## Alejandro Pérez Yus

Scene structure recovery from omnidirectional and depth cameras for assistive computer<br>vision

Departamento
Informática e Ingeniería de Sistemas

Director/es
GUERRERO CAMPO, JOSE JESUS LOPEZ NICOLAS, GONZALO

# SCENE STRUCTURE RECOVERY FROM OMNIDIRECTIONAL AND DEPTH CAMERAS FOR ASSISTIVE COMPUTER VISION 

Autor
Alejandro Pérez Yus

Director/es
GUERRERO CAMPO, JOSE JESUS
LOPEZ NICOLAS, GONZALO

## UNIVERSIDAD DE ZARAGOZA

Informática e Ingeniería de Sistemas

# Scene structure recovery FROM OMNIDIRECTIONAL AND DEPTH CAMERAS FOR ASSISTIVE COMPUTER VISION 

## Alejandro Pérez Yus

Ph.D. Dissertation

AdVISORS $\mid$ Gonzalo López Nicolás<br>José Jesús Guerrero Campo

Programa de Doctorado en Ingeniería de Sistemas e Informática Universidad de Zaragoza - Spain

MAY 2018

## Acknowledgments

I would like to dedicate this thesis to those who have taken part in this journey, whether directly (work matters) or indirectly (by keeping me sane).

First, I must give my most sincere gratitude to Josechu Guerrero and Gonzalo López. Thanks for encouraging and giving me a chance, for providing me time and valuable advice, and for making me enjoy the ride during all these years.

Secondly, I have to manifest my profound gratitude to the committee members Óscar Reinoso, Pablo F. Alcantarilla, Jesús Bermúdez, Daniel Gutiérrez y Eduardo Fernández; and the international reviewers Renato Martins and José M. Carranza. Thanks for spending your time evaluating my work, I hope it wasn't too boring to read!

This thesis has been possible thanks to the funding by the Spanish Ministry of Economy, Industry and Competitiveness, with the scholarship that I was conceded (BES-2013-065834), and the projects DPI2012-31781, DPI2014-61792-EXP, and DPI2015-65962-R (MINECO/FEDER, UE). I'm especially grateful for the opportunities they gave me to attend to conferences and perform abroad stays in two renowned research centers: the University of Washington and INRIA Sophia-Antipolis, with none others than Dieter Fox and Patrick Rives. Apart from the academic value, these stays were amazing personal experiences, thanks to the wonderful people who crossed my path, like Luis Puig, Eduardo, Renato, and pinche Rocío.

And I cannot forget about the people from my lab (and the other slightly less awesome labs on this floor). I've been here a long time, which has given me the possibility of meeting so many extraordinary people, that I can't even attempt to name them all. I will always remember drawing movies with the old guard in the early years, suffering Friday mornings with the new guard more recently, or some Venezuelan guy who was there every single day. There was also a Mexican who ate all my Serrano ham, and the truest Italians that I will ever meet. Some even dare to collaborate with me, like Jesús, Daniel, or my young padawan Clara. To all of you, let the friendship that coffee breaks have created never end!

To conclude, I have to dedicate this to all my friends from the outside world, and my family. To the marqueses, the beautiful ones, the park's table, and the engineers around the world, for allowing me to disconnect and have fun. To Marina, and her satellites Jefa and Argui, for becoming my new favorite living beings. To King Crimson, for being the OST. And to my parents Alfredo y María, for being as supportive as they were during all these years. And yes mom, I'm happy.

## Resumen

La visión por computador es un campo de investigación en continua expansión, que tiene como principal objetivo desarrollar algoritmos para extraer información del entorno a partir de imágenes. Las aplicaciones de este campo son innumerables, y continuos avances muestran un prometedor futuro por delante. En la actualidad, la visión por computador se encuentra presente en muchas áreas, entre las que pueden incluirse, por ejemplo, la conducción autónoma, el control de drones o la robótica. En particular, nuestra motivación está dentro del contexto de la asistencia personal, en donde las técnicas de visión por computador se desarrollan para estar al servicio de las personas, y especialmente de personas con discapacidades. En esta tesis, nos centramos en el problema de desarrollar un asistente basado en sistemas de visión artificial que permita a las personas con discapacidad visual desplazarse de manera segura y eficiente. Existen dispositivos ampliamente extendidos para la ayuda de ciegos, como el bastón blanco o el perro lazarillo. Si bien este tipo de dispositivos son de utilidad en situaciones simples y de corto alcance, creemos que ha llegado el momento de utilizar los avances tecnológicos para mejorar la capacidad y autonomía del usuario al proporcionarle información adicional. La aparición reciente en el mercado de cámaras RGB-D de bajo coste, así como sus posibilidades de miniaturización, demuestran que el desarrollo de sistemas portables con dicha tecnología es más que plausible en un futuro próximo. En esta tesis el uso de cámaras RGB-D es un elemento central, ya que permiten capturar simultáneamente información tridimensional junto con el color, siendo especialmente útiles para detectar formas y obstáculos. Para superar algunas limitaciones de este tipo de cámaras, se ha investigado su uso combinado con cámaras omnidireccionales, creando nuevos sistemas que permitan capturar más información del entorno desde un único punto de vista.

Un objetivo general de esta tesis ha sido desarrollar diversos métodos para obtener información relevante de la escena que pueda ayudar en la navegación humana. De la amplia gama de posibles tipos de información que se pueden obtener del entorno, nos centramos en la extracción de información estructural representativa de cualquier entorno construido por el hombre. Particularmente, hemos desarrollado un sistema capaz de encontrar el suelo y la orientación de la escena, así como los posibles obstáculos presentes en el entorno. Este sistema de navegación, que está basado en el uso de la percepción de profundidad de la escena, ha sido desarrollado dando como resultado la contribución de un algoritmo de detección, modelado y recorrido de escaleras. El método
propuesto es capaz de detectar escaleras ascendentes y descendentes, obtener su orientación y dimensiones, así como re-localizar continuamente al sujeto durante su recorrido utilizando un algoritmo de odometría visual que funciona en paralelo.

Con el objetivo de proporcionar un mejor entendimiento visual de la escena, proponemos estimar la distribución estructural de la habitación, lo cual podría ayudar en tareas como navegación, reconocimiento de escenas o detección de objetos. Sin embargo, una de las principales limitaciones de las cámaras RGB-D es su reducido campo de vista. Para superar este inconveniente, se presenta la siguiente contribución que consiste en el desarrollo y calibración de un novedoso sistema de visión que permite extender el campo de vista mediante cámaras omnidireccionales. Concretamente, se ha desarrollado un nuevo sistema híbrido con lente de ojo de pez y cámara de profundidad, así como el necesario método de calibración del sistema. Además, para extender el método desarrollado a otros sistemas, se ha elaborado un segundo método de calibración basado en observaciones de líneas que nos permite calibrar múltiples combinaciones de cámaras, sin requerimiento de solapamiento de campos de vista y sin necesidad de construir ningún patrón de calibración. El sistema híbrido de cámaras desarrollado permite vislumbrar nuevas posibilidades, que motivan la siguiente contribución: el diseño de un método de estimación de la estructura del entorno que permite obtener reconstrucciones 3D a escala de la escena. Este método se fundamenta en el uso del amplio campo de vista del ojo de pez y de la percepción 3D de la cámara de profundidad.

Si bien la extracción de información relevante del entorno es un problema relevante, también lo es la comunicación de la información obtenida al usuario. Éste es un problema complejo que no es tratado en la mayoría de sistemas de asistencia basados en visión por computador de la literatura. Dados los nuevos avances en visión prostética, consideramos viable el contribuir en este área con las técnicas de visión por computador desarrolladas en esta tesis. Actualmente, pacientes con prótesis visuales son capaces de ver una distribución de puntos de luz, llamados fosfenos. Sin embargo, actualmente el conjunto de fosfenos tienen resolución espacial y rango dinámico limitados. Nuestra siguiente contribución es el desarrollo de algoritmos de codificación de la información percibida en patrones de fosfenos, mediante una representación icónica que hace posible entender la escena para su navegación. Partiendo de la información de profundidad proporcionada por el sensor, el método desarrollado permite al usuario percibir a través de la codificación en fosfenos algo tan necesario como es la sensación de movimiento y profundidad del entorno, siendo la clave en técnicas de representación icónica propuestas.

## Abstract

Computer vision is an impressively expanding field of research that aims to develop algorithms to extract information of the environment using images. The applications of such field are endless, and continuous advances show a bright future ahead. Nowadays computer vision is applied to many areas, such as autonomous driving, drones or robotics. In particular, our motivation lies within the context of human assistance, where computer vision techniques are developed to be of service of people, especially people with disabilities. In this thesis, we particularly focus on the problem of developing a camera-based assistant that allows visually impaired people to move safe and efficiently. Popular low-tech aids for the blind, such as the white cane, work well for simple, short-range situations, but we believe technological advances can enhance the overall experience by providing useful information. Particularly, the advent a few years ago of consumer RGB-D cameras and its current miniaturization possibilities paves the way for a wearable system with such type of camera quite affordable and plausible in the near future. We use RGB-D cameras as a central element of our system, since they provide three-dimensional information alongside color, making them especially useful to detect shapes and obstacles. Additionally, we have also explored their combination with omnidirectional cameras, to create new powerful systems able to capture more information of the environment at once.

A main goal of this thesis is to develop a variety of methods to recover relevant information of the scene that would help in human navigation. Since this problem is too general, we focused on addressing the extraction of common structural information that can be found in any man-made environment. Particularly, we developed a system able to find the floor and the orientation of the scene, in which obstacles along the way could be detected using depth perception. We extended this navigational system leading to our first major contribution, which is the elaboration of a stair detection, modeling and traversal algorithm. This proposed method is able to detect ascending and descending staircases, obtain their orientation and dimensions, and continuously re-localize the subject during the traversal using a visual odometry algorithm running in parallel.

Additionally, in order to enable a better understanding of the scene, we propose to estimate the layout of the room, which could help in tasks such as navigation, scene recognition or object detection. However, one of the main limitations of the RGB-D cameras is their narrow field of view. In order to overcome this limitation, our next contribution was to develop and calibrate novel camera systems to extend the field of view by means of omnidirectional cameras. In particular, we developed a fisheye and depth
hybrid camera system and the corresponding method of calibration for these type of systems. Moreover, to extend the possibilities to other systems, we developed a second calibration method based on line observations able to calibrate multiple camera combinations, without overlapping field of view requirements and without needing to build a calibration pattern. With our hybrid camera system we open new possibilities such as the design of a layout estimation method, able to obtain full-scaled 3D reconstructions of the scene, benefiting from the wide field of view of the fisheye and the data from the depth camera, which is also a relevant contribution.

The communication of the perceived information to the user is a complex problem not directly treated in many of the assistive computer vision systems of the literature. Given the new advances in prosthetic vision, we investigated the application of the developed computer vision techniques to this area. Patients with visual prosthesis are able to see an array of light dots, called phosphenes. However, nowadays phosphene arrays have limited spatial resolution and dynamic range. Our next major contribution was the challenging task of codifying the information extracted to phosphene patterns, with an iconic representation that makes possible to understand the scene with the limitations given. In particular, we use the depth camera to extract free walking space in the scene, and using an iconic-based approach allows to provide comfortable and informative visual cues that other existing methods usually lack.

## Contents

1 Introduction ..... 1
1.1 Motivation ..... 2
1.2 System framework ..... 3
1.2.1 Assistive systems from the literature ..... 4
1.2.2 Proposed framework ..... 11
1.3 Goals and contributions ..... 13
1.3.1 Stairs detection, modeling and traversal ..... 14
1.3.2 Unconventional camera systems and calibration ..... 15
1.3.3 Scaled layout from wide field of view RGB-D camera ..... 16
1.3.4 Iconic phosphenic representation for human navigation with visual prosthe- ses ..... 17
1.3.5 Complementary contributions ..... 18
1.4 Outline ..... 21
2 Stairs detection, modeling and traversal ..... 23
2.1 Introduction ..... 24
2.2 Related work ..... 26
2.3 Scene segmentation and classification ..... 31
2.3.1 Segmentation ..... 32
2.3.2 Scene orientation ..... 34
2.3.3 Classification of regions ..... 37
2.4 Stair detection, modeling and traversal ..... 38
2.4.1 Stair detection ..... 39
2.4.2 Stair modeling ..... 42
2.4.3 Visual odometry ..... 46
2.4.4 Stair traversal ..... 49
2.5 Experiments ..... 51
2.5.1 Stairs detection with public dataset ..... 52
2.5.2 Quality of the modeling ..... 54
2.5.3 Computation time analysis ..... 57
2.5.4 Evaluation of the traversal algorithm ..... 58
2.6 Discussion ..... 60
3 CALIBRATION OF HYBRID CAMERA SYSTEMS ..... 63
3.1 Introduction ..... 64
3.2 Related work ..... 66
3.3 A novel hybrid camera system with depth and fisheye ..... 69
3.3.1 System description ..... 70
3.3.2 Depth-fisheye camera calibration ..... 72
3.3.3 Experiments ..... 77
3.4 Extrinsic calibration of multiple RGB-D cameras from lines ..... 81
3.4.1 Line extraction and matching ..... 82
3.4.2 Extrinsic calibration from line observations ..... 86
3.4.3 Observability ..... 88
3.4.4 Experimental validation ..... 89
3.5 Discussion ..... 96
4 Scaled Layout recovery with wide field of VIEW RGB-D ..... 97
4.1 Introduction ..... 98
4.2 Related work ..... 100
4.3 Depth and fisheye images processing ..... 102
4.3.1 System calibration ..... 103
4.3.2 Line extraction in the fisheye image ..... 104
4.3.3 Estimation of the vanishing points ..... 105
4.3.4 Depth information processing ..... 107
4.3.5 Classification of lines ..... 108
4.4 Layout estimation ..... 108
4.4.1 Corner extraction ..... 109
4.4.2 Layout hypotheses generation ..... 117
4.4.3 Evaluation of the hypotheses ..... 123
4.4.4 Scaling of hypotheses ..... 124
4.5 Experiments ..... 124
4.5.1 Corner extraction ..... 125
4.5.2 Layout estimation ..... 129
4.5.3 Advantages of the camera pairing ..... 135
4.5.4 Results with Google Tango ..... 135
4.6 Discussion ..... 136
5 ICONIC REPRESENTATION FOR NAVIGATION WITH PROSTHETIC VISION ..... 139
5.1 Introduction ..... 140
5.1.1 Background on prosthetic vision ..... 140
5.1.2 Models of phosphene patterns ..... 141
5.1.3 Simulated Prosthetic Vision ..... 141
5.1.4 Problem definition ..... 145
5.1.5 Geometry and notation details ..... 146
5.2 Perception of free space and scene pose ..... 146
5.2.1 Relative movement of the user in the scene ..... 147
5.2.2 Orientation of the scene ..... 147
5.2.3 Perception of free space ..... 148
5.3 Iconic representation of free space ..... 150
5.3.1 Plucker polygon-ray intersection method ..... 152
5.4 Experiments of the iconic representation ..... 153
5.4.1 Experiments in simulated environments ..... 153
5.4.2 Experiments with real images ..... 157
5.5 Discussion ..... 165
6 CONCLUSIONS AND FUTURE WORK ..... 167
Bibliography ..... 173

## 1

## Introduction

The field that aims to make computers extract information from images is called computer vision. This field has been active for over half century, but it is increasingly growing as years go by, with continuous advances and growing number of areas of interest. The explosion of popularity is happening in the world of academia research, but also in the many incarnations of the industry. For instance, nowadays it is widely used in applications such as autonomous driving, robotics, augmented reality, aerial supervision, security cameras, social media platforms or automatic inspection in manufacturing processes. The main reason for this to happen is due to the astonishing results that the state of the art is achieving: now it is possible to accomplish tasks that were unthinkable a few decades back. In addition, the expectation of the field is to keep improving in the future, as newer methods arise and computational or technical enhancements happen. At this day and age, one could say computer vision is one of the most promising and exciting areas to work on.

In this thesis, the methods and algorithms proposed have been developed within this context, and thus they represent new contributions to the state of the art of computer vision. To present a more detailed description about the contents of the thesis, the following sections are going to answer the most important questions. In particular: Why? We describe the motivation of this thesis and the global aim that stimulates the research carried out through all these years (Section 1.1); How? the description of the system framework that is necessary to address the established problem (Section 1.2); and What? With the global aim in mind and the set of tools defined, we enumerate the particular goals we strive to achieve (Section 1.3). Alongside the goals, the particular contributions with its correspondent publications are also pointed out. In addition, Section 1.4 shows a brief description about the contents to come during the rest of the document.

### 1.1. Motivation

Many of the applications of computer vision techniques intend to directly or indirectly improve the quality of life of people. But some people have special needs, such as the disabled, or the elderly people. Here, of the many applications of computer vision, we focus on the development of algorithms and techniques in the framework of what is called assistive computer vision. In particular, our main concern has been to deal with visually impaired people, which are the ones that may benefit the most of having visual aid, since cameras perceive the type of stimuli that people with that disability cannot perceive properly. Visual impairment impacts many aspects of the life of the people affected by it, such as the ability to read text or interpret signs, the recognition of objects and people, or problems related to mobility. According to [Bourne et al., 2017], in 2015 there were 253 million of visually impaired people worldwide. Of them, 36 million are blind and 217 have moderate to severe vision impairment. Due to the growth and ageing of the world's population, the number of blind people could rise to 115 million in 2050 [Bourne et al., 2017]. This large amount of people is not negligible and shows the need of portable, practical, and highly functional assistive devices to help out people under these circumstances. Additionally, the problem aggravates in low-income countries and it is important to note that $81 \%$ of people living in blindness are aged 50 or above, so the system to develop should not be high priced or technologically complex. Giving assistance to the visually impaired has been a relevant topic for many years in the computer vision community, and many advances have been accomplished in some specific tasks. However, it still remains hard to integrate them naturally. The continuous improvements in electronics, increasing capacity and portability, makes the problem of creating assisting devices more relevant now than ever.

In this thesis, we pay special attention to the mobility problems derived from visual impairment. Safely moving from one point to another poses some difficulties, such as the presence of (moving) obstacles, traffic, orientation issues or the detection of curbs or doors. Some of these problems are usually overcome by their remaining senses, a previous knowledge of the environment or with the help of low-tech mobility aids such as the white cane or guide dogs. Nevertheless, even when moving in familiar environments or using a mobility aid, visually impaired people are still prone to be involved in accidents. According to the survey performed in [Manduchi and Kurniawan, 2011], $7 \%$ of the respondents experienced falls while walking at least once a month. Moreover, the frequency of accidents has nothing to do with the type of mobility aid or the number of times going out along unfamiliar routes.

Although the reliability, feedback, simplicity and price of mobility aids like the white cane seem hard to beat, we believe with modern sensors and techniques the navigating experience can be enhanced comfortably. The idea is to complement rather than replace, increasing and improving the amount of information they can already receive by other means. For example, while a white cane does a good job at detecting obstacles
and providing cues in short range, it is not as useful in mid and large range. And most importantly, when users are moving, obstacle avoidance is not their only concern: they would need as well to acquire some extra knowledge about the environment that allow them to know where to go or which path is the best to take. This necessity becomes even bigger when they are in an unknown environment with hardly any previous notion about it, since there is not cognitive map of the environment already acquired. Without appropriate orientation and position cues they might not be able to head to the shortest or safest path unless they get external help, which is an important limitation. The underlying challenge is to think about the information the user might want or need to receive. This is where the problem differs from the equivalent in robotics, since humans have their own decision-making process, having a more complex and spontaneous way of thinking, and no restrictions for any predetermined set of rules or behaviors. However, there are certain structures that are common in man-made constructions worldwide, like the stairs, doors or walls, that are essential in human navigation and thus required to be detected regardless of the environment the user is moving in. In this thesis, we have paid especial attention to those kinds of structures, since they are the foundation of any navigational aid such as the one we sought to create.

In essence, the main motivation of this thesis is to make the navigation of visually impaired people be an overall better experience, by enhancing the safety, efficiency and completeness they can get with current assistive devices. It is also important to note that, while our main concern is to improve the quality of life of those with visual impairment, the idea of enhancing the experience with cameras could be of use in other applications, such as people in especially demanding works (e.g. firemen, policemen), or tourists in unknown spaces. Besides, the methods and techniques here proposed may also be beneficial in other fields of research, such as robotics, augmented reality or SLAM (Simultaneous Localization and Mapping). For instance, some of the perception problems related to mobility of people are also shared in the field of service robotics (e.g. home telepresence robots), since in both cases the person or robot needs to move in the same type of environments. The possibility of generalizing our methods to other fields was encouraging, and served as additional motivation throughout the realization of the thesis.

### 1.2. SYSTEM FRAMEWORK

In the previous section it was stated the general problem we intend to solve in this thesis, which is providing assistance to the visually impaired people, with main focus on mobility issues. Not all cases of visual impairment are similar, since they may have different cause and different degree of vision loss, which may range from moderate low vision to total blindness. The scope of this thesis does not lie on attempting to restore the ability to see properly, which is an issue that concerns other areas of research. Instead, we believe that using modern assistive technologies we can improve the quality of life
of those affected by this condition, by helping them get through some limitations and recover some lost functionality for their daily life. The methodology resides in developing a system able to extract useful information from the environment and transmitting it to the user by other means than the visual system.

A system such as the one we aim to develop can be divided in two main modules or parts: perception and communication. The perception problem is the process of extracting information from the environment and interpreting it. This problem is usually approached with sensors that are able to capture raw data of the scene (e.g. laser scanners, sonars, cameras) and then processing this data in a computer for which appropriate algorithms have been developed. The communication problem is the process of transmitting the information already interpreted to the user. This problem is approached by taking advantage of the remaining senses of the user. For example, two widespread ways to achieve that are by sending audio signals or via haptic feedback. Recently, visual prostheses are starting to be implanted in some cases of blindness. During the realization of this thesis, we studied both problems, since they are connected and equally relevant: having one of them functional in a real system would be not truly useful without the other.

The demand for assistive technologies for the visually impaired is not a new issue, but it has been increasing in the last decades. Many systems addressing this problem have already been proposed in the literature, and even some have been commercialized in the last few years. Before explaining our ideas, we think it is important to provide an insight of the most influential works for the thesis here presented. Thus, the next section briefly reviews the state of the art on this subject. Then we will be ready to motivate and describe the main aspects of our system.

### 1.2.1. Assistive systems From the literature

Probably the earliest technological aids developed for the blind were sonar-based. Sonar sensing, while being an old technology, can be helpful to detect the presence of obstacles in front of the user. Two classic examples, coming from the University of Michigan, are the Navbelt [Shoval et al., 1998] and the GuideCane [Borenstein and Ulrich, 1997]. They exemplify, respectively, two common configurations of assistive devices for the visually impaired: a wearable system, and an enhanced version of the white cane. The Navbelt uses an array of ultrasonic sensors arranged radially in a belt, so that they can detect obstacles in the surroundings of the subject without having to actively point at any particular direction (Figure 1.1a). The navigation suggestions were communicated to the user via stereo audio interface, with beeps and continuous sounds. On the other hand, the GuideCane attempts to enhance a regular white cane by adding an array of ultrasonic sensors covering about $120^{\circ}$, and by using wheels to roll on the floor, with a motor that steers the system avoiding obstacles (Figure 1.1b). As the authors themselves analyze in [Shoval et al., 2000], the GuideCane is much more intuitive to use as it directs


Figure 1.1. Three examples of sonar-based assistive devices for the visually impaired. (a) NavBelt [Shoval et al., 1998]. (b) GuideCane [Borenstein and Ulrich, 1997]. (c) A commercial system called Ultracane.
the user as it moves before him, like a guide dog. Unlike the Navbelt, it is possible to achieve normal walking speeds without needing any training. Besides, it does not use audio signals, which block audio cues from the environment that are very important for blind people. However, the information these systems provide is very limited: basically two dimensional occupancy mapping, with poor angular resolution and unable to detect more complex features and objects. More modern approaches such as the already commercialized UltraCane ${ }^{1}$ solve some of these limitations, also attempting to enhance the white cane. In particular, besides a downward facing ultrasonic sensor that detects usual obstacles, it additionally incorporates another one facing upwards to prevent obstacle collision at chest/head height (Figure 1.1c). The feedback to the user is provided via a tactile interface, felt by the user when holding the cane. While undoubtedly a good enhancement to a regular white cane, it is still unable to perceive and communicate more meaningful information than the mere presence of obstacles.

Another type of sensors used in assistive systems is the laser scanner. Compared to the sonar, they are able to get much more rich and accurate information from the environment; but they are often expensive, heavy and involve high power requirements. One example of wearable system to aid blind people is shown in [Ueda et al., 2006], where the main sensor is a 3D laser scanner chest-mounted, used to detect obstacles and then warn the user via audio interface. Laser scanning is also used in [Capi and Toda, 2011], where they propose a robotic system consisting on one PC, two laser range finder, a camera, a microprocessor and a joystick potentiometer, all of it mounted in a trolley

[^0]walker.
Some other common sensors used in this context are RFID [Kulyukin et al., 2004] and GPS. The latter provides very useful navigation cues whenever you are outdoors and want reach another destination in the city. It is completely implemented in devices such as smartphones, that most people already possess; and with the help of modern text-to-speech (TTS) and speech-to-text (STT) capabilities, and the accurate maps that most widely used services provide (e.g. Google Maps), we could say that this technology already provides useful navigation cues rather comfortably. Nonetheless, this technology only works outdoors, and it does not solve other problems that users may come across during outdoor navigation, such as traffic, curbs and collision with people and other obstacles. On the other hand, in order to take advantage of the RFID technology, it is necessary to have a previously prepared environment, which is something we cannot normally assume. For example, [Faria et al., 2010] tested a system with RFID tags along the path and in specific points of interest. The RFID reader is in the white cane, at the lower part, to read the tags placed in the ground; and the transmission of information is via a combination of TTS and Bluetooth headphones and haptic feedback. Alternatively, [Ahmetovic et al., 2016] suggest a turn-by-turn navigational assistant that uses a smartphone, but instead of RFID, GPS or Wi-Fi they use as landmarks a set of pre-installed Bluetooth low energy beacons to localize the user. Receiving this kind of information is useful and reliable, but the lack of infrastructures to set out a system like this, is a very strong drawback.

Recently, the most promising assistive systems use cameras as main type of sensors. There are many reasons to adopt cameras as centerpiece for this kind of problem. First, because of their functionality. The amount of tasks that can be achieved with computer vision is unbeaten: you can go beyond obstacle detection and use it for recognizing objects, people, traffic signs, and so on. Being multifaceted is a major advantage, since it makes possible to perform most of the tasks with only one sensor. The extensive research in this field in this and other related topics such as robotics or autonomous driving has paved the way for this type of sensors. Besides, the costs of using cameras are minimal: they are already available, portable and cheap. Every year digital cameras are able to produce better image quality with smaller size of sensors at cheaper price, as the extensive usage of cameras in portable devices such as smartphones is currently putting in evidence. The apparition of the first smart-glasses also shows that it is possible to miniaturize the sensors and use them in an unobtrusive wearable system. The price point is also extremely relevant: according to [Bourne et al., 2017], the $89 \%$ of the world's visually impaired live in low- and middle-income countries. An example of how monocular vision can be helpful to detect specific features, such as text signs, doors, elevators or cabinets is shown in [Tian et al., 2013] (Figure 1.2a). There are also commercial systems, like OrCam ${ }^{2}$ : a small device that can be attached to any glasses, and that possess a camera able to recognize text, faces, colors, objects and bank notes. The information is

[^1]

Figure 1.2. (a) Example of door detection and text information extraction with conventional cameras from [Tian et al., 2013]. (b) A woman using OrCam, a commercial visually impaired aid attached to the glasses that using a camera performs tasks such as reading text aloud.
transferred to the user by saying it aloud. To avoid overwhelming the user, it only says things aloud when it points to where they want to have the text recognized (Figure 1.2b). In [Yoshida et al., 2011], edges extracted in the image are codified in sounds so that a trained user can interpret them and recognize shapes.

Unfortunately, there are some limitations to take into account when considering cameras. While they are able to capture similar kind of information to what the human eyes can do, it is still impossible to reach the performance and level of abstraction of the brain to interpret such input. The images that can be acquired during the normal use are highly variable and it is not an easy task to develop an algorithm able to understand the scenes everywhere, with other adjacent problems such as occlusions or changeable lightning conditions. In addition, cameras may have physical problems such as being out of focus to the important part of the image, motion blur due to the movement of the person, or simply capturing poorly framed images that leave the relevant features out of sight. Besides, sometimes computer vision algorithms are highly time consuming, but for this task they must execute in real time in order to be useful. Some of these issues can be overcome with what we call unconventional cameras, i.e. camera systems more complex than a regular monocular camera that may include other artifacts (e.g. lenses, projectors, mirrors, other cameras) that allow them to retrieve more information, or information of different nature.

