localized first. Thus, in our second approach, the RGB-D SLAM approach has been chosen for 3D mapping of the environment, so that, localized RFID tagged item information can be projected on the 3D map view.

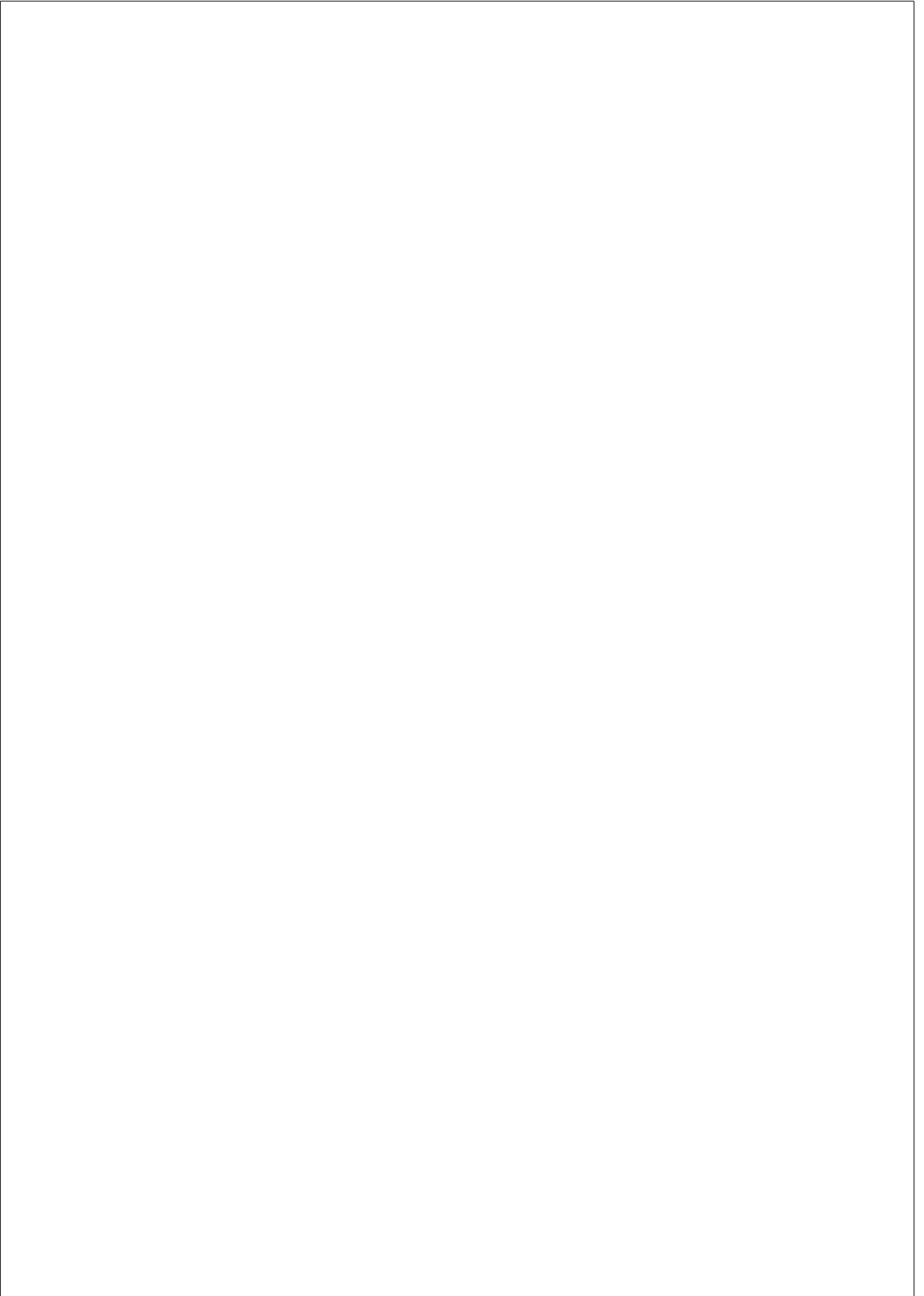
Resum

En aquesta tesi, dos enfocaments de presentació de la informació a la vista d'interior s'han presentat mitjançant identificació per radiofreqüència (RFID) i sensors de la càmera en un robot. L'objectiu és capturar imatges de l'ambient interior i la creació d'una vista 3D, de manera que els usuaris poden veure, navegar i localitzar un producte a la vista. RFID és un 'sistema' Auto ID, que és capaç d'identificar un objecte etiquetat des d'una distància remota sense una línia de visió directa. Per altra banda, un RFID sistema pot ser configurat per adquirir ubicacions aproximades d'objectes RFID etiquetats. En el primer enfocament, s'han presentat una creació vista interior com Google Street View i la projecció d'informació de productes obtinguts per RFID. I, en el segon enfocament, la projecció de la informació de productes obtinguts per RFID en una vista de núvol de punts 3D s'ha presentat usant un econòmic RGB-D (Red, Blue, Green, and Depth) sensor de càmera i RGB-D SLAM. En aquest estudi, hem explorat com es pot crear un sistema 'Store view' connectant imatges en una imatge panoràmica per després inter-connectar-les en un 'Street View' de la botiga. L'estudi també explora com es pot utilitzar la programació dinàmica web per recuperar i projectar la informació obtinguda pel RFID del producte en la vista. La localització simultània i mapatge (SLAM) és un mètode per localitzar un robot, al mateix temps que es mapeja l'entorn. Amb un sistema de RFID instal·lat en un robot mòbil, tots els elements dins de l'entorn poden ser identificats i poden ser localitzats quan la ubicació del robot es localitza primer. Per tant, l'enfocament SLAM RGB-D ha estat elegit per al mapatge en 3D de l'entorn perquè la informació de la localització de l'article RFID etiquetat pot ser projectada a la vista del mapa.

Preface

The research presented in this thesis has been conducted under Dr. Rafael Pous, director, Ubiquitous Computing Applications Lab (UbiCA Lab), DTIC, UPF. The focus of the research was to project RFID acquired inventory information on the indoor view using camera sensors on a robot. Two automated approaches have been presented in chapter 3 and chapter 4. In chapter 3, an automated Google Street View-like 360° synthetic view of a retail store has been presented where RFID-obtained product information is presented on the panoramic view. In chapter 4, a 3D synthetic view using RGB-D SLAM has been used for creating an indoor 3D point cloud view on a robot. To show RFID information on the 3D point cloud maps, rviz (ROS Visualization), and PCL (Point Cloud Library) have been used.

The study results of the approach presented in chapter 3, have been published in the **Intelligent Systems, IEEE (Volume:30, Issue: 6), DOI: 10.1109/MIS.2015.90.**



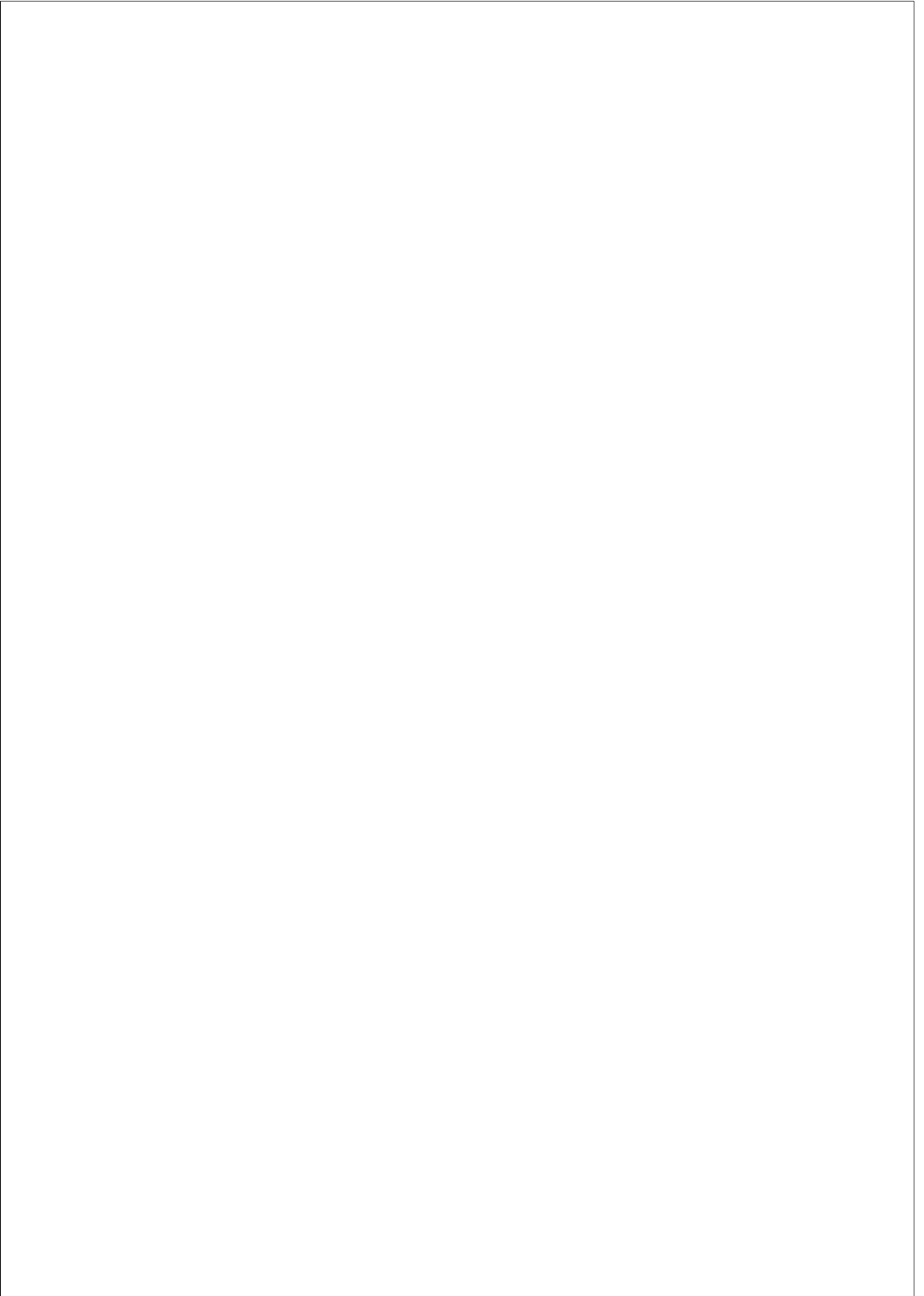
Contents

List of Figures	xiv
List of Tables	xv
1 Introduction	1
1.1 Motivation	1
1.2 Research Objective	4
1.3 Research Methodology	4
1.4 Approaches	5
1.5 Structure of this dissertation	5
2 State-of-the-art review	7
2.1 Radio-Frequency Identification	7
2.1.1 RFID Tag	8
2.1.2 RFID Reader	9
2.2 Feature Detection and Matching Algorithms	9
2.2.1 SIFT	10
2.2.2 SURF	10
2.2.3 FAST	11
2.2.4 BRIEF	11
2.2.5 ORB	12
2.3 Panorama Stitching	12
2.4 ROS	13
2.5 Review of Patents	14
3 Indoor 360° synthetic view using camera and RFID sensor	17
3.1 Introduction	18
3.2 Related Work to Indoor 360° Synthetic View	18
3.3 The Proposed System	19
3.3.1 Overview and Architecture	19

Contents

3.3.2	Implementation	20
3.4	Experimental Results	28
3.4.1	Synthetic View of a Real Store	28
3.4.2	User Evaluation	31
3.5	Chapter Summary	34
4	3D synthetic view using RGB-D SLAM and RFID sensors	35
4.1	Introduction	36
4.2	Simultaneous Localization and Mapping (SLAM)	36
4.3	Depth Measurement Techniques	37
4.3.1	Time-of-flight	37
4.3.2	Phase-shift	38
4.3.3	Triangulation	38
4.3.4	Structured light	38
4.4	Laser Scanner	39
4.4.1	2D laser scanner	39
4.4.2	3D laser scanner	40
4.5	RGB-D Camera	40
4.6	Related Work to RGB-D SLAM Mapping	43
4.7	Related Work to 3D Map Annotation	46
4.8	Point Cloud	47
4.8.1	Point Cloud Map	47
4.8.2	Point Cloud Library	47
4.8.3	Point Cloud File Formats	48
4.9	Displaying 3D Maps	49
4.9.1	Rviz	50
4.9.2	Offline Map Publisher	50
4.10	Methodology	52
4.11	3D Mapping with RTAB-Map	52
4.11.1	Using Visual Odometry	53
4.11.2	Using Robot Odometry	54
4.12	Projecting RFID-obtained Information	54
4.13	Experiments and Results	55
4.14	Chapter Summary	61
5	Conclusion	63
5.1	Concluding Remarks	63
5.2	Contributions	64
5.3	Future Works	64
5.4	Other Research and Development	65

	Contents
5.4.1 Speech Interaction Application	65
A Appendix	69
Bibliography	71



List of Figures

1.1	Research objective	4
2.1	RFID UHF tags	8
2.2	RFID readers	9
2.3	SIFT features matched between two images, a demonstration of OpenSIFT	10
2.4	Scale and rotation in-variance in object detection with SURF	11
2.5	Keypoint matching using ORB	12
2.6	Hugin control point detection	14
3.1	The “Store view”system architecture	20
3.2	RFID system on the robot	21
3.3	A panorama of the actual retail store created by Hugin shell script (panorama 1, store section 1)	21
3.4	The ϕ and ψ calculation by the equations 3.2 and 3.3	24
3.5	Panoramas of different sections of the store	26
3.6	AdvanRobot is reading RFID tags and their approximate locations by going around the retail store.	28
3.7	Panorama (1-7) capture points labelled as P (1-7)	29
3.8	RFID inventory (black dots) read by the robot, where $pos.X$, $pos.Y$, and $pos.Z$ are the length, width, and height coordinates.	29
3.9	The Store view user interface	30
3.10	Virtual browsing - a user is evaluating the ‘virtual browsing’ feature and locating products with the mouse.	31
3.11	A user is evaluating the ‘click and display’ feature and locating products with the mouse.	34
4.1	Structured light depth measurement parameters	39
4.2	2D laser scanners	40

List of Figures

4.3	3D laser scanners	41
4.4	The overview of RGB-D mapping proposed by Henry et al. .	43
4.5	The overview of RGB-D SLAM system proposed by Endres et al.	45
4.6	The RTAB-Map memory management model presented by Labbé and Michaud.	46
4.7	RGB-D SLAM 3D map viewing approaches	50
4.8	Rviz is showing depth-registered points from Xtion Pro Live camera.	51
4.9	Active node of the map publisher	51
4.10	RGB-D mapping using xBox 360 Kinect camera by RTAB-Map using visual odometry	55
4.11	RGB-D mapping with AdvanRobot using robot odometry . .	56
4.12	The AdvanRobot RFID antennas in different heights.	57
4.13	Marker Display in X,Y,Z coordinates	57
4.14	RGB-D mapping using visual odometry	58
4.15	RGB-D mapping using robot odometry	59
4.16	Camera position, tilt angles, and camera focus in different heights (in meters)	60
5.1	The speech interaction application architecture	66
A.1	Shop point cloud map (rendered by Potree) to view on web page	69
A.2	Shop point cloud (rendered by Potree) zoom view on web page	69

List of Tables

3.1	The user evaluation and the system performance results . . .	32
4.1	RGB-D camera comparison	41
4.2	PCL point types	48
4.3	A PCD file FIELDS declarations	49
4.4	3D Map rendering parameters and values	53



Acronyms

AIDC	Automatic Identification and Data Capture. 1
API	Application Programming Interface. 19, 20, 23–25
BRIEF	Binary Robust Independent Elementary Feature. 11
CSS	Cascading Style Sheets. 25
DOM	Document Object Model. 25
EPC	Electronic Product Code. 8
FAST	Features from Accelerated Segment Test. 11
GPU	Graphics Processing Unit. 27, 46
HDR	High-dynamic-range imaging. 13
IMU	Inertial Measurement Unit. 57
IoT	Internet of Things. 1
LiDAR	Light Detection and Ranging. 66
LRF	Laser Range Finder. 41
PCD	Point Cloud Data. 50
PCL	Point Cloud Library. vii, 49
PHP	Hypertext Preprocessor. 5, 19, 20, 26
PLY	Polygon File Format. 51, 52
RANSAC	Random Sample Consensus. 13, 55

Acronyms

RFID	Radio-Frequency Identification. v, 1, 2, 5, 7, 8, 17–20, 23, 27, 36–38
RGB-D	Red, Blue, Green, and Depth. v, vi, 37, 49
RMSE	Root Mean Square Error. 46
ROS	Robot Operating System. 14, 53
RSSI	Received Signal Strength Indicator. 9
rviz	ROS Visualization. vii, 53
SIFT	Scale-invariant Feature Transform. 11
SLAM	Simultaneous Localization and Mapping. 37, 45, 55
SURF	Speeded Up Robust Feature. 10, 11
ToF	Time-of-Flight. 39, 40
TORO	Tree-based netwORk Optimizer. 48
VTK	Visualization Toolkit. 49
WebGL	Web Graphics Library. 52, 66

1

Introduction

“Specialized elements of hardware and software, connected by wires, radio waves and infrared, will be so ubiquitous that no one will notice their presence.”

- Mark Weiser, Scientific American, 1991.

1.1 Motivation

Today, computing is far more ubiquitous than ever before. Ubiquitous computing is embracing every aspect of our daily lives from personal to professional computing anywhere, anytime. With the advent of smart hand-held devices (e.g. smartphone, PDA, etc.), and with the ubiquitous access to the Internet, the usefulness of timely information is now an important part of our busy daily life. Computers are now more connected with physical objects; where, the objects or ‘things’ are embedded with electronics, software, sensors, and connected to the Internet to collect and exchange data. Ubiquitous Computing (UbiComp) and the Internet of Things (IoT) paradigms drastically change the way we use computers now. Computers are more sensor equipped, and thus smart applications are providing more location and context-aware services than before.

The identification of an entity is the first step of locating it within the context or the environment. Automatic Identification and Data Capture (AIDC) is the family of technologies for automatically identifying objects, collecting and entering the data into a computer system without any human intervention [1]. Typical AIDC (also called Auto-ID) systems are the barcode, RFID, smart card, and biometric systems. RFID stands for Radio-Frequency Identification and is a real time wireless identification and tracking of enti-

Chapter 1. Introduction

ties using radio frequency. RFID has been recognized as a promising technology for retail, supply chain management, customer interaction, etc. [2,3]. Unlike a barcode system, with an RFID system, multiple entities or items can be uniquely identified from a remote distance. In an RFID deployed retail store, an RFID tag is attached to an item’s label, so that, the item can be scanned by RFID system to identify, track, and the retrieve related item information almost instantly. As a result, stock keeping is automated, faster, and almost error free. Although, RFID system significantly improve the up-to-date inventory management, the RFID system alone does not present a graphical representation of the objects or entities it is sensing; and of course, it does not provide any visual feedback of the environment within which the objects are located. So far, a user experience of online (or offline) searching for an item of a store is limited to catalog searching. A catalog may or may not be fully updated. In either case, the catalog also does not provide users any graphical feedback of the items’ physical location inside the store.

In April 2010, Google started a pilot project, aiming at extending the Street View service inside business shops [4]. The primary goal was to give this service to their trusted businesses to show the 360° panoramic view of the interior like a virtual tour as if the visitors were walking around inside the store. Google indoor Street View for business still requires a manual process, where, Google certified photographers manually snap photographs with a tripod and upload these photos to a Google server to create the panoramic virtual tour. Besides, Google indoor Street View for business does not have any service for searching and locating a product item with product related information on the panoramic view of the store. Google My Business ¹ is another related initiative of connecting business with local customers that encourages business owners to add indoor photos, update business address and opening hours to promote their products or services through Google map service. An up-to-date indoor view with entity information on that view can be desirable in many scenarios such as retail [5]. For instance, a shopper would like to view, navigate, and browse up-to-date product information on the 3D-like view of the retail store either from in-store or out-store. To better understand the impact of online information and smartphones usage on in-store shopping, Google conducts a survey in association with Ipsos MediaCT ² and Sterling Brands ³ brand research company

¹<https://www.google.com/business/>

²<http://www.ipsos.com/connect/>

³<http://www.sterlingbrands.com/design/about.php>

1.1. Motivation

to understand customer expectations. The survey reports that two-thirds of shoppers could not find the details they needed while visiting the store, and one-third of surveyed customer actually prefer to find additional information using the smartphone rather asking a store employee [6].

A major limitation of currently available Google indoor Street View is the scalability and the lack of continuously updated content. With the current scheme of Google Street View, (1) many businesses may not update their indoor images frequently (as it requires photographers to snap photos and upload these photos manually to the Google server), and (2) present images do not contain any additional information about the products or services, only a panoramic view. An innovative convergence of state-of-the-art technologies can be integrated to bridge the gap of presently available indoor Street View technology towards a real-time or continuously updated indoor 360° synthetic view using a camera and RFID sensors.