One of the biggest challenges of using cameras in navigation is the geometric reconstruction of the environment, which is often solved using SLAM algorithms or camera systems that are able to retrieve depth information from the scene. These camera systems are called range cameras, i.e. camera systems able to capture 2D images in which each pixel represents the distance to a point in the scene, called range images or depth images. For example, a system that has two cameras calibrated and a shared field of view


Figure 1.3. Examples of stereo-vision used for assistance for the visually impaired.
is called a stereo camera. With stereo vision it is possible to retrieve 3D information of the scene, similar to the depth perception of the human pair of eyes. This perception can easily be used to improve the detection of obstacles or specific shapes, being a clear advantage over monocular cameras. In 1998, Molton et al. proposed a wearable system composed mainly by a stereo camera to help blind people avoid obstacles [Molton et al., 1998]. The electronics and computing devices as well as the sensors are mounted in a backpack, with the two cameras placed over the shoulders (Figure 1.3a). A more modern approach using a head-mounted stereo vision system is the one shown in [Pradeep et al., 2010]. It joins in the same mobility system a SLAM algorithm along with obstacle detection. With the visual odometry and the mapping they perform a traversability analysis of the environment and a tactile interface situated on a vest steers the subjects away from obstacles along the path. Rodriguez et al. in [Rodriguez et al., 2012] proposed another obstacle avoidance system using stereo cameras, with acoustic feedback instead of vibration interface (Figure 1.3b).

Nevertheless, stereo vision has some problems capturing depth in texture-less or repeated areas, which could make them not completely reliable in some relatively simple situations such as looking at a plain wall or to the floor. To overcome these limitations, there are other types of range cameras, such as the Time-Of-Flight (TOF) or the Structured-Light cameras (SL). Both types of range cameras work with laser projection to the scene (with usually infrared light) and are able to infer the depth either by emitting light pulses and measuring the per-pixel delay after reflecting in the scene (TOF), or by projecting a light pattern to the scene and measuring its distortion (SL). Since such sen-


Figure 1.4. Examples of visual aids from the literature with TOF cameras.
sors emit its own light, the problem of texture-less and repeated areas is eliminated, and additionally work well in low light and even complete darkness. However, the sun also emits infrared light, causing interferences that prevents those sensors to extract reliable information at the light of day. Therefore, its use is mostly restricted to indoor environments. Time-Of-Flight is used in this context in works such as [Bostelman et al., 2006] and [Lee et al., 2012], with main application for object detection (Figure 1.4a). In [Qian and Ye, 2013] the TOF sensor is attached to an otherwise plain white cane, and it is used to extract 3D planes and recognize staircases (Figure 1.4b).

Since 2010, the introduction in the consumer market of the Microsoft Kinect (Figure 1.5a), along with its developer counterpart the Asus Xtion Pro (Figure 1.5b), made a huge impact in the field. The Kinect is a system that combines structured-light range sensing with a synchronized conventional camera, allowing to benefit from the strengths of both worlds and overcome some drawbacks. For example, some objects of certain surface materials or with thin or complex shapes often do not return good depth measurements. Nevertheless, in some tasks the color image can be helpful to overcome that limitation. We studied this problem in [Perez-Yus et al., 2018c], where we demonstrate how objects that return poor depth data can be tracked with RGB camera and the depth information of the surroundings can be used to improve the performance. These types of systems are commonly referred to as RGB-D cameras (i.e. color, RGB, and depth, D). The Kinect was sold as a video game device, used to track your body and movements, and thus its price was not very high (around \$150). For that price, having depth images


Figure 1.5. RGB-D cameras widely used in the literature due to their affordable price.
of good resolution $(640 \times 480)$ was groundbreaking, considering how fast and reliable it was to obtain reconstructions of the scene with this system. Quickly, developers started working on applications with such sensors. That is the case of [Zöllner et al., 2011], which proposed a head-mounted configuration or [Bernabei et al., 2011], which uses a waist-mounted configuration along with an accelerometer to detect the movement of the user, floor and obstacles. [Mann et al., 2011] introduces the use of a vibro-tactile helmet to communicate with the user besides the head-mounted RGB-D sensor. Also Lee et al. in [Lee and Medioni, 2011] improved what was published for Pradeep et al. in [Pradeep et al., 2010] by replacing the stereo camera with an RGB-D camera. Aladren et al. [Aladren et al., 2016] improved the way-finding by fusing data from the depth and RGB camera; the latter being used to extend the information recovered by the depth planar segmentation. In this case, the presence of obstacles or walls is communicated to the user with audio signals. More modern approaches integrate machine learning techniques for segmentation of the scene with RGB-D, like [Wang et al., 2014b]. Newer RGB-D devices have been appearing with enhanced features. For example, the Kinect v2 has a TOF camera that provides much higher resolution. The Google Tango project showed the miniaturization possibilities of these systems, by adding them to smartphones and tablets. In [Li et al., 2016], a Tango device is used for localization inside a semantic map of the scene and obstacle avoidance. In 2017, the iPhone X included a depth front-facing camera to recognize the face and unlock the phone automatically.

Throughout this section, we have cited many systems from the literature that feature a communication interface to transmit the information to the user, mainly via audio or tactile signals. Nevertheless, in some cases it is possible to communicate with the user using the visual system with prosthetic vision. Visual prostheses consist of retinal or cortical implants that apply electrical stimulation of the visual cortex or other parts of the visual pathway (such as retina) and cause patients to perceive bright dots of light called phosphenes [Brindley and Lewin, 1968]. Using an array of electrodes it is possible to generate a grid of phosphenes that resemble a low resolution image made of dots. Typically, these systems are paired with a camera that acquire images and a portable


Figure 1.6. Argus II Retinal Prosthesis from Second Sight. (a) Photo of the implant and representation of its main components. (b) External equipment, including glasses with the camera, the antennas to communicate wirelessly to the implant, and the Video Processing Unit (VPU). (c) The images captured by the camera are transformed into an electronic coded signal that activates the electrode array provoking the visualization of dots of light called phosphenes (right).
computer that converts the image data into an electronic coded signal that is transferred to the electrode array via wireless interface. Experimental results demonstrate that patients with this kind of devices can detect phosphenes at individual electrodes and they were able to develop coordination in using their visual prosthetic device [Ahuja et al., 2011]. Currently there are systems commercially available, such as the Argus II Retinal Prostheses System, from Second Sight ${ }^{3}$ (Figure 1.6).

### 1.2.2. PROPOSED FRAMEWORK

After gathering information of what has been done in the past about assistance for the visually impaired, we decided a few features of how a system developed by us should be.

[^2]

Figure 1.7. Different sensor placements we have used in this thesis: (a) Head-mounted (b) Chest-mounted

The main sensor of our system is an RGB-D camera. Having 3D perception can help recognizing the basic structures of the environment as well as detecting obstacles along the path, enabling a more robust and secure navigation for the user. The localization of the user can be tackled with SLAM techniques that use these sensors. The main drawback of RGB-D cameras is the poor performance outdoors. In this thesis, our experiments focus on indoor applications, though the approaches would still be valid outdoors if the sensor is substituted for a similar one able to work in sunlight.

The usage of additional sensors is convenient and advisable. RGB-D cameras have limitations, like the limited field of view. For example, adding omnidirectional cameras to the system allows to capture the whole scene around, so the user does not need to be accurately pointing to a precise direction. In this thesis, we have calibrated an omnidirectional camera with an RGB-D camera in a hybrid system. Other sensors like Inertial Measurement Units (IMU) have proven to help in SLAM techniques, and although they were not utilized in this thesis, they could be a useful enhancement for the system.

We decided our system should be wearable, meaning all the sensors, hardware and communication parts should be worn by the user. This allows the system to work autonomously without depending on external infrastructures, such as RFID tags. Since our idea is to complement and not replace, this leaves the usage of additional aids such as the white cane to the preference of the user.

An important consideration is the location of the camera worn by the user. We consider an egocentric point of view, or first person view, where the camera points to the scene emulating the visual field of view that a person typically has. Specifically, in this work we have considered two possibilities: chest-mounted and head-mounted (Figure 1.7).

Wearing the camera in the chest has more stability and is always pointing at the region in front of the user, causing safer navigation in terms of detecting obstacles. However, the field of view of wearing the camera on the head is larger, less likely to be obstructed, and the user can stand and scan the environment more comfortably and intuitively than having to move the whole body. Mayol-Cuevas et al. in [Mayol-Cuevas et al., 2009] provided a more extended analysis on the matter, analyzing social aspects as well. In particular, it claims a chest-mounted configuration is likely the most social acceptable one as it interferes the least in social interaction. Nevertheless, the perspectives of miniaturization of RGB-D cameras in the near future may made glass-mounted cameras the less intrusive ones. Without a clear winner, our decision was to make the methods to develop able to work independently of location of the camera during normal usage, with the only possible exception of initialization of the algorithm.

About the communication user interface, our group -as many others- has used audio responses in previous works [Aladren et al., 2016]. In this thesis, we decided to focus on the problem of visual prostheses. While the information provided by these systems is still limited (low resolution and dynamic range), our efforts were destined to design iconic representations that may provide relevant navigation cues despite these limitations. In this case, the work was performed by simulating prosthetic vision, due to the impossibility of carrying out experiments with real patients.

### 1.3. GoALS AND CONTRIBUTIONS

Once we have asserted our motivation and the framework of our thesis, it is time to define the specific goals to pursue, and the extent of the work developed in these four years in those lines of research. In particular, the thesis can be divided in four main lines of work. The first three are related to the perception of the environment, meaning the extraction of relevant information that the user might need to receive. We focus on extracting basic information that allows the user to move safely (e.g. avoiding obstacles), but also providing additional contextual information that enables to perform higher level tasks, such as presence and location of staircases or recognition of the shape of the room layouts. For some of these tasks we also propose the usage of a novel hybrid device that provides depth perception and wide field of view images at the same time. The last line of work is about conveying the perceived information to users, in this case, focusing on blind people that use visual prostheses.

The next sections detail each specific line of work followed, pointing out the contributions and associated publications that came out as a result of our work.

### 1.3.1. Stairs Detection, modeling and traversal

The users need to be able to move safely in any environment, whether it is known or unknown. This includes the detection for avoidance of obstacles and other hazards that may be dangerous for a safe navigation. While a white cane does a good job at detecting obstacles in short range, our goal attempts to address mid and large range obstacle detection. RGB-D cameras and computer vision algorithms may be used to that end. However, when users are moving in an unknown environment, obstacle avoidance is not their only concern: they would want to acquire knowledge of the presence of certain structures that allow them to move efficiently from one place to another. Independently of the environment, the most widespread structures that any person may come across in its daily life are stairs and curbs. Detecting them is crucial for both safety reasons and for communicating them the possibility of moving up or down the building.

Thus, our first goal was to create a perception system that allows the user to move in any environment safely and efficiently, by including the detection of stairs in a more general obstacle detection framework.

## MAIN CONTRIBUTIONS:

- We have developed a stair detection algorithm from point clouds obtained from an RGB-D camera that surpassed the state of the art.
- Besides the detection, our algorithm models the stair dimensions and retrieves the position and orientation with respect to the user.
- Additionally, by running a visual odometry algorithm in parallel, we have also developed a stair traversal algorithm that allows to model the full staircase and retrieve the step where the user stands.
- Reciprocally, during traversal the current view is used to correct the drift of the visual odometry.


## Associated publications:

[1] Perez-Yus, A., Gutierrez-Gomez, D., Lopez-Nicolas, G., and Guerrero, J. J. (2017b). Stairs detection with odometry-aided traversal from a wearable RGB-D camera. Computer Vision and Image Understanding, 154:192-205
[2] Perez-Yus, A., Lopez-Nicolas, G., and Guerrero, J. J. (2015). Detection and modelling of staircases using a wearable depth sensor. ECCV 2014 Workshops, Part III, LNCS 8927(3):449-463
[3] Guerrero, J. J., Perez-Yus, A., Gutierrez-Gomez, D., Rituerto, A., and LopezNicolas, G. (2015). Human navigation assistance with a RGB-D sensor. In VI Congreso Internacional de Diseño, Redes de Investigación y Tecnología para todos (DRT4ALL), pages 285-312

## InTERNATIONAL COMMUNICATIONS:

- Oral presentation at the 2nd Assistive Computer Vision and Robotics workshop (ACVR), in conjunction with European Conference on Computer Vision (ECCV), 2014, Zurich (Switzerland). Related publication: [2]
- Poster presentation at the International Computer Vision Summer School (ICVSS) 2015, Sicily (Italy). Title: Stair detection and modelling from a wearable depth camera.


### 1.3.2. UNCONVENTIONAL CAMERA SYSTEMS AND CALIBRATION

One of our main concerns was about choosing the means we use to perceive in our system. We manifested our primary interest in RGB-D cameras as main sensor, because of the numerous advantages they have. However, these cameras have some limitations that could be overcome with some additional sensors. We wanted to explore the possibility of extending the field of view that most of these cameras usually have by using unconventional camera systems. In particular, we devised a novel hybrid camera system composed by a depth and a fisheye camera. To use such system it is required to perform calibration, and there were no existing methods suitable for this device. Therefore, it was necessary to develop new calibration methods that allowed to calibrate that system.

Our second goal was to explore new ways to enhance RGB-D cameras, particularly focusing on the expansion on the field of view. For this, it is necessary to propose novel unconventional camera systems and methods to calibrate them easily and with as fewer constraints as possible.

## Main contributions:

- We have proposed a novel hybrid camera system, composed by a depth and a fisheye camera, that attempts to combine the advantages of both systems (3D perception and wide field of view).
- A calibration procedure to calibrate such system has been developed, considering the particularities of the problem.
- To provide a more general solution to the calibration problem, another method based on line correspondences was proposed, able to calibrate multiple configurations of RGB-D based systems, without needing to build a calibration device or to have overlapping field of view among the cameras.
- This newer method was successfully tested with the aforementioned device and others, including a rig of 8 RGB-D cameras arranged radially to provide omnidirectional field of view.
[4] Perez-Yus, A., Fernandez-Moral, E., Lopez-Nicolas, G., Guerrero, J. J., and Rives, P. (2018a). Extrinsic calibration of multiple RGB-D cameras from line observations. IEEE Robotics and Automation Letters, 3(1):273-280
[5] Perez-Yus, A., Lopez-Nicolas, G., and Guerrero, J. J. (2016a). A novel hybrid camera system with depth and fisheye cameras. In IAPR International Conference on Pattern Recognition (ICPR), pages 2789-2794

INTERNATIONAL COMMUNICATIONS:

- Oral presentation at IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017, Vancouver (Canada). Related publication: [4]
- Poster presentation at ICPR, 2016, Cancun (Mexico). Related publication: [5]


### 1.3.3. ScALED LAYOUT FROM WIDE FIELD OF VIEW RGB-D CAMERA

With depth cameras it is relatively simple to detect obstacles, if obstacles are simply considered as groups of 3D points that could obstruct the way of a walking person. However, in this work we aim to enhance the navigation experience of people with vision problems, and one way of doing this is providing context. One of the most basic information that we can extract to contextualize a scene is the spatial layout of the room the user is in. That information is useful and essential to enable or facilitate many tasks including navigation, localization, object detection or scene recognition. While layout estimation is not a new issue, existing methods usually have major limitations. For example, they are usually up to scale if they are recovered on single image, and the field of view is often too small, so the recovered layout may be wrong because of lack of spatial information, or may not produce relevant information for higher level operations. One of the keys to tackle these limitations is the usage of unconventional cameras, such as our novel device with fisheye and depth camera, designed to have wider field of view while providing synchronized 3D information.

Our third goal was, therefore, to develop a method to estimate the layout of the scene combining information coming from a fisheye and a depth camera, and thus, obtaining scaled 3D reconstructions of a large extent of the room, benefiting from the wide field of view of the fisheye and the scale and range information from the depth camera.

## MAIN CONTRIBUTIONS:

- We have proposed a method to recover the full-scaled 3D layout of the scene in one single shot.
- It corresponds with the first application of the novel hybrid camera we introduced, with a fisheye and a depth camera. From the fisheye we are able to extract lines that are used to find relevant corners and generate layout hypotheses. The depth camera enhances the process and incorporates scale to the final reconstruction.
- As a result, our returned 3D reconstruction works as an extension of the 3D data captured by the depth sensor to over 180 degrees of field of view.


## Associated publications:

[6] Perez-Yus, A., Lopez-Nicolas, G., and Guerrero, J. J. (2018b). Scaled layout recovery with wide field of view RGB-D. Submitted to a journal
[7] Perez-Yus, A., Lopez-Nicolas, G., and Guerrero, J. J. (2016b). Peripheral expansion of depth information via layout estimation with fisheye camera. In European Conference on Computer Vision (ECCV), pages 396-412. Springer

## International communications:

- Poster presentation at ECCV 2016, Amsterdam (Netherlands). Related publication: [7]
- Oral presentation at the International workshop on Lines, Planes and Manhattan Models for 3-D Mapping (LPM) in conjunction with IROS 2017, Vancouver (Canada). Title: Wide RGB-D for Scaled Layout Reconstruction.


### 1.3.4. ICONIC PHOSPHENIC REPRESENTATION FOR HUMAN NAVIGATION WITH VISUAL PROSTHESES

Besides perception tasks, in this thesis we aim to work on the communication problem taking advantage of new advances in visual prostheses. Recent research demonstrates that visual prostheses are able to provide visual perception to people with some kind of blindness. In visual prostheses, image information from the scene is transformed to a phosphene pattern to be sent to the implant. This is a complex problem where the main challenge is the very limited spatial and intensity resolution. Moreover, depth perception, which is relevant to perform agile navigation, is lost, and codifying the semantic information to phosphene patterns remains an open problem. One way to tackle these limitations is to design an iconic representation of the environment that could be displayed with phoshpene patterns. That way, even with the low resolution, useful information could be transferred to the users.

Hence, our fourth goal was to design and implement a new iconic representation of the environment, particularly dealing with navigation purposes, where the information to transfer via phosphene patterns would be useful to move in indoor environments avoiding obstacles. Given the difficulties of working with real people with visual prostheses, at this point our goal stayed confined to simulated prosthetic vision.

## Main contributions:

- We have developed an RGB-D based method that is able to perceive the free space in front of the user and provide the walkable floor area.
- We have designed an iconic representation with phosphene patterns that realistically simulates the information provided with visual prostheses. This iconic representation consists in showing the walkable polygon of the floor with a chess pattern that provides sense of depth and movement and therefore, makes the navigation more comfortable and informative, while additionally indicating the presence of obstacles.
- Experiments have been performed in simulation environments and with real data from indoor environments obtained with a head-mounted RGB-D camera.


## ASSOCIATED PUBLICATIONS:

[8] Perez-Yus, A., Bermudez-Cameo, J., Lopez-Nicolas, G., and Guerrero, J. J. (2017a). Depth and motion cues with phosphene patterns for prosthetic vision. In IEEE International Conference on Computer Vision Workshops (ICCVW), pages 15161525

## International communications:

- Poster presentation at the 5th International Workshop on Assistive Computer Vision and Robotics (ACVR), in conjunction with IEEE International Conference on Computer Vision (ICCV), 2017, Venice (Italy). Related publication: [8]


### 1.3.5. COMPLEMENTARY CONTRIBUTIONS

During this thesis, two research stays have been carried out. In addition, some Master and Bachelor thesis have been supervised.

### 1.3.5.1. Research stay at University of Washington, Seattle, USA

- Supervisor: Dieter Fox
- Dates: Sept. 17 - Dec. 15, 2015 (90 days)
- Summary: During this stay, the core element of the research was RGB-D cameras, matching the framework of the thesis. However, in this case, we focused on studying and improving the limits of perception of RGB-D systems when facing difficult or complex objects and environments. Particularly, there are certain objects whose surface properties or complex shapes prevents the depth sensor from
returning good depth measurements, and only color-based methods can be applied. We show how the depth information of the surroundings of the object can still be useful in the object pose tracking with RGB-D even in this situation. Specifically, we propose using the depth information to handle occlusions in a state of the art region-based object pose tracking algorithm. Experiments with recordings of humans naturally interacting with difficult objects have been performed, showing the advantages of our contribution in several image sequences.
- Results: Publication of a paper presented in a workshop:
[9] Perez-Yus, A., Puig, L., Lopez-Nicolas, G., Guerrero, J. J., and Fox, D. (2018c). RGB-D based tracking of complex objects. Understanding Human Activities through 3D Sensors Workshop (UHA3DS), in conjunction with ICPR 2016, LNCS 10188


### 1.3.5.2. Research stay at INRIA Sophia-Antipolis, France

- Supervisor: Patrick Rives
- Dates: Sept. 1 - Nov. 29, 2016 (90 days)
- Summary: During this stay, the line of work was focused on the calibration of unconventional camera systems, particularly developing the extrinsic calibration method based on line observations. In these three months I could benefit from the expertise of Patrick Rives and Eduardo Fernandez-Moral, and also make use of the multi-camera sensors they developed in their lab.
- Results: The journal publication [4], also presented in an international robotics conference (IROS 2017).


### 1.3.5.3. Supervision of Bachelor and Master thesis

- Student: Clara Fernandez Labrador

In this Master thesis we have developed a method for 3D layout recovery of indoor scenes from a single 360 degrees panoramic image. This method has the main novelty of combining geometric reasoning on computer vision and deep learning techniques adapted to the proposed image geometry. Our method uses the extraction of structural corners as a starting point to construct layout hypotheses assuming Manhattan World and without any prior information about the room shape. In particular, corners are extracted as intersections of lines that are orthogonal in 3D space. This process has been enhanced with a Convolutional Neural Network that detects structural edges and allows filtering lines belonging to other non-relevant objects. From these possible corners we draw layout hypotheses and choose the best fitting solution to the normals' map extracted with another CNN.

We show results of 3D layouts recovered from images of the SUN360 public dataset. We demonstrate the effectiveness of our method with respect to existing works and the advantages of the introduction of deep neural networks in the pipeline of the process. As a result of this work, a research paper was written and submitted to a journal.
[10] Fernandez-Labrador, C., Perez-Yus, A., Lopez-Nicolas, G., and Guerrero, J. J. (2018). Layouts from panoramic images with geometry and deep learning. Submitted to a journal
[a] Fernández Labrador, C. (2017). Estimation of the 3D layout of indoor environments from panoramic images. Master thesis, Universidad de Zaragoza. Supervised by J.J. Guerrero and A. Perez-Yus

- Student: Enrique Otero Moliner

In this bachelor thesis, we developed a framework for evaluation of iconic phosphene representations in simulated prosthetic vision. In particular, we created several realistic virtual indoor environments in which a simulated robot with a RGB-D camera can be externally controlled by a subject. The subject must navigate trying to avoid collisions only viewing an iconic representation with phosphene arrays similar to what a real user of prosthetic vision would see. Several students collaborated in the experiments and we collected the data and provide qualitative and quantitative analysis of the validity of the representation approach and the overall performance with different parameters of representation.
[b] Otero Moliner, E. (2017). Development of simulated prosthetic vision in virtual environments. Bachelor thesis, Universidad de Zaragoza. Supervised by G. Lopez-Nicolas and A. Perez-Yus

- Student: Daniel Cantón Toro

In this bachelor thesis, we developed a new and improved framework for evaluation of iconic phosphene representations in simulated prosthetic vision. In particular, with respect to previous work, the main difference here is that we consider more realistic movements on simulation, including movements of the head, which translates in more complexity on the segmentation of the scene and thus the system overall.
[c] Cantón Toro, D. (Expected Sept. 2018). Virtual reality for simulated prosthetic vision with phosphene patterns. Bachelor thesis, Universidad de Zaragoza. Supervised by G. Lopez-Nicolas and A. Perez-Yus

### 1.4. Outline

From here on, the following chapters describe, respectively, each of the four main blocks of the thesis, with a final conclusions chapter at the end. Each of the four main chapters have a brief introduction to the problem and the state of the art related to the matter. Then they continue with the extended explanation of the method to end with some experiments and final discussion. In more detail, the rest of the document is organized as follows:

- In Chapter 2, we describe our stair detection, modeling and traversal algorithm. The chapter includes specific related work about the matter and experiments.
- In Chapter 3, we present our new hybrid camera system with depth and fisheye camera and two calibration methods: one specific to that camera system, and another more general to calibrate this and other unconventional camera systems that include RGB-D. Related work about extrinsic calibration is provided, along with experiments of both methods presented.
- In Chapter 4 we introduce our scaled layout recovery algorithm that uses the hybrid camera system to obtain complete 3D reconstructions of rooms of variable shapes.
- In Chapter 5, we show our approach of iconic representation with phosphene patterns, to enable human navigation to people with visual prostheses. Experiments in simulation and with real images have been performed in a simulated prosthetic vision framework.
- Finally, in Chapter 6 we conclude this thesis, with some final remarks and ideas for future work.


## 2

## STAIRS DETECTION, MODELING AND TRAVERSAL

Stairs are one of the most common structures present in human-made scenarios, and also one of the most dangerous for those with vision problems. In this chapter we propose a complete method to detect, localize and parametrize stairs with a wearable $R G B-D$ camera. Our proposal uses the depth data to determine if the horizontal planes in the scene are valid steps of a staircase judging their dimensions and relative positions. As a result, we obtain a scaled model of the staircase with the spatial location and orientation with respect to the subject. The visual odometry is also estimated to continuously recover the current position and orientation of the user while moving. This enhances the system giving the ability to come back to previously detected features and providing location awareness of the user during the climb. Simultaneously, the detection of the staircase during the traversal is used to correct the drift of the visual odometry. A comparison of results of the stair detection with other state-of-the-art algorithms was performed using public dataset. Additional experiments have also been carried out, recording our own natural scenes with a chest-mounted RGB-D camera in indoor scenarios. The algorithm is robust enough to work in real-time and even under partial occlusions of the stair.


### 2.1. InTRODUCTION

We have defined one of our main goals as the development of a perception system that allows the user to move safely and efficiently in any environment. Probably the first task that comes to mind within this goal is obstacle avoidance: forewarning the user to avoid collisions, with sufficient time in advance to make the navigation efficient. Modern RGB-D cameras are extremely helpful in this task providing 3D information at mid-range, and thus there are many works addressing this problem with this type of sensor in the literature [Zöllner et al., 2011, Bernabei et al., 2011, Mann et al., 2011, Lee and Medioni, 2011, Pradeep et al., 2010]. In this chapter, we go beyond obstacle detection and focus on the also immensely relevant task of stair detection. Stairs are basic structures in human-made constructions that allow humans to displace vertically between different floors. They are omnipresent inside buildings and besides they exist in many outdoor man-made environments worldwide. However, they are a potential source of serious accidents. RGB-D cameras can assist enormously, as stairs usually have a similar and distinctive shape which can be perceived by these instruments: a sequence of steps consisting in shifted parallel planes, occasionally with perpendicular risers joining one another. The case of single steps, or curbs, must be also considered, since it is also a potential dangerous structure.

Here, we present a method which takes advantage of RGB-D cameras in the tasks of stair detection, modeling and traversal. Each task is defined as follows:

- Detection: to notice the presence of stairs, avoiding false positives and negatives. We consider the possibility of detecting both ascending and descending staircases, of any number of steps.
- Modeling: for any positive detection, to provide the dimensions of the steps, number of them in sight and position and orientation of the staircase with respect to the user. This task is useful for navigation and validation of the stair detection.
- Traversal: to be able to follow the user while going up or down the staircase, obtaining real-times updates of the position of the user during the way (e.g. number of step on which stands), completing the full model of the stair, and noticing the end of the stair.