A detailed 3D visual model can provide extensive knowledge about the environment. In general, a 3D modeling is processing intensive and time-consuming. The 3D mapping paradigm consists of image and depth data collection, processing, and construction of the 3D geometric model with added texture maps in a virtual environment [7]. A large indoor environment 3D modeling may not be a real-time automated process. A recent development of indoor 3D modeling is 3D point cloud mapping using RGB-D SLAM [8]. The RGB-D SLAM utilizes inexpensive RGB-D depth cameras such as Kinect camera⁴ to capture RGB images with per-pixel depth information. The term SLAM refers to Simultaneous Localization and Mapping, is an active research field in robot mapping. A 3D point cloud map is then generated by utilizing either the robot pose (position) information or the visual odometry computed using an RGB-D camera. However, creating a large 3D point cloud map is still a challenging task as it requires extensive knowledge about different approaches of RGB-D SLAM, different camera sensors, and robotics in general.

Although, an accurate 3D model can depict the actual environment as it is, a 3D model alone may not be useful without an up-to-date entity information on the view. In general, a 3D model annotation is also a challenging task, in particular when an environment view constantly changes (e.g. in a retail

⁴<http://www.xbox.com/en-US/xbox-one/accessories/kinect-for-xbox-one>

Chapter 1. Introduction

shop). Sensors like RFID and RGB-D camera on a robot can be used to create the automated indoor 3D view with item information on it.

1.2 Research Objective

The key research objective is - “to use a robot equipped with camera and RFID sensors to create an automated indoor synthetic view of a retail store, so that, users can browse, navigate, and locate the product items with related information on that view.”

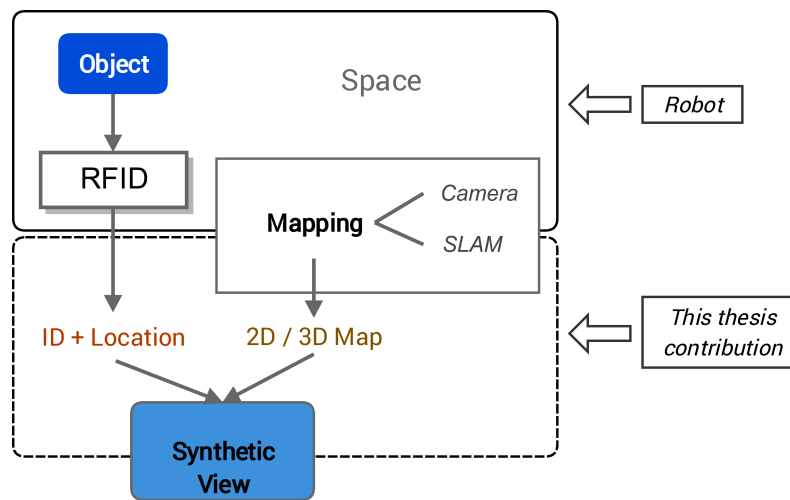


Figure 1.1: Research objective

The research aims at merging object and space information obtained by a robot into a synthetic view that user can navigate as if he or she were in the real store. A mobile RFID system on a robot can be utilized to identify RFID-tagged objects in a space periodically. A camera sensor on the robot can also be used to generate the real time or updated map. A synthetic view can be created by combining the environment (2D or 3D map) with object (ID + location) information, so that, users can browse, search, locate and view items on that view (Figure 1.1).

1.3 Research Methodology

To achieve the research objective above, we first explore the different state-of-the-art technologies related to image processing, and 3D point cloud

1.4. Approaches

mapping methods. Here, we explore different types of cameras that are suitable for map view creation, their strengths and limitations. Then we explore different tools, techniques and methods of projecting RFID-obtained information on the map view. Here, we explore HTML DOM parsing Ajax (for client-side processing) and PHP (for server side processing), the point cloud library functionalities, and ROS visualization packages etc. Finally, the implementation approaches are focused as below.

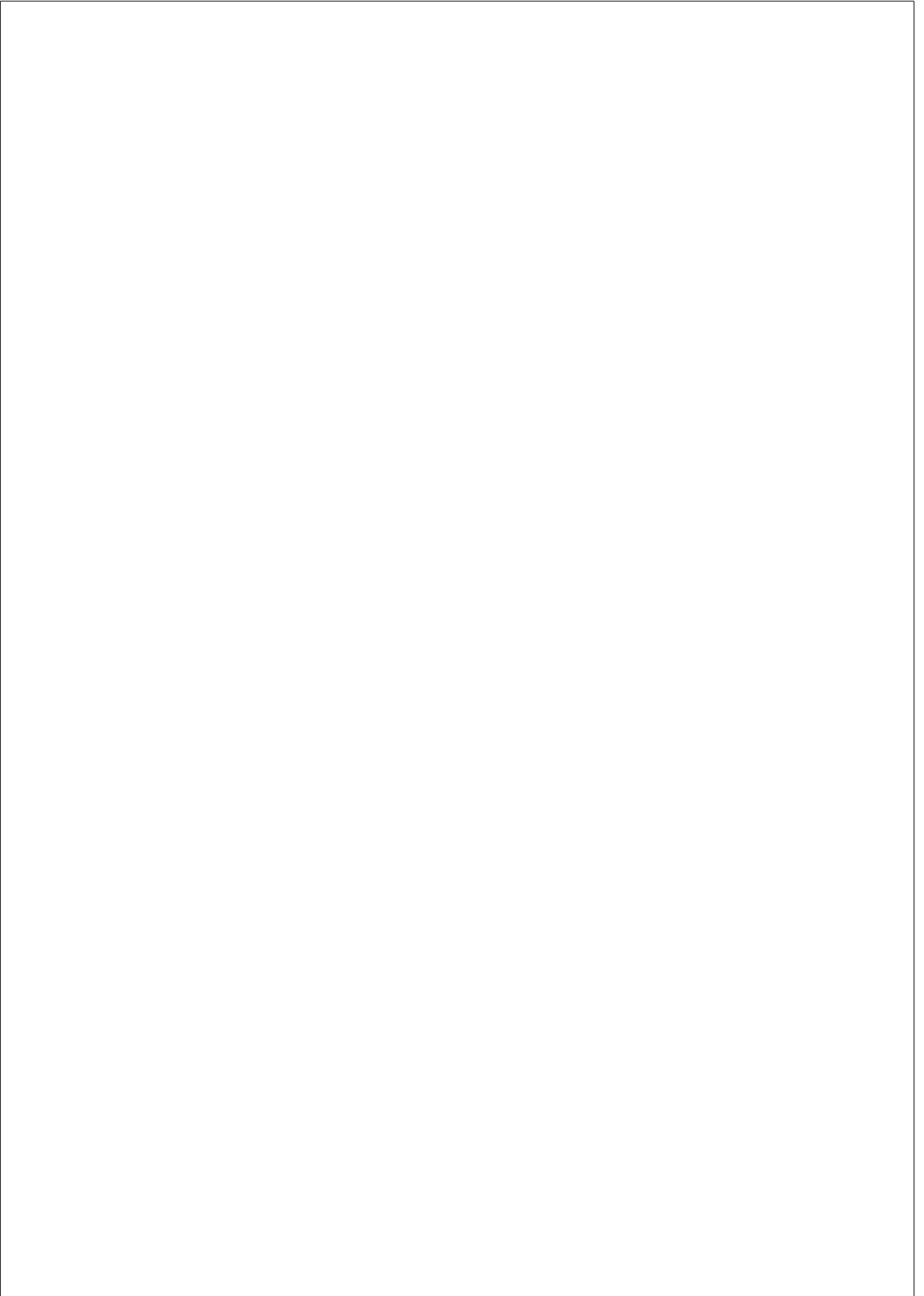
1.4 Approaches

To implement the research objective, we focus on two approaches - 1) Google Street View-like indoor view, and 2) 3D indoor view using camera sensor on a robot. The implementation approach of Indoor 360° synthetic view using camera and RFID sensor is presented in chapter 3, and the implementation approach of 3D point cloud map using RGB-D SLAM is presented in chapter 4.

1.5 Structure of this dissertation

The remainder of this dissertation is structured as follows:

- **Chapter 2:** State-of-the-art review. This chapter outlines the related state-of-the-art review of RFID and 3D indoor mapping.
- **Chapter 3:** Indoor 360° synthetic view using a camera and RFID sensors on a robot. This chapter describes the proposed solution of street View-like indoor view and projecting RFID obtained information on that view.
- **Chapter 4:** 3D synthetic view using RGB-D SLAM and RFID sensors on a robot. This chapter describes the proposed system of creating indoor 3D point cloud map using RGB-D SLAM and projecting RFID obtained information on that 3D view.
- **Chapter 5:** Conclusion.



2

State-of-the-art review

This chapter outlines the state-of-the-art reviews related to indoor environment mapping by a robot using RFID and camera sensors where the users would be able to view, identify, browse, and locate an RFID-obtained product information. At first, we explore the RFID technology. Then we review the different 3D robot mapping tools, techniques, and algorithms. Then we consider the different feature detection algorithms and their applications. Finally, a review of open-source image stitching tools is briefly presented.

2.1 Radio-Frequency Identification

Radio-Frequency Identification (RFID) is the automatic identification and tracking of tags attached to a product, animal, or a person using radio frequency [9]. Unlike a barcode system, an RFID system does not require any direct line-of-sight and identifies an item uniquely rather identifying the item type. Moreover, an RFID system can scan multiple items at a time, rather than scanning items one by one with a barcode system. Besides the entity identification, an RFID systems can also be configured to determine an approximate location of the entity within an environment. Important RFID applications include - the smart labels for products, the smart key for vehicle ignition, vehicle toll collection, rail car and shipping container identification, library books identification, passport identification, cattle and pets tracking, medical appliance identification, smart cards personal identification and payment, etc. An RFID system consists of RFID tags, RFID reader, and a server to update RFID inventory in a database table.

Chapter 2. State-of-the-art review

2.1.1 RFID Tag

An RFID tag (also called a transponder) is an integrated circuit (IC) with an antenna inlay to receive radio-frequency and respond back the identifier number stored in its memory to an RFID reader. The unique identifier is called an electronic product code (EPC) [10]. An EPC identifies a unique RFID tagged entity, object or asset. An RFID tag can be passive, semi-passive or active depending on its power supply. A passive tag does not require any battery power to operate; rather it uses the electromagnetic field (EMF) of the reader. Usually, a passive tag works up to a short range as it does not have any power supply. A semi-passive tag is battery powered to run its chip circuitry. The battery power improves the range of a semi-passive tag. An active RFID tag is self-powered by a battery and continuously able to broadcast its signal to a reader or other active tags depending on the class of tag it is. The passive UHF RFID tags (WEB G2iL [11], MINIWEB [12], and SHORTDIPOLE [13]) are shown in Figure 2.1.



(a) SmartTrac UHF WEB G2iL tag



(b) SmartTrac UHF MINIWEB tag



(c) SmartTrac SHORTDIPOLE UHF tag

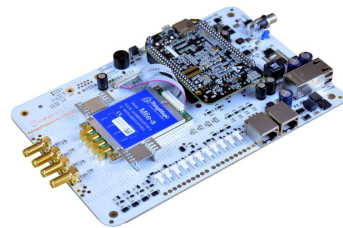
Figure 2.1: RFID UHF tags

RFID operates in different frequency bands depending on different RFID tag class or the country radio frequency standard. The High Frequency (HF) tags usually operate in between 138 KHz to 13.95 MHz frequency band. The Ultra High Frequency (UHF) typically operates between 860 MHz to 960 MHz frequency band with a read range of approximately 10 meters. Passive UHF RFID tags are usually used in retails due to its small size, no battery power requirements with a suitable read range.

2.2. Feature Detection and Matching Algorithms

2.1.2 RFID Reader

The RFID reader (also called the interrogator) uses antenna/s to send and receive radio signals to and from the RFID tags. When a signal is received from a tag, the tag’s EPC is added in the scanned EPC list. All EPCs scanned by the RFID reader are usually saved with other information such as time of reading, reader’s ID, read count, received signal strength indicator (RSSI) value, etc. RFID readers can be either fixed (e.g. AdvanReader 150 UHF RFID Reader [14] or handheld (e.g. AdvanScan Handheld UHF RFID Reader [15] as shown in Figure 2.2. A fixed RFID reader may have multiple antenna ports, and each port can further be extended with multiplexers if needed. The handheld RFID readers are usually used manually, where the fixed readers affixed with a permanent structure such as at the back of a shelf.



(a) Keonn AdvanReader 150 UHF RFID Reader



(b) Keonn AdvanScan Handheld UHF RFID Reader

Figure 2.2: RFID readers

2.2 Feature Detection and Matching Algorithms

A feature is a point of interest of an image, typically used for matching or comparing the image with other images. Feature detection is the process of finding interest points in an image; and the algorithm that detects features is called a feature detector. Matching features across images requires to computing the feature description or computing patches around a point of interest. The process of computing feature descriptors is called feature extraction. The image matching is then can be computed by matching feature descriptor values. Image matching is the fundamental aspect in computer vision applications such as scene recognition, image stitching, 3D reconstruction, object or motion tracking, etc.

Chapter 2. State-of-the-art review

2.2.1 SIFT

Scale Invariant Feature Transform (SIFT) (2004) by Lowe [16], is a popular feature detector and descriptor for extracting invariant features from images. The keypoints are usually extracted by the SIFT detector, and their descriptors are computed by the SIFT descriptor. A SIFT feature (keypoint) descriptor is computed by sampling the image gradient magnitudes and orientation around the keypoint using the level of Gaussian blur of the image. The SIFT keypoint descriptors are highly distinctive as it able to find its correct match in a large dataset of features. The orientation invariance is achieved by rotating the gradient relative to the keypoint orientation. An example of OpenSIFT ¹ in Figure 2.3 is showing the SIFT features matched between two images.

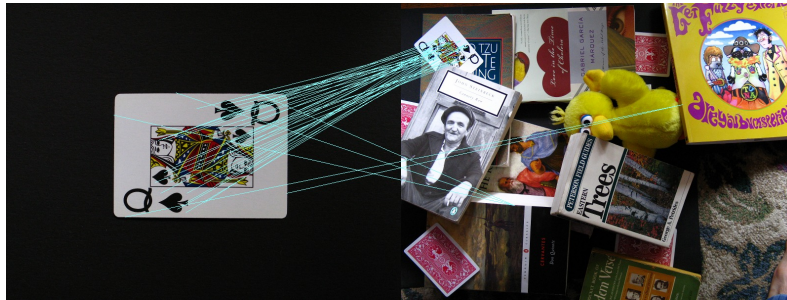


Figure 2.3: SIFT features matched between two images, a demonstration of OpenSIFT

2.2.2 SURF

The Speeded Up Robust Feature (SURF) (2006) by Bay et al. [17] is an SIFT-like feature detector and descriptor, however, with the concern of speeded up performance by lowering computational time. SURF lowers the computational time through an efficient use of Hessian matrix approximation for the detector and sums of approximated 2D Haar wavelet responses for the descriptor. Unlike SIFT, SURF uses stack without downsampling the images at higher levels of a pyramid that results in having images in the same resolution. SURF uses integral images that allow the computation of rectangular box filters in a near constant time. The author presents the results of performance improvement over SIFT. An example of real-time scale and rotation invariance object detection with SURF can be seen as Figure 2.4 [18].

¹<https://robwhess.github.io/opensift/>

2.2. Feature Detection and Matching Algorithms

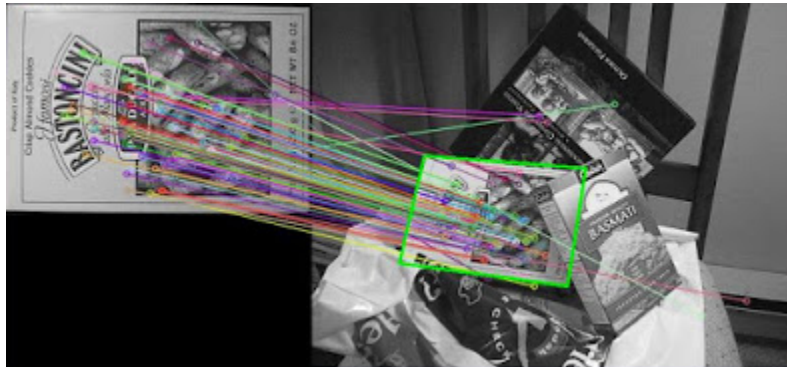


Figure 2.4: Scale and rotation in-variance in object detection with SURF

2.2.3 FAST

Features from Accelerated Segment Test (FAST) (2006) is a corner detection technique presented by Rosten & Drummond [19] for real-time full-frame feature detection using machine learning. The FAST detector uses a circle of 16 pixels to classify whether a candidate point, p is a corner. If a set of N contiguous pixels in the circle is brighter than the intensity of that point p plus a threshold value t or all darker than the intensity p minus threshold value t , then p is classified as a corner. However, there are known limitations to the algorithm. Firstly, for $N < 12$, the number of interest points detected are very high. Secondly, the speed of the algorithm is determined by the order in which the 16 pixels are queried. The Machine learning is added to the approach, to overcome these limitations. The fast computational speed makes the FAST detector suitable for large video frame processing for real-time tracking, SLAM applications, etc.

2.2.4 BRIEF

The Binary Robust Independent Elementary Feature (BRIEF) (2010) by Calonder et al. [20] is a short, fast and efficient feature descriptor that directly computes binary strings from image patches. BRIEF is aimed at faster to compute, faster to match, and memory efficiency. The binary string comparisons are done with Hamming distance that can be extremely fast with modern CPU instruction sets. The use of a BRIEF descriptor supposes the key points are already detected, this can be done with a detector such as SIFT or SURF. BRIEF can be quite useful when there are limited computational resources to compute large datasets.

2.2.5 ORB

As the name indicates, Oriented FAST and Rotated BRIEF (ORB) is based on FAST detector and BRIEF descriptor presented by Rublee et al. [21]. FAST excels in its performance. However, FAST features do not have orientation component. Rublee et al. added the Harris corner measurement to filter FAST features at each level in the pyramid for computed orientation of keypoints. FAST with Harris corner filtering is called oFAST (oriented FAST). To make the BRIEF descriptor invariant to in-plane rotation, rather than computing each set of rotations and perspective wraps of each image patch (which is expensive to compute), a more efficient method is to steer BRIEF according to the orientation of the keypoints is called rBRIEF (rotated BRIEF) [21]. The combination of the oFAST detector and the rBRIEF descriptor is called ORB. A typical matching result using ORB on images with viewpoint change can be seen in Figure 2.5 [21].

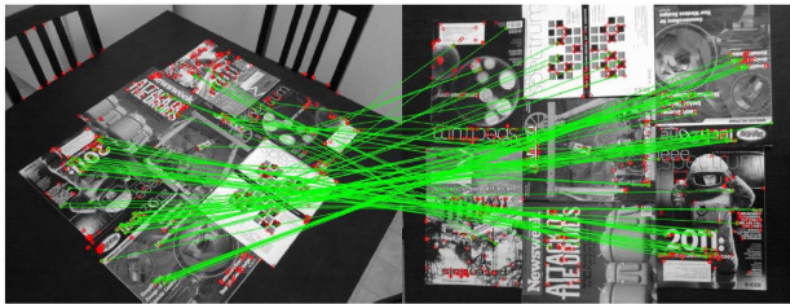


Figure 2.5: Keypoint matching using ORB

2.3 Panorama Stitching

To implement the street view-like indoor view and navigation approach (section 1.4), an automated panorama stitching is required. Panorama (image) stitching is the process of combining multiple images with overlapping fields of view to a single large image. Typically, an image stitching process consists of - feature (keypoint) detection, image registration, and image calibration. The keypoint detection is the process of finding correspondences between overlapped images by a feature detector. The image registration involves matching features in a set of images or using direct alignment to minimize the sum of absolute differences between overlapping images using RANSAC (Random Sample Consensus) algorithm. The image calibration

2.4. ROS

aims to minimize the differences between an ideal camera lens model with the camera lens it was used for taking images. Usually, distortion, exposure, vignetting, etc., are corrected in this phase with correcting alignment and blending adjustments. AutoStitch and PTGui are two proprietary image stitching software tools. AutoStitch ² is a commercial software developed by Lowe & Brown, that uses SIFT and RANSAC algorithms. PTGui ³ is a proprietary panoramic image stitching software for Windows and Mac OS X. Both AutoStitch and PTGui require manual image manipulation and do not provide a command-line interface to automate the stitching process. Hugin PanoTools scripts are open source alternative to AutoStitch or PTGui. The PanoTools or Panorama Tools is a set of open-source program and libraries for immersive imaging, originally written by professor Helmut Dersch, University of Applied Sciences Furtwangen [22]. libpano13 is the latest PanoTools library for projecting and blending multiple images into an immersive panorama. Hugin ⁴ is the open source cross-platform front-end of PanoTools ⁵ and enblend/enfuse ⁶ developed by Pablo d’Angelo and others for panoramic stitching and HDR merging (Figure 2.6). Hugin stitching process can be automated and enhanced by bash scripting by calling its command-line interface that allows the flexibility of parametric customization (e.g. field of view, projection, canvas geometry, blending, exposure fusion, etc.). Hugin control point detector is called *cpfnd* for generating control points for both single and multi-row image stitching. The other essential tools are - *cpclean* for control point pruning, *autooptimiser* for project geometry optimizing, *nona* is the hugin rendering engine, *enblend* for merging overlapped images with multi-resolution splines, and *enfuse* for merging images with exposure fusion. Hugin supports all major projections including - rectilinear, cylindrical, fish-eye, Mercator, equirectangular projection, etc.

2.4 ROS

The Robot Operating System (ROS) ⁷ is a set of open-source libraries and tools to build robot applications, primarily developed at Willow Garage ⁸, a

²<http://matthewalunbrown.com/autostitch/autostitch.html>

³<https://www.ptgui.com/>

⁴<http://hugin.sourceforge.net/>

⁵<http://panotools.sourceforge.net/>

⁶<http://enblend.sourceforge.net>

⁷<http://www.ros.org/>

⁸<https://www.willowgarage.com/>

Chapter 2. State-of-the-art review

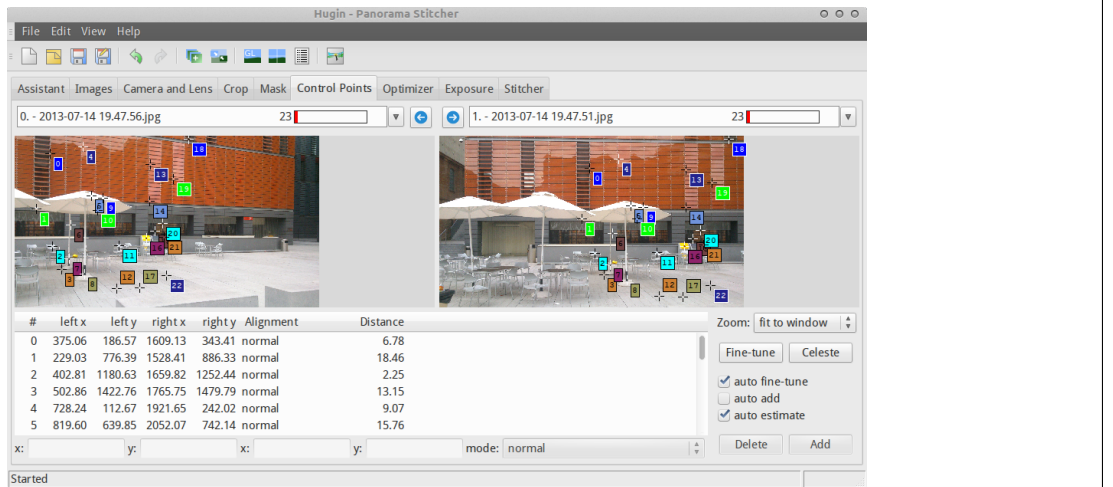


Figure 2.6: Hugin control point detection

robotic research institute. ROS provides operating system-like functionalities including hardware abstraction, device drivers, libraries, visualizers, distributed architecture of message-passing, package management, etc. ROS is language independent, and the main client libraries are written in C++, Python, and LISP. At present, ROS has become a de facto standard for robotic application development. ROS starts with an ‘ROS master’, the main control program that lets other ROS software components, called ‘Node’ to find and communicate with other Nodes by sending (publishing) and receiving (subscribing) messages through a reference bus called a ‘Topic’. As ROS architecture is highly modular and distributed, one or more computer running different nodes can listen to a particular ROS master and accomplish the task. For example, a camera perception can be acquired by a robot onboard computer, and the 3D map can be subscribed and viewed by other computers by listening to robot ‘ROS master’ using a network connectivity. ROS launch files are used to configure and run many nodes at once. From powering up a robot to performing a particular task such as RGB-D SLAM mapping (presented in chapter 4) requires ROS to be installed and configured in the robot onboard computer first.

2.5 Review of Patents

- **US Patent 20150170258**

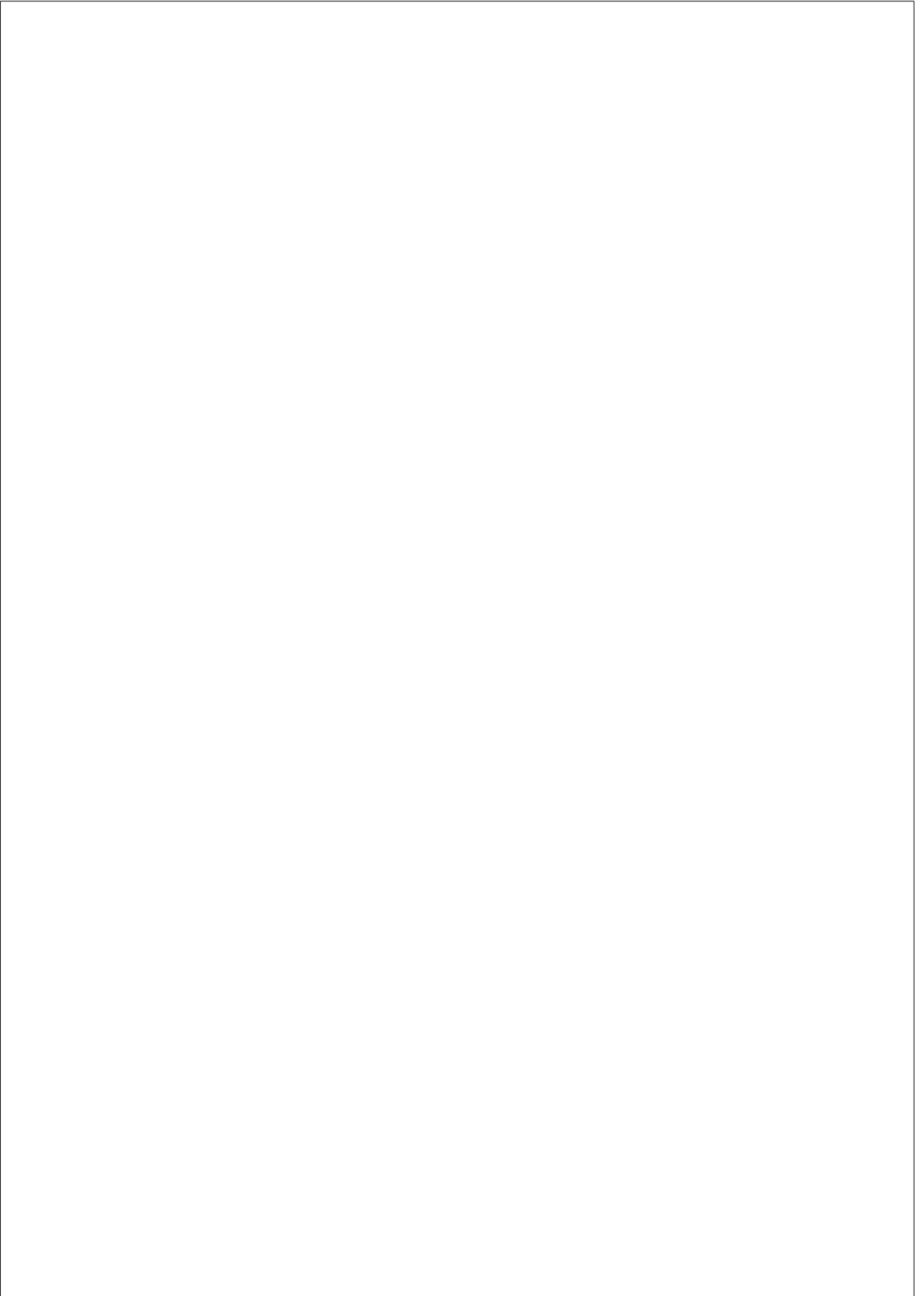
Kulig et al. [23] presents a method for searching and location prod-

2.5. Review of Patents

uct information using a mobile device on a generated map from retail and planogram data. The method would require downloading vendor supplied product location and merchandising fixture data to the mobile device. The product location sought by the user is indicated by a location visualization.

- **US Patent 20150312722**

Samsung presents a method of location determination and mapping through crowdsourcing [24]. Crowdsourcing is a business term which means the process of obtaining services or contents from a large group of online people or users. The patent proposes a method of indoor map generation by crowdsourcing that includes indoor location determination using radio frequency (e.g. Wi-Fi) data from a communication device, showing one or more routes of the map.



3

Indoor 360° synthetic view using a camera and RFID sensors

A 360° panoramic view provides an all-around view of the environment giving a feeling of presence that users can roam around the place and explore. Google Street View is one of such experiences that let people explore 360 ° panoramas of outdoor places. Like an outdoor place, technology assisted indoor view, and information browsing may be a need for many scenarios such as shopping. For instance, shoppers would like to search and browse products on the indoor view of the store with product information such as price, colors, sizes, approximate product location within the store, being or not being there physically. RFID can be used to obtain information about objects present in a physical space, including their approximate location. Handheld RFID readers, smart shelves, zenithal antennas, and robots can be used to obtain information with varying time and space resolutions. In this chapter, we present a system that projects this information on a panoramic view of a retail store, allowing users to navigate virtually around the store, but instead of being given static information, obtaining quasi-real-time information about the products as they are in the store. When a user clicks on the image of a shelf in the panorama, information is shown about the products that were at or near that position the last time an RFID-based inventory was obtained. A novel system implementation, integration, and test results in a real retail store are presented in this chapter.

3.1 Introduction

Map applications have become an essential part of daily life for locating streets, buildings, shopping stores, transportation, etc. At present, users can search and view an outdoor place without being there physically. Google Maps have enhanced the map service with 360-degree panoramic views of the streets. Google has also started a commercial service called Google Business View [25], which shows 360-degree virtual tour of the business interior environment and a non-commercial service called Google Art Project [26], which shows historically significant places with brief static annotations. However, these indoor virtual browsing services are not designed to update panoramas and annotations frequently. In a retail store, inventory information changes constantly and technologies like the barcode and RFID are being used for up-to-date stock keeping. RFID has become a promising technology in retail, inventory management, supply chain management, and retail marketing, etc. [2]. RFID systems can detect the presence or absence of an object and can also be configured to determine the approximate location of the object within an environment. However, an RFID systems alone can not present a graphical representation of the products they identify and of course, they can not provide any visual feedback of the environment within which they are located. In this chapter, we present the implementation and the integration of the “Store view” system for retail store virtual browsing by creating a 360-degree Street View-like panoramas and utilizing a robot equipped with an RFID system that allows users to search, navigate and locate products on the panoramic view of the physical store.

3.2 Related Work to Indoor 360° Synthetic View

Colbert et al. point out that without additional capture devices, images can not be precisely geo-located, and they present a method of iterative vision based pose estimation of business indoor images to get geo-located images for panorama making [27]. A vision-based pose estimation can be statistically optimized; however, their proposed system still requires user-guided optimization and does not indicate how to display object information on a panorama. NavVis is a Google Street View-like mapping system that utilizes a trolley equipped with two laser scanners and six cameras to capture photos for creating maps allowing users to navigate virtually through the indoor such as in museums with smartphones [28]. The trolley is re-

3.3. The Proposed System

quired to be pushed by a human operator, and it also does not provide any context information on panoramas. Another alternative to the panorama based services is the computer vision-based systems. Even so, a smooth operation of vision based systems is challenging in real-world environments due to many factors including lighting changes, pose changes, motion blur, etc. [29]. In particular, in a retail scenario, identifying products with the same color but in different sizes or distinguishing folded and unfolded apparels can be difficult. Retail inventorying by a robot such as MetaBot [30], is designed to navigate autonomously in retail environments and to execute micro-merchandising tasks commanded by retail staffs or scheduled at pre-defined time-intervals. The current implementation of the “Store view” system presented in this chapter is the continuation of the research published by Carreras et al. [5] which combines the ideas of the mobile robotic RFID inventorying with the product information projection on the 360° panoramic view of retail environments.

3.3 The Proposed System

3.3.1 Overview and Architecture

The objective of the “Store view” system is to enable a panorama based store view, navigation and projection of RFID obtained product information on the 360° panorama by integrating with the information obtained by an RFID-equipped robot and camera sensor. The “Store view” system consists of both the client and the server side components. The server side components are RFID host software with an inventory database and a panorama image provider. The client-side component is the user interface of “Store view”; which is programmed with HTML5, JavaScript and Google Map JavaScript API (v3) [31]. The RFID software is responsible for updating RFID tag information to the inventory database. The panorama provider module is responsible for creating all panoramas of the retail store using the Hugin shell script [32]. To implement the dynamic client-side features, an AJAX updater and a JavaScript DOM parser has been implemented. In the server-side, PHP scripts have been implemented for the database connectivity and server-side processing. The “Store view” system architecture is presented in Figure 3.1.

Chapter 3. Indoor 360° synthetic view using camera and RFID sensor

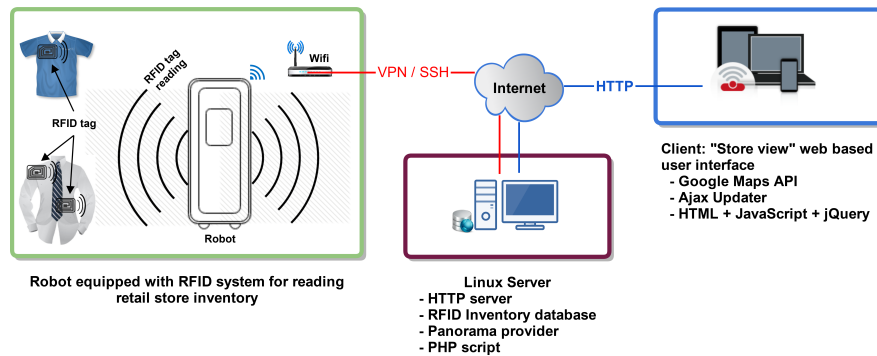


Figure 3.1: The “Store view” system architecture

3.3.2 Implementation

To implement the proposed system, at first, an automated panorama stitching process is implemented. Then, a custom indoor street view is implemented using Google Street View JavaScript API. Here, we incorporate the RFID obtained product information with the store panoramic view by HTML DOM parsing, JavaScript, and PHP programming. The connectivity between the robot and the server computer is through Wi-Fi, and the web server can be accessed from any computer using HTTP web browser. To locate a product on the store panorama, we implement a scheme of the store and product coordinates in the form of X, Y, Z that can be applied to any store. The implementation and integration details are outlined below.

RFID Hardware in AdvanRobot

AdvanRobot¹ is RFID system equipped robot manufactured by Keonn Technologies. AdvanRobot is equipped with twelve RF antennas, two multiplexers, and an RFID reader. The extended base structure of the robot holds all antennas on two sides, left and right side of the robot at different heights. Antennas (Advantenna-P22) are connected to multiplexers (AdvanMux-8), and the multiplexers are connected to RFID reader (AdvanReader-100) [33] (Figure 3.2). A laser sensor enables the robot to generate the indoor navigation map in (X, Y) parameters. All antennas are measured and attached to get the location of the Z value which is the height at which an antenna is fixed with the robot structure. The current position of the robot gives the

¹<http://keonn.com/systems/view-all-2/inventory-robots.html>

3.3. The Proposed System

X , Y and the “ahead” orientation of the robot determines which side (left or right) and at which Z height each antenna senses the tags. The reference store coordinate $(X, Y) = (0, 0)$ is configurable from any corner of the store. As a result, the robot module can determine the current X , Y position of the robot and the approximate location of a tag, on each side at different Z height value.

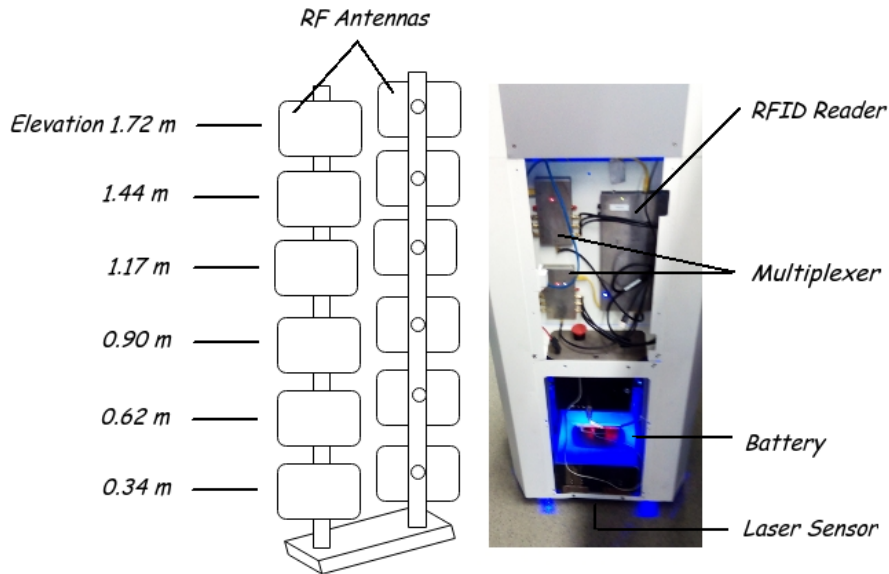


Figure 3.2: RFID system on the robot

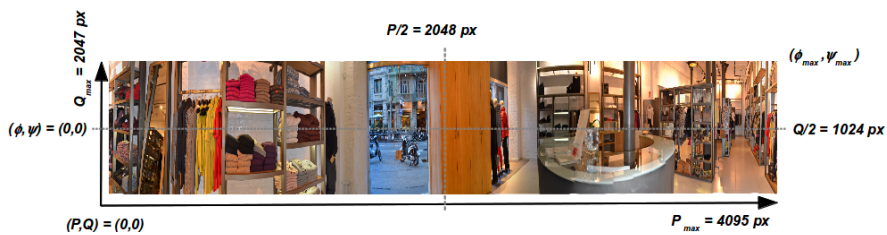


Figure 3.3: A panorama of the actual retail store created by Hugin shell script (panorama 1, store section 1)

Automating 360° Panorama stitching

The panoramic view is created by taking overlapping photographs of all around the place throughout the 360 degrees (with 15-degree clockwise ro-

Chapter 3. Indoor 360° synthetic view using camera and RFID sensor

tation steps) and stitching them together. Although, images photographed can be stitched together manually with commercial tools such as PTgui, Autostitch etc., we take an automated approach and thus chose to use Hugin shell script with the ImageMagick command line scripts which is useful for any size store. We use the command *nona* for remapping, and *enblend* for blending the remapped photos into a finished TIFF image. We utilized hugin *cpfind* for identifying control points between overlapped images which is an alternative to SIFT implementation *autopano-sift-c*. From the output point of view, both *cpfind* and *autopano-sift-c* generate similar results while *cpfind* accepts more parameters. Then we optimize the process by running *autooptimizer* to optimize photometric parameters such as alignment, output projection angles, etc. All the above commands are pipelined as a batch process with a Linux shell script as stated in the following self-descriptive pseudo-process (similar to the process of Brown and Lowe [34]).

Pseudo-process: 360° Panorama stitching

Input: n overlapped images

- I. **create** project file by reading all input images using *pto_gen*
- II. **find** control points using *cpfind*
 - (a) **do** pairwise and multirow matching
 - (b) for each images
 - i. **find** m candidate matching images with maximum number of feature matches
 - ii. **apply** RANSAC to solve homography between pair of images
- III. **optimize** control points using *cpclean*
- IV. **find** vertical lines using *linefind*
- V. **do** optimize image positions, photometric optimization, straighten panorama using *autooptimiser*
- VI. **set** panorama output projection (360°, 180°), optimal crop and output size using *pano_modify*
- VII. **create** stitching makefile

3.3. The Proposed System

VIII. **make** panorama, apply *enblend* and *enfuse*

Output: 360° Panorama

The output panorama of the above process is an equirectangular projected (aspect ratio 2:1) 360° panorama. An output panorama is presented in Figure 3.3. When all panoramas of the store are created, we inter-link them by custom JavaScript programming with the Google Street View API [31]. All inter-linked panoramas with the Google Maps API enable users to navigate through the “Store view”.

Store and Object Coordinates

A retail store can be measured with X , Y , and Z coordinates representing the length, width, and the height of the store in measurable units such as in centimeters (Figure 3.4). Any retail product has its location that can also be represented with XYZ coordinates. In the Figure, all product locations inside the store ranges from $(X, Y, Z) = (0, 0, 0)$ cm to $(750, 1750, 300)$ cm. Let’s say, the product “Short Ski Jacket” location coordinates $(X_1, Y_1, Z_1) = (433, 60, 170)$ are recorded by the RFID system of the robot. And if we would like to locate this “Short Ski Jacket” from a panorama capture center point such as $(X_0, Y_0, Z_0) = (212, 225, 135)$, in spherical coordinate system, we need to calculate the triplet of (ρ, ϕ, ψ) which gives the radial distance, azimuth angle and elevation angle calculated by the equations (3.1), (3.2), and (3.3) respectively. A panorama capture point (X_0, Y_0, Z_0) is the panorama center point where the robot stops for capturing images all around for a particular panorama. At present, all panorama capture points are manually indicated in the robot navigation plan and recorded in the database for a particular store.

$$\rho = \sqrt{(X_1 - X_0)^2 + (Y_1 - Y_0)^2} \quad (3.1)$$

$$\phi = \arctan\left(\frac{|Y_1 - Y_0|}{|X_1 - X_0|}\right) \quad (3.2)$$

$$\psi = \arctan\left(\frac{|Z_1 - Z_0|}{\rho}\right) \quad (3.3)$$

By applying equation (3.1), (3.2), and (3.3), we get $\phi = 39.5^\circ \approx 40^\circ$, and the $\psi = 5.5^\circ \approx 6^\circ$. In Google Street View API, the azimuth angle is called

Chapter 3. Indoor 360° synthetic view using camera and RFID sensor

heading, and the elevation angle is called *pitch*. The *heading* and the *pitch* angles define the Street View point-of-view (POV). Thus, by calculating the POV value we can manipulate the rotation angles to show the position of the object on the panorama. The default POV *pitch* $\psi = 0^\circ$ is always the panorama height divided by 2. A positive pitch value shows the upper half and a negative pitch value shows the lower half of the panorama. We adjust the POV *heading* angle to make sure that the angle $\phi = 0^\circ$ and $\psi = 0^\circ$ starts to the left of the panorama image in Figure 3.3. Next, we describe how to map the object coordinates to the panorama pixel coordinates.