In our current framework, we use a wearable RGB-D camera with the computations performed in a computer (Figure 2.1a). Consequently, given the limitations of RGB-D cameras in sunlight, we restrain our experiments to indoor environments. Without loss of generality, we have used a chest-mounted camera placement to record the sequences, with the camera slightly looking downwards (e.g. about $45^{\circ}$ ) to ensure not missing the staircases. Given the $45^{\circ}$ vertical field of view of the RGB-D camera used, should be enough to locate the obstacle-free path in front of the subject and detect stairs. Additionally, we have included the estimation of the visual odometry [Gutierrez-Gomez et al., 2015] during the navigation with the RGB-D camera. This allows us to maintain loca-


Figure 2.1. (a) Photo of the current wearable camera system. The camera is placed on the chest, pointing downwards. The computations are performed in a laptop carried in a backpack. (b) Block diagram of one iteration of the main loop of the proposed method.
tion awareness and to know the relative position to relevant features in the scene even when they are not in the current view. The visual odometry is also used to accomplish the traversal task. On the other hand, visual odometry methods have drift that keeps accumulating through time. Our stair traversal method has the double duty of correcting such drift by using online stair observations. In this work, we also make use of the Manhattan World assumption [Coughlan and Yuille, 1999], according to which the world is organized following three orthogonal main directions. Thus, we retrieve those Manhattan directions to make a faster and simpler stair modeling, considering staircases most likely follow that convention in indoor man-made environments.

A diagram summarizing how the system works is shown in Figure 2.1b. The sensor provides RGB and Depth images which are used to compute a 3D point cloud [Rusu and Cousins, 2011] and to feed the visual odometry estimation. With the 3D point clouds we perform a planar segmentation of the scene, compute the relative transformations of the user to the environment and obtain a classification of the planes according to their orientation. Unidentified vertical planes and clusters of points are classified as obstacles to be avoided. The horizontal planes classified as step candidates are run through the stair detection and modeling algorithm. When the system detects that the user has reached the proximity of the staircase the system proceeds with the stair traversal algorithm, able to recover the position of the user along the stairway using the user pose estimated with
the visual odometry module. Simultaneously, the live information from the camera is used to correct the visual odometry drift. Experiments to test our system's detection ratio, quality of the model, temporal performance and drift correction during traversal have been carried out and included at the end of this chapter.

### 2.2. RELATED WORK

Different approaches with different types of sensors have been used for detecting stairs. For instance, conventional cameras were used by Se and Brady [Se and Brady, 2000]. They use grayscale images to detect and estimate the orientation and slope of staircases in order to help partially sighted people by using traditional computer vision: Gabor filtering, vanishing point detection and homographies (Figure 2.2a). In the same vein, Hernandez et al. [Hernandez and Jo, 2010] propose to find the diagonal lines corresponding to the handrails to select stair candidates (Figure 2.2b), which seems hardly applicable in scenarios where staircases do not have handrails or those are hard to detect. Hesch et al. [Hesch et al., 2010] focus on detecting descending staircases for small ground robots in both far and near distance, being the latter the most interesting. It uses line detection combined with optical flow computation in order to detect the edges of the first step-downs, which is very useful since it is by far the most dangerous situation. However, it is designed for autonomous tracked vehicles with small size and the camera situated close to the floor, which is a different problem than a wearable system for human navigation (Figure 2.2c). Wang [Wang and Wang, 2009] used Real AdaBoost for training a cascaded classifier (Figure 2.2d). In [Carbonara and Guaragnella, 2014], an algorithm to detect ascending stairs in frontal and lateral views is proposed and tested with images coming from a smartphone. In this case, they solve the detection recognizing the periodic structure of staircases analyzing the frequency spectrum.

Stairs are usually characterized by a distinctive shape formed by a flight of steps, which probably makes the sensors capable of measure depth the most appropriate for the task, as they provide three-dimensional information. That is the case of stereo cameras. Gutmann et al. proposed a stair detection algorithm for humanoid robots in [Gutmann et al., 2004] where the stereo vision system segments the scene into planar surfaces. Lu et al. [Lu and Manduchi, 2005] combine the use of the geometry information provided by a stereo system with the RGB data to make the system less prone to error. Pradeep et al. [Pradeep et al., 2008] use stereo vision to estimate normals and planes of the scene. However, they provide the basis for stair finding but not the recognition itself. Recently, new approaches address this problem with stereo vision producing outperforming results. That is the case of [Harms et al., 2015], which detects ascending staircases extracting concave and convex 3D edges of stairs (Figure 2.3a). Apart from detection, they incorporate modeling of the staircase and line tracking while traversing the stair as well. In [Schwarze and Zhong, 2015], another head-mounted stereo system is used to


Figure 2.2. State-of-the-art stair detection algorithms using conventional monocular cameras.
retrieve models of ascending stairs that keep growing as traversing (Figure 2.3b).
Most of the authors who use laser scanning for finding stairs focus on robot navigation [Albert et al., 2001, Mihankhah et al., 2009, Bansal et al., 2011, Park et al., 2011]. In [Qian and Ye, 2013] they use a laser sensor attached to a white cane. Plane extraction with a NCC-RANSAC procedure provides a good segmentation for stair detection. In [Ishiwata et al., 2013], a small laser range sensor attached to the chest of the subject is used to develop a visually impaired assistant. With this laser scanner, a segmentation of the scene is performed and the coordinates are classified into horizontal or vertical segments, after which the system judges whether there are steps or not. While similar to our proposal, in this case they only return positive detection of ascending and descending staircases, without further computation of the model. Recently, Stahlsmidth et al., proposed two methods for ascending [Stahlschmidt et al., 2015a] and descending [Stahlschmidt et al., 2015c] staircases, that use information coming from a


Figure 2.3. (a) Point cloud from the stereo system in [Harms et al., 2015], with highlighted convex (red) and concave (green) edges. (b) Model of the staircase while traversing from [Schwarze and Zhong, 2015]. (c) Graph-based stair detection (left) and an example of stair detection (right) from a 3D laser scan including railing (green points) from [Westfechtel et al., 2016]. (d) Stair modeling with laser data from humanoid robot [Oßwald et al., 2011b]. (e) During traversal of the humanoid robot, the edges from the camera view are used to refine the pose of the stair [Oßwald et al., 2011a].

TOF camera. In [Stahlschmidt et al., 2015b] the same author proposed an algorithm to extract the relevant parameters of the staircase. An interesting approach is shown in [Westfechtel et al., 2016], where a graph-based method for point cloud data allows to classify the points among treads, risers and railing system (Figure 2.3c). The railing system, while often overlooked, could be really useful in the field of assistive computer vision. The traversal of staircases has not been treated thoroughly regarding visually impaired aids with laser scanners, but there have been some approaches in humanoid robotics. Oßwald et al. in [Oßwald et al., 2011b] studied two planar segmentation approaches to model staircases using the laser range data acquired by tilting the head of a humanoid robot (Figure 2.3d). In [Oßwald et al., 2011a] they show how the model of the stair is used as input for the traversal of a spiral staircase combined with the laser data, the joint encoders, an IMU, and a camera pointing downwards to help the robot refine his pose for stair climbing. The model is projected to the camera images, and the pose of the robot with respect to the stair is refined with the edges extracted that should match the model (Figure 2.3e). In [Oßwald et al., 2012] they improve the method by additionally adding chamfer matching to refine the pose of the model of the staircase. In our work, only the RGB-D sensor is used to compute the model, the pose, and the close-range refinement.

The appearance in the consumer market of modern RGB-D cameras such as Microsoft Kinect or Asus Xtion Pro Live have become a new and powerful election to work on this topic. Some authors make use of these sensors applying machine learning algorithms to perform staircase detection. In the case of [Filipe et al., 2012], neural networks are used to detect the presence of obstacles and classify scenes captured by the depth camera among ascending staircase, descending staircase, or none. Wang and Tian used a similar approach, where the groups of parallel concurrent lines in the RGB image detected by the Hough transform are classified between stairs and pedestrian crosswalks using the depth information [Wang and Tian, 2012]. More recently, [Munoz et al., 2016] train an SVM for similar purposes.

Other authors preferred the usage of geometrical reasoning instead of machine learning to detect staircases with RGB-D. This is the approach we also consider to solve this problem, trying to overcome some of the shortcomings of these methods and to enhance the functionality of the system. For example, the method of [Tang et al., 2012] needs an accelerometer to help identifying the ground plane orientation, whereas our algorithm does it automatically with just the depth information. Their algorithm to find steps runs by looking for planes one by one at certain heights, which is prone to fail as it does not take into account any other shape constraint except having sufficient points. This may cause false positives and that the inliers of the planes correspondent to the steps include more points than the ones the step actually has (Figure 2.4a). Besides, they considered there is a staircase if there are three or more steps, ignoring the possibility of less than three steps, which is also quite common in doorways or other special constructions. Another similar approach that improves the results and some of these shortcomings


Figure 2.4. (a) Stair detection image from [Tang et al., 2012], where the points correspondent to each step extend beyond the actual step. (b) Model of the staircase from [Delmerico et al., 2013].
is shown in [Vlaminck et al., 2013], but it still has some important limitations. They find the floor as the biggest plane parallel to the floor plane, which might not be true if there are some other dominant planes, such as big table. Besides, they use a RANSAC algorithm every time they want to find any specific plane, but they might not even be on the scene, being highly time-consuming. But the main shortcoming of these two works is focusing on the detection and ignoring the modeling. With the modeling, the actual measurements of the steps can be obtained, information which can be used to verify the detection, to give indications to the user or to analyze the traversability of the staircase. Tang et al. provided a dataset of some staircases both ascending and descending and other common indoor scenes to detect false positives [Tang et al., 2012]. A comparison of our results with the methods from [Tang et al., 2012] and [Vlaminck et al., 2013] will be shown later in this chapter. Particularly, this comparison shows that the absence of the retrieval of the fully measured model of the staircase in [Tang et al., 2012, Vlaminck et al., 2013] leads to more false positives.

On the other hand, Delmerico et al. proposed an ascending stairway modeling that introduces some interesting ideas [Delmerico et al., 2013]. Their goal is to localize and model stairways to check for traversability and enable autonomous multi-floor exploration. The model of the staircase is an inclined plane inside a bounding box containing the stairway, and the measurements of the steps and the whole staircase (Figure 2.4b). In order to build up a complete model of the stairway they align the point clouds from different views relying on the robot's estimated pose, which is typically more complicated in human navigation. In addition, the stair edge detection, which is the starting point of their algorithm, is based on abrupt changes in depth that only appears in ascending staircases when the sensor is lower than the steps, i.e. a small robot. That collapses with


Figure 2.5. (a) Stair detection with wide FOV camera, showing directions to follow and audio cues [Romić et al., 2017]. (b) Stair detection with a radar scanner designed for a wheel chair from [Abdulatif et al., 2017]
our idea of a chest-mounted configuration. Moreover, the incapacity of the algorithm to detect descending stairs and their requirement of a minimum of three steps for detecting a stairway leaves a margin of improvement.

New works using RGB-D appeared in the last few years. For example, [Chan et al., 2017] proposes a method that combines 2D features (Hough lines extraction) embedded with 3D data. A new method uses an RGB-D sensor coupled to mobile devices [Ciobanu et al., 2017]. They initially extract patches parallel to the floor plane given the gravity direction from the IMU, and then use those to detect ascending and descending staircases. This method is considerably faster than other similar alternatives. Also, wide angle cameras were used for this task in [Romić et al., 2017], in a system that included sound guidance to communicate the best direction to follow (Figure 2.5a). In [Abdulatif et al., 2017] they introduce a mirror-based two-dimensional frequency-modulated continuous-wave radar scanner for wheelchairs to enable stair detection (Figure 2.5b).

### 2.3. SCENE SEGMENTATION AND CLASSIFICATION

In any visual assistant, in order to perform any complex task it is necessary to recognize features in the surroundings. Before the recognition, a partition of the environment in different segments must be performed, and that is called segmentation. The starting point of this method is the segmentation of the scene in planes and clusters (Section 2.3.1). To give context to these segments it is necessary to calculate how the scene is oriented with respect to the user (Section 2.3.2). Once we have calculated the main transformations of the scene it is possible to classify the planar segments according to their orientation and localize the step candidates (Section 2.3.3).

### 2.3.1. Segmentation

In most human-made scenarios, the basic structure of the scene is a combination of planes at different orientations. Range sensors have proven to be extremely helpful for planar segmentation, and many algorithms have been developed through the years [Hoover et al., 1996]. We use the algorithm from [Rabbani et al., 2006], integrated in the Point Cloud Library (PCL) [Rusu and Cousins, 2011]. This region-growing algorithm outputs delimited regions of points with similar normal orientations and spatially lying on a unique 3D plane. We prefer this approach instead of using direct plane detection algorithms, such as RANSAC, because with region-growing the planes found form already closed regions corresponding to one single element and are not a set of uncorrelated points scattered in the scene (e.g. Figure 2.4a). Prior to this phase the point cloud is filtered to reduce the amount of data and then a normal extraction algorithm is applied. Our complete segmentation procedure has the following stages:

### 2.3.1.1. DownsAMPLING

Each point cloud has a large quantity of points ( $640 \times 480$ pixel) which provides redundant information and makes further computations highly time-consuming (Figure 2.6a). Thus, the first operation is downsampling. We apply the 3D voxel grid filter from [Rusu and Cousins, 2011] to the point cloud, i.e. a 3D division of the space in a grid of 3D boxes (voxels) inside of which there is only one point (the centroid) instead of the initial set of points contained. The size of the edges of the voxels is determined by balancing time consumption and accuracy. Big voxels improve the performance rate but reduces the accuracy of the models. Typically, a size of voxels of about $3-4 \mathrm{~cm}$ worked well for us. This is a common algorithm widely used for downsampling point clouds, which also helps removing noise and smoothing the surfaces. As a result of the downsampling we define the point cloud $P^{C}=\left\{\mathbf{p}_{1}^{C}, \mathbf{p}_{2}^{C}, \ldots, \mathbf{p}_{i}^{C}, \ldots \mathbf{p}_{n-1}^{C}, \mathbf{p}_{n}^{C}\right\}$, where $\mathbf{p}_{i}^{C}=\left(x_{i}^{C}, y_{i}^{C}, z_{i}^{C}\right)$ represents each of the $n$ points in the scene in the camera reference $C$ (Figure 2.6b).

### 2.3.1.2. NORMAL ESTIMATION

The surface normal estimation is based on the Principal Component Analysis (PCA) [Rusu and Cousins, 2011], consisting in the analysis of the eigenvectors and eigenvalues of a covariance matrix created from the nearest neighbors of every point $\mathbf{p}_{i}$. The eigenvector associated with the smallest eigenvalue corresponds to the normal direction $\mathbf{n}_{i}$ (Figure 2.6c). We considered the neighbors in a small radius ( 5 cm around the points) to reduce the computation time and be able to detect sharper edges. In this process the curvature of the surfaces $c_{i}$ is also computed to feed the following stage.


Figure 2.6. Example of the segmentation process used in this work: (a) Initial colored point cloud as retrieved by the camera. (b) Cloud after applying a voxel grid filter. (c) Normal extraction. (d) After the region-growing algorithm, where the planar regions are colored in random colors. (e) Non-planar clusters from the Euclidean Cluster Extraction algorithm colored randomly. (f) Classification of planes (yellow = vertical, blue $=$ horizontal, green $=$ floor, red $=$ others )

### 2.3.1.3. REGION-GROWING

This algorithm [Rabbani et al., 2006] starts from a seed, which is the point with minimum curvature, and then expands the region towards the neighboring points that have small normal deviation and similar curvature value. The neighboring points which satisfy the normal and curvature threshold become the new seeds and the process is repeated until the region cannot expand any more. Then, a new initial seed is chosen among the remaining points, and the process starts over until the regions found are smaller than a certain pre-established threshold. The thresholds we set in normal's deviation and curvature values are respectively $6^{\circ}$ and 0.5 . The minimum size of a region is set to 50 points.

We get as output a set of regions $R^{C}=\left\{R_{k}^{C}\right\}$ where $R_{k}^{C} \subseteq P^{C}$ (Figure 2.6d). Usually all $\mathbf{p}_{i}^{k}=\left(x_{i}^{k}, y_{i}^{k}, z_{i}^{k}\right) \in R_{k}^{C}$ lie on a plane $A^{k} x_{i}^{k}+B^{k} y_{i}^{k}+C^{k} z_{i}^{k}+D^{k}=0$ where ( $A^{k}, B^{k}, C^{k}, D^{k}$ ) are the plane coefficients of $R_{k}^{C}$. However, the points of the region might as well form a curved surface with smooth transitions, which would satisfy the thresholds above. As the ground, walls, doors or steps are all planes, it is important to verify this possibility. Thus, a RANSAC algorithm seeks for the biggest plane in each region. If most of the points are inliers (we set more than $80 \%$ ), it is considered a planar surface with the plane equation obtained, and therefore normal vector $\mathbf{n}_{k}^{C}=$ $\left(A^{k}, B^{k}, C^{k}\right)$ and distance to the origin $D^{k}$.

### 2.3.1.4. EUCLIDEAN CLUSTER EXTRACTION

The points still not belonging to any region go through a cluster extraction algorithm which establishes connections and forms separate entities just by looking at their Euclidean position in the scene, ignoring normal orientations. In this operation we group in the same instance all the points that form isolated objects in the scene in a set of clusters $C^{C}=\left\{C_{k}^{C}\right\}$ where $C_{k}^{C} \nsubseteq R^{C}$ (Figure 2.6e).

### 2.3.2. SCENE ORIENTATION

As the points of the cloud are referenced to the camera, a change of the reference system is necessary to determine the position with respect to the user in order to provide a more natural way to understand the environment and the movements of the person. Besides, the fact that most human-made indoor scenarios are composed by planes situated in three dominant orientations can be used in our benefit.

### 2.3.2.1. CAMERA TO FLOOR

The information from the camera is not very useful on its own when reasoning about the scene if the relative position of the camera to the world is unknown. We move the reference frame from the camera to the floor by computing the transformation ${ }^{C} \mathbf{T}_{F}$


Figure 2.7. Main frame transformations of the algorithm. ${ }^{C_{0}} \mathbf{T}_{C_{k}}$ : from the initial camera reference frame $\left(C_{t=0}=C_{0}\right)$ to the camera in $t=k\left(C_{k}\right) .{ }^{C} \mathbf{T}_{F}$ : from the camera reference frame $(C)$ to the floor $(F)$.
${ }^{F} \mathbf{T}_{M}$ : Transformation from the floor reference frame $(F)$ to match the Manhattan directions (M).
shown in Figure 2.7. That way all the planes will be properly oriented and the height of the points with respect to the floor allows to draw conclusions about the nature of the objects in the scene.

The computation of this transformation requires to find the floor plane (in Figure 2.7 the green plane with normal $\mathbf{v}_{F}$ ). As no other sensor has been used for this task, the only previous knowledge is the approximate location of the camera on the chest. A RANSAC procedure [Fischler and Bolles, 1981] is used to find the biggest planes one by one, and the relative distance and orientation of each plane with respect to the camera are analyzed to determine whether it is floor or not. Note that, simply considering the largest plane to be the floor will not be true whenever there are more dominant planes in the scene (e.g. a table, a wall). The rules to verify that a plane is floor or not are the following:

- The orientation of the normal must be coherent with the orientation of the camera in the chest. For example, in Figure 2.7 it would be an approximate rotation of $\approx 180^{\circ}$ in $\mathbf{z}_{C}$ and of $\approx 45^{\circ}$ in $\mathbf{x}_{C}$ so the $\mathbf{y}_{F}$ matches the floor normal $\mathbf{v}_{F}$.
- The distance of the plane to the camera should be within a provided valid range which depends on the height of the user.
- It is very likely that the floor appears close to the subject, as the camera points down. So we can consider for floor detection points closer than a certain threshold in $\mathbf{z}_{C}$.


Figure 2.8. Projection of the point clouds to the floor reference frame from a real case scenario, where the white points on the floor are those which form the best floor candidate plane and the yellow arrow is the corresponding normal

This computation requires relaxed thresholds to not discard valid portions, as conditions can vary due to the movement and the height of the subject; but they should quickly discard planes belonging to instances such as walls or tables. In our implementation the inclination of the camera is not restricted to $45^{\circ}$, but within a valid range of $20^{\circ}-70^{\circ}$ and the height of the camera from the floor within $1-1.6 \mathrm{~m}$. The distance to consider points as inliers in the RANSAC is the voxel edge size. The last condition could be useful to discard the planes which are extremely close to the floor, such as steps, which could deceive the algorithm. The algorithm typically start searching within $\mathbf{z}_{C}=1 \mathrm{~m}$ and progressively increasing the threshold until a valid plane is found. In Figure 2.8 there is an example with a real point cloud where the points within the initial threshold are colored in white.

### 2.3.2.2. Floor to Manhattan

We assume that most indoor scenes satisfy the Manhattan World assumption, i.e. most planes have normals in three mutual orthogonal directions [Coughlan and Yuille, 1999]. This simplifies the reasoning about the environment, and it is widely used in the literature since it is usually correct in indoor environments. In our case of study we are going to use it to retrieve the directions of the stairs, as most certainly lie in that convention. The stair must be properly oriented to get the model that fits the points better.

To acquire the Manhattan directions $\left(\mathbf{m}_{x}, \mathbf{m}_{y}, \mathbf{m}_{z}\right)$ we can take advantage of the previous transformation floor ${ }^{C} \mathbf{T}_{F}$, as it already matches the vertical direction ( $\mathbf{m}_{y}=$ $\mathbf{y}_{F}$ ). The problem is then reduced to find $\mathbf{m}_{x}$ and $\mathbf{m}_{z}$. After the segmentation we have a set of planes with their normal directions and their number of points. The two orthogonal directions which satisfy the greater number of points are the $\mathbf{m}_{x}$ and $\mathbf{m}_{z}$. To solve the


Figure 2.9. Excerpt from the Technical Edification Code of Spain (Código Técnico de la Edificación, CTE, DB-SUA, section 1.4.2., 2006) showing measurements that stairs must have.
ambiguity of these directions we choose as $\mathbf{m}_{z}$ the one that has the smallest angle with the vector pointing out to the front of the user $\left(\mathbf{z}_{F}\right)$. Once we have the Manhattan directions we define the reference frame $M$. In Figure 2.7 the $\mathbf{x}_{M}$ share directions with the blue and purple planes and $\mathbf{z}_{M}$ with the orange one. Throughout iterations the ${ }^{F} \mathbf{T}_{M}$ is recalculated but maintaining the orientation convention assumed.

### 2.3.3. CLASSIFICATION OF REGIONS

Once we have the planar regions with their normals and the transformations to the Manhattan directions, we can transform the regions $R^{C}$ to the new reference frame $R^{M}$. The regions are then classified regarding their orientation and relative position. To consider a plane parallel or perpendicular to another a threshold of $10^{\circ}$ is considered in every case.

The planes which are perpendicular to the floor are classified as vertical (e.g. walls, doors, risers, furniture). These are the planes used to obtain the Manhattan directions of the environment. As we have these directions, extended reasoning about the orientation of the walls can be done and new subcategories of vertical planes can be added (left, right, frontal).

The planes which are parallel to the floor are horizontal. The distance of the planes to the floor $\left(H_{k}\right)$ is considered for further analysis, considering the dimensions regulated by the Technical Edification Code ${ }^{1}$ (Figure 2.9). According to the Code the vertical distance between two consecutive steps ranges from a minimum $H_{\text {min }}=13 \mathrm{~cm}$ to a maximum $H_{\max }=18.5 \mathrm{~cm}$. Horizontal regions are considered as step candidates if they are situated above (in ascending stairways) or below (in descending ones) $H_{\text {min }}$ from the floor (Figure 2.10). The floor must have height zero given the transformation ${ }^{F} \mathbf{T}_{M}$ and

[^3]

Figure 2.10. Classification of planes regarding orientation and height. Horizontal planes are classified among Floor (green), Obstacles (red) or Step candidates (blue) depending their distance to the floor. Vertical planes (yellow) are used to compute ${ }^{F} \mathbf{T}_{M}$.
other planes whose height does not fit above descriptions (i.e. less height than $H_{\text {min }}$ ) are considered obstacle. To evaluate this condition it is necessary to add a threshold as the measurements can be noisy. Other size and shape restrictions are kept to a minimum at this point because they could discard valid portions of steps which might be useful for a better modeling of staircases. From this operation we get a set of step candidates $S=\left\{S_{1}, S_{2}, \ldots, S_{i}, \ldots S_{n_{s}-1}, S_{n_{s}}\right\}$ where $S_{i}=\left\{R_{k}\left|\left\|\mathbf{n}_{k} \times \mathbf{y}_{F}\right\| \approx 0,\left|H_{k}\right| \geq H_{\text {min }}\right\}\right.$ and $n_{s}$ is the number of step candidates. The existence of a set of at least one step candidate activates the stair detection algorithm.

The planes which are not perpendicular nor parallel to the ground, along with the non-parallel regions and clusters (Section 2.3.1) are kept as obstacles and removed from further analysis. This information could be used to obtain the obstacle-free walkable area. Note that, the fact that the floor plane is visible does not always mean it is walkable area, since there could be some obstacle above, like a table. Two examples of removing the obstacle area from the floor plane are shown in Figure 2.11. In Figure 2.6f there is a complete example of the classification.

### 2.4. Stair detection, modeling and TRAVERSAL

The step candidates obtained in Section 2.3.3 are the input of the stair detection and modeling algorithm, whose output consists in the detection and retrieval of a scaled model of the staircase. At this moment, the algorithm is functional with both ascending


Figure 2.11. In green, the portion of the ground which can be walked over as it has no obstacles.
and descending staircases. Isolated single steps can also be detected. The algorithm is able to overcome partial occlusions of a stairway splitting steps in more than one region in the detection. Spiral staircases can be detected but the modeling part has not been addressed yet.

### 2.4.1. Stair DETECTION

The detection consists in determining the presence of a staircase in the scene. That is, to determine whether the $n_{s}$ step candidates ( $S$ ), or a subset of those, form a staircase or not. To verify this, we analyze if the disposition of the step candidates resembles an actual staircase, i.e. they form groups of planes that are uniformly separated vertically by regulated distances, and shifted horizontally so they are not on top of each other. The algorithm establishes connections among the candidates forming a graph-like structure, such as the ones shown in Figure 2.12. The connectivity between step candidate regions $S_{A}$ and $S_{B}$ has been computed considering there is at least a minimum pre-established number of points from $S_{A}$ inside a valid range of distances from $S_{B}$. In this case, we set a radius of 0.5 m for the Kd -tree nearest neighbor search and a minimum amount of 10 points to set a valid connection. Connected steps are organized in levels that measure the distance to the floor in steps. For example, level 1 means first step upwards, whereas level -2 means two steps down the staircase. The floor plane is the reference, at level 0 . Creating connections as described allow us to discard unconnected candidates and to overcome occlusions that may create separate planar patches of the same step, like in the second example of Figure 2.12.

The candidates are analyzed one by one starting from the closest to the floor, and verifying the connections to the previously established levels every time (Figure 2.13a). We define a set of levels $L=\left\{L_{j}\right\}$, where $j=0 . . n_{L}$ and $n_{L}$ is the highest level detected. The level zero is occupied by the floor, so initially $n_{L}=0$. The candidates whose


Figure 2.12. On the left, two examples of stair detection with both ascending and descending stairs (top) and with more than one region per level (bottom). The connectivity is traced with white arrows. At the right, the graph structure that summarizes the connections between regions and its organization per level.
centroid is between $H_{\min }$ and $H_{\max }$ to the ground constitute the first step candidates (Figure 2.13b). The first step must be connected to the floor if it is present in the image (Figure 2.13c). If no first step candidate satisfies neighboring conditions, the algorithm determines there is no staircase. Otherwise, $L_{1}=S_{1}$ (where $H_{\min } \leq H_{1} \leq H_{\max }$ ) is established, and $n_{L}=1$.