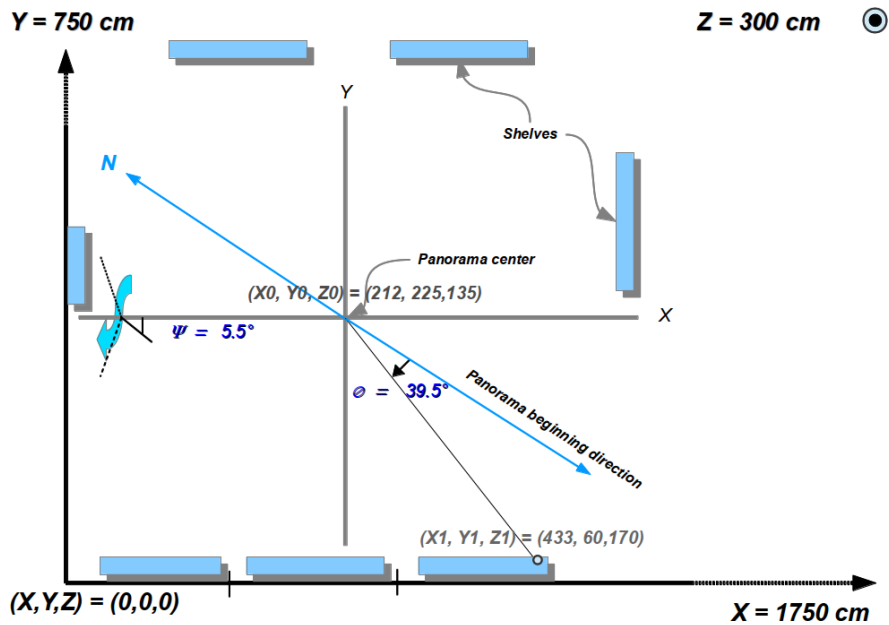


Figure 3.4: The ϕ and ψ calculation by the equations 3.2 and 3.3

Panorama Coordinates

To implement the virtual browsing of products, we need a way to calculate the panorama coordinates in accordance with Street View API. Google Street View service has coined the term “world size”, which is the size of the panorama. We set the panorama size to 4096 pixels by 2048 pixels world size. As the panorama starts from (0, 0) pixel value, the P_{max} value is 4095, and Q_{max} value is 2047 as shown in Figure 3.3. A panorama image can be measured in width and height pixel differences while matching with

3.3. The Proposed System

the street view *heading* and *pitch* rotations. As a result any point (P, Q) can be located by calculating the location P pixel and Q pixel of the panorama with the equations (3.4) and (3.5).

$$P = \left(\frac{\phi - \phi_{min}}{360} \right) \times (P_{max} - 1) \quad (3.4)$$

$$Q = \left(\frac{Q_{max} + 1}{2} \right) \times \left(1 + \frac{\tan \psi}{\tan \psi_{max}} \right) \quad (3.5)$$

$$\text{where, } \psi_{max} = \arctan \frac{h}{2f}$$

with, $h = \text{height of the camera CCD,}$

$f = \text{focal length of the camera}$

These equations are trivial from the point of computer graphics; however, they play an important role with Street view API. For any object location in terms of (X, Y, Z) , we can compute corresponding (P, Q) of the panorama and show inventory information on that position.

Locating objects on the panorama

By using both the object coordinates and the store coordinates, the “Store view” system computes the P and Q coordinates of the panorama as well as the ϕ and ψ angles to rotate the POV to highlight the product. A real store possibly has more than one panorama (Figure 3.5, 3.7). In this case, the “Store view” system calculates the nearest panorama from the product location and thus, any point on any panorama can be located and displayed.

Projecting Information Using HTML DOM Parsing

In previous sections, we have discussed how to locate the object on the panorama. In this section, we discuss how to fetch information from the database and display the product information using HTML DOM parsing with JavaScript. Document Object Model (DOM) is a cross-platform, language-independent convention for representing and interacting with HTML, XHTML and XML documents. JavaScript is a client-side scripting language that can run within the Internet browsers and accomplish many tasks such as dynamically manipulating HTML DOM to display dynamic text and images etc.

Chapter 3. Indoor 360° synthetic view using camera and RFID sensor



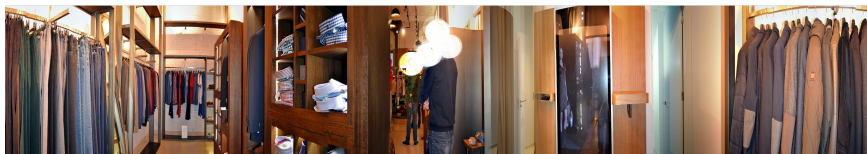
(a) Panorama 2, store section 2



(b) Panorama 3, store section 3



(c) Panorama 4, store section 4



(d) Panorama 5, store section 5



(e) Panorama 6, store section 6



(f) Panorama 7, store section 7

Figure 3.5: Panoramas of different sections of the store

3.3. The Proposed System

A web page is essentially a combination of HTML for the structure of the page and CSS (Cascading Style Sheet) for the presentation of the HTML elements. As JavaScript can be programmed to access all the HTML DOM elements, the CSS of the page can also be manipulated and updated instantly on the browser to achieve any dynamic information on a web page. To fetch information from the database a combination of PHP (programmed to run in the web server) and the asynchronous JavaScript calls (programmed in the client browser) can be utilized. An Ajax function is used to fetch updated product list from the database and create HTML ‘li’ elements for each item by browser asynchronously.

When the user moves around with the mouse or touchpad, the movement is calculated as panorama *heading* and *pitch* value change which in turns are calculated as P , Q value of the panorama coordinate. The current panorama can always be identified by panorama providers’ *getPano()* method. Also, an HTML DOM element can be returned by the standard JavaScript methods named *getElementById()* and *getElementsByName()*. The P_{min} , P_{max} , Q_{min} , Q_{max} are the ranges of panorama coordinates (P , Q) discussed in earlier sections. The HTML DOM parsing that is essential for the virtual product browsing in “Store view” system is presented in Algorithm 1.

Algorithm 1: Pseudo algorithm for highlighting a product in the virtual mode

```

1: procedure HIGHLIGHTELEMENT
2:   for  $i < \text{list.length}$  do
3:      $\text{div} = \text{document.getElementById('browser')}$ ;
4:      $\text{list} = \text{m.getElementsByTagName('li')}$ ;
5:      $\text{item} = \text{list}[i].\text{id}$ ;
6:      $\text{tokens} = \text{item.split("delim", length)}$ ;
7:     if
8:        $\text{tokens}[\text{panoIdIndex}] == \text{panoid} \ \&\&$ 
9:        $(\text{tokens}[\text{itemP}] > P_{min} \ \&\& \ \text{tokens}[\text{itemP}] < \text{maxP}) \ \&\&$ 
10:       $(\text{tokens}[\text{itemQ}] > Q_{min} \ \&\& \ \text{tokens}[\text{itemQ}] < Q_{max})$ 
11:       $\text{highlightBox.innerHTML} = \text{'<div}$ 
12:       $\text{style='...'.itemInfo}$ 
13:       $\text{'>'}$ ;
14:      return; then
15:    end if
16:  end for
17: end procedure

```

3.4 Experimental Results

3.4.1 Synthetic View of a Real Store

We conduct the experiments at the Roberto Verino’s Barcelona store where 1065 product items were tagged with UHF RFID tags. The store geometry is 17.5 meters (X) long, 7 meters wide (Y), and 3 meters high (Z). The robot was set to move at the speed of 10 cm per second on a predefined walkway in the store to record RFID tags and their approximate locations (Figure 3.6). A 3D scatter plot depicts the store-wide RFID inventory in different locations of the store in Figure 3.8.



Figure 3.6: AdvanRobot is reading RFID tags and their approximate locations by going around the retail store.

The computer hardware used for panoramas stitching is Intel Core i5 quad-core processor with 8GB RAM and without any GPU hardware. Each panorama was stitched with 24 photos, taking approximately 25 minutes to complete stitching with a single thread. With a multi-threaded panorama stitching script, we achieved approximately one hour for three simultaneous (≈ 20 minutes for each) panorama stitching with the above-mentioned computer configuration. With a simultaneous processing, with free RAM takes less time to complete the task due to less hard disk swapping. In the stitching script, we make sure that there was at least five keypoints to match images to maintain acceptable quality panorama stitching.

3.4. Experimental Results

The “Store view” user interface (UI) presented in Figure 3.9b and 3.9c is created with 7 panoramas covering the whole store, where users are able to search, locate and virtually browse products of the store. The “Store view” UI highlights the product location with product details based on the RFID inventory location recorded by RFID system on the Robot. As the clothes can be of different sizes and shapes, and also can be displayed on a hanger, or may be stacked on the shelves, the “Store view” system highlights the approximate product location of the product with a blue highlight box where the center of the box indicates the product location.

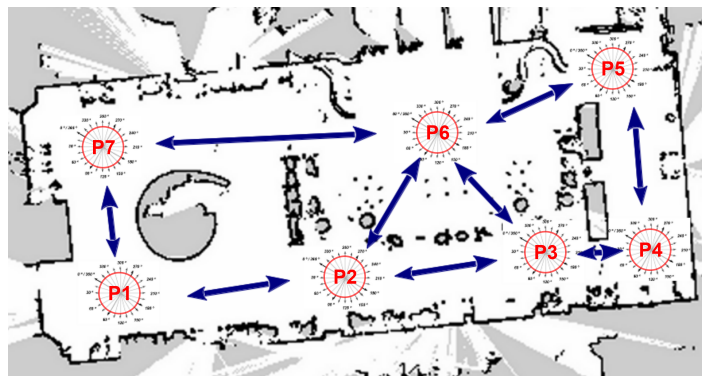


Figure 3.7: Panorama (1-7) capture points labelled as P (1-7)

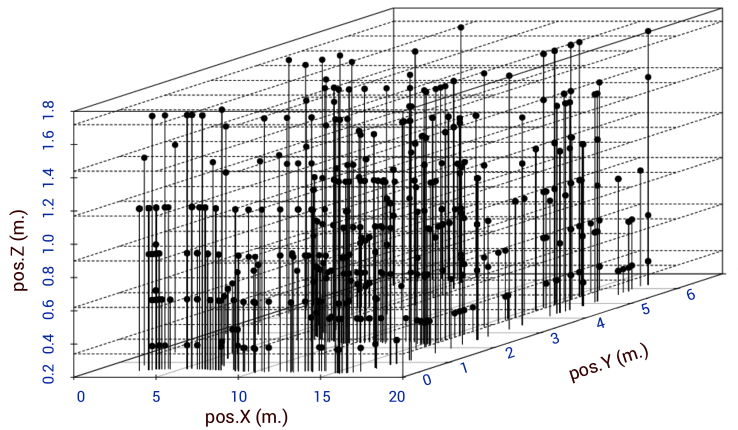


Figure 3.8: RFID inventory (black dots) read by the robot, where $pos.X$, $pos.Y$, and $pos.Z$ are the length, width, and height coordinates.

Chapter 3. Indoor 360° synthetic view using camera and RFID sensor



(a) “Store view” link from Google Street View.



(b) Panorama to panorama navigation links.



(c) Search and click - displaying the RFID obtained product (here, “Short Ski Jacket”) related information on the panorama.

Figure 3.9: The Store view user interface

3.4. Experimental Results

3.4.2 User Evaluation

The Store view system was evaluated by 21 users (subjects). The subjects were given in total 357 product EPCs which are randomly distributed, mutually exclusive sets of EPCs, to find and locate these (product) items using the Store view user interface (Figure 3.10, 3.11). We found that 80.3% product items were identified by the users accurately, and they were able to find and locate items in the Store view. Within these 80.3% of EPCs, the mean deviation of product location estimated by the users from the ground truth (X, Y, Z) was 0.53 meter. The remaining 19.7% of items were deemed to be outside of the current view. This could either be due to the item not being in the current view or the subject not being able to identify the object amongst other items, for instance, in the case of their being multiple occluding items. This problem could be improved by taking more photos at lower spatial separation which is currently 5 meters. This would reduce the view angle between the fronto-parallel view and the target product. The subjects were then given questionnaires to assess the speed and performance of the system. The results are summarized in Table 3.1, from which we would like to outline Q16 and Q17 reporting that the speed and the reliability of the system was graded 4.5 over 5. The questions Q1 to Q15, related to the usability of the system, also received positive feedback. The table shows a complete set of statistics for each question for a thorough analysis and interpretation of the evaluation.

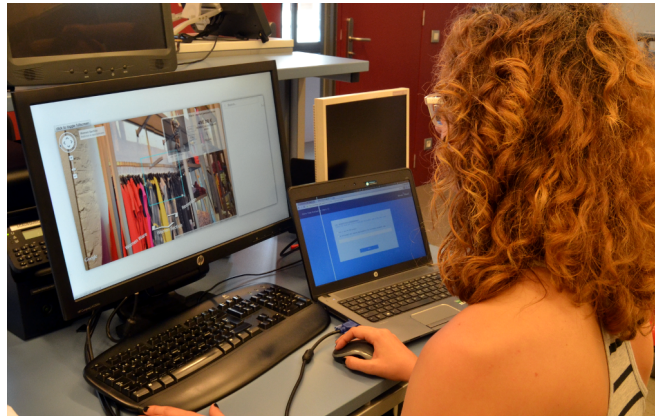


Figure 3.10: Virtual browsing - a user is evaluating the ‘virtual browsing’ feature and locating products with the mouse.

Chapter 3. Indoor 360° synthetic view using camera and RFID sensor

Table 3.1: The user evaluation and the system performance results

No. of EPC evaluated:	No. of Subjects:	User evaluation questionnaire					
		Trials	Lowest score	Highest score	Mean (μ)	Mode (Mo)	Std (σ)
Q1.	It was simple to use.	21	3	5	4.4	5	0.66
Q2.	I am able to search a product quickly using this system.	21	3	5	4.4	5	0.66
Q3.	I feel comfortable using this system.	21	3	5	4.5	5	0.67
Q4.	It was easy to learn to use this system.	21	3	5	4.5	5	0.71
Q5.	I believe I became productive quickly using this system compared to other web based catalog.	21	3	5	4.5	4	0.73
Q6.	Whenever I make a mistake using the system, I recover easily and quickly.	21	2	5	4.1	4	0.85
Q7.	It is easy to virtually browse product on panorama.	21	3	5	4.5	5	0.71
Q8.	The organization of information on the system screens is clear.	21	2	5	3.9	5	0.92

(cont.)

3.4. Experimental Results

Table 3.1: The user evaluation and the system performance results (*cont.*)

Q9.	This system has all the functions and capabilities I expect it to have.	21	3	5	4.2	4	0.83
Q10.	It is fun to use, something new.	21	2	5	3.7	5	1.01
Q11.	It was easy to read texts on the screen.	21	2	5	4.3	5	0.78
Q12.	A highlight box was visible to show approximate product location.	21	3	5	4.3	5	0.83
Q13.	The product information like name, size, color etc. was displayed when available.	21	4	5	4.3	5	0.82
Q14.	I would recommend it to a friend.	21	2	5	4.3	4	0.93
Q15.	Overall, I am satisfied with this system.	21	2	5	4.1	4	0.85
Q16.	Speed [1=slow, 5=fast enough]	13	3	5	4.5	5	0.78
Q17.	Reliability (failure / crash free operation) [1=unreliable, 5=reliable]	13	3	5	4.5	5	0.66
Accurately identified items: 80.3%.							
Mean deviation from ground truth (X, Y, Z): 0.53 m.							

Chapter 3. Indoor 360° synthetic view using camera and RFID sensor

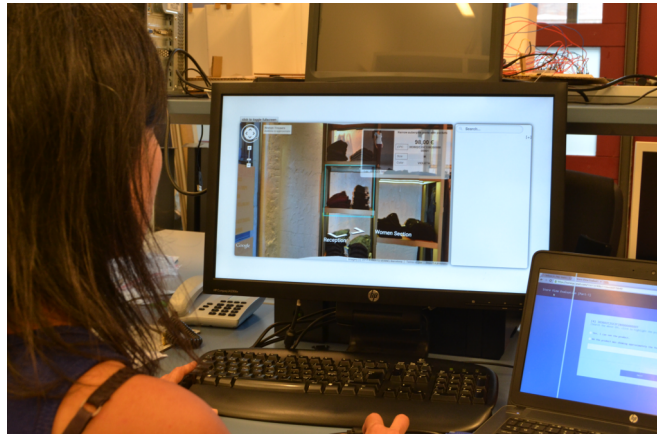


Figure 3.11: A user is evaluating the ‘click and display’ feature and locating products with the mouse.

3.5 Chapter Summary

In this chapter, we present the projection of RFID-obtained product information on the panoramic view of an actual retail store. In our approach, we first describe the RFID integration on the robot, then we present the automated panorama making process. Next, we present a Street View-like indoor visual model by inter-linking panoramas and its navigation scheme. We also present a method for locating a product on the panorama of the retail store. Finally, we present the up-to-date retail inventory information projection on panoramas store-wide. The idea of inventory by RFID system on a robot enables efficient space resolution by reducing RFID antennas, readers, and the overhead installation cost. The web based user interface of “Store view” brings the up-to-date inventory information accessible anytime, anywhere bridging the gap between offline and online commerce. Finally, we evaluate the “Store view” user interface with the users and present the user evaluation results. The “Store view” system enables independent shopping for the shoppers with improved store management for the shopkeepers. The integration of numerous technologies, sensors and a Robot is the novelty of the “Store view” system. Indoor 3D modeling is an alternative to panorama based services [35,36]. In general, 3D modeling is more processing intensive and time consuming than a panorama stitching process. In chapter 4, we present a point cloud based indoor 3D mapping and information projection on the map using RGB-D SLAM.

4

3D synthetic view using RGB-D SLAM and RFID sensors

The term ‘3D’ normally refers to three-dimensional space, the physical universe. Unlike 2D, where ‘seeing’ an object is perceived by analyzing the colors and textures, with a 3D view, the real world objects are perceived by spatial and temporal relationships in the scene. A 3D view clearly distinguishes the foreground and background objects with depth information while allowing all around view. An ideal 3D view gives the exact look and feel of a physical environment. A detailed and accurate models can provide extensive knowledge about the environment and the objects it contains. 3D models of indoor environments have great potentials and interesting applications. Typically, a large 3D model is computing intensive and time-consuming. However, inexpensive depth (RGB-D) cameras made it possible to construct 3D models much easier than with high configuration cameras and computer graphics hardware. An RGB-D camera or depth camera captures per-pixel depth information with RGB color information. RGB-D mapping involves creating 3D point cloud maps with RGB-D camera sensors and Simultaneous Localization and Mapping (SLAM) by a robot. A useful application of robot mapping would be - creating a 3D view of the indoor environment while RFID inventorying to create a 3D synthetic view of the indoor with up-to-date object information on it. In this chapter, a novel synthetic view of indoor is presented by utilizing RGB-D SLAM and RFID sensors.

4.1 Introduction

While a textured 3D map provides interior details, the map itself is not very useful without contextual information. Different approaches have been utilized to annotate 3D maps in different contexts. However, 3D map annotation or semantic modeling is a laborious, time-consuming, and error prone process [37]. Due to heavy graphics rendering requirements, a complete automation of 3D map annotation or semantic modeling in real time may not be possible for a large indoor. Also, in a large environment such as in a retail store the products or items are prone to change its location quite frequently. An up-to-date object information projection on the 3D indoor map requires a solution that can update the object information with a system like RFID. With RFID readers on a mobile robot, it is possible to localize the RFID tagged objects using the robot position and orientation information with RFID tag reading time synchronization. In this chapter, we present projection of RFID information on top of a 3D point cloud map of the indoor environment.

4.2 Simultaneous Localization and Mapping (SLAM)

While fixed robots have been serving in industrial automation, mobile robots are becoming smarter with sensing technologies where it can build their map and move around the environment for specific tasks. However, in an unknown environment, a robot neither knows its position nor the environment setting. A method which can simultaneously build the map of the environment and localize the robot position within the map is called Simultaneous Localization and Mapping (SLAM).

The SLAM problem can be mathematically expressed as -

Given, the robots control, $u_1 : T = u_1, u_2, u_3, u_4, \dots, u_T$

and the observations, $z_1 : T = z_1, z_2, z_3, z_4, \dots, z_T$

Need to build the map, m

and the path of the robot, $x_0 : T = x_1, x_2, x_3, x_4, \dots, x_T$.

Thus, the estimation of the robot path and the map is the probability,

$$P(x_0 : T, m \mid z_1 : T, u_1 : T)$$

When the robot knows its position and the structure around it, it can move

4.3. Depth Measurement Techniques

around and perform its given task. To do so, SLAM algorithms use the depth or distance information of its surroundings. A traditional state-of-the-art technology of acquiring depth information is the laser scanner. A laser scanner provides long range depth information with higher accuracy than sonar or other existing technology. A laser scanner can be a 2D or 3D laser scanner. A 2D laser scanner provides x, y and theta, where theta is the orientation angle. A 3D laser scanner can only produce 3D maps with pixel color intensity as it is not built for capturing texture. However, laser scanners are expensive especially 3D laser scanners. Instead of using commercial 3D laser scanners, a recent trend is the use of depth camera to capture both the depth information and the textures of the surroundings for creating a 3D map of the environment. Depth cameras such as Microsoft Kinect, Asus Xtion pro etc. are inexpensive, however they have limitations too. A depth camera has typical range of 4 meters or less and the depth accuracy is not as accurate as a laser scanner. As a result, a hybrid approach has also emerged where the depth information is sensed by laser scanner and the textures are captured using depth camera and combine the result for estimating robot pose from 2D laser scanner and building a more accurate 3D point cloud maps.

4.3 Depth Measurement Techniques

The depth measurements can be obtained by microwaves, light waves, and ultrasonic waves. With light waves different methods are used, such as time-of-flight, structured light, triangulation, and laser scan. The existing depth measurement techniques are briefly described as below.

4.3.1 Time-of-flight

The Time-of-flight (ToF) measurements consists of measuring the time delay of an emitted signal that returns back to the receiver. This time value can be used to measure the depth of each and every pixel in the image. With velocity of the signal, v and the signal traveled time, t , the travel distance,

$$S = v \times t$$

In a ToF camera, the sender and the receiver are close to each other. As a result, the distance from sender to the object is equals to the distance from object to the receiver. Therefore, the distance to the object -

$$D = v \times \frac{t}{2}$$

Chapter 4. 3D synthetic view using RGB-D SLAM and RFID sensors

The major advantage of ToF camera is that only one camera is required for the acquisition of 3D geometry in real-time.

4.3.2 Phase-shift

When a continuous wave, such as a laser light is emitted, the amplitude of the carrier signal modulated by sinus signals of different frequencies can also be used to measure distance. The reflected signal is compared to the currently sent one; and the phase shift $\Delta\varphi$ can be measured. Since the phase shift is proportional to the distance,

$$D = \frac{\Delta\varphi \lambda}{4\pi} = \frac{\Delta\varphi v}{4\pi f}$$

Where, λ is the wavelength of the modulated signal and f is the frequency.

4.3.3 Triangulation

Triangulation is the process of determining the location of a point by measuring angles to it from known points [38]. The point can then be fixed as the third point of a triangle with one known side and two known angles. Thus distance can be measured applying triangulation.

4.3.4 Structured light

Projecting a pattern of infrared (IR) light on a 3D surface produces a line of illumination that appears distorted when looked from a perspective different from the projector’s perspective. This distortion of light pattern allows computing the depth and the 3D structure of the scene. Microsoft Kinect uses structured light imaging where the projector projects a known pattern called ‘Speckles’ in near-infrared light, and the CMOS IR camera observes the scene [39]. The depth in Kinect is calculated by the triangulation of each ‘Speckle’ between the virtual image and the observed pattern where each ‘Speckle’ corresponds to each point of the image [40]. The similar principle is used in Xtion PRO LIVE cameras [41].

Figure 4.1 illustrates the geometric relationship between the imaging sensor, the structured light projector, and an object surface point. The distance or depth D , from camera to the object surface thus can be expressed as a triangulation principle -

4.4. Laser Scanner

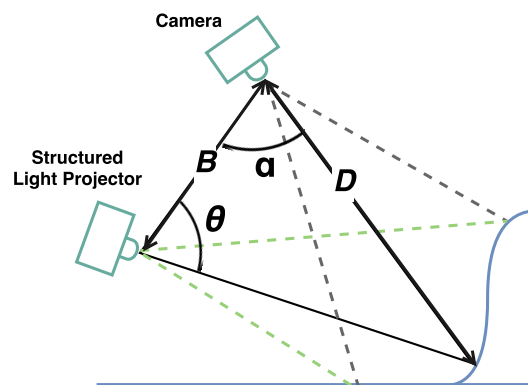


Figure 4.1: Structured light depth measurement parameters

$$D = B \frac{\sin(\theta)}{\sin(\alpha + \theta)}$$

4.4 Laser Scanner

The laser scanner also called a laser range finder (LRF) is an optical device that measures the distance to an object in a scanning field using a pulsed laser beam [42]. A laser (light amplification by stimulated emission of radiation) beam is a single wavelength, same phase, a high-density light beam that can travel a straight line longer distance with a narrow beam. A laser scanner uses a rotating mirror to change the direction of the laser beams. When both a laser scanner rotates all around while rotating mirror rotates up and down, the result is a systematic 3D sweeping of beams into the scanning area. The laser scanner measurement is based on Time-of-Flight principle, when the emitted laser beam hits an object, part of the beam reflects back to the scanner detector, the time is measured. Laser scanners are widely used in robotic applications such as obstacle avoidance, object tracking, map building, feature extraction, or self-localization, etc [43].

4.4.1 2D laser scanner

A 2D laser scanner provides distances and angles to the surrounding objects by scanning the environment in a plane, usually parallel to the ground. Widely used, industry standard 2D laser scanners Sick LMS100 and Hokuyo UTM-30LX are shown in Figure 4.2. Sick LMS100 2D laser scanner has 270° field of view with scan range of 0.5m - 20m, and the light source is

Chapter 4. 3D synthetic view using RGB-D SLAM and RFID sensors

infrared 905 nm wavelength [44]. Hokuyo UTM-30LX 2D laser scanner also has 270° field of view, with longer scan range of 0.1m to 60m, and the light source is semiconductor laser diode with 905nm wavelength [45].



Figure 4.2: 2D laser scanners

4.4.2 3D laser scanner

3D laser scanners are also known as terrestrial laser scanners, traditionally used for mapping roads, building structures, etc. In a 3D laser scanner, the scanning area can be extended by moving the scanner from different vantage points to cover the whole area of scan interest. Multiple scans can also be merged by post processing of point clouds. Two professional 3D laser scanners Faro focus^{3D} X 130 and Trimble TX8 3D laser scanner is shown in Figure 4.3. Faro Focus X 130 has 300° / 360° field of view with scan range 0.6 m to 130 m and the light source is Laser class 1, 1550 nm wavelength [46]. Trimble TX8 3D laser scanner has 360° / 317° field of view with scan range 0.6 m to 120 m, and the light source is also Laser class 1, 1500 nm wavelength [47]. Thus, 3D laser scanners can produce a very dense point clouds maps covering a large scan area with high precision.

4.5 RGB-D Camera

RGB-D cameras are sensing systems that capture RGB images along with per-pixel depth information. The underlying sensing techniques of RGB-D cameras can be range-gated ToF, RF-modulated ToF, pulsed-light ToF or projected-light stereo. An RGB-D camera is also called a range camera as it produces 2D images showing the distance to points in a scene. RGB-D cameras are inexpensive and compact in compared to other systems such as a stereo vision or a triangulation systems or a laser scanning system.

4.5. RGB-D Camera

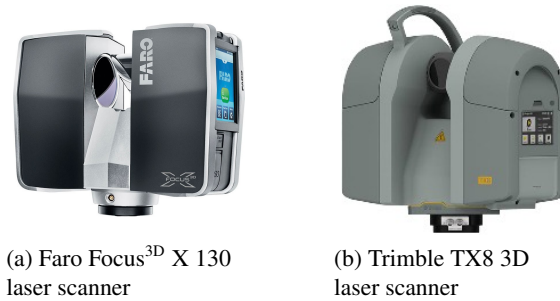





Figure 4.3: 3D laser scanners

Moreover, using RGB-D cameras, it is possible to acquire video stream and depth data for many robotic applications such as 3D mapping and localization, path planning, navigation, object recognition, people tracking [48] etc. RGB-D cameras can operate from as near as 0.1 meter from the closest object for object detections and for the indoor mapping as near as 0.5 meter. In contrast to a laser scanner, an RGB-D camera systems do not require any mechanical moving parts as the processing is done with a whole scene in real-time rather than single point by point scanning. Three widely used RGB-D cameras are Microsoft xBox 360 Kinect, Asus Xtion PRO LIVE, and SoftKinect DepthSense DS325. A comparison of these camera attributes are presented in Table 4.1.

Table 4.1: RGB-D camera comparison

Attributes	xBox 360 Kinect	Xtion PRO LIVE	DepthSense DS325
			
Field of View	57° H, 43° V	58° H, 45° V, 70° D	74° H, 58° V, 87° D
Frame rate	30 FPS	30 FPS	30, 60 FPS
Resolution	RGB and depth: VGA (640 x 480)	RGB and depth: VGA (640x480)	RGB: HD (1280x720), Depth: QVGA (320x240)

Chapter 4. 3D synthetic view using RGB-D SLAM and RFID sensors

Microsoft xBox 360 Kinect is an integrated sensor with an RGB camera, an infra-red projector, a CMOS sensor, and a microphone [39]. Microsoft Kinect sensors are jointly manufactured by Microsoft and PrimeSense [49]. The xBox 360 Kinect depth camera operates on 640x480 pixel resolution at 30 FPS (frames per second). The recent xBox 360 Kinect sensors do not require any camera calibration as they come with the factory calibration. The xBox 360 Kinect sensors works with open-source, platform-independent OpenNI driver [50].

Xtion PRO LIVE manufactured by Asus is a color infrared sensor capable of detecting adaptive depth when user or camera is in movement and does not require any additional power source as it works on USB 2.0 [41]. Like Microsoft xBox 360 Kinect, Asus Xtion RPO LIVE depth camera operates on the 640x480-pixel resolution at 30 FPS. Xtion Pro Live requires OpenNI2 driver [50]. Its lightweight, adaptive depth sensing and no additional power requirements makes it attractive for robotic applications such as 3D mapping and obstacle detection.

DepthSense DS325 is manufactured by SoftKinect excels in near mode dense depth as close as 15 cm at 60 FPS [51]. DepthSense DS325 is built with CMOS 3D sensors and dual microphones. Like Asus Xtion PRO LIVE, DepthSense DS325 is also a USB powered depth camera that works with SoftKinect driver.

RGB-D cameras have limitations regarding depth scanning range, field-of-view and image resolution. The effective maximum depth scanning range of RGB-D cameras are approximately 4 meters, and the filed-of-view is approximately 58° , which is far more constrained than laser scanners field-of-view ($\sim 300^\circ / 360^\circ$). Also, the RGB and the depth images of RGB-D cameras are typically VGA (640x480 pixels) and thus the final point cloud is indeed a mid-resolution output. Despite these limitations, a natural application of depth camera is the Simultaneous Localization and Mapping (SLAM) RGB-D mapping.

In this chapter, we present a 3D synthetic view system created by RGB-D camera, SLAM and RFID sensors on a robot.

4.6 Related Work to RGB-D SLAM Mapping

The term ‘RGB-D SLAM’ refers to the SLAM using RGB-D image data. Traditionally, the RGB-D SLAM approaches used the RGB data from the camera and the depth data from the laser scanner. The newer approach is the use of inexpensive Kinect style camera’s RGB and depth data instead of expensive cameras and laser scanners.

The RGB-D mapping was first published by Henry et al. (2010), proposing a solution by combining the visual odometry and pose-graph estimation with color and depth information to create dense point cloud maps of indoor environments [8, 52]. The term ‘odometry’ refers to the use of motion sensor data to estimate the change in position over time. The goal of the research was to create dense 3D point cloud map of indoor environments using inexpensive depth camera such as Microsoft Kinect (Figure 4.4) [8]. Unfortunately, there are no source code or executables available to evaluate the system.

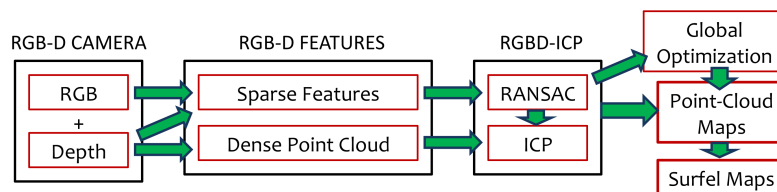


Figure 4.4: The overview of RGB-D mapping proposed by Henry et al.

KinectFusion (2011) [35] is a real-time 3D reconstruction and interaction system using a standard Kinect camera. In their approach, only the depth data was used to track the 3D pose of the camera to reconstruct the 3D model of the scene with high-performance GPU hardware. As the size of the voxel grid has cubic influence on the memory usage, KinectFusion can be used for small scale mapping only. The core use of KinectFusion was demonstrated as a low-cost handheld scanner for real-time 3D modeling that can be leveraged for geometry-aware augmented reality (AR) and physics-based interactions, where the virtual world more realistically interacts with the real world.

Kintinuous (Spatially Extended KinectFusion) (2012) [53] introduced a technique called ‘cyclical indices’ for moving the origin of the space that is rela-

Chapter 4. 3D synthetic view using RGB-D SLAM and RFID sensors

tive to volume rather than fixing the volumetric space to reduce the increasing GPU memory usage of the KinectFusion. The approach was further improved with a GPU-based implementation of an existing dense RGB-D visual odometry algorithm and real-time surface fused coloring published in 2013 [54]. Although the research presented with evaluations looks promising, there was no available source code to evaluate the system.

Endres et al. (2012) [55, 56] proposed an environment measurement model (EMM) to improve the accuracy and the robustness of the RGB-D SLAM system. The EMM was introduced to validate the transformation estimated by the iterative-closest-point (ICP) algorithm. Endres et al. point out that, in case of low overlap between frames or few visual features, e.g., due to motion blur, occlusions, or lack of textures, the RANSAC and the ICP may lead to unreliable feature detection failure. Therefore, a beam based EMM was proposed to verify a transformation estimate, independent of the estimation calculation method used. To compute a globally consistent trajectory, g^2o framework [57] have been used to optimize the nonlinear robot pose graph errors (Figure 4.5) [55]. The experimentation evaluations conducted offline, have been presented with a RGB-D benchmark datasets. The authors evaluated the main stream feature detectors such as SIFT [16], SURF [17], ORB [21], and a combination of Shi-Tomasi [58] and SURF detectors. Except SIFT, the author tested with the OpenCV (GPU based) implementation and presents that ORB and Shi-Tomasi SURF combination is suitable for performance with limited hardware resources, however, with the trade-off of accuracy and robustness. The author also states that, with the GPU, SIFT was clearly the best choice with highest accuracy, which was the median RMSE of 0.04m. With the SURF detector algorithm, the author also presented that, with some dataset, the feature matching was improved, however for most sequences the improvement was not significant.

Real-Time Appearance-Based Mapping (RTAB-Map) proposed by Labbé and Michaud (2013) is a real-time online RGB-D Graph SLAM, based on a global Bayesian loop closure detector. In appearance-based localization and mapping [59, 60], the loop closure detection process is used to determine if the current comes from a previously visited location or a new observation. The objective of appearance-based mapping is to propose a localization and mapping that is independent of time and size. RTAB-Map has introduced a memory management mechanism to ensure satisfaction of real-time large scale mapping regardless of the size of the environment. A global loop clo-

4.6. Related Work to RGB-D SLAM Mapping

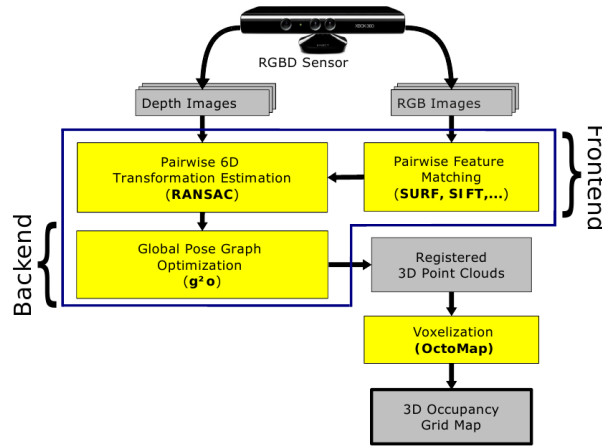


Figure 4.5: The overview of RGB-D SLAM system proposed by Endres et al.

sure detection is the process of comparing new location with previously visited locations and adding the new location if there is no match found. With this approach the required time for processing new observations increases as the number of locations increases. RTAB-Map resolves this problem by a combination of the Working Memory (WM) and the Long-Term Memory (LTM). The WM keeps the most recent and frequently observed locations and when a match is found between the current location and the WM location; the associated locations stored in LTM is then remembered and updated (Figure 4.6) [59].

RTAB-Map uses bag-of-words [61] approach for vision-based mapping where each image corresponds to visual words extracted by local feature descriptors such as SURF [17], SIFT [16], ORB [21] etc., and kept in a visual dictionary. The visual dictionary is incrementally constructed online using a randomized forest of kd-trees. In the bag-of-words approach, an image signature Z_t is represented by a set of visual words at the time t , when the image is acquired. The perception module acquires an image from RGB-D camera and sends it to the Sensory Memory (SM). The SM evaluates the image signatures for useful features to The short term Memory (STM). The STM is used to observe similarities through time between consecutive images with a weighted update. The STM size T_{STM} is set based on robot velocity and the rate at which the locations are acquired. A memory location L_t is then created with the signature Z_t as a bi-directional graph and

Chapter 4. 3D synthetic view using RGB-D SLAM and RFID sensors

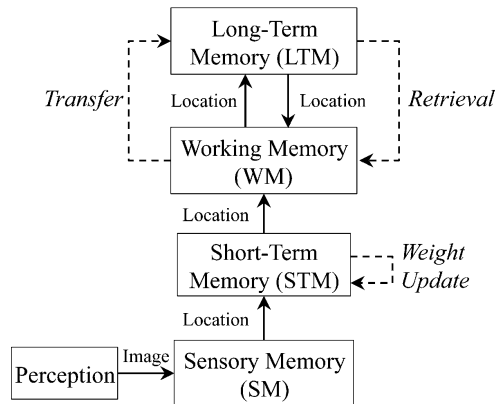


Figure 4.6: The RTAB-Map memory management model presented by Labbé and Michaud.

the memory location is stored in WM and in STM. When a transfer from WM to LTM is necessary, the lowest weight is selected. In case of multiple same lowest weight of many locations, the oldest one is transferred. A discrete Bayesian filter is used for estimating the probability of the current location L_t matches with the already visited location stored in the WM to keep track of loop closure hypotheses. Tree-based netWORk Optimizer (TORO) [62] graph optimization method is used in order to correct the map for odometry errors. TORO is also used to optimize multiple session map with multiple roots.

4.7 Related Work to 3D Map Annotation

Rusu et al. [63] present a system for mapping objects for household environments given the partial view of the environment as point clouds. The author contributes in point cloud improvement for object recognition (e.g. kitchenware) within the point cloud by classifying high-level features. The approach seems to work for a static or known environment where items are limited and has distinct features. However, in constantly changing environment such as in a retail environment, there are possibilities to have thousands of items with similar shapes, colors, folded, unfolded, packed, unpacked items etc. Moreover, to provide the user an up-to-date mapping and item information, the retail store map may need be done periodically on a daily basis. Tian et al. [64] present a small scale 3D models’ labeled data set to label unlabeled 3D models using semi-supervised semantic la-

bel propagation technique. The approach is interesting, however, for a large retail store with large number of items, matching 3D models with environment map may turn out to be difficult. And, of course, the 3D models for each item has to be created first.

4.8 Point Cloud

4.8.1 Point Cloud Map

Point cloud is the convenient form of map representation of any types of RGB-B SLAM. A point cloud is a set of data points in some coordinate system. In a three-dimensional coordinate system, points are usually defined by X , Y , and Z coordinates of an underlying surface if possible. Mathematically, a point p is represented as an n -tuple, e.g., $p_i = \{x_i, y_i, z_i, r_i, g_i, b_i, dist_i, \dots\}$. A Point Cloud P is represented as a collection of points p_i , e.g., $P = \{p_1, p_2, \dots, p_i, \dots, p_n\}$. Point clouds are usually scanned by a RGB-D cameras, stereo cameras, 3D laser scanners, time-of-flight cameras. The point cloud library is used for processing a point cloud map.