The algorithm takes the remaining step candidates by height and starts testing connectivity and height conditions to determine whether they belong to a new (Figure 2.13d) or to the current level (Figure 2.13e). Note that we consider the height of the centroid of the regions. In case they belong to the current level (a step candidate is considered the same height if it is within $\pm 3 \mathrm{~cm}$ ), the step candidate regions fuse in one single point cloud forming the level. If they have no connection to previous levels (e.g. a horizontal plane correspondent to a table) they are classified as obstacles (Figure 2.13f). As a result, a set of connected regions corresponding to different levels is obtained ( $\mathrm{Fi}-$ gure 2.12). The algorithm is summarized in Algorithm 1. When all the candidates have been checked and the number of levels is greater than one, the system proceeds with the


Figure 2.13. Explicative sketches for the stair detection algorithm. (a) Select the candidates in order. (b) First step must be in the valid range of heights. (c) First step must be connected to the floor if it is visible. (d)(e) The connectivity to previous levels is checked doing a neighbor search. (f) If the candidate is not connected to the previous level, it is not part of the staircase.

```
Algorithm 1: Stair detection algorithm
    Step candidates list \(S=\left\{S_{1}, S_{2}, \ldots, S_{i}, \ldots S_{n_{s}-1}, S_{n_{s}}\right\}\);
    Levels list \(L=\{ \}\);
    \(L_{0}=R_{\text {floor }} ; \quad n_{L}=0 ;\)
    \(S=\operatorname{SORTBYHEIGHT}(S)\);
    if \(\left(S_{1}\right.\). height \(\left.>H_{\max }\right)\) then stair \(=\) false;
    else
        stair \(=\) true;
        for \(i=1: n_{s}\) do
            \(S_{i}\). is_connected \(=\) false;
            for \(j=0: n_{L}\) do
                if (ARECONNECTED \(\left.\left(S_{i}, L_{j}\right)\right)\) then \(S_{i} . i s \_\)connected \(=\)true;
            if \(S_{i}\).is_connected then
                if (Si.height \(\approx L_{n_{L}}\). height) then \(L_{n_{L}} \cdot\) points \(\cup S_{i}\).points;
                    else \(n_{L}=n_{L}+1 ; \quad L_{n_{L}}=S_{i} ;\)
    return stair;
```



Figure 2.14. Three examples of detections of curbs and floor at another level.
modeling of the staircase.
A special case occurs when there is only one step candidate. It might either actually be the first step of a staircase, or be just a single curb on the way. Curbs are a singular case, sometimes omitted by stair detection algorithms, but dangerous as well. Here, more strict area and shape analysis can be applied in order to determine in which case we are. In particular, we do not consider shape restrictions for the region like with stairs, since the surface area beyond the curb can be of any size, and the regulations for staircases do not apply. Thus, we consider there is a valid curb and thus there is floor at another level if the surface area of the region is big enough to be walked over, or an obstacle if it is small, like an object on the ground. In Figure 2.14 there are three examples of positively detected curb.

### 2.4.2. Stair modeling

Not all staircases are equal, even if we just consider the rectangular case. For example, they may or may not have risers, or the risers could be inclined or perpendicular to the floor orientation. For the modeling of a staircase, we are going to consider a unified model for all the possible cases, where the steps are defined by a horizontal rectangular plane of $l_{\text {width }} \times l_{\text {length }}$ and a vertical rectangular plane of $l_{\text {width }} \times l_{\text {height }}$ which links the horizontal plane to the previous level. The line where two planes intersect is called the edge of the step. Every staircase is also oriented according to three orthogonal directions whose pose with respect to the user is relevant to guide the subject towards it. In the modeling phase we are going to retrieve the $l_{\text {width }}, l_{\text {length }}, l_{\text {height }}$ and ${ }^{C} \mathbf{T}_{S}$ as depicted in Figure 2.15. The model can be then drawn for the number of levels detected in the previous stage $\left(n_{L}\right)$. If the traversal of the staircase is then performed, the final number of levels can be obtained, with the procedure explained in Section 2.4.4.

The extraction of the measurements is correlated with the extraction of the ${ }^{C} \mathbf{T}_{S}$, for which first we need to define the three main directions of the stair, $\left(\mathbf{x}_{S}, \mathbf{y}_{S}, \mathbf{z}_{S}\right)$. We have developed two methods:

1. Manhattan World based method: Considering stairs are oriented according to the Manhattan assumption, i.e. $\left(\mathbf{x}_{S}, \mathbf{y}_{S}, \mathbf{z}_{S}\right)=\left(\mathbf{x}_{M}, \mathbf{y}_{M}, \mathbf{z}_{M}\right)$.


Figure 2.15. Parameters to compute for the modeling: $l_{\text {width }}, l_{\text {length }}, l_{\text {height }}$ and ${ }^{C} \mathbf{T}_{S}$.
2. PCA based method: Method based on the Principal Component Analysis of the step candidates.

The better option of the two depends on the type of scene: scenes populated with big structural planes (coming from a side wall or the risers, for instance) work better with the first one, whereas scenes with mainly horizontal planes (e.g. isolated stairs, no risers) work better with the second. The Manhattan World option is used when there are enough planes with normals in horizontal directions that allow us to call the scene "Manhattan World scene". We determine this circumstance with a threshold empirically set at a percentage of Manhattan points (i.e. points whose normal follows one of the Manhattan directions, see Section 2.3.2.2) of the total number of points. The PCA option is more time consuming, so only is used when there is not enough evidence of the Manhattan directions of the scene. This method can always be applied, as it is based on the step candidates already detected (i.e. if there is no step candidates the modeling does not even start). However, when the observation of the steps is partial due to occlusions it could lead to erroneous solutions (hence, it is the second choice). From here on the PCA method is detailed as follows:

From the last stage we have a level-organized set of points corresponding to each visible step. We can perform a Principal Component Analysis (PCA) in every set of points to retrieve their main directions and initial estimate of their measurements by defining the bounding rectangle which encloses all the points in these directions. From this procedure we get a collection of principal components and measurements which presents high variance (Figure 2.16a). To solve this we choose one step as best initial


Figure 2.16. (a) Principal components for each step colored in order (blue-green-red) and bounding rectangle in white. (b) Illustrative sketch of the different components involved in the PCA algorithm. (c) Example of the rotation of the selected axis to minimize the area of the bounding box with the points of one single step. It needs to be done with all the steps at the same time in order to obtain the final direction that fits best the staircase.
guess: the one with greater extent. We define extent as the ratio of the area of the concave hull and the area of the rectangle as defined in Figure 2.16b. The principal components of this step are rotated twice:

1. To match the vertical direction (i.e. the normal of the floor plane).
2. Until the sum of areas of the bounding rectangles computed using these directions of all steps at the same time is minimized.

With this last operation you make sure that the axis of the model fits the points of the stair in the best way. In Figure 2.16c there is a visual example of how the directions are rotated so the area of one step is minimized.

Once the directions of the staircase have been computed, either using Manhattan World method or PCA, the bounding rectangles of each step present different dimensions, so the final stage consists in defining the global dimensions of the staircase. The


Figure 2.17. (a) In the case of ascending (left) and descending stairs (right) the measured length from the bounding rectangles of the steps, $d$, does not match the real length of our model, $D$ (which assumes orthogonal continuous steps). (b) Two stairs where this circumstance can be observed, i.e. non-existent risers in ascending stair (left) or any descending stair (right). (c) Our results of the modeling after the correct length estimation.
$l_{\text {width }}$ can be chosen as the largest value of all, whereas the $l_{\text {height }}$ is considered the average of vertical distances between the centroids of consecutive steps. The $l_{\text {length }}$ cannot be computed as an average of the measured lengths of the steps, since the vertical projection of the bounding rectangles of two consecutive steps usually overlaps in ascending staircases due to inclining or non-existent risers, or leaves a gap in descending staircases due to self-occlusions (Figure 2.17). It causes that the length you see in ascending staircases is more than what we use for our model, whereas in descending staircases some valid portion of the step is hidden. Thus, we compute the $l_{\text {length }}$ as the average horizontal distance between the edges of every two consecutive steps (see model length $D$ in Figure 2.17a).

Once we have all the parameters, we can use them to validate the staircase detection or discard false positives, as we know the appropriate dimensions steps must have by regulations. The ${ }^{S} \mathbf{T}_{C}$ can be obtained using as rotation matrix the three stair directions as described, and as translation vector the coordinate in the camera reference frame of the center of the edge of the first step of the staircase (as depicted in Figure 2.15). Alternatively, depending on the design of the assistive system, the translation part could direct the subject to other desirable part of the stair, such as one of the ends of the edge of the first step, where the handrails are expected to be.

### 2.4.3. VISUAL ODOMETRY

Odometry is the process of using the data from sensors to retrieve the change of position and orientation over time. When the sensors being used are cameras it is called visual odometry. In this work we estimate the visual odometry using the information from both RGB and depth cameras. Calling the initial reference frame of the camera $C_{0}$, and the reference frame in the instant $k$ as $C_{k}$, we define the transformation provided by the visual odometry as ${ }^{C_{0}} \mathbf{T}_{C_{k}}$ (Figure 2.7). In theory, this transformation could be used to displace any feature captured at any given time to a common geometrical reference. This information can be of great help, since it allow us to move on from single image processing by adding some spatial memory to the system. In practice, the sensor localization typically has some drift issues that increase through iterations. As our goal is far away from creating an accurate reconstruction of the environment, we use the odometry as a rough estimate of the position of past features that might be no longer visible, such as the floor, or the stairs. For the visual odometry from RGB-D there are many approaches [Raposo et al., 2013, Taguchi et al., 2013]. We use the method presented by Gutierrez-Gomez et al. [Gutierrez-Gomez et al., 2015], where visual odometry is obtained in real time from the dense RGB and inverse depth maps by establishing pixel-wise constraints through the flow equations. The method can be summarized as follows:

Let us denote two camera frames as $A$ and $B$, at instants $t$ and $t+\Delta t$ respectively. Given the intensity images $\mathcal{I}_{A}$ and $\mathcal{I}_{B}$, and inverse depth maps $\mathcal{W}_{A}$ and $\mathcal{W}_{B}$ defined over the image domain $\Omega \subset \mathbb{P}^{2}$, for an image point $\mathbf{p}=(u, v, 1)^{\top} \in \Omega$ in frame $A$, the
following photometric and geometric constraints hold:

$$
\begin{align*}
\mathcal{I}_{B}(\mathbf{p}+\Delta \mathbf{p}) & =\mathcal{I}_{A}(\mathbf{p})  \tag{2.1}\\
\mathcal{W}_{B}(\mathbf{p}+\boldsymbol{\Delta} \mathbf{p}) & =\frac{1}{\mathbf{e}_{\mathbf{z}}^{\top} \mathbf{X}_{B}} \tag{2.2}
\end{align*}
$$

where $\mathbf{X}_{B}$ is the 3D point lifted from pixel $\mathbf{p}+\boldsymbol{\Delta} \mathbf{p}$ in frame $B, \Delta \mathbf{p}=(\Delta u, \Delta v, 0)^{\top}$ is the displacement of one point from frame $A$ to $B$, and $\mathbf{e}_{\mathbf{z}}^{\top}=(0,0,1)$. The photometric constraint assumes constant illumination of one scene point over time. The geometric constraint is the measurement model of the depth sensor at frame $B$ in inverse depth parametrization.

Assuming small pixel displacements between frames we compute the flow equations from (2.1) and (2.2):

$$
\begin{align*}
\nabla \mathcal{I}_{A}(\mathbf{p}) \Delta \mathbf{p}+\mathcal{I}_{B}(\mathbf{p}) & =\mathcal{I}_{A}(\mathbf{p})  \tag{2.3}\\
\nabla \mathcal{W}_{A}(\mathbf{p}) \Delta \mathbf{p}+\mathcal{W}_{B}(\mathbf{p}) & =\frac{1}{\mathbf{e}_{\mathbf{z}}^{\top} \mathbf{X}_{B}} \tag{2.4}
\end{align*}
$$

where the gradient operators $\nabla \mathcal{I}=\left(\frac{\partial \mathcal{I}}{\partial u}, \frac{\partial \mathcal{I}}{\partial v}, 0\right)$ and $\nabla \mathcal{W}=\left(\frac{\partial \mathcal{W}}{\partial u}, \frac{\partial \mathcal{W}}{\partial v}, 0\right)$.
Using the camera projection and inverse projection models, $\mathbf{p}=\pi(\mathbf{X})=\mathbf{K} \frac{\mathbf{X}}{\mathbf{e}_{\mathbf{z}} \mathbf{X}}$ and $\mathbf{X}=\pi^{-1}(\mathbf{p})=\frac{1}{\mathcal{W}(\mathbf{p})} \mathbf{K}^{-1} \mathbf{p}$, with $\mathbf{K}$ being the conventional calibration matrix, and with the same assumption of small pixel displacement we get (using first order Taylor expansion):

$$
\begin{align*}
& \frac{1}{\mathbf{e}_{\mathbf{z}}^{\top} \mathbf{X}_{B}}=\frac{1}{\mathbf{e}_{\mathbf{Z}}^{\top} \mathbf{X}_{A}}-\frac{1}{\left(\mathbf{e}_{\mathbf{z}}^{\top} \mathbf{X}_{A}\right)^{2}} \mathbf{e}_{\mathbf{z}}^{\top} \boldsymbol{\Delta} \mathbf{X}_{\mathbf{p}}+\mathcal{O}\left(\left\|\mathbf{e}_{\mathbf{z}}^{\top} \boldsymbol{\Delta} \mathbf{X}_{\mathbf{p}}\right\|^{2}\right) \\
& \approx \mathcal{W}_{A}(\mathbf{p})-\mathcal{W}_{A}^{2}(\mathbf{p}) \mathbf{e}_{\mathbf{z}}^{\top} \Delta \mathbf{X}_{\mathbf{p}}  \tag{2.5}\\
& \begin{aligned}
\Delta \mathbf{p} & = \\
& \mathbf{K} \frac{\mathbf{X}_{B}}{\mathbf{e}_{\mathbf{z}}^{\top} \mathbf{X}_{B}}-\mathbf{K} \frac{\mathbf{X}_{A}}{\mathbf{e}_{\mathbf{z}}^{\top} \mathbf{X}_{A}} \\
& =\mathbf{K X}_{B}\left(\mathcal{W}_{A}(\mathbf{p})-\mathcal{W}_{A}^{2}(\mathbf{p}) \mathbf{e}_{\mathbf{z}}^{\top} \Delta \mathbf{X}_{\mathbf{p}}\right)-\mathbf{K} \mathbf{X}_{A} \mathcal{W}_{A}(\mathbf{p}) \\
& =\mathcal{W}_{A}(\mathbf{p})\left(\mathbf{K}-\mathbf{p} \mathbf{e}_{\mathbf{z}}^{\top}\right) \Delta \mathbf{X}_{\mathbf{p}}
\end{aligned}
\end{align*}
$$

where $\Delta \mathbf{X}_{\mathbf{p}}$ is the 3D flow associated to each pixel, $\Delta \mathbf{X}_{\mathbf{p}} \doteq \mathbf{X}_{B}-\mathbf{X}_{A}$. Substituting in (2.3) and (2.4), we get the linear constraints on this pixel-wise 3D flow.

$$
\begin{align*}
& \mathcal{W}_{A}(\mathbf{p}) \nabla \mathcal{I}_{A}(\mathbf{p})\left(\mathbf{K}-\mathbf{p} \mathbf{e}_{\mathbf{z}}^{\top}\right) \Delta \mathbf{X}_{\mathbf{p}}+\mathcal{I}_{B}(\mathbf{p})-\mathcal{I}_{A}(\mathbf{p})=0  \tag{2.7}\\
& \mathcal{W}_{A}(\mathbf{p})\left(\nabla \mathcal{W}_{A}(\mathbf{p})\left(\mathbf{K}-\mathbf{p e} \mathbf{e}_{\mathbf{z}}^{\top}\right)\right.\left.+\mathcal{W}_{A}(\mathbf{p}) \mathbf{e}_{\mathbf{z}}^{\top}\right) \Delta \mathbf{X}_{\mathbf{p}}+ \\
&+\mathcal{W}_{B}(\mathbf{p})-\mathcal{W}_{A}(\mathbf{p})=0 \tag{2.8}
\end{align*}
$$

Assuming that the scene is rigid, the 3D flow map $\Delta \mathbf{X}_{\mathbf{p}}$ is produced only by a small interframe camera motion, described by the rotation translation pair $\left({ }^{B} \mathbf{R}_{A},{ }^{B} \mathbf{t}_{A}\right) \in$ $S E(3)$ :

$$
\begin{align*}
\boldsymbol{\Delta} \mathbf{X}_{\mathbf{p}} & ={ }^{B} \mathbf{R}_{A} \mathbf{X}_{A}+{ }^{B} \mathbf{t}_{A}-\mathbf{X}_{A} \\
& =\left(\mathbf{I}+\left[{ }^{B} \boldsymbol{\theta}_{A}\right]_{\times}\right) \mathbf{X}_{A}+{ }^{B} \mathbf{t}_{A}-\mathbf{X}_{A}+\mathcal{O}\left(\left\|\left[{ }^{B} \boldsymbol{\theta}_{A}\right]_{\times}^{2} \mathbf{X}_{A}\right\|\right) \\
& \approx^{B} \mathbf{t}_{A}-\left[\pi^{-1}(\mathbf{p})\right]_{\times}{ }^{B} \boldsymbol{\theta}_{A}=\mathbf{M}(\mathbf{p})^{B} \boldsymbol{\xi}_{A} \tag{2.9}
\end{align*}
$$

where $[\cdot]_{\times}$denotes the antisymmetric matrix from a vector and ${ }^{B} \boldsymbol{\theta}_{A}$ is the logarithmic map of ${ }^{B} \mathbf{R}_{A}$. Note that ${ }^{B} \boldsymbol{\xi}_{A}=\left({ }^{B} \mathbf{t}_{A} ;{ }^{B} \boldsymbol{\theta}_{A}\right)$ is not a twist, i.e. ${ }^{B} \boldsymbol{\xi}_{A} \notin \mathfrak{s o}(3)$, since ${ }^{B} \mathbf{t}_{A}$ is the translation part of the rigid motion. Eq. (2.9) leads to a well-posed problem with 6 unknowns corresponding to the camera motion parameters for nearly $W_{i m} H_{i m}$ constraints (where $W_{i m}$ and $H_{i m}$ are the width and height of the image respectively), excluding pixels without depth measurements, with the following residuals:

$$
\begin{align*}
r_{\mathcal{I}}(\mathbf{p}, \boldsymbol{\xi})= & \mathcal{W}_{A}(\mathbf{p}) \nabla \mathcal{I}_{A}(\mathbf{p})\left(\mathbf{K}-\mathbf{p} \mathbf{e}_{\mathbf{z}}^{\top}\right) \mathbf{M}(\mathbf{p}) \boldsymbol{\xi}+\mathcal{I}_{B}(\mathbf{p})-\mathcal{I}_{A}(\mathbf{p})  \tag{2.10}\\
r_{\mathcal{W}}(\mathbf{p}, \boldsymbol{\xi})= & \mathcal{W}_{A}(\mathbf{p})\left(\nabla \mathcal{W}_{A}(\mathbf{p})\left(\mathbf{K}-\mathbf{p e}_{\mathbf{z}}^{\top}\right)+\mathcal{W}_{A}(\mathbf{p}) \mathbf{e}_{\mathbf{z}}^{\top}\right) \mathbf{M}(\mathbf{p}) \boldsymbol{\xi}+ \\
& +\mathcal{W}_{B}(\mathbf{p})-\mathcal{W}_{A}(\mathbf{p}) \tag{2.11}
\end{align*}
$$

which can be straightforwardly minimized by standard Gauss-Newton least squares.
In practice we do not apply conventional least squares. Instead we use a robust cost function by applying iteratively reweighted least squares algorithm [Holland and Welsch, 1977]. We also follow a coarse-to-fine approach using a 3 level image pyramid, performing a number of 10 iterations on a pyramid level before stepping down to the next finer level. The incremental motion estimate at each iteration $\gamma$ is computed as:

$$
\begin{equation*}
{ }^{B} \boldsymbol{\xi}_{A}^{(\gamma)}=\underset{\boldsymbol{\xi}}{\operatorname{argmin}} \sum_{\mathbf{p} \in \Omega} \omega\left(\frac{\breve{r}_{\mathcal{I}}(\mathbf{p})}{\sigma_{r_{\mathcal{I}}}}\right) \frac{r_{\mathcal{I}}^{2}(\mathbf{p}, \boldsymbol{\xi})}{\sigma_{r_{\mathcal{I}}}^{2}}+\omega\left(\frac{\breve{r}_{\mathcal{W}}(\mathbf{p})}{\sigma_{r_{\mathcal{W}}}}\right) \frac{r_{\mathcal{W}}^{2}(\mathbf{p}, \boldsymbol{\xi})}{\sigma_{r_{\mathcal{W}}}^{2}} \tag{2.12}
\end{equation*}
$$

where $\breve{r}_{\mathcal{I}}(\mathbf{p})$ and $\breve{r}_{\mathcal{W}}(\mathbf{p})$ denote the initial residuals computed after warping intensity and inverse depth maps in frame $B$ towards frame $A$ with the estimated camera motion up to current iteration ${ }^{k} \mathbf{T}_{k+1}^{(\gamma+1)} \cdot \omega(x)=\frac{6}{5+x^{2}}$, since we use an estimator based on the Student's t-distribution with $\nu=5$ as in [Kerl et al., 2013], which shows in general better performance than other candidates. The scaling parameters are fixed to $\sigma_{r_{I}}=5$ and $\sigma_{r_{w}}=0.0025 m^{-1}$ based on tests on static sequences and the disparity measurement model of RGB-D sensors [Konolige and Mihelich, 2015].

After each iteration, the motion estimation between frames $k$ and $k+1$ is updated by the current incremental estimate:

$$
{ }^{k} \mathbf{T}_{k+1}^{(\gamma+1)}=\left(\begin{array}{cc}
\exp \left(\left[{ }^{B} \boldsymbol{\theta}_{A}^{(\gamma)}\right]_{\times}\right) & { }^{B} \mathbf{t}_{A}  \tag{2.13}\\
0 & 1
\end{array}\right)^{-1}{ }^{k} \mathbf{T}_{k+1}^{(\gamma)}
$$

Camera motion at first iteration ${ }^{k} \mathbf{T}_{k+1}^{(0)}$ is initialized assuming a constant velocity, i.e., ${ }^{k} \mathbf{T}_{k+1}^{(0)}={ }^{k-1} \mathbf{T}_{k}$.

### 2.4.4. Stair traversal

In the stair detection and modeling stage, all of the computations could be done on one single image at a time. However, during the traversal, the context of the problem changes, and thus the operations to perform need to be different. For instance, while the user is traversing the staircase instead of searching for stairs, the application should extract information about the current state of the traversal, e.g. in which step the user stands, which and where is the next step or how many steps are left to reach the other end. It is impossible to recover this kind of information in one single image analysis because all the steps usually look the same. The stair traversal is a continuous process extended in time. Hence, we introduce the visual odometry module described in Section 2.4.3 in the process.

During normal execution of our algorithm, the stair detection and modeling should be running normally. Assuming the user is approaching the staircase, there will be a point when the user stands so close to the staircase that the edge from the first step is no longer visible, and the transformation ${ }^{C} \mathbf{T}_{S}$ cannot be computed as described. Then it can be estimated using the visual odometry transformation. Calling $k$ the last iteration when the ${ }^{C_{k}} \mathbf{T}_{S}$ could be properly computed, it is possible to retrieve the ${ }^{C_{k+n}} \mathbf{T}_{S}$ in the iteration $k+n$ with the transformations ${ }^{C_{k}} \mathbf{T}_{S}$ and ${ }^{C_{0}} \mathbf{T}_{C_{k}}$ :

$$
\begin{equation*}
{ }^{C_{k+n}} \mathbf{T}_{S}=\left({ }^{C_{0}} \mathbf{T}_{C_{k+n}}\right)^{-1}{ }^{C_{0}} \mathbf{T}_{C_{k}}{ }^{C_{k}} \mathbf{T}_{S} \tag{2.14}
\end{equation*}
$$

The current pose of the camera with respect to the stair can be computed at any time and it shows the translation with respect to the initial point. Since $l_{\text {height }}$ and $l_{\text {length }}$ of the current staircase are known from the modeling (Section 2.4.2), the 3D position of the centroid of every step in sight transformed to the stair reference frame would reveal which step it is (Figure 2.18). With this information, every time a step of a level higher than the current $n_{L}$ is detected, the number of levels of the staircase model is updated ( $n_{L}:=n_{L}+1$ ). Similarly, transforming the estimated centroid of the body of the user to the stair reference frame can be used to know the step in which the user is, whether by directly looking at the $\mathbf{z}_{S}$ value or by computing the $\mathbf{y}_{S}$ value minus the estimated height of the camera (equal to the distance of the camera to the floor from previous computations). For example, in Figure 2.18, the user currently stands on Step 2. When the user step estimation using the height is higher than the one provided by the length, it means that the user is currently climbing the step to the following one.

In practice, it does not work as well as expected, since the visual odometry has a noticeable drift, especially in situations like this where the camera is constantly moving and a few centimeters can cause mismatches. However, we can turn the drift problem


Figure 2.18. The centroids of the steps and the estimated centroid of the body can be used to retrieve the numbers of the steps and the step the user is on by transforming the points to the stair reference frame.
around by recognizing known structures in the image. In particular, we use online observations of the stair, whose 3D model is well known (Section 2.4.2), to correct the drift. The transformation ${ }^{C_{k+n}} \mathbf{T}_{S}$ reveals an estimation of how the user has moved, or looking at it the other way, an estimation of how the stair is posed with respect to the user. The pose correction problem lies in computing the rotation and translation needed to match where the stair really is, i.e. as it is seen by the camera. Therefore, during the traversal we simultaneously apply a drift correction to improve the recovery of the odometry. It requires some slight changes of some previously commented algorithms. To compute the vertical direction, instead of looking for the floor, all step planes are used to compute the resulting normal as we can certainly say they are horizontal planes (Figure 2.19a). The rotation in $\mathbf{y}_{s}$ can be computed as described, either trusting Manhattan estimation or the directions from the PCA (Figure 2.19b). The rotation needed to move the estimated stair axes to the new ones is the correction in orientation. The translation part can be calculated by looking at the height of the centroids of the steps in $\mathbf{y}_{s}$ and the edgepoints of the steps can reveal the translation in $\mathbf{z}_{s}$ (Figure 2.19c). In the $\mathbf{x}_{s}$ direction the translation cannot be retrieved and the odometry needs to be trusted.

Calling ${ }^{S} \mathbf{T}_{S^{\prime}}$ the drift correction transformation, the final transformation from the current camera reference frame to the stair reference frame is:


Figure 2.19. To correct the stair pose estimated by the odometry we use the live depth information, drawn as the grey stair. (a) Correction of the orientation to match the vertical normal. (b) Correction of the orientation to match the edges of the steps in sight. (c) Correction in the position to match the centroids and edges of the steps.

$$
\begin{equation*}
{ }^{C_{k+n}} \mathbf{T}_{S^{\prime}}={ }^{C_{k+n}} \mathbf{T}_{S}{ }^{S} \mathbf{T}_{S^{\prime}} \tag{2.15}
\end{equation*}
$$

Using the correction, the computation of the position of the subject is more reliable. The dimensions of the last step found are computed in order to determine when they are significantly larger than the step length, because that would mean that it is the last step. Once these last step is detected, the model of the staircase is updated with the final number of steps of the stair. Notice that, instants before the user is about to reach the other end the stair is not in the view of the camera. This shows again how in traversing situations single image is not useful, and thus the need of the visual odometry and stair traversal algorithm. When the user finally stands on the floor at another level, the stair traversal algorithm stops and the algorithm proceeds as usual by performing the detection and modeling.

### 2.5. EXPERIMENTS

The experiments were carried out in a 3.4 Ghz computer with a GPU Nvidia GeForce GT730 running Ubuntu 12.04, ROS Hydro and the library PCL version 1.8. We use data both collected by ourselves and from public datasets. In particular, Tang et al. compiled a dataset in [Tang et al., 2012] which includes 148 captures acquired with a Microsoft Kinect sensor. 90 of them include RGB and depth snapshots of a set of staircases from different poses and the other 58 are normal indoor scenes to test for false positives. We have also analyzed the quality of our modeling, the computation time of the whole system and the stair traversal method.

Table 2.1. Comparison of false negatives and false positives between our work and the one presented in [Tang et al., 2012, Vlaminck et al., 2013]

| Method | False negative | False positive |
| :--- | ---: | ---: |
| Tang [Tang et al., 2012] | $5.07 \%$ | $1.02 \%$ |
| Vlaminck QVGA [Vlaminck et al., 2013] | $0.00 \%$ | $8.62 \%$ |
| Vlaminck VGA [Vlaminck et al., 2013] | $0.00 \%$ | $3.45 \%$ |
| Our work | $0.00 \%$ | $0.00 \%$ |



Figure 2.20. Four examples of images without staircases including deceiving parallel planes which are detected and then discarded by our algorithm due to impossible stair models.

### 2.5.1. Stairs DETECTION WITH PUBLIC DATASET

We tested for false positives and false negatives using this dataset and compared our results with the ones from [Tang et al., 2012] and [Vlaminck et al., 2013] (Table 2.1). The first thing that catches the attention of these results is the generally low percentage of both false positives and false negatives. That shows how appropriate RGB-D cameras are for the task. Our method achieves $0 \%$ of false negatives (FN) and false positives (FP), being the method that stands out in both metrics. In FN, our results were equal to [Vlaminck et al., 2013], meaning we detect all stairs from the dataset. However, [Vlaminck et al., 2013] had the highest percentage of FP, meaning their method is less reliable and detects stairs where there are none more often than the rest. On the other hand, [Tang et al., 2012] has less FP than [Vlaminck et al., 2013] although having higher percentage of FP, meaning their method is probably more conservative and less prone to affirm the presence of stairs when there is margin of error. The main reason of us being completely successful was that we include a modeling stage that allow us to add an extra validation step that discard invalid staircases for reasons like being too narrow, or having too small or too big steps. If we had not include the modeling in our method and the subsequent validation step, circumstances such as a bad floor detection or structures composed by parallel planes such as shelves would have caused the detection of a false positive. Some examples of false positives that our method detects but discards afterwards are shown in Figure 2.20. Some successful results of our stair detection and modeling from several images of the dataset are shown in (Figure 2.21).

We studied the step detection ratio according to the position of the step in the staircase using Tang's dataset (Figure 2.22). The behavior changes when we are facing an as-


Figure 2.21. Several examples of results obtained with Tang's dataset [Tang et al., 2012]. The model of the staircases retrieved is superimposed in the point cloud for visual verification.
cending staircase or a descending one. Since the images from the dataset were collected with a head-mounted camera, the viewpoint shifts from almost completely front-facing to approximately looking halfway down. Thus, standing before a descending staircase allows us to see the whole staircase. However, the self-occlusion of consecutive steps and the quality of the measurements decreasing with the distance, harm the detection of steps farther than the third position. The example from Figure 2.17 (right) shows this circumstance with the whole stair model drawn for visualization: after the third step there are fewer points and more noise. In ascending staircases the ratio of detection diminishes in a less prominent way, because the steps remain almost as close to the subject as they rise, with the penalty of having less and less visual angle. Steps higher than the seventh position are out of the field of view of the camera.

### 2.5.2. QUALITY OF THE MODELING

In general, the modeling usually provides qualitatively good results with rectangular staircases unless there are severe occlusions, there is a strong influence of the sun or the stairs present atypical constructions (e.g. Figure 2.23). We have quantitatively analyzed the resemblance of the model to the real staircase. We have excluded the width from the analysis as the view of the stairs may be partial and it is not as relevant as the other measurements. After computing the height and length of a staircases, in both ascending and descending perspectives, from different viewing angles, the results were compared to the real measurements of the steps, as shown in the Table 2.2. Besides, half of the experiments were conducted with a person going up and down the stairs, to evaluate the robustness under natural occlusions of the stairway. As we can observe, the values do not have strong deviation even though the model is computed with one single frame. Considering the point cloud has been downsampled with a voxel size of 4 cm , mean errors and standard deviation of less than 2 cm are within the expected margin, and in any case accurate enough for our intended task. Several frames capturing the same staircase could be potentially used to improve the retrieved dimensions, or even for an online update of these dimensions during traversal. The presence of obstacles partially occluding the view of the staircase does not adversely affect the quality of the model and we get similar results in terms of average measurements. In fact, against all odds, our experiment from Table 2.2 shows slightly better standard deviation in the presence of occluding obstacles. Nevertheless, this circumstance is coincidental and not direct consequence of our method. Some pictures of the experiments with people climbing up/down the staircase can be seen in Figure 2.24.


Figure 2.22. Step detection rate with the step position in the staircase.


Figure 2.23. Two examples of complex stairs where the modeling fails because of atypical shapes (left) or non-Manhattan directions of the edges with respect to the wall (right).


Figure 2.24. Example of a person partially blocking the view of the staircase during ascent or descent.

Table 2.2. Average and standard deviation (in centimeters) of the length and height measured with and without obstacles.

|  | No obstacles |  | Obstacles |  | Real |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  | $\overline{\mathbf{x}}$ | $\sigma$ | $\overline{\mathbf{x}}$ | $\sigma$ | $\mathbf{x}_{\mathbf{r}}$ |
| Length | 29.003 | 2.013 | 29.389 | 1.887 | 30 |
| Height | 15.399 | 1.364 | 15.561 | 0.593 | 17 |

### 2.5.3. COMPUTATION TIME ANALYSIS

The computation time was also tested to analyze the performance of the system in the current state of development in the computer described above. The four main parts in which the algorithm is divided are the Visual Odometry estimation (VO), the Segmentation and Classification (SC), the Stair Detection and Modeling (SDM) and the Stair Traversal (ST); the yellow blocks from Figure 2.1b. The VO and SC part run every iteration. When the SC raises the existence of a set of step candidates, the SDM part is executed. The ST part runs when the user is climbing the stairs, instead of SDM (they both never run in the same iteration). The VO stage has a detailed explanation about the computation time depending on the configuration used in [Gutierrez-Gomez et al., 2015]. In this implementation, we have removed the dense volumetric mapping, and we manage to estimate the visual odometry in an average time of 15.397 ms per iteration. Unlike the VO part, SC stage's runtime is scene dependent. It takes longer to compute when the scene is bigger or more complex, as both the normal estimation and regiongrowing algorithms need to iterate in a larger amount of points. The stair-related part is also scene dependent, as the SDM stage is only executed when there is stair in the image and the ST when the user is traversing. To cover all situations and provide results from this part, we have performed an experiment where the user approaches the staircase from far away (no visible stair) until he reaches the first step (during half of the time) and then move upstairs it until the other end of the stair is in sight (the other half of the time).

The following numeric results come from using a voxel grid of size 4 cm , which provides a good compromise between accuracy and speed. Excluding the VO part from the computation, each iteration takes a median time of 39 ms ( 25 Hz ), with 46 ms during the first half and 29 ms during the second half; and a maximum of 77 ms . The second half takes less time due to the simpler scene (points are close to the camera and among themselves so the voxel grid returns fewer points to deal with) and to the execution of the ST instead of the SDM algorithm (ST takes a median time of 1ms whereas SDM takes 7 ms ). In Figure 2.25 there is a box plot showing the median and quartiles of the different stages. The segmentation part appears broken down in the four biggest time consumers (voxel grid, normal estimation, region-growing and cluster extraction) discarding stages which takes less than 1 ms . Voxel grid is the slowest part but it presents lesser variability than the others, where it is more noticeable. Cluster extraction is usually small as most


Figure 2.25. Box plot of the computation time of the most relevant parts of our algorithm. For each column, the box is limited by the 25 th and 75th quartile with the median inside. The whiskers reach the most extreme points and the outliers are marked with a cross.
points of the scene belong to planes already segmented in the region-growing stage. The stair-related times are only represented when a stair is visible (otherwise time is zero). In general, this timing should be considered fast enough for indoor navigation assuming walking speeds around $1-1.5 \mathrm{~m} / \mathrm{s}$. Modern laptops or even smartphones and tablets should have nowadays enough processing power to run this system. In case it was necessary to improve the performance rates, some pre-processing parts could be optimized by using more efficient algorithms (we used some standard implementations included in widely common open source libraries) or by running them in GPU (at this point only the visual odometry takes advantage of the GPU), but the optimization of the system has not been subject of our research at this point.

### 2.5.4. Evaluation of the traversal algorithm

For the stair traversal algorithm, we have tested several video sequences with and without the drift correction implemented. A qualitative analysis consisted in visualizing the 3D mapping obtained by fusing some intermediate key frames compared to the stair template generated in the modeling. An example is shown in Figure 2.26. In (a) we can see that, although the first step matches the model perfectly in both cases, as the camera rises up the stair the 3D map diverges from the template, due to the drift. It is more prominent in the last few steps, where the drift is big enough to cause the detection of


FIGURE 2.26. Visual comparison of the 3D map composed by key frames obtained during the traversal with the stair model without (a) and with the correction (b).
one step more than the stair actually has. With the drift correction the 3D map succeed in matching the stair template, recovering the correct number of steps of the stair.

A more quantitative analysis leads to the graphs in Figures 2.27, 2.28, and 2.29. In Figure 2.27 there is the $y_{s}-z_{s}$ trajectory of the user in the first seven steps and the corresponding stair profile. The trajectories have a wave form during the traversal, where each peak corresponds with the instant the user reaches the height to walk on the following step. With the drift correction every peak is consistently paired with the edge of every step. However, without it the peak is reached increasingly farther, being more than a half step in the seventh step. In Figure 2.28 a closer look to this gap can be observed in both cases compared to the ground truth. In the seventh step the trajectory without odometry correction reaches the step with 16.6 cm of gap, whereas with the correction there is almost no deviation with respect to the expected trajectory. When the stair is large enough that gap can provoke the misdetection of more steps than the stair has. For instance, in Figure 2.26 the algorithm detects an extra 13th step non existent in the real staircase. In Figure 2.29 it is displayed the three rotation angles of the camera during the traversal, with and without the correction. As it occurs with the translation, the orientation drift also increases through iterations, but it is corrected with our approach.

### 2.6. DISCUSSION

In this chapter, we have presented a stair-aimed perception module of a wearable personal assistant oriented to visually impaired people, although it may have applications in other fields such as robotics, especially in the case of humanoids. For this we have developed an algorithm covering operations such as the detection of stairs, the retrieval of the location and measurements of the stairs and the continuous self-localization during the traversal. Our algorithm includes a visual odometry module which provides location awareness to the system, enabling the possibility of going back to places not currently visible and the traversal of staircases. Moreover, we use the information of the camera during the traversal to correct the drift that the visual odometry has. The experiments prove that the model quality and the computing time are good enough to be used in real-time. The algorithm overcomes some limitations existing in related works, such as the possibility of single step detection or full modeling with partial occlusions caused mainly by other people traversing the staircases.

In this chapter we have made the first contribution towards a computer vision based assistant for the visually impaired. We have particularly shown how effective RGB-D cameras are for this kind of tasks. In the following chapter we explore the possibilities of creating less conventional camera systems in order to increase the field of view and enable us to see all the environment at once.


Figure 2.27. Trajectory of the person during the climb with and without drift correction.


Figure 2.28. Distance in $z_{S}$ the moment the user reaches the first seven steps.


Figure 2.29. Rotation angles of the person with respect to the stair reference frame in the three stair directions.

## 3

## CALIBRATION OF HYBRID CAMERA SYSTEMS

RGB-D cameras are the main device we use to extract information from the environment, but current conventional RGB-D sensors have problems such as limited range, noise, poor resolution or small field of view. One of our endeavors during this thesis has been to enable the usage of new unconventional camera systems able to overcome some of these limitations and augment the concept of RGB-D. In this chapter, we introduce a novel hybrid camera configuration composed by a fisheye camera attached to an $R G B-D$ system, that allow us to take advantage of both devices, i.e. large field of view and $3 D$ scaled perception. To use this system we have developed a calibration procedure specifically designed for this configuration that lets the depth data be accurately mapped to the wide angle image of the fisheye camera. To overcome the lack of generality of this method, we later propose a new method that allows to estimate the relative poses between any $R G B$ and depth cameras without the requirement of an overlapping field of view, thus providing flexibility to calibrate a large variety of sensor combinations. In this approach, we extract and match lines of the scene in the $R G B$ and depth cameras, and impose geometric constraints to find the relative poses between the sensors. We have validated our method with systems as diverse as our hybrid camera system or a camera rig with 8 RGB-D cameras arranged to achieve 360degree horizontal field of view, demonstrating that our approach achieves good accuracy and is very simple to apply.


### 3.1. INTRODUCTION

The development of cheap RGB-D cameras in the consumer market has been a breath of fresh air in fields like robotics or assistive computer vision. With them it is possible to retrieve the 3D of the scene simultaneously with the RGB image, with a single device and no extra computational cost. However, the camera specifications of most of the RGBD cameras that dominate the market can fall a bit short depending on the application: limited depth range that provides unusable data in very short or far distances, noise on the depth camera, not high enough resolution or limited field of view. For example, this kind of limitations led some researchers to enhance the system with an external camera in order to get higher resolution images [Herrera et al., 2012].

In this chapter, our main concern is the small field of view (FOV). In the vast majority of vision-based applications related to mobility (e.g. autonomous driving, robotics, assistive computer vision), having a large FOV is necessary or provides important advantages [Soheilian et al., 2013, Martins et al., 2015, Perez-Yus et al., 2016b]. It is particularly interesting when the information of the sensor is an RGB-D camera, since they straightforwardly provide three-dimensional and scaled data alongside color. However, given the limitations mentioned above, emerges the idea of using more sophisticated systems, that may include other cameras, lenses, projectors, mirrors, etc. These kinds of systems, which we may refer to as unconventional, are devised so they can retrieve more information from the environment, or just different kind of information alongside each other. For example, a system that includes several RGB-D cameras with different viewpoints may be used to extend the FOV. Such system needs to be calibrated in order to fuse all the data in the same reference frame. This process is called extrinsic calibration, and consists in estimating the relative poses between the cameras.

Following this idea, we worked on the design of a new unconventional camera that would allow us to extend the FOV of an RGB-D camera. In particular, we propose to substitute the conventional RGB camera (Figure 3.1a) with a fisheye camera that has much wider FOV, as can be observed in Figure 3.1b, forming a new hybrid camera system. The images in Figure 3.1 have been taken simultaneously with the same device, consisting of a fisheye camera rigidly attached to a conventional RGB-D camera (Figure 3.1c). To our knowledge, this is the first time this configuration has been used as a single unity, since in most RGB-D cameras the FOVs of both components are intended to be as coincident as possible. However, the interest in such sensor pairing is clear in the recent Google's Tango project: the so-called motion tracking camera is indeed a wide angle camera. As a consequence of the novelty, currently there are no available tools to calibrate this hybrid camera system: fisheye cameras require the usage of different projection models due to their extremely wide FOV and the distortion of the images. Thus, existing approaches of extrinsic calibration between RGB and depth cameras are not applicable to this problem. In Section 3.3, we present a complete procedure to perform the calibration of this new device, including the intrinsic and extrinsic calibration of both


Figure 3.1. (a) Scene view from a conventional RGB-D camera. (b) Same scene view from a fisheye camera. (c) Our proposal: hybrid camera system with Depth and Fisheye. (d) After the calibration we can map the depth to the whole scene image.
fisheye and depth cameras. We present some experiments demonstrating the accuracy of the method, and showing some examples with real images where the depth has been mapped to the fisheye image (Figure 3.1d). In Chapter 4 we show an example of what can be accomplished with this system.

While our calibration approach was functional with the hybrid system proposed, the lack of applicability of the method to other camera systems encouraged us to develop other algorithm that would provide us flexibility to calibrate this and other unconventional camera systems. In Section 3.4 we propose an original method to perform extrinsic calibration of an RGB-D multi-camera system based on line observations. Our method has important advantages with respect to other approaches in the literature:

- No overlapping fields of view are required among the sensors, and thus it is perfectly suitable for extending FOV of the system.
- It can be used to calibrate different combinations of 3D range and image sensors, as long as one of them is a depth camera.
- It avoids needing to build a calibration pattern: since it is based from line observations, they can be extracted in daily life scenes from both imaging and range sensors.

We performed experiments in simulation and with real images with different camera combinations. These experiments show the validity of our method and test the accuracy and real-world usability of the approach. We demonstrate the calibration of: an RGBD sensor from a public dataset consisting on common indoor scenes, our novel hybrid system with fisheye and depth, two non-overlapping RGB-D cameras, and a rig of 8 RGB-D cameras arranged in a radial configuration for omnidirectional FOV.

Before describing our two calibration methods, in the following section we comment on the related works that led us to develop new alternatives.

### 3.2. RELATED WORK

Many calibration approaches for different type of camera systems have been proposed in the literature. For conventional cameras, the most traditional methods use the detection and matching of control points, generally with a checkerboard pattern as a calibration device [Zhang, 1999], although other proposed the usage of circular control points instead [Heikkilä, 2000]. The extrinsic calibration has been classically solved as well through the detection and matching of control points that are detected in the overlapping regions of the different cameras [Szeliski and Shum, 1997]. Line features detected by conventional cameras have also been used in a similar way to recover the essential matrices among uncalibrated cameras [Hartley, 1993], the relative poses of calibrated cameras [Lee, 2016, Přibyl et al., 2017], or the intrinsic and extrinsic parameters of a number of them [Habib et al., 2002, Zhang et al., 2016]. However, the overlap requirement constitutes a very strong constraint. Besides, even when some overlap exists, it is generally more complicated to match features in range images than in intensity images.

The interest in fusing visual information with range data has been approached for more than ten years now [Baltzakis et al., 2003]. Since the beginning, the most popular sensors were the 2D laser range finder (LRF). For instance, the work of Zhang and Pless [Zhang and Pless, 2004] presented a method of camera - 2D LRF extrinsic calibration using several views of a checkerboard (at least five) and solving a minimization problem (Figure 3.2a). Recent similar approaches such as [Vasconcelos et al., 2012,Zhou, 2014] propose minimal solutions for the same problem with outperforming results and fewer needed correspondences. Apart from the checkerboard, the use of other calibration patterns as a resource has been employed as an $a d$ hoc solution for very specific problems [Ha, 2012, Dong and Isler, 2017], like the ones shown in Figure 3.2b and Figure 3.2c. The lack of generality of these solutions for any configuration of cameras is an important limitation. Also the need to create the 3D calibration pattern itself means significant additional work.

There are some works that renounce to use calibration patterns and use common features in man-made scenes instead. For example, in that line, [Scaramuzza et al., 2007] proposed another method of extrinsic calibration of a 3D laser scanner (a 2D


Figure 3.2. Several examples of calibration of cameras to laser range finders.

LRF mounted on a rotating platform already calibrated) and any central camera (perspective or omnidirectional). Though it does not require using any calibration pattern, a set of at least four point correspondences between the two images (depth and color) must be selected manually. Other works use particular arrangements or large geometric features that occur systematically in man-made scenarios, such as perpendicular planes, or trihedrons. That is the case of [Fernandez-Moral et al., 2015], that proposes a method to calibrate several LRF from perpendicular plane observations. Gomez-Ojeda et al. [Gomez-Ojeda et al., 2015] use structural corners to perform extrinsic calibration between a 2D LRF and a camera (Figure 3.2d). Briales and Gonzalez-Jimenez proposed a similar method that produces a minimal solution instead [Briales and GonzalezJimenez, 2015]. More recently, Hu et al. [Hu et al., 2016] proposed another solution able to work with a trihedron as well, but only requiring one shot at a trihedron instead of three [Gomez-Ojeda et al., 2015].

The recent advent of consumer RGB-D cameras has caused the apparition of novel methods for that configuration. Though most drivers for these cameras include precalibrated parameters to map the information between both sensors, it is often recom-


Figure 3.3. (a) Depth camera calibration to a higher resolution color camera [Herrera et al., 2012]. (b) Single-shot calibration of camera to range sensor [Geiger et al., 2012].
mended to calibrate the RGB-D camera due to small differences in the manufacturing process [Zhang and Zhang, 2011]. To improve the quality of the color images, some researchers use high resolution external cameras. In this vein Herrera et al. [Herrera et al., 2012] proposed a method to calibrate both the intrinsic and extrinsic parameters of an RGB-D plus external camera system (Figure 3.3a). They also noted the absence in other methods of the correction of depth distortion not included in the default calibration and they proposed a method using the observed disparity instead of the metric coordinates as [Smisek et al., 2013] did. In [Geiger et al., 2012], a toolbox that calibrates both laser range sensor or a Kinect to a color camera in one single shot is presented (Figure 3.3b). Mikhelson et al. [Mikhelson et al., 2014] ease the process of calibrating the extrinsics of a depth-color camera pair by proposing an online method which removes the need to recalibrate the intrinsics once they are known.

Recently, some alternatives investigate extending the FOV of depth cameras using additional elements. For example, [Endres et al., 2014] uses two planar mirrors as a catadioptric extension of the RGB-D device to view to the front and to the back of the robot (Figure 3.4a). More generally, [Iglesias et al., 2016] proposes a framework for omnidirectional catadioptric RGB-D camera calibration. A consumer set of wide angle lens is used in [Tomari et al., 2012]. Returning to the concept of using large geometric features for extrinsic calibration, [Fernandez-Moral et al., 2014] proposed a solution based on 3D plane observations which works with non-overlapping RGB-D cameras (Figure 3.4b). This solution only requires the co-observation of planar surfaces by the different sensors and it is easy to apply, but it does not allow to include color cameras in the process. These approaches are either expensive to build, hard to calibrate [Endres et al., 2014, Iglesias et al., 2016], or do not provide good enough depth maps [Tomari et al., 2012]. There are commercial systems available, such as the Matterport camera,


Figure 3.4. (a) Catadioptric system to increase the FOV of a depth camera to look at the front and back of the robot [Endres et al., 2014]. (b) Extrinsic calibration of depth cameras with planar observations [FernandezMoral et al., 2014].
that have been used to create the large scale indoor dataset from [Armeni et al., 2017]. However, this camera system is too expensive and unpractical to be used in situations that require mobility.

A more general approach not depending on the geometric configuration of the sensors is based on ego-motion to match the camera trajectories, which are tracked independently. For that, simultaneous localization and mapping (SLAM) or visual odometry (VO) techniques are applied [Brookshire and Teller, 2013, Heng et al., 2013, Schneider et al., 2013]. However, this kind of solution is laborious to apply, requiring robust SLAM or VO in controlled environments. Besides, they may not be able to fully observe the calibration parameters depending on the movement restrictions, e.g. the translation in the vertical axis is unobservable in the case of planar motion, so common for a wheeled robot or autonomous vehicle.

### 3.3. A NOVEL HYBRID CAMERA SYSTEM WITH DEPTH AND FISHEYE

In this section we introduce a new camera system that includes a fisheye and a depth camera. In order to fuse the information coming from both cameras it is necessary to perform the calibration of the system. Given the absence of methods available for such unconventional system, we developed a suitable method. In particular, using an explicitly designed camera model to calibrate the fisheye camera alone is an important prerequisite. We propose using the Scaramuzza's omnidirectional calibration method to this end [Scaramuzza et al., 2006]. To calibrate the whole system we propose and evaluate two alternative methods. The first consists in calibrating the intrinsics of both cameras separately, compute the extrinsics and finally perform the calibration of the
depth measurements and distortion. In the second only the fisheye is required to be calibrated on its own, and the depth camera is jointly calibrated with the relative pose of both devices with a non-linear minimization of the reprojection error in both fisheye and depth images. In Section 3.3.1 we describe the system and camera models and in Section 3.3.2 the calibration procedure. Additionally, in Section 3.3.3 we present results comparing both methods and showing their accuracy on real images.

### 3.3.1. SYSTEM DESCRIPTION

We have created a hybrid camera system by rigidly coupling a high resolution camera with a fisheye lens to an Asus Xtion Pro Live RGB-D sensor (Figure 3.1c). The difference in field of view is large, as Figure 3.1 shows. On the one hand, the FOV of the RGB-D camera may be too small for many applications, especially in close distances. Several works have shown the advantages of wide field of view (or omnidirectional) cameras in robotics and computer vision applications [Rituerto et al., 2010, BermudezCameo et al., 2014]. On the other hand, the depth perception can help detecting obstacles, providing scale or enhancing the recognition at least in one portion of the scene. Next we describe the camera models used in this work before moving to the calibration.

### 3.3.1.1. FISHEYE CAMERA MODEL

We choose the parametric camera model described by Scaramuzza et al. in [Scaramuzza et al., 2006], which considers the omnidirectional image as a highly distorted image. The calibration consists in retrieving the parameters of the polynomial that describes this distortion. With this model it is not necessary to provide a specific model of the sensor and works with all kind of projective, catadioptric or dioptric cameras. Although in this work we focus on fisheye cameras, the usage of this model makes our approach valid for the other types of cameras.

Using that model, the world points $\mathbf{X}^{F}=(X, Y, Z)$ have the origin of coordinates in the optical center of the camera $O^{F}$, where the coordinate system have the $z^{F}$ component following the axis of the (cata) dioptric system (Figure 3.5). Orthogonal to the $z^{F}$ axis it is the sensor plane $\left(x_{s}, y_{s}\right)$, a theoretical plane where the coordinates are still metrical. The images are represented in the image plane $\mathbf{u}=(u, v)$, where the position of the points is expressed in pixels. It is assumed that there is misalignment and deformation between the image plane and the sensor plane, given by an affine transformation $\mathbf{x}_{s}=\mathbf{A} \mathbf{u}+\mathbf{t}$, where $\mathbf{t}=\left(u_{0}, v_{0}\right)$ is the image center. The vector $\mathbf{p}$ pointing at the world point $\mathbf{X}$ from $O^{F}$ follows the equation:

$$
\lambda \cdot \mathbf{p}=\lambda \mathbf{g}(\mathbf{A} \mathbf{u}+\mathbf{t})=\mathbf{P} \cdot \mathbf{X}, \quad \lambda>0
$$

where $\mathbf{P}$ is the perspective projection matrix and the function $\mathbf{g}\left(x_{s}, y_{s}\right)$ is defined as follows:

$$
\begin{equation*}
\mathbf{g}\left(x_{s}, y_{s}\right)=\left(x_{s}, y_{s}, f\left(x_{s}, y_{s}\right)\right)^{\top}=\left(x_{s}, y_{s}, f\left(\rho_{s}\right)\right)^{\top} \tag{3.2.2}
\end{equation*}
$$



Figure 3.5. Coordinate systems, parameters of the cameras and its interaction with the world. In red the parameters to calibrate.

$$
\begin{equation*}
f\left(x_{s}, y_{s}\right)=a_{0}+a_{2} \rho_{s}^{2}+\ldots+a_{N} \rho_{s}^{N} \tag{3.2b}
\end{equation*}
$$

where the function $f$ is modeled as a Taylor expansion defined by the polynomial whose coefficients are $a_{0}, a_{2}, \ldots, a_{N}$, and where $\rho_{s}=\sqrt{x_{s}^{2}+y_{s}^{2}}$. These coefficients along with the matrix $\mathbf{A}$ and vector $t$ are the parameters needed to calibrate the intrinsics.

### 3.3.1.2. DEPTH CAMERA MODEL

The proposed intrinsic model of the depth camera is based in [Herrera et al., 2012]. Basically, this is a standard projective camera model (as the depth camera works with a conventional IR camera) with radial and tangential distortion correction. The images captured by the depth sensor have in every pixel a value of disparity (in disparity units, $d u)$ which increases with depth. Some drivers provide an automatic conversion of these values to metric distances given an internal calibration of the camera during the manufacturing process. However, to increase the accuracy of the measurements, we do not only compute the parameters mapping these pixels to the real world, but also recover the metric distances per-pixel.

The IR camera model, based on [Heikkilä, 2000], transforms coordinates from the Image Plane to the Sensor Plane via the focal distances $f_{x}^{D}$ and $f_{y}^{D}$, the position of the center $C_{I}^{D}=\left(u_{0}^{D}, v_{0}^{D}\right)$ and some distortion coefficients $k_{D}=\left\{k_{1}, k_{2}, k_{3}, k_{4}, k_{5}\right\}$
(Figure 3.5). If we have a metric value of depth per-pixel $z^{D}$, the metric 3D coordinates of the point $\mathbf{X}^{D}=\left(X^{D}, Y^{D}, Z^{D}\right)$ in the world with respect to the optical center $O^{D}$ are:

$$
\begin{equation*}
\mathbf{X}^{D}=\left(X^{D}, Y^{D}, Z^{D}\right)=\left(x_{s}^{D} z^{D}, y_{s}^{D} z^{D}, z^{D}\right) \tag{3.3}
\end{equation*}
$$

However, we must transform the disparity values of the images received from the camera to the metric values, which is done in two phases. First, in [Herrera et al., 2012] it is mentioned the existence of a fixed error pattern that distorts the depth image with a per-pixel offset which is obtained from the intrinsic calibration. In the case of the sensor we use, it follows the function:

$$
\begin{equation*}
d_{u}=d_{d}+\mathbf{D}_{\delta}\left(u_{d}, v_{d}\right) \cdot \exp \left(\alpha_{0}-\alpha_{1} d_{d}\right) \tag{3.4}
\end{equation*}
$$

where $d_{u}$ is the resulting undistorted disparity and $d_{d}$ the distorted disparity. Second, to convert the values of the pixels given in disparity units ( $d u$ ) to metric units, it is necessary to get the following coefficients $c_{1}$ and $c_{0}$ which forms the following equation:

$$
\begin{equation*}
z^{D}=\frac{1}{c_{1} \cdot d_{u}+c_{0}} \tag{3.5}
\end{equation*}
$$

The set of parameters obtained from the intrinsic calibration of the depth camera are: the focal lengths $\left(f_{x}^{D}, f_{y}^{D}\right)$, the image center $\left(u_{0}^{D}, v_{0}^{D}\right)$, the distortion coefficients $k_{D}$, the disparity-depth transformation coefficients $\left(c_{1}, c_{0}\right)$, the matrix $\mathbf{D}_{\delta}\left(u_{d}^{D}, v_{d}^{D}\right)$ and the exponential decay parameters $\left(\alpha_{0}, \alpha_{1}\right)$.