4.8.2 Point Cloud Library

The Point Cloud Library (PCL) is a cross-platform open-source library of algorithms for point cloud processing and 3D geometry processing [65]. PCL contains numerous state-of-the art algorithms including filtering, feature estimation, surface reconstruction, registration, model fitting and segmentation [66]. PCL is fully integrated with ROS for 3D point cloud and robot perception processing. PCL has its own visualization library which offers methods for rendering and drawing of based on Visualization Toolkit (VTK) [67]. VTK is an open-source programmable software system for 3D computer graphics, modeling, image processing, volume rendering, scientific visualization, and information visualization. The PCL visualization library offers methods for rendering and setting visual properties such as colors, point sizes, opacity etc., methods for drawing of the 3D shapes such as cylinder, spheres, lines, polygons etc., methods for specifying the dimension and color of the points in 3D Cartesian space as point cloud. The basic data type in PCL is a *PointCloud*. The *PointCloud* data type is a C++ template class which contains *width(int)*, *height(int)*, *points(std::vector<PointT>)*, *is_dense(bool)*, *sensor origin(Eigen::Vector4f)*, *sensor orientation(Eigen::Quaternionf)*. In terms of data structures, a point cloud P is declared as *Point[]*

Chapter 4. 3D synthetic view using RGB-D SLAM and RFID sensors

points or *std::vector<Point>points*. In C++, where *Point* is the structure/-data type representing a single point *p*. The relevant PCL *PointCloud* point types are listed in Table 4.2.

4.8.3 Point Cloud File Formats

PCD and PLY are two popular point cloud file formats described as follows.

Table 4.2: PCL point types

Point Type	Data
PointXYZ	float x, y, z
PointXYZI	float x, y, z, intensity
PointXYZRGB	float x, y, z, rgb
PointXYZRGBA	float x, y, z, uint32 t rgba
PointNormal	float x, y, z, normal[3], curvature

PCD - Point Cloud Data

Point Cloud Data (PCD) is a native PCL file format for storing multi-dimensional (n-D) point cloud data. It consists of a header in ASCII, followed by the data in ASCII or in binary. A PCD file header specifies the properties of the point cloud data which includes - PCD VERSION, FIELDS, SIZE, TYPE, COUNT, WIDTH, HEIGHT, VIEWPOINT, POINTS, and DATA. Each header property must be written as ASCII and separated by a new line. The VERSION specifies the PCD file version, the FIELDS specifies the name of each dimension/field of a point, the SIZE - specifies the size of each dimension in bytes, the TYPE specifies the type of each dimension as a *char*, the COUNT specifies how many elements each dimension has, the WIDTH specifies the total number of points in the cloud for unorganized point cloud and for organized point cloud this the total number of points in a row, the HEIGHT specifies the total number of or rows for organized point cloud and for unorganized the value is 1, the VIEWPOINT specifies the viewpoint for points in terms of translation (tx, ty, tz) and quaternion (qw, qx, qy, qz), the POINTS specifies the total number of points in the cloud, and the DATA specifies the data type of the point cloud data is stored which is the ASCII or the Binary data type. Different FIELDS specification is listed in Table 4.3.

4.9. Displaying 3D Maps

Table 4.3: A PCD file FIELDS declarations

Dimension	Declaration
XYZ data	FIELDS x y z
XYZ and colors	FIELDS x y z rgb
XYZ and surface normals	FIELDS x y z normal_x normal_y normal_z

PLY - Polygon File Format

Polygon File Format (PLY), also known as Stanford Triangle Format is designed to store large-scale 3-D geometric information such as 3D model from 3D scanners. A PLY file consists of a header followed by a list of vertices and then a list of polygons with other properties [68]. The properties can be saved including color and transparency, surface normals, texture coordinates and data confidence values. A PLY file can either be written as ASCII or binary, however, the header is always written in ASCII. A typical PLY file starts with a magic keyword ‘ply’, followed by the version of ASCII text, number of vertex elements and vertex elements’ properties, number of face elements and face elements’ properties, number of edges and edge elements’ properties etc. The vertex element properties can be X, Y, Z and red (r), green (g), blue (b) colors. Like a vertex element, an edge element can also have ‘rgb’ color property in a PLY file.

4.9 Displaying 3D Maps

The RGB-D SLAM 3D maps are essentially a point cloud map which can be viewed with the RGB-D SLAM interface. However, RGB-D SLAM interface or PCL viewer is less flexible to show any additional information on top of the map. In addition to RGB-D SLAM interface, ROS Visualization tool (RViz) can be used to view point cloud maps. The layered display architecture of rviz allow us to display all types of sensor data including point cloud map and thus suitable for showing additional information about the objects or entities visible on the map. In addition to rviz, large scale point clouds can also be displayed with Web Graphics Library (WebGL) programming [69]. To display the map in web sites, a WebGL renderer is Potree [70] that converts PLY file into octree that can be viewed with WebGL. Different

Chapter 4. 3D synthetic view using RGB-D SLAM and RFID sensors

possibilities of map displaying is depicted in Figure 4.7.

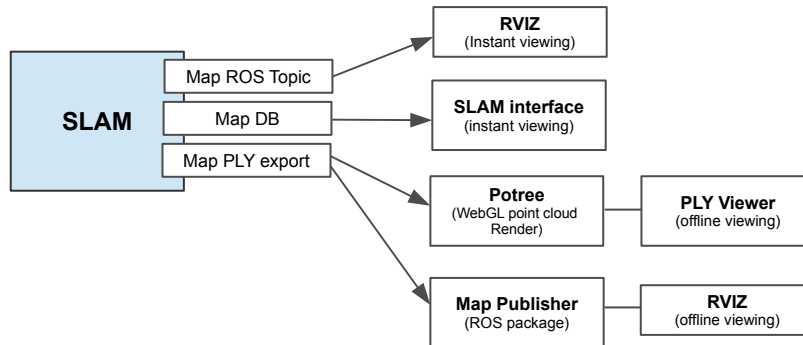


Figure 4.7: RGB-D SLAM 3D map viewing approaches

4.9.1 Rviz

Rviz is a 3D visualizer for displaying sensor data and robot state information from ROS [71]. Rviz supports different types of display types for different visualization. For example, the ‘Axes’ display current X , Y , and Z of robot pose in respect to *base_link* or *map* frame. The ‘TF’ display shows the *tf* transform hierarchy of joint frame states, the ‘Image’ display shows the current frame from the camera sensor, and the ‘PointCloud2’ displays the point cloud sensor messages. The Grid display is used to show the grids of the 3D space with current view distance, yaw, pitch, and focal point value (Figure 4.8).

4.9.2 Offline Map Publisher

A map can be displayed and saved as PLY file format for later reusing. The following section describes the implementation techniques of loading and displaying the saved offline 3D pointcloud map to rviz. To view a point cloud map to rviz, a ros node is needed to create for reading PLY files and publishing it as a *sensor_msgs/PointCloud2* message. The ROS visualization tool rviz supports *sensor_msgs/PointCloud2* display type called PointCloud2 which can subscribe the ros topic published by executing the launch file in listing 4.1. After launching the roslaunch as listing 4.1, the ros topic *point_cloud* is published as active node presented in Figure 4.9. The purpose of the map publisher is to read map source file as PLY format and publish a ROS topic as *PointCloud2* sensor message. The map publisher is programmed in C++ and the purpose of it to read PLY file map source file

4.9. Displaying 3D Maps

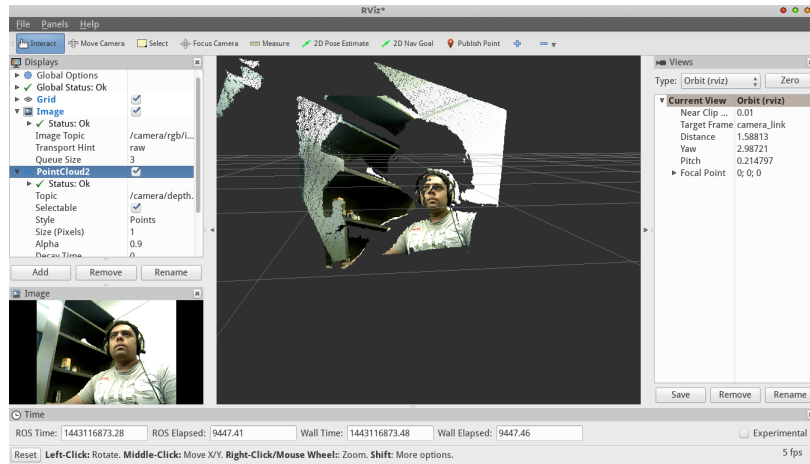


Figure 4.8: Rviz is showing depth-registered points from Xtion Pro Live camera.

and use PCL library to process point clouds so that it can be published as a sensor message. The sensor message is then subscribed with rviz to display the map. The map_publisher launch file is run like a standard ROS launch -

```
roslaunch map_publisher map_publisher.launch
rosviz rviz rviz
```

After publishing the *map_publisher* topic the corresponding rqt_graph of active nodes are shown in Figure 4.9.

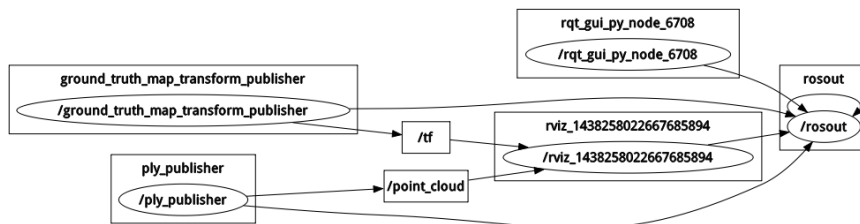


Figure 4.9: Active node of the map publisher

Chapter 4. 3D synthetic view using RGB-D SLAM and RFID sensors

Listing 4.1: map_publisher.launch

```
<launch>
  <node pkg="map_publisher" type="map_publisher" name="
    map_publisher" output="screen">
    <param name="file_path" value="/home/kmn/catkin_ws/src/
      map_publisher/ply_file_to_publish/lanao-cloud.ply" /
    >
    <param name="topic" value="/point_cloud" />
    <param name="frame" value="/map" />
    <param name="rate" value="5.0" />
  </node>

  <node pkg="tf" type="static_transform_publisher" name="
    ground_truth_map_transform_publisher"
    args="0.0 0.0 1.1 0.0 0.17 0.0 /base_link /map 100" />
</launch>
```

4.10 Methodology

First we explored different RGB-D SLAM systems (described in section 4.6), their strengths and limitations. Then we explore different sensor message visualization techniques to project RFID obtained entity information on top of the 3D map view. Here we utilize PCL libraries to fetch the RFID inventory information on top of the 3D map in rviz using the RFID system scanned entity locations (X, Y, Z).

4.11 3D Mapping with RTAB-Map

To achieve this goal, we choose the RTAB-Map for 3D mapping of the environment and the rviz for data visualization on that 3D map. A RGB-D mapping can either be done using the visual odometry or using the robot odometry. A visual odometry is based on motion sensor in the camera whereas, the robot odometry is based on the 2D laser scanner, sonar, or a wheel encoder etc. Here we present the 3D point cloud mapping with RTAB-Map using both the visual and the robot odometry. In the next section we explain both ways of mapping we performed with RTAB-Map.

4.11. 3D Mapping with RTAB-Map

4.11.1 Using Visual Odometry

The visual odometry utilizes the depth images and motion estimation of the RGB-D camera and creates the `nav_msgs/odometry`. In a visual odometry based mapping with RTAB-Map is shown in Figure 4.10. The `rtabmap_ros` is a ROS package for publishing `MapData` (map graph and latest node data) as `PointCloud2` sensor messages and to subscribe map data in `rviz` through `MapCloud` plugin. The `rgbd_odometry` node wraps the RGB-D odometry which is computed using visual features from the RGB images with their depth information from the depth images. A RANSAC approach computes the transformation between the consecutive images using the feature correspondences between the images. With a visual odometry (`rgbd_odometry`), no laser scanner odometry is required as it creates the odometry from the camera. As a result, with a visual odometry, a free-hand mapping is also possible.

The `rgbd_odometry` node subscribes to the `rgb/image` topic for RGB/Mono rectified images, the `rgb/camera_info` for RGB camera metadata, and the `depth/image` for the registered depth images. The output of the visual odometry (`nav_msgs/Odometry`) is then published as `/odom` topic which can be used for SLAM. With visual odometry based mapping, the important topic `rgb/image` is mapped to `/camera_front/rgb/image_rect_color` topic, `depth/image` is remapped to `/camera_front/depth_registered/image_raw`, and the `rgb/camera_info` is remapped to `/camera_front/depth_registered/camera_info`. In addition to the above mentioned parameters, the 3D rendering parameters used in our experiments are listed in Table 4.4. Ideally, the lower the voxel size and cloud filtering radius, the dense is the point cloud map, provided that camera is able to capture noise free texture and depth information. Here, we kept the maximum depth to 4.0 meter both xBox kinect and Xtion pro live cameras maximum depth in practice is not more than 4 meters. The mesh smoothing algorithm MLS integrated with RTAB-Map generated more accurate maps, however with the cost of post processing rendering time.

Table 4.4: 3D Map rendering parameters and values

Parameters	Values
Cloud filtering:	radius 0.004 m, angle 30 degree (<i>cont.</i>)

Chapter 4. 3D synthetic view using RGB-D SLAM and RFID sensors

Table 4.4: 3D Map rendering parameters and values (*cont.*)

3D voxel size:	0.001 m
Cloud decimation: (1-2-4-8-..)	4 (number of images before adding to poin cloud of each node)
Maximum depth:	4.0 m
Cloud point size:	1
Optional Mesh smoothing by MLS	0.04 m

4.11.2 Using Robot Odometry

Mapping with robot odometry is similar to visual odometry except, the /odom input is taken from the laser scanner or IMU, wheel encoders (shown in Figure 4.11). AdvanRobot ¹ has got the laser, gyroscope, and the wheel encoder. A 2D laser scanner outputs sensor_msgs/LaserScan messages and an IMU or wheel encoders outputs nav_msgs/Odometry messages. RTAB-Map can be configured to use either the sensor_msgs/LaserScan messages or the nav_msgs/Odometry messages or both. With a robot odometry based mapping, we set the frame_id to base_link and set true to both the depth and scan registration parameters. the /odom topic is remapped to /base_controlle r/odom/ and the scan topic is remapped to /base_scan. The other default parameters are kept as it is. The specified value 2 of LccIcp parameter means that ICP is done with 2D laser scan.

4.12 Projecting RFID-obtained Information

The new AdvanRobot has 12 RFID antennas, 6 in each side, are connected with 3 RFID readers for simultaneously reading RFID inventory. The robot odometry provides the (X, Y) of the RFID tag location and the antenna height provides the Z value which comprises the capability of locating each tagged item location in 3-dimension (X, Y, Z) inside a space. The height of all antennas including their connectivity with RFID readers are shown in Figure 4.12.

¹<http://keonn.com/systems/view-all-2/inventory-robots.html>

4.13. Experiments and Results

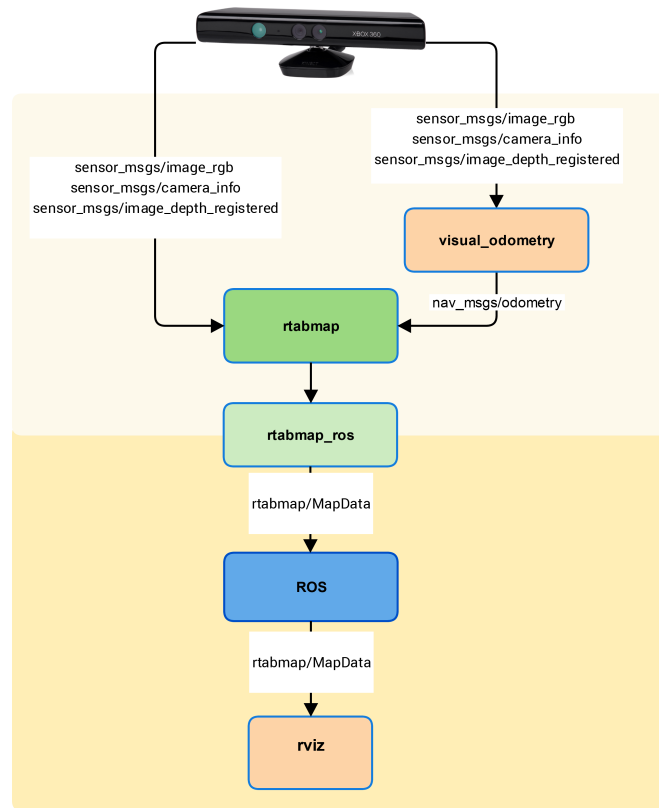


Figure 4.10: RGB-D mapping using xBox 360 Kinect camera by RTAB-Map using visual odometry

We program a marker display ROS package for displaying RFID EPC information on the 3D view. The marker display package creates one marker per EPC and show related information of it from the inventory database or a csv file. The format of the marker display input file consists of EPC, time, X, Y, Z, and a label value. The marker display is programmed to load as many EPC is scanned by the robot RFID system. An example 3x3 marker display in X, Y, Z positions are shown in Figure 4.13 (Orbit view). We will see EPC information display on markers in experiments and results section.

4.13 Experiments and Results

For the purpose of the experiments, we choose three ground truth indoor environment described as below.

Chapter 4. 3D synthetic view using RGB-D SLAM and RFID sensors

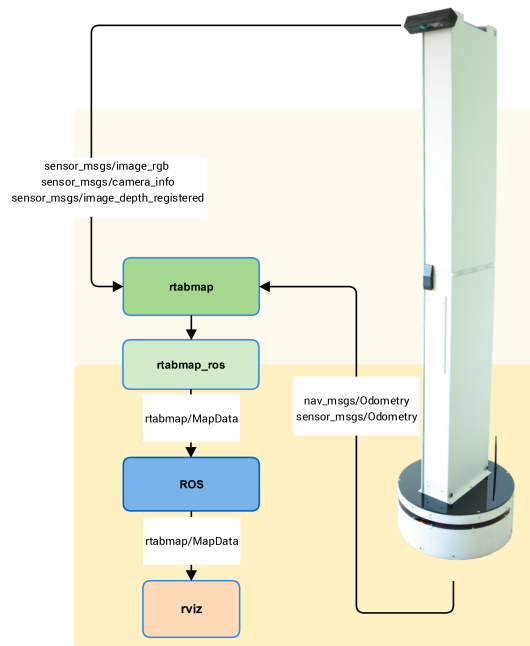


Figure 4.11: RGB-D mapping with AdvanRobot using robot odometry

Test case 1: UbiCA Lab, La Nau building

The UbiCA Lab at La Nau building is 6.8 meter long, 3.9 meter wide, and 3.5 meter high. The map have been taken by going around the side of the lab.

Test case 2: A Retail Store

The retail store was 12 meter long, 2 meter wide, and 2.5 meter high. The map of this store have been taken by going around the customer walking path.

Test case 3: Poblenou Campus Library, 2nd floor

The floor 2 of Poblenou campus library contains 8 rows of book shelves. The maps have been taken by going around and between the shelves.

After few trials, we chose to use SURF over SIFT as SURF was comparatively faster than SIFT as expected. However, we did not find any differences between SURF and ORB while mapping. Keeping the default loop closure threshold value 0.11, weight update value 0.6, and the STM size of 10, we tested different detection rates of 30 Hz, 5 Hz, 1Hz, and 0 (auto

4.13. Experiments and Results

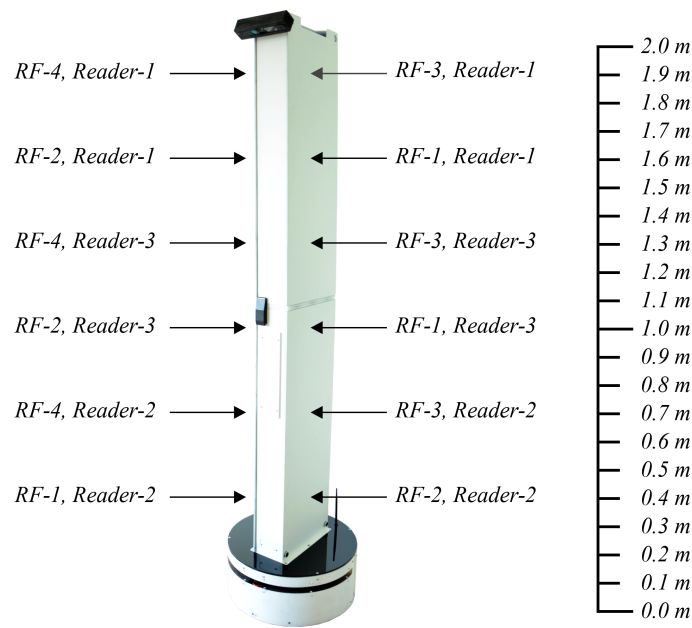


Figure 4.12: The AdvanRobot RFID antennas in different heights.