### 3.3.2. DEPTH-FISHEYE CAMERA CALIBRATION

Our algorithm needs a checkerboard on a flat surface as the main calibration pattern. The calibration requires the capture of several images of the planar checkerboard from different points of view, watching carefully that both the checkerboard in the external camera and the planar surface supporting the checkerboard in the depth image are observed at the same time. That is not a trivial issue: to calibrate properly the intrinsic parameters of the fisheye camera it is necessary to fill the fisheye images with the checkerboard as much as possible (e.g. Figure 3.6a) but doing that requires to pose the camera too close to the board and the depth sensor cannot retrieve information. Placing the camera pair at a reasonable distance makes the system prone to fail in the estimation of the intrinsic parameters of the fisheye camera because the distortion cannot be properly perceived (Figure 3.6b).

To handle this situation, the intrinsic calibration of the fisheye is performed separately from the rest of the algorithm, with their own set of close-range images (similar to Figure 3.6a). The fisheye camera can be easily calibrated by using the toolbox released by [Scaramuzza et al., 2006]. For the rest of the process, an alternative set of mid and long-range images have been collected. The dimensions of the pattern must be large enough that the corners of the checkerboard can be detected in the fisheye image taking


Figure 3.6. Example of images from the sets of fisheye intrinsic calibration (a) and depth intrinsic/extrinsic calibration (b).
into account the resolution of the camera. In our case, we used a DIN-A2 checkerboard. The dimensions are known and used as input for the algorithm.

With this setup we have tackled this calibration problem from two alternative approaches:
A) Calibrating the intrinsic parameters of the fisheye and then performing a joint calibration of the rest of parameters involved in the system all at once.
B) Separating the process in stages, performing first the intrinsic calibration of the cameras separately, then retrieve the extrinsic parameters and finally compute the distortion correction and conversion from disparity to metric units.

The two alternate methods to perform the calibration of our system are described in the following sections.

### 3.3.2.1. JOINT CALIBRATION OF FISHEYE AND DEPTH CAMERAS

The first calibration methods for structured light-based RGB-D systems used images from the IR camera to perform traditional extrinsic calibration [Zhang, 1999]. However, as [Herrera et al., 2012] pointed out, the depth images are not perfectly aligned with the IR images. The work [Herrera et al., 2012] uses disparity images to calibrate the
intrinsics of the depth camera as well as the extrinsics of the system. Our method is inspired by this approach, introducing some modifications to make it suitable for our system.

We assume that the checkerboard is on a planar surface to relate in each image pair the pose of the fisheye camera from the checkerboard to its plane equation. As the checkerboard is not observed by the depth camera, we manually select the vertices of the polygon which contains the disparity pixel values of the board in each image. To relate the pose of the checkerboard as seen by the fisheye camera to the depth measurements, we need a rough estimation of the depth camera parameters. When the board has known shape, e.g. rectangular, an initial estimate can be obtained selecting the corners from the board and using homographies as explained in [Zhang and Pless, 2004]. The calibration of the final values of the depth intrinsics will be done jointly with the extrinsics solving the minimization problem, so the values set at this point facilitate the convergence. In the following sections the main parts of the method are detailed:

### 3.3.2.1.1. RETRIEVING CAMERA POSE FROM FISHEYE IMAGES

This section describes how, for each calibration image $I_{i}$, we obtain the pose of the pattern with respect to the fisheye camera ${ }^{F} \mathbf{T}_{I_{i}} \in \mathbb{R}^{3 \times 4}$ (Figure 3.5). It is formed by the rotation matrix ${ }^{F} \mathbf{R}_{I_{i}} \in \mathbb{R}^{3 \times 3}=\left[\mathbf{r}_{1}, \mathbf{r}_{2}, \mathbf{r}_{3}\right]$ and the translation vector ${ }^{F} \mathbf{t}_{I_{i}} \in$ $\mathbb{R}^{3 \times 1}$. Calling $\mathbf{X}_{i j}=\left[X_{i j}, Y_{i j}, Z_{i j}\right]$ the 3D coordinate of every of the $j$ points from the pattern in the pattern coordinate system and $\mathbf{u}_{i j}=\left[u_{i j}, v_{i j}\right]^{\top}$ the correspondent pixel coordinates in the Image Plane, from (3.1) we get the following equation:

$$
\lambda_{i j} \cdot \mathbf{u}_{i j}=\lambda_{i j} \cdot\left[\begin{array}{c}
u_{i j}  \tag{3.6}\\
v_{i j} \\
a_{0}+\ldots+a_{N} \rho_{i j}^{N}
\end{array}\right]=\left[\mathbf{r}_{1}^{i}, \mathbf{r}_{2}^{i}, \mathbf{r}_{3}^{i}, \mathbf{t}^{i}\right] \cdot\left[\begin{array}{c}
X_{i j} \\
Y_{i j} \\
Z_{i j} \\
1
\end{array}\right]
$$

We can assume without loss of generality that $Z_{i j}=0$ for all points in the pattern, as they all belong to a planar surface in $X_{i j}-Y_{i j}$. If we multiply both sides vectorially by $\mathbf{u}_{i j}$ :

$$
\lambda_{i j} \cdot \mathbf{u}_{i j} \wedge \mathbf{u}_{i j}=\left[\begin{array}{c}
u_{i j}  \tag{3.7}\\
v_{i j} \\
a_{0}+\ldots+a_{N} \rho_{i j}^{N}
\end{array}\right] \wedge\left[\mathbf{r}_{1}^{i}, \mathbf{r}_{2}^{i}, \mathbf{t}^{i}\right] \cdot\left[\begin{array}{c}
X_{i j} \\
Y_{i j} \\
1
\end{array}\right]=0
$$

Which gives the following system for every image $I_{i}$ :

$$
\begin{gather*}
v_{j} \cdot\left(r_{31} X_{j}+r_{32} Y_{j}+t_{3}\right)-f\left(\rho_{j}\right) \cdot\left(r_{21} X_{j}+r_{22} Y_{j}+t_{2}\right)=0  \tag{3.8a}\\
f\left(\rho_{j}\right) \cdot\left(r_{11} X_{j}+r_{12} Y_{j}+t_{1}\right)-u_{j} \cdot\left(r_{31} X_{j}+r_{32} Y_{j}+t_{3}\right)=0  \tag{3.8b}\\
u_{j} \cdot\left(r_{21} X_{j}+r_{22} Y_{j}+t_{2}\right)-v_{j} \cdot\left(r_{11} X_{j}+r_{12} Y_{j}+t_{1}\right)=0 \tag{3.8c}
\end{gather*}
$$

In these equations we know the position of the points in both the world and the image ( $X_{j}, Y_{j}, u_{j}, v_{j}$ ) and the function $f(\rho)$ from the previous calibration of the fisheye camera. Having these three equations for each of the $n$ points $\mathbf{u}_{j}$ in the pattern we need at least $n=3$ points to solve the system. The checkerboards always contain more than three points, which generates an overdetermined system. To solve this system we reformulate it as follows:

$$
\mathbf{M} \cdot \mathbf{H}=\left[\begin{array}{c}
\mathbf{M}_{1}  \tag{3.9}\\
\ddot{\mathbf{M}_{j}} \\
\ddot{\mathbf{M}_{L}}
\end{array}\right] \cdot \mathbf{H}=0
$$

where

$$
\begin{align*}
\mathbf{H} & =\left[r_{11}, r_{12}, r_{21}, r_{22}, r_{31}, r_{32}, t_{1}, t_{2}, t_{3}\right]^{\top}  \tag{3.10a}\\
\mathbf{M}_{j} & =\left[\begin{array}{ccc}
0 & f\left(\rho_{j}\right) X_{j} & -v_{j} X_{j} \\
0 & f\left(\rho_{j}\right) Y_{j} & -v_{j} Y_{j} \\
-f\left(\rho_{j}\right) X_{j} & 0 & u_{j} X_{j} \\
-f\left(\rho_{j}\right) Y_{j} & 0 & u_{j} Y_{j} \\
v_{j} X_{j} & -u_{j} X_{j} & 0 \\
v_{j} Y_{j} & -u_{j} Y_{j} & 0 \\
0 & f\left(\rho_{j}\right) & -v_{j} \\
-f\left(\rho_{j}\right) & 0 & u_{j} \\
v_{j} & -u_{j} & 0
\end{array}\right] \tag{3.10b}
\end{align*}
$$

whose solution can be computed using SVD. The rest of the elements of the matrix $\mathbf{R}_{i}$, i.e. $\mathbf{r}_{3}=\left[r_{13}, r_{23}, r_{33}\right]$ are obtained with the cross product of $\mathbf{r}_{1}$ and $\mathbf{r}_{2}$. This is repeated for each image $I_{i}$, computing all the transformations ${ }^{F} \mathbf{T}_{I_{i}}$.

### 3.3.2.1.2. NON-LINEAR MINIMIZATION

The optimization is a minimization of the weighted sum of squares of the reprojection errors over all parameters, which we perform using a cost function similar to the one from [Herrera et al., 2012]. The cost function has two terms, one for each camera. The costs of the fisheye camera are the sum of the Reprojection Error $R E^{F}$, defined as the Euclidean distance between the pixel position of the corner $\mathbf{p}_{i j}$, and the position of its reprojection $\hat{\mathbf{p}}_{i j}$ given the pose of the checkerboard in every image ${ }^{F} \mathbf{T}_{I_{i}}$ and the intrinsic parameters of the fisheye:

$$
\begin{equation*}
R E_{i j}^{F}=\left\|\hat{\mathbf{p}}_{i j}-\mathbf{p}_{i j}\right\|_{2} \tag{3.11}
\end{equation*}
$$

which is measured in pixels. Although the intrinsic parameters of the camera are precomputed, it is necessary to include this error because of the optimization of the poses ${ }^{F} \mathbf{T}_{I_{i}}$. The costs of the depth camera are the sum of the depth Reprojection Error $R E^{D}$, defined as the L2-norm of the difference between the measured disparity from the image $d_{d}$ and its predicted disparity $\hat{d}_{d}$ for every board pixel $k$ in image $i$ :

$$
\begin{equation*}
R E_{i k}^{D}=\left\|\hat{d}_{i k}-d_{i k}\right\|_{2} \tag{3.12}
\end{equation*}
$$

which is measured in disparity units (du). To compute the predicted disparity, we know from Section 3.3.2.1.1 the pose of the camera with respect to a group of points lying on a plane for every fisheye image $I_{i}$. Introducing the extrinsic transformation ${ }^{D} \mathbf{T}_{F}$, we get the poses of the points with respect to the depth camera, ${ }^{D} \mathbf{T}_{I_{i}}={ }^{D} \mathbf{T}_{F} .{ }^{F} \mathbf{T}_{I_{i}}$, and therefore the plane the checkerboard is on. The normal $\mathbf{n}_{i}$ and the distance to the origin $b_{i}$ are:

$$
\begin{align*}
\mathbf{n}_{i} & =\mathbf{r}_{3}^{i}  \tag{3.13}\\
b_{i} & =\mathbf{r}_{3}^{i}{ }^{\top} \mathbf{t}^{i} \tag{3.14}
\end{align*}
$$

where $\mathbf{r}_{3}^{i}$ and $\mathbf{t}^{i}$ are the third column of the rotation matrix ${ }^{D} \mathbf{R}_{I_{i}}$ and the translation vector ${ }^{D} \mathbf{t}_{I_{i}}$ respectively. The plane equation from the fisheye camera frame is then:

$$
\begin{equation*}
\mathbf{n}_{\mathbf{i}}^{\top} \mathbf{X}-b_{i}=0 \tag{3.1}
\end{equation*}
$$

for each image. With the rough initial estimate of the parameters we can compute the predicted disparity and therefore the depth cost. To avoid including the whole matrix $\mathbf{D}_{\delta}\left(u_{d}, v_{d}\right)$ which adds many parameters to the optimization (in our case $640 \times 480$ ), it is optimized independently as done in [Herrera et al., 2012]. Because of that, the depth cost is computed with the disparity values from the Undistorted Image Plane.

The main cost function is the following, weighted by the inverse of the measurement variance due to the difference in the units of the terms:

$$
\begin{equation*}
J=\beta \frac{\sum R E_{i j}^{F}}{\sigma_{F}^{2}}+\frac{\sum R E_{i k}^{D}}{\sigma_{D}^{2}} \tag{3.16}
\end{equation*}
$$

where $\beta$ is an additional weighting factor to give equal importance to both terms regardless of the number of points in each of them. The optimization has two phases: First, the main minimization, with the cost function from (3.16), where the parameters to minimize are ${ }^{F} \mathbf{T}_{I_{i}},{ }^{D} \mathbf{T}_{F}, c_{1}, c_{0}, f_{x}^{D}, f_{y}^{D}, \mathbf{C}_{I}^{D}$ and $k_{D}$. Second, minimization of the distortion offset given by [Herrera et al., 2012], where the parameters to minimize are $\alpha_{0}, \alpha_{1}$ and $\mathbf{D}_{\delta}(u, v)$. We have used an iterative non-linear minimization using the Levenberg-Marquardt algorithm for both processes.

### 3.3.2.2. STEPWISE CALIBRATION OF FISHEYE AND DEPTH CAMERAS

The idea behind this approach is to separate the calibration of the whole system in stages instead of performing a single global optimization of the parameters. Herrera et al. [Herrera et al., 2012] mention the improvement of the results in the calibration when both the intrinsic and extrinsic parameters are optimized simultaneously. However, the optimization may get stuck in a local minimum if the estimation of the seed values of some parameters is not good enough. We propose to perform following stages instead:

- Intrinsic calibration of the fisheye $(f(\rho), \mathbf{A}$ and $\mathbf{t})$.
- Intrinsic calibration of the IR camera ( $\mathbf{f}^{D}, \mathbf{C}_{I}^{D}$ and $\mathbf{k}_{D}$ ), which can be solved using standard methods [Zhang, 1999].
- Obtain the poses of the checkerboard with respect to the cameras ( ${ }^{F} \mathbf{T}_{I_{i}},{ }^{D} \mathbf{T}_{I_{i}}$ ) and with them, the extrinsic calibration $\left({ }^{D} \mathbf{T}_{F}\right)$.
- Global optimization to compute disparity-depth correction parameters $\left(c_{1}, c_{0}\right.$, $\mathbf{D}_{\delta}\left(u_{d}^{D}, v_{d}^{D}\right), \alpha_{0}$ and $\left.\alpha_{1}\right)$ and refine the previously computed parameters.

The stages not included in the description of the first approach are detailed as follows:

### 3.3.2.2.1. Computation of the extrinsics of the system

Having computed the relative poses of the checkerboard from both cameras ( ${ }^{F} \mathbf{T}_{I_{i}}$ from Section 3.3.2.1.1 and ${ }^{D} \mathbf{T}_{I_{i}}$ from [Zhang, 1999]), for each image pair we have:

$$
\begin{gather*}
{ }^{D} \mathbf{R}_{F}^{\left(I_{i}\right)}={ }^{D} \mathbf{R}_{I_{i}} \cdot{ }^{I_{i}} \mathbf{R}_{F}  \tag{3.17a}\\
{ }^{D} \mathbf{t}_{F}^{\left(I_{i}\right)}={ }^{D} \mathbf{t}_{I_{i}}-{ }^{D} \mathbf{R}_{F}^{\left(I_{i}\right)} \cdot{ }^{F} \mathbf{t}_{I_{i}} \tag{3.17b}
\end{gather*}
$$

Averaging the rotations (turned into rotation vectors) and the translations provides a good estimate of the extrinsics of the system. To refine the extrinsic parameters (6 DOF), we can minimize the reprojection error of the corner points from one reference frame to the other backprojected to the image ( $\hat{\mathbf{p}}_{i j}$ ) with respect to the measured point ( $\mathbf{p}_{i j}$ ), respectively:

$$
\begin{equation*}
\arg \min \left\|\hat{\mathbf{p}}_{i j}^{F}-\mathbf{p}_{i j}^{F}\right\|+\left\|\hat{\mathbf{p}}_{i j}^{D}-\mathbf{p}_{i j}^{D}\right\| \tag{3.18}
\end{equation*}
$$

where $\hat{\mathbf{p}}_{i j}^{c_{a}}=\operatorname{proj}\left({ }^{c_{b}} \mathbf{T}_{c_{a}} \cdot \mathbf{X}_{i j}^{c_{b}}\right)$, proj the projection function (which changes depending on the type of camera) and $c_{a}, c_{b}$ note two different camera frames (in this case, $F$ and $D$ ).

### 3.3.2.2.2. Global optimization

The global optimization proposed in this approach has the same formulation (Section 3.3.2.1.2). In this case, we do not need to perform the calibration of the intrinsics of the IR camera and the extrinsics of the system. These parameters can be fixed and ignored in the optimization process, and only estimate the values of $c_{1}, c_{0}, \mathbf{D}_{\delta}\left(u_{d}^{D}, v_{d}^{D}\right), \alpha_{0}$ and $\alpha_{1}$. However, if we include all the parameters as in Section 3.3.2.1.2 we can use this optimization as a global refinement of the already estimated parameters, taking into account the depth image instead of the IR image.

### 3.3.3. EXPERIMENTS

In the experiments, we use the proposed hybrid camera system shown in Figure 3.1b: The RGB-D sensor is the Asus Xtion Live Pro, which provides a depth image resolution of $640 \times 480$ pixels at 30 Hz . The fisheye camera is a uEye UI-3580CP of $2560 \times 1920$

TABLE 3.1. MRE in the fisheye and depth image for both sets of images $A$ and $B$ with four calibration results.

|  | Evaluated Set A |  | Evaluated Set B |  |
| :---: | :---: | :---: | :---: | :---: |
|  | Fisheye (px) | Depth (du) | Fisheye (px) | Depth (du) |
| 1) Joint calib. A | 0.185 | 0.839 | 0.187 | 1.477 |
| 2) Step calib. A | 0.185 | 0.806 | 0.187 | 1.514 |
| 3) Joint calib. B | 0.165 | 1.322 | 0.183 | 0.856 |
| 4) Step calib. B | 0.165 | 1.321 | 0.182 | 0.835 |

pixels at 15 Hz with a fisheye lens Lensagon CF5M1414, with a field of view of $182^{\circ}$. The images have been captured synchronized using Robot Operating System (ROS). The calibration pattern is a $9 \times 7$ checkerboard printed in a DIN-A2 sized paper attached to a rectangular piece of wood. A set of fisheye images without depth information has been used to calibrate the intrinsics of the fisheye, resulting in a mean reprojection error of less than 0.2 pixels, which can be considered an accurate calibration for such high resolution.

For the depth intrinsics and camera pair extrinsic calibration we used two sets of images for comparison purposes. Set A has 25 image pairs and set B has 28. Both datasets are similar in terms of variability of poses. We use the Mean Reprojection Error (MRE) as quantifiable parameter to evaluate the results, defined by the arithmetic mean of (3.11) and (3.12). In Table 3.1 it is shown the MRE for both sets of images using the results from four calibration procedures:

1) Joint Calibration of Set A (Section 3.3.2.1).
2) Stepwise Calibration of Set A (Section 3.3.2.2).
3) Joint Calibration of Set $B$ (Section 3.3.2.1).
4) Stepwise Calibration of Set B (Section 3.3.2.2).

The MRE for each set with its own calibration results is, in the fisheye camera, less than 0.2 pixels, whereas in the depth camera it is less than 1du. Using the calibration values of extrinsics and depth intrinsics from the other set, the MRE of the depth increases to 1.5 du . These values can be considered highly satisfactory considering the complexity of calibrating two cameras of such different kind. We can see how the difference among the methods is marginal, making both approaches equally valid for the task. The stepwise calibration would be preferable if the IR images are accessible as it is less prone to fall in local minimum. However, the joint calibration procedure has less steps requiring human supervision.

To measure the quality of the calibration we have also used a 3D pattern. It consists of three metal plates screwed and secured at $90^{\circ}$ with calibration patterns attached to them (Figure 3.7a). We have accurate ground truth of the 3D position of the points of these patterns from a photogrammetric reconstruction by bundle adjustment. Obtaining


Figure 3.7. (a) 3D pattern consisting of three orthogonal planes. (b) Point cloud obtained using the calibration.

Table 3.2. Angular difference between planes with respect to the ground truth in the fisheye and depth image.

|  | Fisheye Image | Depth Image |
| :---: | :---: | :---: |
| $\alpha_{12}(\mathrm{deg})$ | -0.3559 | 1.3715 |
| $\alpha_{23}(\mathrm{deg})$ | 1.7045 | -0.3339 |
| $\alpha_{31}(\mathrm{deg})$ | 1.2229 | 0.1908 |
| Mean $(\mathrm{deg})$ | 1.0944 | 0.6321 |

the pose of the camera from the fisheye image and computing the planes from the depth image we can compare the angles between the metal plates to see how good the calibration is. Table 3.2 shows the results of the experiment, with an error of $\approx 1^{\circ}$ in the fisheye image and $\approx 0.6^{\circ}$ in the depth image. The quality of the mapping of the depth information can also be qualitatively analyzed superposing the depth maps in the fisheye image (Figure 3.8). It can be observed how the borders in the depth coincide with the borders in the image, even at large distances. It is also possible to reconstruct the point cloud with this data, where the high accuracy can also be appreciated (Figure 3.7b).

An application of the calibrated system is shown in Chapter 4, where the depth information is extended to the whole fisheye image via layout estimation.


Figure 3.8. Semi-transparent depth map superimposed over the fisheye image to illustrate the quality of the calibration.

### 3.4. EXTRINSIC CALIBRATION OF MULTIPLE RGB-D CAMERAS FROM LINES

In this section, we propose a method for flexible extrinsic calibration of RGB-D cameras. Current solutions for this problem are time consuming and/or may require building a specific calibration pattern. Besides, there are some additional challenges, such as trying to combine different type of cameras, or finding the calibration when the cameras have no overlap in their FOV. The approach presented here is inspired by the kind of solutions that use common features in man-made scenes, such as planes [Fernandez-Moral et al., 2014], or corners [Gomez-Ojeda et al., 2015]. In particular, it is based on the observation and matching of lines in the scene from the different sensors, which are used to formulate constraints on the relative poses of the cameras. Our method allows to calibrate any system with:

1. One sensor able to extract the parameters to completely define a line in 3D space (e.g. a depth camera).
2. One (or several) sensors able to extract the lines in the projective plane (e.g. a standard camera, other depth camera).

Thus, our method calibrates Color-Depth pairs $\{C, D\}$, Depth-Depth pairs $\left\{D_{1}, D_{2}\right\}$ and larger systems with $N^{C}$ color cameras and $N^{D}$ depth cameras $\left\{D_{1} . . D_{N^{D}}, C_{1} . . C_{N^{C}}\right\}$ whenever $N^{D} \geq 1$ and $N^{D}+N^{C} \geq 2$.

Before the calibration we describe our line extraction process in color images (from both conventional and omnidirectional cameras) and from range data (Section 3.4.1). In particular, from range data the lines are found as plane intersections. RGB-D sensors are a special case where we can use line extraction in the RGB image, and then use the depth to get the 3D line. This situation is illustrated in Figure 3.9, where the line with sub-index 1 corresponds to the intersection of two planar surfaces ( $\Pi_{a}$ and $\Pi_{b}$ ) and thus, its 3D parameters are observable by a depth camera, while the 3D parameters of the line with sub-index 2 (which is contained in the plane $\Pi_{a}$ ) are only observable by an RGB sensor. After line extraction, we propose a robust method to find line matchings via a RANSAC approach.

The main contribution of this work is a novel method for extrinsic calibration of an RGB-D multi-camera system based on line observations, introduced in Section 3.4.2. Our method has important advantages with respect to other approaches in the literature: $i$ ) no overlapping fields of view are required among the sensors; $i i$ ) it can be used to calibrate different types of cameras; and iii) it avoids needing to build a calibration pattern. We have additionally included an analysis of the observability of the problem, where we discuss the minimum amount of line-matchings necessary and degenerate cases (Section 3.4.3). We performed experiments in simulation and with real images with


Figure 3.9. Observation of lines in the scene by a pair of RGB-D and conventional cameras rigidly joined with non-overlapping field of view. Line correspondences are used to formulate geometric restrictions to compute the relative pose $T$ between the cameras.
different camera combinations (Section 3.4.4). These experiments show the validity of our method and test the accuracy and real-world usability of the approach. We demonstrate the calibration of: an RGB-D sensor from a public dataset consisting on common indoor scenes; a fisheye camera and a depth camera; two non-overlapping RGB-D cameras; and a rig of 8 RGB-D cameras arranged in a radial configuration for omnidirectional FOV.

### 3.4.1. LINE EXTRACTION AND MATCHING

We use line to refer to the line in 3D space, and segment to refer to the set of collinear points found in a conventional image. Mathematically, a line is the set of points $\mathbf{p} \in \mathbb{R}^{3}$ that satisfy the following equation:

$$
\begin{equation*}
\mathbf{p}=\mathbf{p}_{0}+\lambda \mathbf{v}=\left(p_{0 x}, p_{0 y}, p_{0 z}\right)+\lambda\left(v_{x}, v_{y}, v_{z}\right), \quad \forall \lambda \in \mathbb{R} \tag{3.19}
\end{equation*}
$$

being $\mathbf{p}_{0} \in \mathbb{R}^{3}$ a point in the line, and $\mathbf{v} \in \mathbb{R}^{3}$ the direction vector of the line (see Figure 3.9). We also define the projective plane of the line $\pi$ as the 3D plane that contains the line and the origin of the reference system (i.e. the optical center of the camera). The normal $\mathbf{n} \in \mathbb{R}^{3}$ of this plane (see Figure 3.9), also known as the moment vector of the line, is the vector perpendicular to $\mathbf{p}_{0}$ and $\mathbf{v}$, i.e. $\mathbf{n}=\mathbf{p}_{0} \times \mathbf{v}$. In the following sections we describe how we extract lines in an image depending on the type of camera used.


Figure 3.10. Examples of common indoor scenes used for calibration, like wall-wall and wall-ceil junctions. The first row shows the RGB images with the lines extracted in green. The second row shows the corresponding depth images with the planar surfaces colored in different colors. Images (1-4) belong to the NYU2 RGB-D dataset [Silberman et al., 2012].

### 3.4.1.1. LINE EXTRACTION IN RGB CAMERA

Due to the projective nature of conventional cameras we cannot compute the direction vector $\mathbf{v}$, nor any 3 D point $\mathbf{p}$. Nonetheless, we can extract segments in the image to retrieve the normal vectors $\mathbf{n}$ of their projective planes. For this, the camera must be intrinsically calibrated in advance. The process of getting segments is a traditional problem in computer vision, which can be solved using widespread algorithms. In particular, our approach goes as follows:

1. Apply a Canny filter [Canny, 1986] to extract edges in the intensity image.
2. The edge points are grouped in boundaries formed by consecutive points in the image.
3. For each boundary, we apply a RANSAC procedure [Fischler and Bolles, 1981] to get the lines in 2D. The 2D lines have a direction vector in the image $1=$ $\left[l_{x}, l_{y}, 0\right]^{\top}$ and a set of $k$ inlier edge points $\left\{\mathbf{u}_{1} . . \mathbf{u}_{k}\right\}$ where $\mathbf{u}_{i}=\left(u_{i}, v_{i}\right)$.
4. The mean of the inlier points $\overline{\mathbf{u}}$ is used to compute the 3D ray:

$$
\begin{equation*}
\mathbf{r}=\left[\left(\bar{u}-c_{x}\right) / f_{x},\left(\bar{v}-c_{y}\right) / f_{y}, 1\right]^{\top} \tag{3.20}
\end{equation*}
$$

with camera's optical center $\left(c_{x}, c_{y}\right)$ and focal length $\left(f_{x}, f_{y}\right)$.
5. The normal $\mathbf{n}$ is $\mathbf{n}=\mathbf{l} \times \mathbf{r}_{i}$

Some examples of line extraction are shown in Figure 3.10. While this approach is appropriate for standard cameras, we can substitute this method to a more suitable one if we need to use a more complex type of camera. The work from Bermudez-Cameo


Figure 3.11. (a) Lines extracted in a fisheye image (in green). The two relevant ones are selected in blue. (b) View from the RGB-D camera. Top: the RGB image (not used for calibration). Bottom: the planar segmentation in the depth image used to extract 3D line from plane intersections. (c) 3D planes projected to the fisheye view with the corresponding lines in blue (from RGB) and red (from depth). (d) After calibration, the projection of the 3D lines and the depth map fits the color image.
et al. [Bermudez-Cameo et al., 2015] presents a line extraction method for uncalibrated omnidirectional cameras with revolution symmetry. With this method we can calibrate a wide range of omnidirectional cameras like the fisheye shown in Figure 3.11a.

### 3.4.1.2. LINE EXTRACTION IN DEPTH CAMERA

A depth camera permits to obtain the 3D line parameters (i.e. we can extract $\mathbf{p}, \mathbf{v}$ and $\mathbf{n}$ from the lines in the image). The strategy to obtain the lines may depend on the sensor. Using only depth information, the simplest way to retrieve 3D lines is by looking for 3D plane intersections. For example, in Figure 3.9, the line $\left\{\mathbf{p}_{1}, \mathbf{v}_{1}\right\}$ is the intersection of the planes $\Pi_{a}$ and $\Pi_{b}$ (e.g. the intersection of two walls). We extract 3D planes using

RANSAC for plane fitting. Some samples of real scenes with the planes extracted are shown in Figure 3.10 and Figure 3.11b. A plane $\Pi_{i}$ has normal $\mathbf{n}_{i}$ and distance to the origin $d_{i}$ so that all points $\mathbf{X}$ belonging to the plane satisfy $\mathbf{n}_{i} \cdot \mathbf{X}+d_{i}=0$. To compute the 3D line between two planes $\Pi_{a}$ and $\Pi_{b}$, we get the direction $\mathbf{v}$ as the cross product of their normals, $\mathbf{v}=\mathbf{n}_{a} \times \mathbf{n}_{b}$. A 3D point of the line $\mathbf{p}_{0}$ is obtained as the closest point to the origin that fulfills the equations $\mathbf{n}_{a} \cdot \mathbf{p}_{0}+d_{a}=0$ and $\mathbf{n}_{b} \cdot \mathbf{p}_{0}+d_{b}=0$.

In the case of an RGB-D camera already calibrated (with per-pixel correspondence between color and depth), the easiest way to proceed is to use the segment extraction described for an RGB camera, and use the depth information to transform the segment points to 3D. RANSAC can be used to remove possible outliers in the 3D points that define the line. This approach has the advantage of being able to extract lines on planes, which is not possible only with depth data. That is the case of the line $\left\{\mathbf{p}_{2}, \mathbf{v}_{2}\right\}$ contained in the plane $\Pi_{a}$ in Figure 3.9, which can be detected from the color changes.

### 3.4.1.3. LINE CORRESPONDENCES BETWEEN CAMERAS

After line extraction, we create a set of $N_{L}$ line correspondences $\mathcal{L}_{i=1 . . N_{L}}$. In our notation, we call $D$ the camera with depth information and $C$ the conventional one (color or monochrome). Every correspondence $\mathcal{L}_{i}$ consists of a fully parametrized 3D line from $D$ and the normal of the projective plane from $C$, i.e. $\mathcal{L}_{i}=\left\{\mathbf{p}_{i}^{D}, \mathbf{v}_{i}^{D}, \mathbf{n}_{i}^{C}\right\}$. For example, in Figure 3.9 we can observe $\mathcal{L}_{1}$ and $\mathcal{L}_{2}$, and a real case with two correspondences in Figure 3.11c.

An automatic procedure to extract line correspondences based on RANSAC is implemented as follows:

1. Extract all the lines in $C$ and $D$ for each image pair independently to create a broad set of correspondence candidates $\mathcal{L}$.
2. Filter $\mathcal{L}$ according to an initial estimate of the relative poses of the cameras and their uncertainty by setting angular and/or distance thresholds to remove outliers.
3. Pull a minimal set (Section 3.4.3) of three random correspondences from $\mathcal{L}$ to perform the calibration as described in Section 3.4.2, and count the number of consistent correspondences in $\mathcal{L}$.
4. Repeat the previous step using RANSAC to obtain the maximal consensus (i.e. maximum number of inliers).
5. The final calibration is computed from the inlier correspondences.

Note that it is easier to perform the calibration from scenes without clutter, see Figure 3.11b, where a few lines can be robustly extracted. Cluttered scenes result in higher number of outlier correspondences which may require a better initial approximation of
the calibration to be filtered out. Nevertheless, since calibration should not be performed very often, the correspondences may also be selected with human supervision to guarantee the correctness of the calibration.

### 3.4.2. EXTRINSIC CALIBRATION FROM LINE OBSERVATIONS

In this section we address the problem of extrinsic calibration of a depth camera $D$ and a color camera $C$. Let ${ }^{C} \mathbf{T}_{D}=[\mathbf{R} \mid \mathbf{t}] \in S E(3)$ be the relative pose between the reference frame of $D$ with respect to $C$. Our goal is to find the maximum likelihood estimation (MLE) for ${ }^{C} \mathbf{T}_{D}$. Since rotation and translation can be decoupled, we separate the process in two stages, computing first the rotation $\mathbf{R}$ and then the translation $\mathbf{t}$. We consider that our line observations are affected by unbiased Gaussian noise $N(0, \sigma)$, uncorrelated between different line correspondences. Under this assumption, the MLE is equivalent to the solution of the least-squares minimization of the geometric errors of the line correspondences for the rotation and the translation.

### 3.4.2.1. Rotation estimation

From the definitions in the previous section, the direction vector of a line $\mathbf{v}$ is orthogonal to the normal vector $\mathbf{n}$. This condition holds between separate cameras $C_{1}$ and $C_{2}$ after applying the corresponding relative rotation to transform both vectors to the same reference frame. Thus, we can use the condition of orthogonality to retrieve the rotation by computing the matrix $\mathbf{R} \in S O(3)$ that satisfies:

$$
\begin{equation*}
\left(\mathbf{n}^{C_{2}}\right)^{\top} \cdot \mathbf{R v}^{C_{1}}=0 \tag{3.21}
\end{equation*}
$$

where in our problem, $C_{1}=D$ and $C_{2}=C$ (we drop these super-indexes for readability). We need at least three line correspondences $\mathcal{L}_{i}$ to estimate the rotation $\mathbf{R}$, which has three degrees of freedom. A more extended discussion about the observability of this problem is provided in Section 3.4.3.

The MLE of the relative rotation is equivalent to the solution of the following nonlinear least squares minimization, with the relative rotation $\mathbf{R}$ represented in a minimal parametrization with the exponential map from Lie algebra:

$$
\begin{equation*}
\underset{\mu}{\arg \min } \sum_{i=1}^{N_{L}}\left(\mathbf{n}_{i}^{\top} \cdot e^{\mu} \mathbf{R} \mathbf{v}_{i}\right)^{2} \tag{3.22}
\end{equation*}
$$

where $e^{\mu}$ is the exponential map of the increment of rotation $\mu$ on $\mathbf{R}$. The vector $\mu$ has three dimensions and it is the axis-angle representation of the rotation on a manifold space of $S O(3)$. We solve this non-linear least squares problem iteratively with GaussNewton. The increment vector $\mu$ is computed as:

$$
\mu=-\mathbf{H}^{-1} \mathbf{g}
$$

where $\mathbf{H}$ and $\mathbf{g}$ are the Hessian and the Gradient of the error function, computed as:

$$
\begin{equation*}
\mathbf{H}=\sum_{i=1}^{N_{L}} \mathbf{J}_{i}^{\top} \mathbf{J}_{i}, \quad \mathbf{g}=\sum_{i=1}^{N_{L}} \mathbf{J}_{i}^{\top} r_{i} \tag{3.24}
\end{equation*}
$$

with the Jacobians and residuals given by

$$
\begin{equation*}
\mathbf{J}_{i}=\left(\mathbf{R} \mathbf{v}_{i} \times \mathbf{n}_{i}\right)^{\top}, \quad r_{i}=\mathbf{n}_{i} \cdot \mathbf{R} \mathbf{v}_{i} \tag{3.25}
\end{equation*}
$$

### 3.4.2.2. TRANSLATION ESTIMATION

Following a similar reasoning, the vector $\mathbf{p}$ representing any point on the 3 D line is perpendicular to the normal vector $\mathbf{n}$. Therefore, the relative pose ${ }^{C} \mathbf{T}_{D}=[\mathbf{R} \mid \mathbf{t}]$ must satisfy:

$$
\begin{equation*}
\left(\mathbf{n}^{C}\right)^{\top} \cdot\left(\mathbf{R p}^{D}+\mathbf{t}\right)=0 \tag{3.26}
\end{equation*}
$$

Assuming that the rotation is already known, we require at least three line correspondences to find a valid solution for $\mathbf{t}$ (see Section 3.4.3 for special degenerate cases). The MLE of the relative translation is equivalent to the solution of the following non-linear least squares minimization:

$$
\begin{equation*}
\underset{\mathbf{t}}{\arg \min } \sum_{i=1}^{N_{L}}\left(\mathbf{n}_{i}^{\top} \cdot \frac{\mathbf{T} \mathbf{p}_{i}}{\left\|\mathbf{T} \mathbf{p}_{i}\right\|}\right)^{2}=\underset{\mathbf{t}}{\arg \min } \sum_{i=1}^{N_{L}}\left(\mathbf{n}_{i}^{\top} \cdot \frac{\mathbf{R} \mathbf{p}_{i}+\mathbf{t}}{\left\|\mathbf{R} \mathbf{p}_{i}+\mathbf{t}\right\|}\right)^{2} \tag{3.27}
\end{equation*}
$$

Note that the point $\mathbf{p}_{i}$, after rotated and translated, must be normalized since, otherwise, points situated farther away from the origin of coordinates would have more influence in the optimization. We also solve the problem with Gauss-Newton, where the Jacobians and residuals are computed as:

$$
\begin{equation*}
\mathbf{J}_{i}=\mathbf{n}^{\top} \cdot \frac{\mathbf{I}-\frac{\mathbf{T} \mathbf{p}}{\|\mathbf{T} \mathbf{p}\|}\left(\frac{\mathbf{T} \mathbf{p}}{\|\mathbf{T} \mathbf{p}\|}\right)^{\top}}{\|\mathbf{T} \mathbf{p}\|}, \quad \mathbf{r}_{i}=\mathbf{n}_{i}^{\top} \cdot \frac{\mathbf{T} \mathbf{p}_{i}}{\left\|\mathbf{T} \mathbf{p}_{i}\right\|} \tag{3.28}
\end{equation*}
$$

### 3.4.2.3. CALIBRATION OF MULTIPLE CAMERAS

Let us assume we have a rig of $N=N^{C}+N^{D}$ sensors, with $N^{C}$ conventional cameras $\left\{C_{1}, C_{2}, . ., C_{N^{C}}\right\}$ and $N^{D}$ depth cameras $\left\{D_{1}, D_{2}, . ., D_{N^{D}}\right\}$, if we perform pair-wise calibration for all the combinations $C_{i}-D_{j}$, the global solution will be inconsistent generally. The solution is to perform a complete non-linear optimization with all the parameters. We can set the global reference frame to one of the sensors without loss of generality, for instance $C_{1}$. Thus, we need to find the MLE for $(N-1)$ rigid transformations.

The problem is formulated as follows, for the rotation:

$$
\begin{equation*}
\underset{\mu_{2} . . \mu_{N}}{\arg \min } \sum_{j=1}^{N^{C}} \sum_{k=1}^{N^{D}} \sum_{i=1}^{N_{L}^{j k}}\left(\left(e^{\mu_{j}} \mathbf{R}_{j} \mathbf{n}_{j i}\right)^{\top} \cdot e^{\mu_{k}} \mathbf{R}_{k} \mathbf{v}_{k i}\right)^{2} \tag{3.29}
\end{equation*}
$$

where $\mu_{x}$ is the increment of rotation to $\mathbf{R}_{x}$ for each camera. $N_{L}^{j k}$ stands for the number of line correspondences between cameras $C_{j}$ and $D_{k}$. The Hessian and gradient, with dimensions $(3 \cdot(N-1) \times 3 \cdot(N-1))$ and $(3 \cdot(N-1) \times 1)$ respectively, are computed following (3.24), where the Jacobians are given by

$$
\begin{align*}
\mathbf{J}_{i}^{(j)} & =\left(\left(\mathbf{R}_{j} \mathbf{n}_{j i}\right) \times\left(\mathbf{R}_{k} \mathbf{v}_{k i}\right)\right)^{\top} \\
\mathbf{J}_{i}^{(k)} & =\left(\left(\mathbf{R}_{k} \mathbf{v}_{k i}\right) \times\left(\mathbf{R}_{j} \mathbf{n}_{j i}\right)\right)^{\top} \tag{3.30}
\end{align*}
$$

and the super-indexes $(j)$ and $(k)$ represent the three-column block corresponding to the parameters $\mu_{j}$ or $\mu_{k}$. For the translation, the formulation of the minimization problem is:

$$
\begin{equation*}
\underset{\mathbf{t}_{2} . . \mathbf{t}_{N}}{\arg \min } \sum_{j=1}^{N^{C}} \sum_{k=1}^{N^{D}} \sum_{i=1}^{N_{L}^{j k}}\left(\mathbf{n}_{j i}^{\top} \cdot \frac{\mathbf{T}_{j}^{-1} \mathbf{T}_{k} \mathbf{p}_{k i}}{\left\|\mathbf{T}_{j}^{-1} \mathbf{T}_{k} \mathbf{p}_{k i}\right\|}\right)^{2} \tag{3.31}
\end{equation*}
$$

where the operation $\mathbf{T}_{j}^{-1} \mathbf{T}_{k} \mathbf{p}_{k i}$ transforms the point $\mathbf{p}_{k i}$ to the reference frame of $C_{j}$ (note that we have employed homogeneous notation for simplicity, and the transformed points are used in its compact form afterwards). Again, instead of using the coordinates of the transformed point, we normalize to have the direction vector of such point. The resulting Jacobians are:

$$
\begin{align*}
& \mathbf{J}_{i}^{(j)}=\mathbf{n}_{i j}^{\top} \cdot \frac{-\mathbf{R}_{j}^{\top}-\frac{\mathbf{T}_{j}^{-1} \mathbf{T}_{k} \mathbf{P}_{k i}}{\left\|\mathbf{T}_{j}^{-1} \mathbf{T}_{k} \mathbf{P}_{k i}\right\|}\left(\frac{\mathbf{T}_{j}^{-1} \mathbf{T}_{k} \mathbf{p}_{k i}}{\left\|\mathbf{T}_{j}^{-1} \mathbf{T}_{k} \mathbf{P}_{k i}\right\|}\right)^{\top}}{\left\|\mathbf{T}_{j}^{-1} \mathbf{T}_{k} \mathbf{p}_{k i}\right\|}  \tag{3.32}\\
& \mathbf{J}_{i}^{(k)}=\mathbf{n}_{i j}^{\top} \cdot \frac{\mathbf{R}_{j}^{\top}-\frac{\mathbf{T}_{j}^{-1} \mathbf{T}_{k} \mathbf{p}_{k i}}{\left\|\mathbf{T}_{j}^{-1} \mathbf{T}_{k} \mathbf{P}_{k i}\right\|}\left(\frac{\mathbf{T}_{j}^{-1} \mathbf{T}_{k} \mathbf{p}_{k i}}{\left\|\mathbf{T}_{j}^{-1} \mathbf{T}_{k} \mathbf{P}_{k i}\right\|}\right)^{\top}}{\left\|\mathbf{T}_{j}^{-1} \mathbf{T}_{k} \mathbf{p}_{k i}\right\|}
\end{align*}
$$

### 3.4.3. ObSERVABILITY

In this section we present the analysis of minimal solutions and possible degenerate cases of our calibration problem. For that, we analyze the shape of the Fisher Information Matrix (FIM) for the parameters of the maximum likelihood estimator (MLE) of the calibration presented in the previous section. The FIM coincides with the Hessian of the least squares problem resulting from the MLE, and its inverse is the covariance of the resulting calibration (which corresponds in turn to the Cramér-Rao lower bound when the MLE is given by an unbiased Gaussian distribution [Fernandez-Madrigal and Claraco, 2013]). When the FIM is singular, the information provided is not sufficient and the MLE does not exist, therefore, the calibration problem has a solution only when the FIM has full rank.

Let us analyze first the rotation estimation problem. From the error function and its Jacobian (eqs. (3.22) and (3.25)), we have that each line correspondence imposes a new constraint between a pair of sensors. Thus, we need at least 3 measurements to compute
the relative rotation. These constraints must be linearly independent, which is the case when the direction vectors $\mathbf{v}^{D}$ of the 3D lines as seen by the depth camera are not all parallel (i.e. two of the three can be parallel). Notice that two parallel lines in 3D are not necessarily parallel in the image, as they may intersect in a vanishing point.

Regarding the estimation of the translation, assuming that the rotation is known, each line correspondence imposes a new constraint between the pair of sensors. In order to get a full rank FIM, the Jacobians of the 3 constraints (3.28) must be linearly independent. For that, at least 2 normal vectors $\mathbf{n}^{C}$ must be linearly independent, which is true for any pair of different lines in the projected image (even for parallel lines in the image). This condition is trivially fulfilled for any three constraints for which the rotation's FIM has full rank. Also, the lines used for calibrating the translation cannot intersect all in the same 3D point (e.g. a trihedron), because the translation along the projection ray which contains the optical center of the camera and the line's intersection would not be observable. Hence, the observation of the three non-parallel lines which do not intersect in the same point provides enough information to localize a conventional camera with respect to a depth camera.

The required line correspondences may be observed in several views or in a single one (e.g. the rightmost image pair of Figure 3.10). Note that the 3 direction vectors do not have to form an orthogonal base, as it was required in other works [Gomez-Ojeda et al., 2015], nor any other special configuration. For a rig with $N$ sensors, the number of line correspondences will depend on the type of sensors of the system. Again, by analyzing the FIM of both rotation and translation estimation, we can guarantee that the system has a solution if there are at least $3(N-1)$ correspondences which fulfill the rotation and translation conditions between pairs of cameras. Depending on the type of sensors in the system, a solution may exist even with less line correspondences, but such analysis is out of the scope of this paper. Note also that we will be interested to obtain considerably more line correspondences than the minimal set in order to improve the accuracy. Nevertheless, the minimal solution is of interest to remove outlier line correspondences using RANSAC (Section 3.4.1.3).

### 3.4.4. EXPERIMENTAL VALIDATION

We have performed experiments in simulation and with real multi-camera systems. With the former we provide an analysis of performance and robustness to noise with regard to the number of line correspondences. Next, the real case scenarios show the validity and applicability of our method to the real world.

### 3.4.4.1. SimULATION

We present the results of calibration of a pair of depth and RGB cameras with a relative pose $\mathbf{T}$ from $D$ to $C$ given by a rotation of $(0.2,-\pi / 4-0.2,0.1)$ in Euler


Figure 3.12. Simulation results of the calibration accuracy for the rotation (left) and translation (right), given by the mean error with respect to the ground truth, over the number of line correspondences at three different noise levels.
angles and a translation of $(-0.06,0.03,0.1)$ in meters, for different numbers of line correspondences and different noise levels in their observations. We have also tested different ground truth poses with similar results. For each experiment, we generate randomly $N_{L}=5 . .200$ lines in 3D space and obtain their observation parameters in both cameras. For the analysis of the rotation we add unbiased Gaussian noise to the vectors $\mathbf{v}$ and $\mathbf{n}$ to rotate them slightly. We analyze the calibration accuracy for three noise levels with standard deviation $\sigma=\{0.1,0.5,1\}$ degrees. For the analysis of the translation, the point $\mathbf{p}$ is also translated with Gaussian noise $\sigma=\{0.001,0.005,0.01\}$ meters. We find these values to be realistic for the case of RGB-D sensors like Asus Xtion Pro Live.

We show the accuracy of the rotation and the translation separately in Figure 3.12, with the accuracy of the calibration measured by the angular error of our estimated rotation (left), and the translation error measured in meters as the norm of the difference with respect to the ground truth (right). The mean errors decrease asymptotically with the number of line correspondences. This behavior was expected, since having more data should improve the performance when the noise in the measurements is Gaussian. We see how the translation error is more sensitive to noise in comparison with the rotation error. This experiment shows promising results, since the mean error values remain small with a reasonably low number of correspondences.

### 3.4.4.2. REAL CASE SCENARIOS

Four sensor combinations are tested to show the successful performance of our method under different challenging situations:

(a)

(b)

Figure 3.13. (a) Fisheye with RGB-D camera system. (b) Omnidirectional camera rig composed by 8 Asus Xtion Pro Live arranged to achieve 360 degree field of view.

### 3.4.4.2.1. FISheye to Depth

This experiment describes the extrinsic calibration of a regular fisheye camera with FOV over $180^{\circ}$ and the depth sensor of an Asus XPL, shown in Figure 3.13a. This system is useful to combine the large FOV of the fisheye camera with the real scale provided by the depth [Perez-Yus et al., 2016b]. We compare our calibration results to the ones obtained using the method from [Perez-Yus et al., 2016a] (Section 3.3), based on a traditional planar checkerboard calibration which we use as ground truth. Note that we do not make use of the color information from the Asus XPL in this experiment in order to be consistent with our comparison with [Perez-Yus et al., 2016a].

We have recorded a set of 65 image pairs from our office desktop which contain a good number of lines, thus constituting a good source of information for our method, (see Figure 3.11). We selected manually 77 line matches to perform calibration and test its accuracy. From the set of line correspondences, we extract random sets of $N_{L}=\{10,20,30,40,50\}$ and measure the average angular and translational errors with respect to the ground truth from 100 calibration runs for each set, using the same metrics of the simulation results. The error values in Table 3.3 seem to corroborate the results from simulation, since the error decreases as number of line correspondences rises. The residuals from the optimization are also quite low, as we could expect since no outliers are introduced with the manual selection of correspondences. In Figure 3.11d there is

Table 3.3. Mean rotation and translation errors with respect to the ground truth and residuals in the calibration case of fisheye and depth cameras. Results for both manual and automatic (via RANSAC) matching of lines. The accuracy is analyzed respectively with the number of lines $\left(N_{L}\right)$ or iterations $\left(N_{\text {iter }}\right)$.

|  | $\mathrm{N}_{\mathrm{L}}$ | Mean error with GT |  | Mean residual error |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | Rotation (degrees) | Translation (meters) | Rotation (degrees) | Translation (meters) |
| < | 10 | 1.9796 | 0.0483 | 0.0132 | 0.0053 |
| Z | 20 | 1.2006 | 0.0288 | 0.0158 | 0.0050 |
| < | 30 | 0.8901 | 0.0216 | 0.0169 | 0.0051 |
| 2 | 40 | 0.7103 | 0.0179 | 0.0172 | 0.0050 |
|  | 50 | 0.5741 | 0.0150 | 0.0176 | 0.0051 |
| $\underset{\sim}{~}$ | $\mathrm{N}_{\text {iter }}$ | Rotation (degrees) | Translation (meters) | Rotation (degrees) | Translation (meters) |
| Z | 100 | 1.1922 | 0.0376 | 0.0009 | 0.0007 |
| 2 | 1000 | 0.7270 | 0.0202 | 0.0009 | 0.0008 |
|  | 10000 | 0.5545 | 0.0127 | 0.0010 | 0.0008 |

the reprojection of the 3D planes and lines on the fisheye image after the calibration.
We also test the performance of the automatic line-matching via RANSAC (Section 3.4.1.3) compared to manual matchings. For that, we only consider as correspondence candidates those pairs of lines with a relative rotation below $5^{\circ}$ (i.e.: $|\mathbf{n} \cdot \mathbf{v}|<\cos ((90-$ $5) * \pi / 180$ )) and a relative translation of 10 cm (i.e.: $|\mathbf{n} \cdot \mathbf{p}|<0.1$ ), where we have used the identity as the initial estimation of the relative pose, obtaining a total of $N_{L}=152$ candidate line matches. Notice that this type of heuristic filtering of candidate correspondences may be applied to any system where we have some rough information about the sensor set-up. We apply RANSAC to remove outliers and perform the calibration, whose results are shown in the lower part of the Table 3.3. We can see that the automatic approach achieves better accuracy in comparison to the case with manually selected correspondences.

### 3.4.4.2.2. Kinect calibration: RGB to Depth

In this case we show the performance of our calibration system using images from the NYU2 RGB-D public dataset [Silberman et al., 2012]. This dataset is thought to be used in segmentation tasks instead of calibration, so all images are from common indoor scenes (e.g. living rooms, kitchens, offices). A few examples of the line and plane extractions are shown in Figure 3.10. We use the provided parameters of extrinsic calibration of the camera Kinect as ground truth to compare our results. The Kinect has a relative pose from the depth camera to the RGB camera close to the identity, with a translation in the X axis of around 2.5 cm . We use the identity as initial rotation matrix,
with initial translation equal to zero.
For the experiment we use a recorded sequence in a study room, which was one of the less cluttered (third image in Figure 3.10). We use the automatic line-matching with prefiltering of $0.5^{\circ}$ for the rotation and 5 cm for the translation, obtaining an initial set of 2229 line-matchings. The RANSAC returns 328 inliers, for which in the optimization we got a rotation error of $0.7578^{\circ}$ and a translation error of 1.27 cm (the translation result in X is 3.16 cm ). The residuals of the optimization are very low (under $10^{-5}$ for the rotation and the translation).

We can consider these results satisfactory considering the difficulty of using images from an external dataset which was thought for another task. In particular, this dataset has very high levels of clutter which introduces many outliers in the set of line correspondences. Besides, many of the useful line correspondences are from far distances, where the plane extraction is less accurate due to the higher levels of noise from the depth camera. Other methods achieve better accuracy (such as [Herrera et al., 2012]), but they need to build a calibration pattern and capture images for this specific purpose.

### 3.4.4.2.3. RGB TO DEPTH WITH NON-OVERLAPPING FOVS

In this experiment we have used an omnidirectional camera rig formed by 8 Asus Xtion Pro Live cameras arranged in a radial configuration, see Figure 3.13b. For this particular experiment we only use two adjacent cameras from the rig to calibrate the RGB of one of the cameras to the Depth of the other. Adjacent cameras have a relative rotation of $45^{\circ}$, which we use as initial estimate of relative pose, and a relative translation of less than 10 cm .

The average value of the residuals after the optimization for different numbers of line correspondences ( $N_{L}$ ) are shown in Table 3.4 (columns 2-3). As in simulation, we observe that a higher number of line correspondences generally improve the results in both rotation and translation. Such improvement stabilizes after a few tens of lines, with a similar trend as the simulation above.

In order to evaluate the accuracy of the system it is desirable to have the ground truth of the calibration of our camera rig. Since this is not available, we employ a big planar checkerboard in a way that each camera observes the portion of the checkerboard not visible by the other camera to evaluate the accuracy of our calibration. First we perform a qualitative evaluation by visualizing the image stitching together with the point cloud reconstructed after calibration from different perspectives (Figure 3.14), showing the consistency of the different views. For a quantitative evaluation, we extract the 3D points of the square corners from the checkerboard and place them into the same reference frame given by the calibration (as it is commonly done for intrinsic calibration [Herrera et al., 2012, Perez-Yus et al., 2016a]). Then, we measure the distance of the corners between both cameras to compare them to the real measurements. We compute the average distance between the most distant corners for each row. The average size of the


Figure 3.14. Above: Visual evaluation of the RGB to depth camera calibration with a checkerboard with the point clouds reprojected to a common reference frame (general and lateral views). Below: a panoramic reprojection of the 3D points where we can see the checkerboard's pattern continuity.
checkerboard squares is of 118.3 mm in the calibrated images, which is similar to the real dimension of 120 mm . We can also estimate the plane equations from each side and compare angles of their normals and the differences in distances to the origin. We obtained an angular difference of $1.7^{\circ}$ and a distance difference of 2.4 mm .