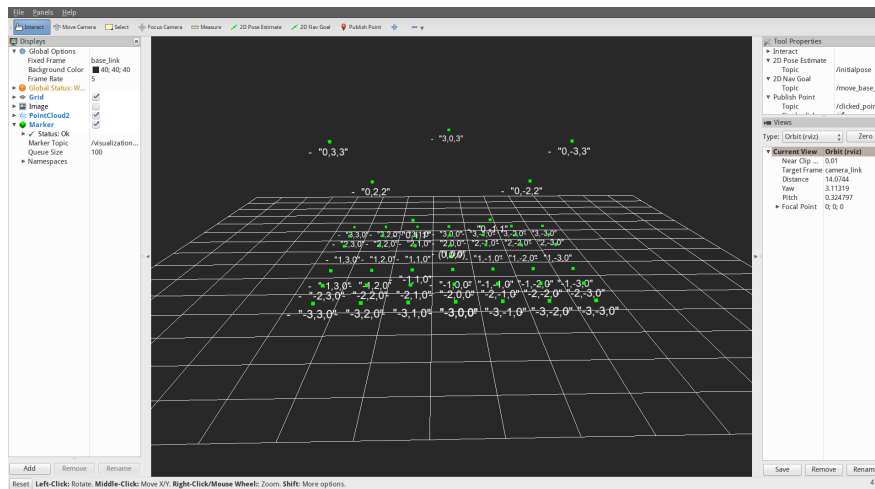
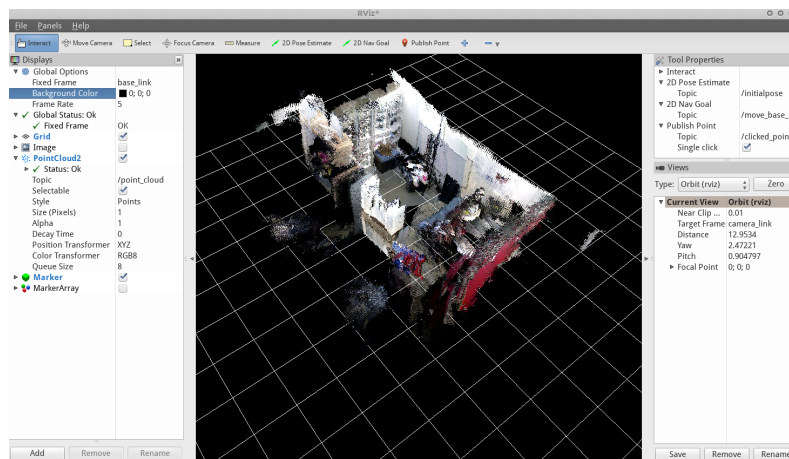


Figure 4.13: Marker Display in X,Y,Z coordinates

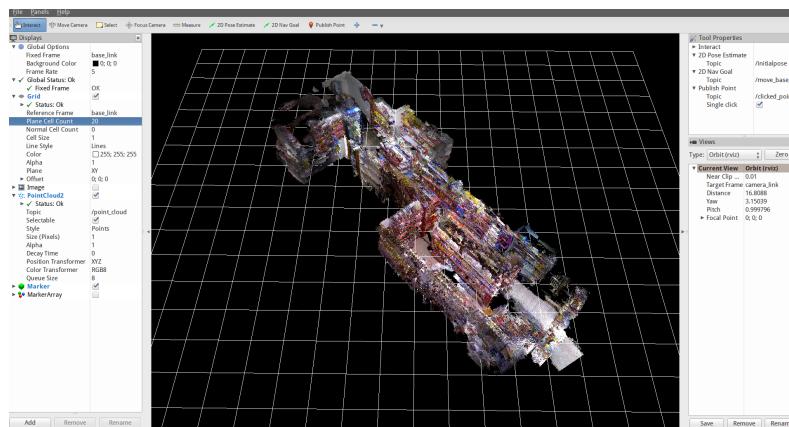
detection rate). With a remote mapping, the value of 5Hz was fast enough to map with fast Wi-Fi (5Mbps) speed, with a lower Wi-Fi speed (1Mbps), when robot was far away, 1Hz was slow yet map able detection rate. When

Chapter 4. 3D synthetic view using RGB-D SLAM and RFID sensors

the mapping was done in the same computer of the camera connected, the value of 30Hz and 0 (auto) did not make any difference as the camera was running at 30FPS. The maps that have been created by visual odometry (subsection 4.11.1) are presented in Figure 4.14. The average camera position change (motion) was approximately 0.5 meter per second, and upto 1 meter per second have been tested with no pause in mapping.



(a) La Nau Lab

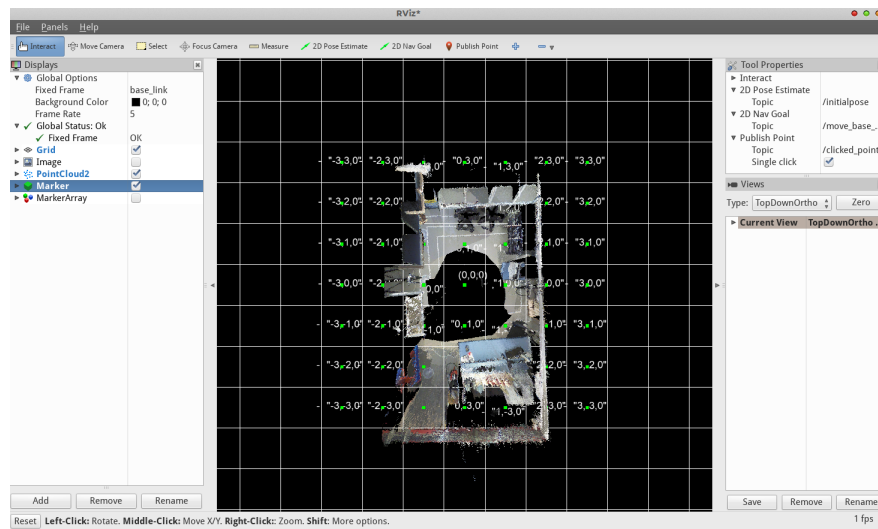


(b) Retail shop

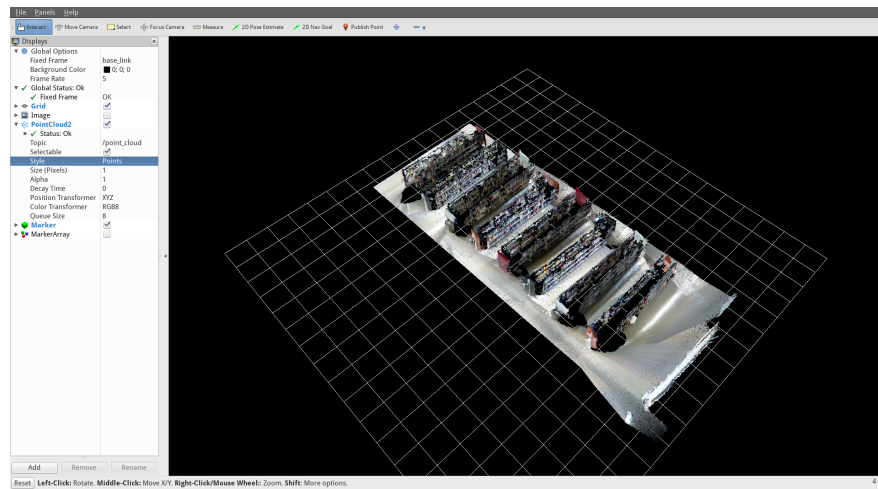
Figure 4.14: RGB-D mapping using visual odometry

The maps that have been created by robot odometry are presented in Figure 4.15.

4.13. Experiments and Results



(a) La Nau Lab Location Marker Display



(b) UPF Poblenu Campus Library (level 2)

Figure 4.15: RGB-D mapping using robot odometry

An optimum camera height with optimum tilt angle is important for better camera coverage area and RGB-D mapping in general. Although the horizontal field of view of RGB-D camera are approximately 58° , in a closed

Chapter 4. 3D synthetic view using RGB-D SLAM and RFID sensors

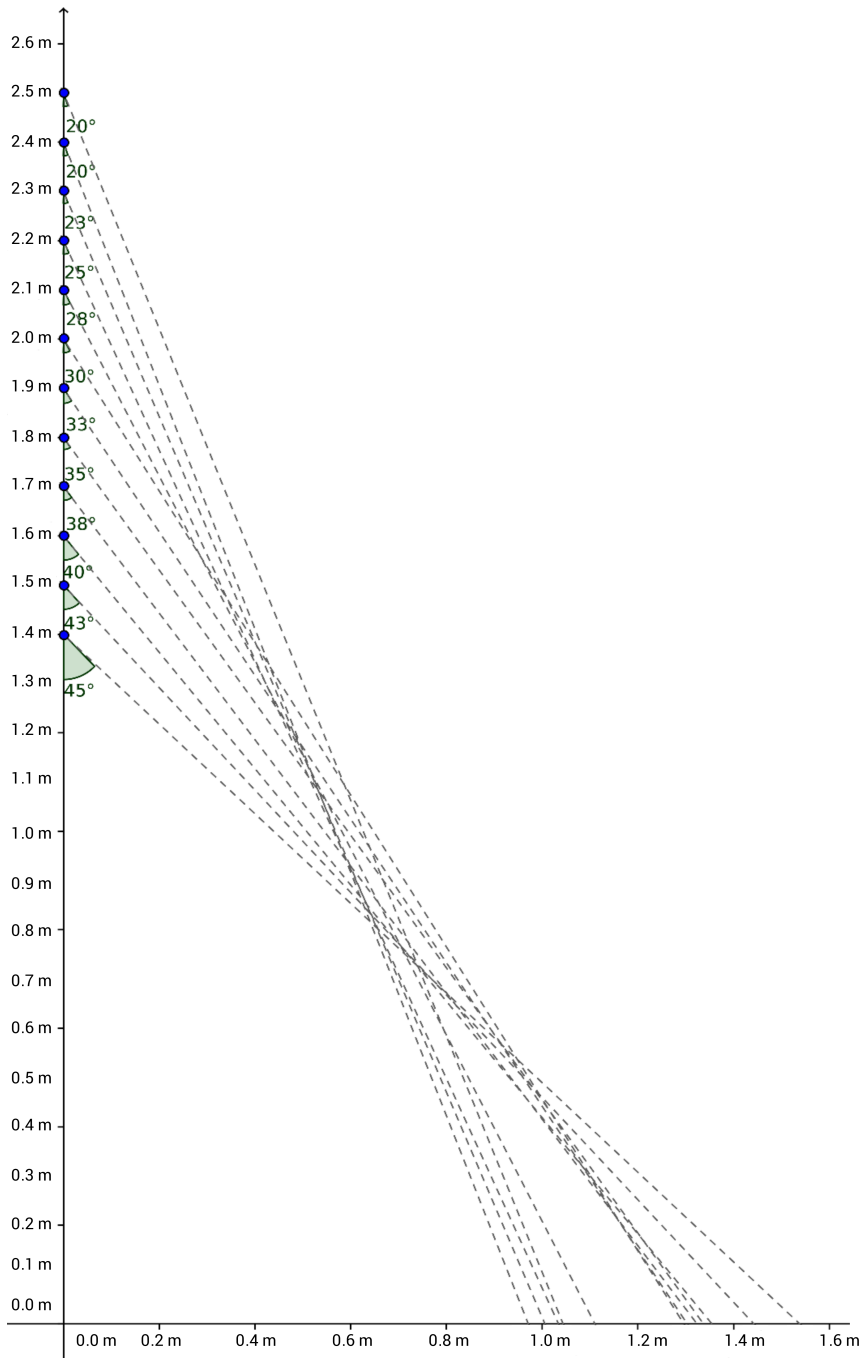


Figure 4.16: Camera position, tilt angles, and camera focus in different heights (in meters)

4.14. Chapter Summary

space, such in a library corridor in between two shelves (1 meter wide), the camera height and the tilt angle did matter in generating maps. The camera coverage coverage of the map with a single camera depends on the height of the camera as well as the tilt angle. It was observed that the higher the camera was the less was the tilt angle and worse the quality of the map generated. The best quality was generated at the height of 160 cm with the tilt angle of 40° facing down to the floor. In other word $(90 - 40) = 50^\circ$ down from the horizon level. However, with this height the upper part of the indoor was not covered in the map. As a result the camera has been placed on the upper part of the robot about 2.0 m up with a $(90 - 30) = 60^\circ$ tilt down from the horizontal level facing the floor. This allowed to cover the upper and lower part of the environment. We also find that the minimum distance from camera to object is required about 0.4 m in a library corridor to get an acceptable textured point cloud map. Figure 4.16 depicts the relationship between the camera tilt angle and the coverage area relationship in different heights.

4.14 Chapter Summary

In this chapter, we have presented the indoor 3D mapping with RGB-D mapping and information presentation with rviz. Here, we first review related work to RGB-D mapping and information visualization on 3D maps. Then we choose RTAB-Map to configure to perform indoor mapping with parametric customization. Here we present both the visual odometry based mapping and robot odometry based mapping. Then we implement an offline map publisher to load map for later use also. After loading the map, we show marker with RFID information on top of the map. The integration of ROS, SLAM and RFID is the novelty of the research.



5

Conclusion

5.1 Concluding Remarks

In this dissertation, two approaches for indoor mapping and information presentation have been presented. The first approach involved 2D image stitching to panoramas and inter-linking them to create the pseudo-3D model. The second approach involved 3D point cloud capturing and information presentation using ROS visualization tools. A 2D modeling is relatively simpler than 3D models; however, a 2D based indoor model do not have any depth information of the scene. In contrast, a 3D model extensively relies on depth information for an accurate volumetric representation of the real world. Although, a 3D model is intended to represent the environment model accurately, the depth measurement has to be highly accurate. RGB-D cameras are inexpensive, and the highest point resolution can be achieved as small as 4 millimeter, which is acceptable in many scenarios in robotics, however for a professional 3D modeling, a high precision 3D scanners or LiDARs with levels of noise filtering would still be required. Because, the quality of a 3D indoor mapping largely depends on the interior features such as textures, light condition, indoor structure, image capture distance from the camera, depth value accuracy, etc. One of the important concern regarding 3D modeling is the scalability. For practical use, point cloud maps should be leaner (smaller file size) to save and retrieve when needed. At present, the 3D maps are vastly large in file size and, in general, it takes a long time to load for viewing them. Although, map segmentation and other techniques can be incorporated to load maps incrementally, at present, there is no easy way of merging the segmented maps into a single map automatically. Manipulating a large map is still a challenging task with currently available desktop (or laptop) hardware. Also, point cloud map viewers, in general, are not

Chapter 5. Conclusion

unified, that is, there is no single standard of map viewer implementation. Researchers and professionals greatly rely on ad-hoc trials and errors, their previous experience and expertise to achieve a better-looking point cloud maps. The trials and errors, in turn, ends up trying vastly different algorithms and then finding the right approach.

Thus, the key for an automated indoor modeling is to focus on the simplest model that is dimensionally accurate and volumetrically correct.

5.2 Contributions

The specific research contributions are -

1. Retail space indoor view creation with Google Maps Street View API,
2. Automated panorama stitching,
3. RFID information presentation on the street view-like indoor view,
4. RGB-D Camera and SLAM integration for viewing RFID tagged object information on the point cloud maps,
5. Offline 3D point cloud (color) map loading in rviz,
6. RFID sensor information presentation by Marker display on point cloud map in rviz, etc.

5.3 Future Works

A number of point cloud map refinement approaches can be taken for improving the quality of a 3D point cloud map. The future works of the currently presented work are outlined as below:

- **MeshLab**¹ / **CloudCompare**²
Meshlab filters such as remove duplicate vertex, merge close vertices etc., are useful for filtering noise in the point cloud. CloudCompare subsampling is another efficient way to remove redundant points of a point cloud. CloudCompare subsampling seems to be faster than

¹<http://meshlab.sourceforge.net/>

²<http://www.danielgm.net/cc/>

5.4. Other Research and Development

Meshlab poisson disk sampling algorithm. Both Meshlab and CloudCompare filters can be applied using their command line interface. CloudCompare explicitly has full set of commands and custom MLX scripts can be programmed for Meshlab.

- **Scan Matching**

3D laser scanners or LiDARs are able to scan the environment surface with higher precision than a RGB-D camera. Usually high precision LiDAR scanners are expensive, however with less expensive LiDAR scanner and RGB-D camera combined scan match may improve the point cloud maps.

- **Point Cloud Segmentation**

Point Cloud Segmentation is a technique to segment a large point cloud to smaller clouds. Point segmentation allows faster loading of point clouds that can be adapted with a point cloud map viewer.

- **Potree**

Potree is a three.js³ WebGL renderer which converts the point clouds to octree to view point clouds incrementally over HTTP protocol by web browser (Figure A.1, A.2, Appendix A). With the WebGL programming, RFID information may also be presented.

5.4 Other Research and Development

In addition to time spent working on this dissertation, the author was also involved in other research and developments briefly mentioned as follows.

5.4.1 Speech Interaction Application

Speech interaction has pros and cons. However, a small set of speech interaction may be a medium of interaction for visually impaired people. In this project, a speech interaction Android application was developed for locating RFID tagged items on smart shelves. A smart shelf is equipped with RFID reader and antenna to scan and update item presence within the shelf. The Android application was developed to make voice query and find approximate location of items on the shelf, specially for visually impaired. A small set of speech dialog was designed using Google Speech API for visually impaired as assistant towards independent living (Figure 5.1).

³<http://threejs.org/>

Chapter 5. Conclusion

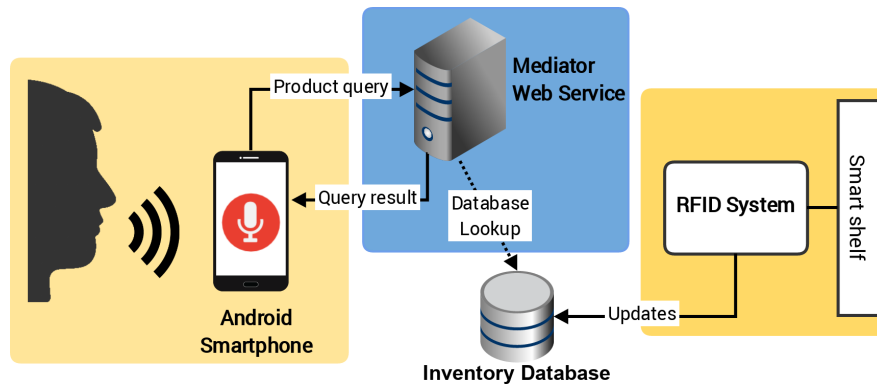


Figure 5.1: The speech interaction application architecture

List of Publications

Year 2015

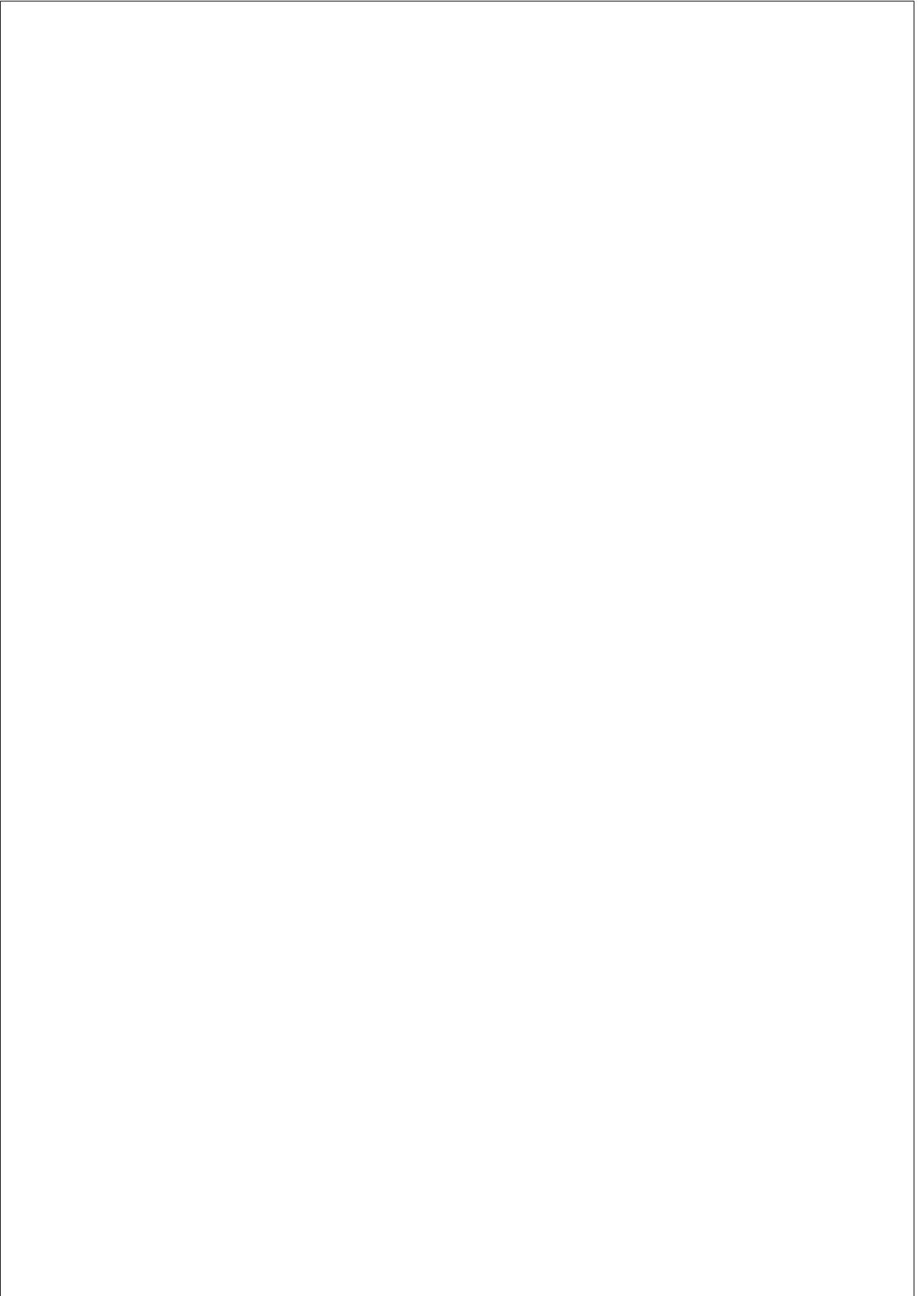
Nur, Kamruddin; Morenza-Cinos, Marc; Carreras, Anna; Pous, Rafael, **“Projection of RFID-Obtained Product Information on a Retail Stores Indoor Panoramas,”** in Intelligent Systems, IEEE , vol.30, no.6, pp.30-37, Nov.-Dec. 2015. DOI: 10.1109/MIS.2015.90

Kamruddin Nur, Zulqarnain Rashid, and Rafael Pous. 2015. **A smart-phone application for voice browsing RFID smart shelves.** In Proceedings of the 2015 ACM conference on Mobile and Ubiquitous Multimedia (MUM). (To appear.)

Year 2013

Anna Carreras, Marc Morenza-Cinos, Rafael Pous, Joan Melià-Seguí, Kamruddin Nur, Joan Oliver, and Ramir De Porrata-Doria. 2013. **STORE VIEW: pervasive RFID indoor navigation based retail inventory management.** In Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication (UbiComp '13 Adjunct). DOI: 10.1145/2494091.2496011.

Zulqarnain Rashid, Kamruddin Nur, Anna Carreras, and Rafael Pous. 2013. **Browsing reality: dynamic contextualization in human scale smart spaces.** In Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication (UbiComp '13 Adjunct). DOI=10.1145/2494091.2494096.



A

Appendix

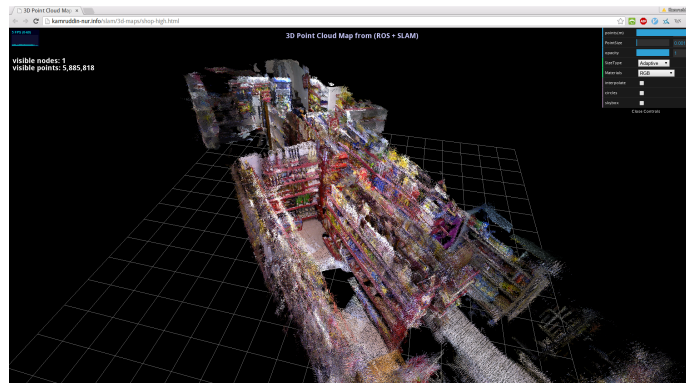


Figure A.1: Shop point cloud map (rendered by Potree) to view on web page

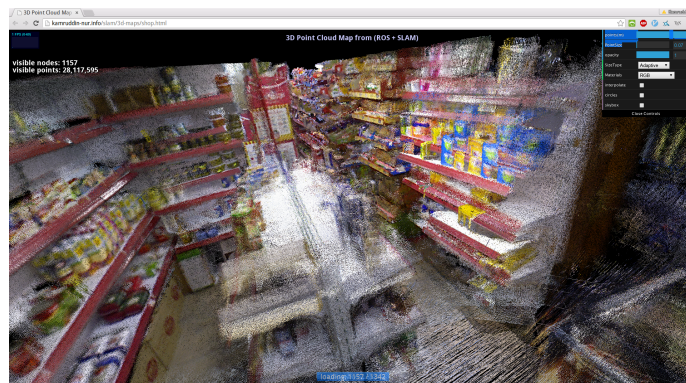
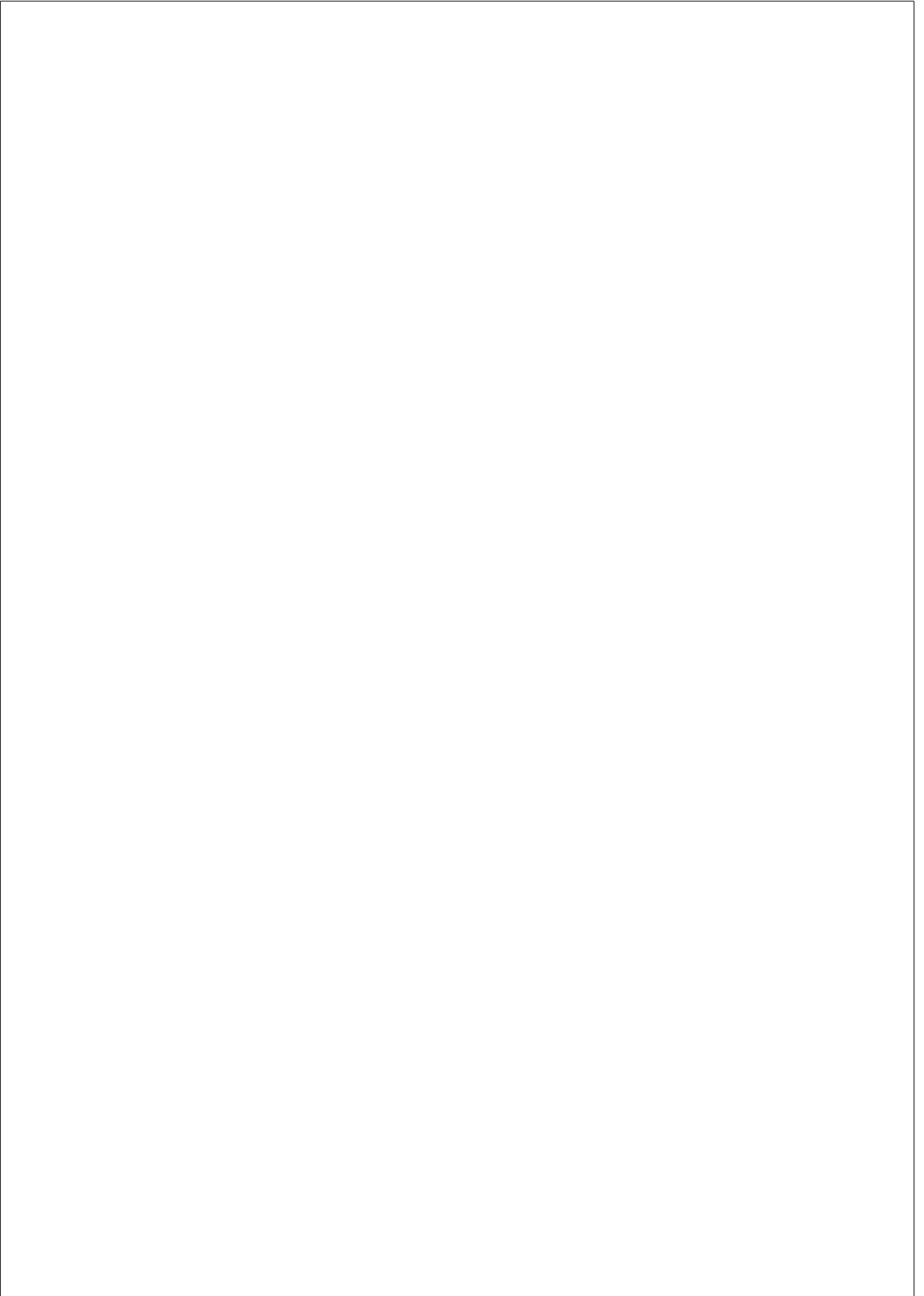


Figure A.2: Shop point cloud (rendered by Potree) zoom view on web page



Bibliography

- [1] J. Symonds, *Auto-identification and ubiquitous computing applications: RFID and smart technologies for information convergence*. 2009. (Cited on page 1).
- [2] J. Kroon, C. Vrijlandt, S. de Ridder, and F. van der Reep, “Rfid in retail: New approaches, new viewpoints,” *European Retail Digest*, vol. 55, p. 33, 2007. (Cited on pages 2 and 18).
- [3] G. Roussos, “Enabling rfid in retail,” *Computer*, vol. 39, pp. 25–30, March 2006. (Cited on page 2).
- [4] PC Magazine, “Google Indoor Business.” <http://www.pcmag.com/article2/0,2817,2395479,00.asp>, 2011. (Cited on page 2).
- [5] A. Carreras, M. Morenza-Cinos, R. Pous, J. Melià-Seguí, K. Nur, J. Oliver, and R. De Porrata-Doria, “Store view: Pervasive rfid & indoor navigation based retail inventory management,” in *Proceedings of the ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*, UbiComp ’13 Adjunct, pp. 1037–1042, ACM, 2013. (Cited on pages 2 and 19).
- [6] S. Samat, “The 3 New Realities of LOCAL RETAIL.” https://think.storage.googleapis.com/docs/how-digital-connects-shoppers-to-local-stores_articles.pdf, 2014. (Cited on page 3).
- [7] S. F. El-Hakim, P. Boulanger, F. Blais, and J.-A. Beraldin, “Sensor based creation of indoor virtual environment models,” in *Virtual Systems and MultiMedia, 1997. VSMM '97. Proceedings., International Conference on*, pp. 50–58, 1997. (Cited on page 3).

Bibliography

- [8] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, “Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments,” *Int. J. Rob. Res.*, vol. 31, pp. 647–663, Apr. 2012. (Cited on pages 3 and 43).
- [9] J.-P. Curty, M. Declercq, C. Dehollain, and N. Joehl, “Introduction to rfid,” *Design and Optimization of Passive UHF RFID Systems*, Springer, pp. 37–48, 2007. (Cited on page 7).
- [10] EPCGlobal Gen2 Specification, *EPC Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860 MHz – 960 MHz*, 1.2 ed., 2015. (Cited on page 8).
- [11] SMARTRAC N.V., “WEB G2iL,” 2013. Available at: https://www.smartrac-group.com/files/content/Products_Services/PDF/Web_G2iL.pdf. (Cited on page 8).
- [12] SMARTRAC N.V., “MINIWEB Monza R6,” 2015. Available at: https://www.smartrac-group.com/files/content/Products_Services/PDF/SMARTRAC_MINIWEB_IMPINJ_MONZA_R6_ETSI.pdf. (Cited on page 8).
- [13] SMARTRAC N.V., “SHORTDIPOLE Impinj Monza 5,” 2013. Available at: https://www.smartrac-group.com/files/content/Products_Services/PDF/ShortDipole_M4.pdf. (Cited on page 8).
- [14] Keonn Technologies, S.L., “Advanreader 150 uhf rfid reader.” <http://keonn.com/rfid-components/readers/advanreader-150.html>, 2015. (Cited on page 9).
- [15] Keonn Technologies, S.L., “Advanscan handheld uhf rfid reader.” <http://www.keonn.com/systems/view-all-2/cloud-based-handheld-reader.html>, 2015. (Cited on page 9).
- [16] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004. (Cited on pages 10, 44, and 45).
- [17] H. Bay, T. Tuytelaars, and L. Van Gool, “Surf: Speeded up robust features,” in *Computer vision—ECCV 2006*, pp. 404–417, Springer, 2006. (Cited on pages 10, 44, and 45).

Bibliography

- [18] S. Brahmbhatt, “Real-time object detection in opencv using surf.” <http://docs.opencv.org/modules/stitching/doc/stitching.html>, 2012. (Cited on page 10).
- [19] E. Rosten and T. Drummond, “Machine learning for high-speed corner detection,” in *Computer Vision–ECCV 2006*, pp. 430–443, Springer, 2006. (Cited on page 11).
- [20] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, “Brief: Binary robust independent elementary features,” *Computer Vision–ECCV 2010*, pp. 778–792, 2010. (Cited on page 11).
- [21] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “Orb: an efficient alternative to sift or surf,” in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 2564–2571, IEEE, 2011. (Cited on pages 12, 44, and 45).
- [22] H. Dersch, “Panorama tools. open source software for immersive imaging. international vr photography conference. berkeley, 2007,” 2007. (Cited on page 13).
- [23] M. Kulig, N. Pettyjohn, E. Saunders, N. Jeffrey, and D. Cannarozzi, “Systems and methods for displaying the location of a product in a retail location.” Patent. US 20150170258 A1. June 2015. (Cited on page 14).
- [24] K. Mori, S. Sudarsan, D. Bennett, Y. Jiang, Y. Pellet, J. Singh, and S. Sidhu, “Location determination, mapping, and data management through crowdsourcing.” Patent. US 20150312722 A1. October 2015. (Cited on page 15).
- [25] Google Inc., “Google Business View.” <https://www.google.com/maps/about/partners/businessview/>, 2014. (Cited on page 18).
- [26] Google Inc., “Google Art Project.” <http://www.google.com/culturalinstitute/project/art-project>, 2013. (Cited on page 18).
- [27] M. Colbert, J.-Y. Bouguet, J. Beis, S. Childs, D. Filip, L. Vincent, J. Lim, and S. Satkin, “Building indoor multi-panorama experiences at scale,” in *ACM SIGGRAPH 2012 Talks*, p. 24, ACM, 2012. (Cited on page 18).

Bibliography

- [28] G. Schroth, “Improved positioning indoors.” <https://www.tum.de/en/about-tum/news/press-releases/long/article/30040/>, 2012. (Cited on page 18).
- [29] K. J. Dana, B. van Ginneken, S. K. Nayar, and J. J. Koenderink, “Reflectance and texture of real-world surfaces,” *ACM Trans. Graph.*, vol. 18, pp. 1–34, Jan. 1999. (Cited on page 19).
- [30] J. Francis, U. Drolia, K. Mankodiya, R. Martins, R. Gandhi, and P. Narasimhan, “Metabot: Automated and dynamically schedulable robotic behaviors in retail environments,” in *Robotic and Sensors Environments (ROSE), 2013 IEEE International Symposium on*, pp. 148–153, IEEE, 2013. (Cited on page 19).
- [31] G. Inc., “Google Maps API Street View Service.” <https://developers.google.com/maps/documentation/javascript/streetview>, 2013. (Cited on pages 19 and 23).
- [32] P. d’Angelo, “Hugin - Panorama photo stitcher.” <http://hugin.sourceforge.net/>, 2013. (Cited on page 19).
- [33] Keonn Technologies, S.L., “Components for your RFID projects.” <http://www.keonn.com/rfid-components/view-all/frontpage.html>, 2012. (Cited on page 20).
- [34] M. Brown and D. G. Lowe, “Automatic panoramic image stitching using invariant features,” *International journal of computer vision*, vol. 74, no. 1, pp. 59–73, 2007. (Cited on page 22).
- [35] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon, “Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera,” in *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST ’11, (New York, NY, USA), pp. 559–568, ACM, 2011. (Cited on pages 34 and 43).
- [36] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, “Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments,” *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 647–663, 2012. (Cited on page 34).

Bibliography

- [37] X. Xiong and D. Huber, “Using context to create semantic 3d models of indoor environments,” in *in Proceedings of the British Machine Vision Conference (BMVC)*, 2010. (Cited on page 36).
- [38] Wikipedia, “Triangulation.” <https://en.wikipedia.org/wiki/Triangulation>, 2015. (Cited on page 38).
- [39] Z. Zhang, “Microsoft kinect sensor and its effect,” *MultiMedia, IEEE*, vol. 19, no. 2, pp. 4–10, 2012. (Cited on pages 38 and 42).
- [40] V. Castaneda and N. Navab, “Time-of-Flight and Kinect Imaging.” Seminar, Technische Universitat Munchen. Online Available: http://campar.in.tum.de/twiki/pub/Chair/TeachingSs11Kinect/2011-DSensors_LabCourse_Kinect.pdf, 2011. Accessed: 2015-09-30. (Cited on page 38).
- [41] ASUSTeK Computer Inc., “Xtion PRO LIVE.” https://www.asus.com/es/3D-Sensor/Xtion_PRO_LIVE/, 2015. (Cited on pages 38 and 42).
- [42] K.-H. Lee and R. Ehsani, “Comparison of two 2d laser scanners for sensing object distances, shapes, and surface patterns,” *Comput. Electron. Agric.*, vol. 60, pp. 250–262, Mar. 2008. (Cited on page 39).
- [43] Llamazares, E. Molinos, M. Ocaña, L. Bergasa, N. Hernández, and F. Herranz, “3d map building using a 2d laser scanner,” in *Computer Aided Systems Theory – EUROCAST 2011* (R. Moreno-Díaz, F. Pichler, and A. Quesada-Arencibia, eds.), vol. 6928 of *Lecture Notes in Computer Science*, pp. 412–419, Springer Berlin Heidelberg, 2012. (Cited on page 39).
- [44] Sick Sensor Intelligence, “LMS100-10000.” https://www.sick.com/media/pdf/1/41/841/dataSheet_LMS100-10000_1041113_en.pdf, 2015. (Cited on page 40).
- [45] Hokuyo Automatic Co. Ltd., “UTM-30LX.” https://www.hokuyo-aut.jp/02sensor/07scanner/utm_30lx.html, 2009. (Cited on page 40).
- [46] Faro, “Faro Focus^{3D} X 130.” <http://www.faro.com/en-us/products/3d-surveying/faro-focus3d/overview>, 2015. (Cited on page 40).

Bibliography

- [47] Trimble Navigation Limited, “Trimble TX8.” http://www.trimble.com/3d-laser-scanning/tx8.aspx?tab=Technical_Specs, 2015. (Cited on page 40).
- [48] W. Choi, C. Pantofaru, and S. Savarese, “Detecting and tracking people using an rgb-d camera via multiple detector fusion,” in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pp. 1076–1083, IEEE, 2011. (Cited on page 41).
- [49] wikipedia, “PrimeSense.” <https://en.wikipedia.org/wiki/PrimeSense>, 2014. (Cited on page 42).
- [50] Occipital, Inc., “OpenNI 2 SDK Binaries & Docs.” <http://structure.io/openni>, 2015. (Cited on page 42).
- [51] SoftKinect, “DS325 Datasheet.” http://www.softkinetic.com/Portals/0/Download/WEB_20120907_SK_DS325_Datasheet_V2.1.pdf, 2012. (Cited on page 42).
- [52] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, “RGB-D mapping: Using depth cameras for dense 3d modeling of indoor environments,” in *The 12th International Symposium on Experimental Robotics (ISER)*, 2010. (Cited on page 43).
- [53] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald, “Kintinuous: Spatially extended kinectfusion,” 2012. (Cited on page 43).
- [54] T. Whelan, H. Johannsson, M. Kaess, J. Leonard, and J. McDonald, “Robust real-time visual odometry for dense RGB-D mapping,” in *IEEE Intl. Conf. on Robotics and Automation, ICRA*, (Karlsruhe, Germany), May 2013. (Cited on page 44).
- [55] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, “An evaluation of the rgb-d slam system,” in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 1691–1696, May 2012. (Cited on page 44).
- [56] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, “3-d mapping with an rgb-d camera,” *Robotics, IEEE Transactions on*, vol. 30, pp. 177–187, Feb 2014. (Cited on page 44).

Bibliography

- [57] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, “G2o: A general framework for graph optimization,” in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 3607–3613, May 2011. (Cited on page 44).
- [58] J. Shi and C. Tomasi, “Good features to track,” in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR’94., 1994 IEEE Computer Society Conference on*, pp. 593–600, IEEE, 1994. (Cited on page 44).
- [59] M. Labbe and F. Michaud, “Appearance-based loop closure detection for online large-scale and long-term operation,” *Robotics, IEEE Transactions on*, vol. 29, pp. 734–745, June 2013. (Cited on pages 44 and 45).
- [60] M. Labbe and F. Michaud, “Online global loop closure detection for large-scale multi-session graph-based slam,” in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pp. 2661–2666, Sept 2014. (Cited on page 44).
- [61] J. Sivic and A. Zisserman, “Video google: a text retrieval approach to object matching in videos,” in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pp. 1470–1477 vol.2, Oct 2003. (Cited on page 45).
- [62] G. Grisetti, S. Grzonka, C. Stachniss, P. Pfaff, and W. Burgard, “Efficient estimation of accurate maximum likelihood maps in 3d,” in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pp. 3472–3478, IEEE, 2007. (Cited on page 46).
- [63] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz, “Towards 3d point cloud based object maps for household environments,” *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 927–941, 2008. (Cited on page 46).
- [64] F. Tian, S. Xu-kun, L. Xian-mei, and X. Hong-tao, “3d model multiple semantic automatic annotation for small scale labeled data set,” in *Virtual Reality and Visualization (ICVRV), 2011 International Conference on*, pp. 193–198, Nov 2011. (Cited on page 46).
- [65] R. Rusu and S. Cousins, “3d is here: Point cloud library (pcl),” in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 1–4, May 2011. (Cited on page 47).

Bibliography

- [66] PCL, “Point Cloud Library.” <http://pointclouds.org/>, 2015. (Cited on page 47).
- [67] VTK, “Overview, year = 2015, howpublished=<http://www.vtk.org/overview/>.” (Cited on page 47).
- [68] G. Turk, “The PLY Polygon File Format.” <http://www.dcs.ed.ac.uk/teaching/cs4/www/graphics/Web/ply.html>, 1994. (Cited on page 49).
- [69] T. Parisi, *WebGL: up and running*, pp. 2–3. ” O’Reilly Media, Inc.”, 2012. (Cited on page 49).
- [70] Markus Schütz, “Potree.” <http://potree.org/>, 2015. (Cited on page 49).
- [71] ros.org, “rviz.” <http://wiki.ros.org/rviz>, 2014. (Cited on page 50).