### 3.4.4.2.4. Omnidirectional Rig of 8 RGB-D cameras

In this experiment we calibrate the relative positions among all cameras from the camera rig shown in Figure 3.13b. Since the cameras have an approximate vertical FOV of $45^{\circ}$, the eight camera rig achieves an horizontal FOV of $360^{\circ}$. We calibrate this rig following Section 3.4.2.3. Table 3.4 (columns 4-5) shows the residuals of the optimization according to the number of correspondences extracted between pairs of adjacent cameras (e.g. $N_{L}=10$ corresponds to 10 correspondences per pair and 80 for the full rig). A comparison with the two-camera case reveals that the residuals are smaller because of the global optimization.

In this case we cannot obtain any ground truth, so we evaluate the accuracy quali-

Table 3.4. Residual errors in the extrinsic calibration of an RGB camera to a depth camera experiment and the omnidirectional RGB-D camera rig for different number of line correspondences.

|  | RGB to Depth no overlap |  | Omnidirectional RGB-D rig |  |
| :---: | :---: | :---: | :---: | :---: |
| $\mathbf{N}_{\mathbf{L}}$ | Rotation <br> residual <br> (degrees) | Translation <br> residual <br> (meters) | Rotation <br> residual <br> (degrees) | Translation <br> residual <br> (degrees) |
| $\mathbf{1 0}$ | 0.0882 | 0.0319 | 0.0378 | 0.0557 |
| $\mathbf{2 0}$ | 0.0849 | 0.0336 | 0.0354 | 0.0405 |
| $\mathbf{3 0}$ | 0.0632 | 0.0271 | 0.0267 | 0.0160 |
| $\mathbf{4 0}$ | 0.0553 | 0.0229 | 0.0205 | 0.0110 |
| $\mathbf{5 0}$ | 0.0489 | 0.0193 | 0.0211 | 0.0193 |



Figure 3.15. Panoramic views from two different scenes obtained with the 8 RGB-D camera rig. The views have been obtained by reprojecting the 3D points transformed to a common reference frame with the relative poses obtained following our multi-camera calibration method.
tatively by visual verification. The different RGB images are stitched into a panorama by projecting the individual 3D point clouds transformed to a common reference frame. Ideally, the images should merge seamlessly for a good calibration. In Figure 3.15 there are two examples of our image stitching, where it can be observed that the relative positions are well recovered. Compared to [Fernandez-Moral et al., 2014], which uses the same camera rig, we got better results in the image stitching. The main reason for that is that we use information coming from the color camera and not only depth.

### 3.5. DISCUSSION

In this chapter, we have addressed one of the main limitations that conventional RGB-D cameras usually have: the narrow field of view. To overcome this limitation, first we presented a new hybrid camera system composed of a depth sensor and a fisheye camera. Many applications could benefit from such configuration, including navigation, SLAM or object detection. The fisheye provides wide field of view while depth provides certainty and scale. However, as a consequence of its novelty, none of the existing methods of calibration can be directly applied to this system. Thus, we have proposed a method which combines state-of-the-art works for this purpose. The method was tested with real images showing promising results in terms of accuracy.

Additionally, in order to provide a more general solution, we developed a novel procedure to calibrate different combinations of range and conventional cameras based on lines. In contrast to previous alternatives, we solve the problem for sensors systems without overlapping FOVs, our solution is considerably easier to apply, it does not have unobservable parameters and it allows to calibrate different sensor combinations with reasonable accuracy. We also present an observability analysis of the problem, providing relevant information regarding the number of observations necessary for our method to perform properly. Our experiments in simulation and real multi-camera systems prove the validity of the method and its applicability to real cases. In particular, we successfully calibrate the aforementioned hybrid camera system with fisheye and depth camera, and a camera rig with 8 conventional RGB-D cameras.

In the following section we introduce the first method that takes advantage of the hybrid camera system presented in this chapter. In particular, the combination of large field of view and 3D perception is used to extract full-scaled models of the layout of the room.

## 4

## Scaled Layout recovery with wide field of view RGB-D

In this chapter, we propose a method that integrates depth and fisheye cameras to obtain a wide $3 D$ scene reconstruction with scale in one single shot. With this system, whose calibration has been addressed in previous chapter, we have a portion of the scene with shared field of view that provides simultaneously color and depth. In the rest of the color image we estimate the depth by recovering the structural information of the scene. Our method finds and ranks corners in the scene combining the extraction of lines in the color image and the depth information. These corners are used to generate plausible layout hypotheses, which have real-world scale due to the usage of depth. The wide angle camera captures more information from the environment (e.g. the ceiling), which helps to overcome severe occlusions. After an automatic evaluation of the hypotheses, we obtain a scaled 3D model expanding the original depth information with the wide scene reconstruction. We show in our experiments with real images from both homemade and commercial systems that our method achieves high success ratio in different scenarios and that our hybrid camera system outperforms the single color camera set-up while additionally providing scale in one single shot.


### 4.1. InTRODUCTION

One of the most important topics in computer vision and robotics has been to perceive the 3D information from the scene. The recent advent of consumer RGB-D cameras has caused a great impact in the field, since having color and depth information synchronized in one single shot is very appealing. Unfortunately, these devices usually have a field of view (FOV) too narrow for certain applications, and it is necessary to move the camera in order to capture different views of the scene. That is often not easy to achieve when cameras are attached to systems with limited mobility and requires using SLAM algorithms or additional sensors to maintain the system well localized.

Here, we propose to use a color camera with wide FOV to extend the depth information in a hybrid camera configuration composed by a depth and a fisheye camera. The FOV of a fisheye is over $180^{\circ}$, in contrast with the usual FOV of $43^{\circ} \times 57^{\circ}$ of consumer depth cameras (Figure 4.1a). Once the cameras are calibrated, the system is capable of viewing over a hemisphere of color information where the central part of the image has also depth data (about $8.7 \%$ of the total number of pixels, as shown in Figure 4.1b). One can think of this configuration inspired in the vision of the human eye, where the central part (fovea) provides richer information than the periphery, and the field of view is slightly over $180^{\circ}$. To our knowledge, this is the first time this configuration has been used, although the interest in such sensor pairing is clear in the recent Google's Tango project. Notice that, although our work uses a fisheye camera, the approach could be extended to other kinds of omnidirectional systems.

In particular, we propose to extend the 3D information in one single shot via spatial layout estimation. Our layout estimation method is based on line segments from the fisheye image, and provides scaled solutions rooted on the depth information. As a result, a final 3D scene reconstruction is provided (see Figure 4.1c). The 3D room layout can be seamlessly merged with the original depth information to generate a 3D image with the periphery providing an estimation of the spatial context to the central part of the image, where the depth is known with good certainty. The collaboration between cameras is bidirectional, since the extension of the scene layout to the periphery is performed with the fisheye, but the depth information is used both to enhance the layout estimation algorithm and to scale the solution.

A scheme of the whole algorithm is shown in Figure 4.2. In detail, the depth camera provides a region of the image with 3D data, from which an initial estimate of the Vanishing Points (VPs) and 3D planes can be recovered. The VPs are used to retrieve the scene orientation necessary to generate layout proposals. Here, we assume scenes are from a Manhattan World [Coughlan and Yuille, 1999], meaning the world is organized according to three orthogonal main directions. This assumption holds for most human-made scenarios, especially indoors. The 3D planes extracted are used to provide scale, impossible to get otherwise with one single shot and no previous knowledge of the scene. The layout is scaled by detecting the floor plane, however, we include a final


Figure 4.1. (a) Field of view of our proposed system composed by a Fisheye and an RGB-D camera. (b) The depth information in the center is extended to the periphery combining information with the line segments that we use to extract the spatial layout of the scene. (c) As a result we obtain a full-scaled 3D reconstruction of the scene.


Figure 4.2. Block diagram of the main stages of the algorithm, starting with the initial Fisheye and Depth images and finishing with the result of the scaled layout.
scaling procedure whenever it is not in the image. Having scale has many advantages in this type of methods which usually require several heuristics. For instance, tuning parameters can be grounded in reality. Depth information is also used to filter hypotheses and reward line segments corresponding to intersections of planes.

The line segments from the wide image are classified according to the three Manhattan directions. The horizontal lines are projected either to the floor plane or the estimated ceiling plane to have the 3D segment position in the real world. Structural corners are then looked for, by considering plausible and simple cases of line distribution. The corners of the map are evaluated by our scoring function, and layout hypotheses are proposed by the probability of these corners to occur in the real world, proportional to these scores. Then, layout hypotheses are generated based on geometrically coherent wall distributions that do not contradict the initial depth information and the visible segments. The algorithm is able to work even under high clutter circumstances due to the combination of lines from both floor and ceiling (visible because of using a large FOV camera), but also because of our generation of Manhattan hypotheses that can estimate hidden corners to complete the layout. For the evaluation stage we propose three alternative methods whose performance is comparable to a well known state of the art method [Lee et al., 2009] while being faster and more efficient.

Experimental evaluation is also provided, for which real images from two camera systems have been used. In particular, we use our hybrid camera system from Figure 4.1 as well as a Google Tango device, showing also the potential of the method with commercial devices. With both systems our proposal shows promising results about the algorithm and the camera configuration. Our method gets good results even when only a few hypotheses are drawn thanks to our corner detection approach. We also show how having depth information helps notably in the layout extraction. In the following section we provide a brief analysis of the state of the art before introducing our method.

### 4.2. Related work

One of the first attempts to recover 3D layout information of indoor environments with single images was [Delage et al., 2006], which uses a Bayesian network model to find the floor-wall boundary. In contrast, Lee et al. [Lee et al., 2009] use line segments to generate layout hypotheses evaluating their fitness to an Orientation Map (OM). Using lines has the advantage of producing results without relying on scene-specific properties such as colors and image gradients. However, while some lines can actually include structural information of the environment (e.g. intersections wall-wall or wall-floor), usually most of them belong to clutter or are useless and misleading. To help with this problem, some assumptions and some set of rules are usually proposed based on geometric coherence. Usually the main assumptions are that all structures in indoor environments are composed by planar surfaces and that these surfaces are oriented according
to three orthogonal directions [Coughlan and Yuille, 1999]. This assumption holds for most indoor environments, and it is almost unanimously used in the literature.

Other works try to simplify the problem by making assumptions about the structure, e.g. assuming that the room is a 3D box, like Hedau et al. in [Hedau et al., 2009]. This work uses a modified version of Geometric Context (GC) [Hoiem et al., 2007] instead of the OM for evaluation of hypotheses which includes a separate clutter category. Similar examples are [Schwing et al., 2012], which applies efficient structured prediction; [Ramalingam et al., 2013], which defines a catalogue of scene corners; and [Chang et al., 2015], whose method is based on line consistency. Some other works perform this type of '3D box' reasoning while performing object detection [Hedau et al., 2010, Lee et al., 2010, Del Pero et al., 2011, Del Pero et al., 2012, Choi et al., 2013, Schwing et al., 2013]. Doing so can have both tasks help each other (e.g, it is impossible for this room to have a certain layout if there is a bed across the wall, or vice versa). However, using the '3D box' assumption does not generalize well in real world scenes, e.g. in corridors or entrances. Other approaches make use of video sequences instead of single images [Flint et al., 2011, Furlan et al., 2013]. While introducing temporal consistency may be beneficial in this task, we focus on getting better layout estimation with single image.

Deep learning has exploded in popularity in the last few years, and recently new interesting approaches in layout estimation are appearing and dominating in challenges such as LSUN Room Layout Estimation ${ }^{1}$. That is the case of [Mallya and Lazebnik, 2015], which has trained a fully convolutional neural network (FCNN) to extract the informative edges of the scene (i.e. those corresponding to structural boundaries and not coming from clutter), and then use those edges to extract the layout. Another similar proposal with better results is shown in [Zhang et al., 2017]. Dasgupta et al. [Dasgupta et al., 2016] propose an alternative method that trains a FCNN which returns the heat maps of each surface class (walls, ceiling, floor). RoomNet [Lee et al., 2017] is an end-to-end neural network which infers the type of room and the main keypoints to build a layout (i.e. corners). These approaches show a great potential of these data-driven techniques. On the other hand, the results are too tied to specific room configurations, not being able to work on complex structures.

The methods mentioned up to this point have in common that all of them use images from conventional cameras. As opposed to that, some recent works use omnidirectional cameras such as catadioptric systems or fisheye cameras. Having greater field of view has many advantages for this task:

- Larger view of the line segments appear in the image, so it is more likely to extract the relevant lines of the scene.
- Allows to perceive better the orientation of the scene and thus a more robust vanishing point estimation.
- Provides better view of the ceiling areas, which usually have less clutter than the

[^4]lower parts in indoor scenes.

- Larger portion of the room is captured at once, which provides wider or even complete room reconstructions.

For example, in [Jia and Li, 2013], they use a fisheye camera to perform layout retrieval, essentially extending the work from [Lee et al., 2009], but with wider FOV. Lopez-Nicolas et al. in [Lopez-Nicolas et al., 2014] perform the closed layout recovery using a catadioptric system mounted in a helmet. Jia and Li [Jia and Li, 2015] use $360^{\circ}$ panorama full-view images, which allows them to recover the layout of the whole scene at once. Fukano et al. [Fukano et al., 2016] propose another method with panoramas, solving the problem as a high order energy minimization. Similarly, PanoContext [Zhang et al., 2014b] uses the same type of images to perform layout retrieval along with a whole-room context model in 3D including bounding boxes of the main objects inside the room. Pano2CAD [Xu et al., 2017] enhances PanoContext by considering not only box-shaped types of room. In [Yang and Zhang, 2016] they propose an alternative graph-based method for panoramas, combining superpixels and line segments. Cabral et al. [Cabral and Furukawa, 2014] gets 3D information from structure from motion with a panoramic image, and then generates the best fitting floor plan reconstruction.

In these approaches the recovered 3D layout is obtained up to a scale, unless previous knowledge about the scene is provided. Modern RGB-D cameras are able to provide depth alongside color, which provides scale information in a single shot. However, most of these cameras have a FOV too narrow for layout estimation. Recently, some alternatives investigate extending the FOV of depth cameras using additional elements [Endres et al., 2014, Iglesias et al., 2016, Tomari et al., 2012, Fernandez-Moral et al., 2014]. Extended discussion about these methods was provided in Section 3.2. Here, we propose to use a depth camera alongside a fisheye in the same calibrated system, and perform the depth extension via spatial layout estimation. Our proposal combines the advantages of omnidirectional cameras (recover wider information) and depth cameras (provide 3D certainty and scale). This is also a camera system that already exists in the market (e.g. Google Tango) or can be easily reproduced with state-of-the-art calibration methods [Perez-Yus et al., 2016a, Perez-Yus et al., 2018a] (see Chapter 3). No restrictions about the shape of the room are considered, i.e. not only '3D box' room shapes or layout estimations tied to specific room configurations. No rigid assumption about the camera pose in the scene is considered, since it is found automatically. Besides, no sequences of images or complex machine learning algorithms are used in our method.

### 4.3. DEPTH AND FISHEYE IMAGES PROCESSING

In this section, we address the initial stages of our method, describing how we extract the information we need to perform the layout recovery (Figure 4.2). First we explain the calibration of the system, needed to map information from the depth camera to the
fisheye camera (Section 4.3.1). Then we describe the line segment extraction algorithm in the fisheye image (Section 4.3.2). To recover the Manhattan directions we use information from depth (for an initial estimate) and lines (for the final values), as described in Section 4.3.3. Then we perform plane extraction in the depth image to find 3D line intersections and to enable scaled layout extraction (Section 4.3.4). At the end, the line segments are classified according to its orientation (Section 4.3.5).

### 4.3.1. System calibration

To map world points $\mathbf{X}$ from the depth camera reference frame $D$ to the fisheye camera reference frame $F$, it is necessary to calibrate the extrinsic parameters ( $\mathbf{R}, \mathbf{t}$ ) and the intrinsic parameters of both cameras (Figure 4.3). The extrinsic calibration of range sensors to cameras is not a new issue, but most related works require manual selection of correspondences or do not support omnidirectional cameras [Zhang and Pless, 2004, Scaramuzza et al., 2007, Geiger et al., 2012]. To obtain the intrinsic parameters of the fisheye camera, we need a specific method with an appropriate camera model [Puig et al., 2012]. In particular, we choose the parametric camera model described by Scaramuzza et al. in [Scaramuzza et al., 2006], which considers the omnidirectional image as a highly distorted image with the distortion modeled as a polynomial. Using this polynomial it is not necessary to provide a specific model of projection and works with all kind of perspective, catadioptric or dioptric cameras. The points in the image $\hat{\mathbf{x}}_{i}=\left[u_{i}, v_{i}\right]$ and the vector $\mathbf{x}_{i}=[x, y, z]$ which points to the world point $\mathbf{X}_{i}$ are related following:

$$
\mathbf{x}_{i}=\left[\begin{array}{l}
x  \tag{4.1}\\
y \\
z
\end{array}\right]=\left[\begin{array}{c}
u_{i} \\
v_{i} \\
f\left(\rho_{i}\right)
\end{array}\right]=\left[\begin{array}{c}
u_{i} \\
v_{i} \\
a_{0}+a_{2} \rho_{i}^{2}+\ldots+a_{N} \rho_{i}^{N}
\end{array}\right]
$$

where the function $f$ is the polynomial that models the distortion with coefficients $a_{0} . . a_{N}$ and $\rho_{i}=\sqrt{u_{i}^{2}+v_{i}^{2}}$.

For the extrinsic calibration we have recently proposed two alternative methods suitable for our system [Perez-Yus et al., 2018a, Perez-Yus et al., 2016a]. While the one in [Perez-Yus et al., 2018a] has the advantages of being more generalizable to other camera systems and does not require to build a calibration device, we use [Perez-Yus et al., 2016a] since it is more accurate for our camera system. This method is inspired by [Herrera et al., 2012], adapted to the fisheye camera model from [Scaramuzza et al., 2006]. During the calibration process, the intrinsic parameters of the depth camera are also computed as defined in [Herrera et al., 2012] to improve the default parameters of the system. The depth images as captured by the sensor are transformed to point clouds using these parameters, and they are rotated and translated to the fisheye camera reference frame, following $\mathbf{X}_{F}=\mathbf{R} \cdot \mathbf{X}_{D}+\mathbf{t}$. From now on, every computation is done in that frame unless specified. A more detailed analysis of our calibration procedure is presented in Section 3.3.


Figure 4.3. Scheme of the system in a 3D world scene with the extrinsic calibration parameters $\mathbf{R}$ and $\mathbf{t}$ and the correspondent depth and fisheye images.

### 4.3.2. Line EXtraction in the fisheye image

For the line extraction in the fisheye camera we use the work from [BermudezCameo et al., 2015], which is compatible with central catadioptric and dioptric systems with revolution symmetry. Unlike conventional cameras, 3D lines in space do not appear as straight lines in the omnidirectional images, but they are projected to curves called line-images. In the schematic scene from Figure 4.3 we highlight a vertical line segment on the sphere model and its projection in the fisheye image. The shape of these line-images changes with the type of omnidirectional camera and its specific camera configuration.

The projection of a line $l_{i}$ in the 3D space can be represented by the normal of the plane $\Pi_{i}$ defined by the line itself and the viewpoint of the system, with normal $\mathbf{n}_{l_{i}}=\left(n_{x}, n_{y}, n_{z}\right)^{\top}$. The direction vector $\mathbf{x}$ of the points $\mathbf{X}$ lying on a 3D line $l$ satisfies the condition $\mathbf{n}_{l}^{\top} \mathbf{x}=0$. From [Bermudez-Cameo et al., 2015] and with (4.1), the constraint for points on the line projection in image coordinates is:

$$
\begin{equation*}
n_{x} u+n_{y} v+n_{z} f(\rho)=0 \tag{4.2}
\end{equation*}
$$

The line-images are non-polynomial and do not have conic shape. To extract them is necessary to solve a minimization problem [Bermudez-Cameo et al., 2015]. In the process, each line $l_{i}$ is associated to a set of contour points, denoted by $c\left(l_{i}\right)$, which are the inliers of the constraint (4.2).

### 4.3.3. Estimation of the vanishing points

As mentioned before, we assume the scenes are organized according to three orthogonal directions, $\left\{\mathbf{m}_{x}, \mathbf{m}_{y}, \mathbf{m}_{z}\right\}$, that define the Manhattan reference frame $M$. Parallel lines in the 3D world intersect in one single point in perspective images, called Vanishing Point (VP). In omnidirectional images, line projections result in curved line-images, and parallel lines intersect in two VPs. The directions $M$ are correlated to the VPs in regard that lines along these directions intersect in their corresponding VPs. Thus, we refer indistinctly to the computation of $M$ and the VPs from now on. We estimate the VPs to classify lines and planar surfaces from the depth information according to the three Manhattan directions.

There are previous approaches to obtain the VPs from omnidirectional images [Bazin et al., 2008]. However, we propose a method to extract the VPs taking advantage of both cameras with a two step optimization problem. Depth information is usually more robust, but less accurate than RGB information. Using fisheye images typically obtain a more accurate VP solution, but the problem may be unable to converge if the initial solution is not good enough. Besides that, a joint optimization is problematic as it needs to weight both terms appropriately. Experiments showed that our two-stage optimization procedure performs well without significant extra computational cost.

To compute $M$, we define a $3 \times 3$ matrix $\mathbf{M}$ that has the three Manhattan directions by columns, i.e. $\mathbf{M}=\left[\mathbf{m}_{x}, \mathbf{m}_{y}, \mathbf{m}_{z}\right]$. The initial solution of $\mathbf{M}$ is set as a trivial three orthogonal vector base ( $\mathbf{M}=\mathbf{I}_{3 \times 3}$ ). The variables to optimize are the roll-pitch-yaw angles ( $\alpha, \beta$ and $\gamma$ ) that form the rotation matrix $\mathbf{R}_{\alpha, \beta, \gamma}$ that after the optimization process should orient the vector base according to the Manhattan directions, $\mathbf{M}=\mathbf{R}_{\alpha, \beta, \gamma} \cdot \mathbf{I}=\mathbf{R}_{\alpha, \beta, \gamma}$. All optimizations are performed with the LevenbergMarquardt algorithm.

### 4.3.3.1. Initial estimate with depth information

The first step is to get the 3D normals of the points in the cloud. The normals $\mathbf{n}_{\mathbf{X}_{i}}$ of every point $\mathbf{X}_{i}$ (Figure 4.3) can be estimated using the method from [Rusu and Cousins, 2011]. To reduce computation time, the cloud can be previously down-sampled (e.g. with a voxel grid filter). In Manhattan scenes, it is likely for a large amount of points to have normals oriented in these directions. Based on this, the vector base is rotated until the angle between the normals of as many points as possible and one of the three vectors from the base is minimized. The minimization problem to retrieve $\mathbf{M}$ is formulated as follows:

$$
\begin{equation*}
\underset{\alpha, \beta, \gamma}{\arg \min } \sum_{i=1}^{N_{x}} \min \left(\left|\arccos \left(\mathbf{R}_{\alpha, \beta, \gamma}^{\top} \cdot \mathbf{n}_{\mathbf{X}_{i}}\right)\right|\right) \tag{4.3}
\end{equation*}
$$



Figure 4.4. (a) Planes from depth classified according to the Manhattan directions (red in $\mathbf{m}_{x}$, green in $\mathbf{m}_{y}$, blue in $\mathbf{m}_{z}$ ), initial extracted vanishing points, horizon line (white dotted line), and 3D intersections in yellow lines. (b) Line-images classified with their contours in white and the vanishing points after the second optimization.
where $N_{x}$ is the number of points from the cloud. The product $\left(\mathbf{R}_{\alpha, \beta, \gamma}^{\top} \cdot \mathbf{n}_{\mathbf{X}_{i}}\right)$ gives the cosine of the normal with respect to each one of the directions, from which the min function only takes the smallest one of the angles in absolute value. The columns of the final rotation matrix $\mathbf{R}_{\alpha, \beta, \gamma}$ are the three Manhattan directions M. An example where the points have been classified according to their normals and with the correspondent VPs is shown in Figure 4.4a.

### 4.3.3.2. Final estimate with Lines in the fisheye image

In this second stage, we use as seed the current value $\mathbf{M}$ from the previous optimization. The vector base is now rotated until the angle between the normals of as many lines as possible and one of the three vectors from the base is as close of being orthogonal as possible. This is based in that, by definition, the normal $\mathbf{n}_{l}$ of every line $l_{i}$ is orthogonal to the direction of the line in the 3D world, and therefore, if a line follows the Manhattan direction $\mathbf{m}_{j}$, then $\mathbf{n}_{l_{i}}^{\top} \cdot \mathbf{m}_{j}=0$. The optimization problem is formulated as follows:

$$
\begin{equation*}
\underset{\alpha, \beta, \gamma}{\arg \min } \sum_{i=1}^{N_{l}} \min \left(\left|\mathbf{R}_{\alpha, \beta, \gamma}^{\top} \cdot \mathbf{n}_{1_{i}}\right|\right) \tag{4.4}
\end{equation*}
$$

where the initial values of $\alpha, \beta$ and $\gamma$ are the values returned from the first minimization and $N_{l}$ is the number of lines in the fisheye image. In Figure 4.4b there is an example where the line-images that support each direction have been colored accordingly.

Our convention is to denote $\mathbf{m}_{y}$ the column whose vector is closest to the gravity vector given an intuition of how the camera is posed (pointing to the front, slightly down-
wards). We choose $\mathbf{m}_{z}$ to be the column pointing to the front and leaving $\mathbf{m}_{x}$ orthogonal to the previous two. The VPs are the points in the image that result of projecting rays following the Manhattan directions according to the intrinsic parameters (Section 4.3.1).

### 4.3.4. DEPTH INFORMATION PROCESSING

In this stage we start from the registered point cloud and extract planes (Section 4.3.4.1) and we determine if the floor is present in the image and provide a final transformation from camera pose to oriented scene pose (Section 4.3.4.2).

### 4.3.4.1. Plane EXtraction

The points from the point cloud $\mathbf{X}$ are classified depending on the orientation of their normals $\mathbf{n}_{\mathbf{X}}$ in the three orthogonal classes, given a certain angular threshold. For each class we perform a RANSAC for planes to recover its plane equations. It can happen that some of these plane equations have inliers in different surfaces separated in space (e.g. wall planes at each side of an open door). To avoid that and recover each separate planar patch instead, a 3D clustering is then performed for each plane to recover the planes $P$ in the image (Figure 4.3). Each plane is defined by its normal $\mathbf{n}_{P}$ and distance to the origin $X_{0}$ so that any point $\mathbf{X}$ belongs to a plane if $\mathbf{n}_{P}^{\top} \cdot \mathbf{X}+X_{0}=0$.

### 4.3.4.2. Floor Detection and scene pose

In this work we assume the floor and ceiling are unique and symmetric. Among the horizontal planes (i.e. with normal $\mathbf{n}_{P}=\mathbf{m}_{y}$ ), the lowest one (i.e. highest $X_{0}$ value below the horizon) is initially chosen as floor plane ( $P_{\text {floor }}$ ). Then we verify if there are a significant amount of points below that plane (considering a threshold due to noise): if there are points below the $P_{\text {floor }}$ then it is not a valid floor plane, but other structure (such as a table). When the floor plane is discarded or not found, a virtual $P_{\text {floor }}^{*}$ with normal equal to $\mathbf{m}_{y}$ and distance to the origin $X_{0}=1$ is created to continue the execution of the algorithm normally. At the end, the rest of the planes are used for scaling (see Section 4.4.4).

We compute the transformation matrix ${ }^{M} \mathbf{T}_{F} \in \mathrm{SE}(3)$ that transforms 3D points from the fisheye reference frame to the Manhattan reference frame. To compute ${ }^{M} \mathbf{T}_{F}$ we create its counterpart ${ }^{F} \mathbf{T}_{M}$ with rotation part the Manhattan directions (M) and the translation vector $\left[0, X_{0}^{F}, 0\right]^{\top}$, where $X_{0}^{F}$ is the height of the camera with respect to the floor. Then, ${ }^{M} \mathbf{T}_{F}={ }^{F} \mathbf{T}_{M}^{-1}$.

### 4.3.5. CLASSIFICATION OF LINES

Those lines $l_{i}$ whose minimum angular distance to their closest Manhattan direction $\mathbf{m}_{j}$ is below a threshold $\theta_{t h}$ are classified as lines in that direction $L_{j}$ :

$$
\begin{equation*}
\left|\measuredangle\left(\mathbf{n}_{l_{i}}, \mathbf{m}_{j}\right)-\frac{\pi}{2}\right|<\theta_{t h} \rightarrow l_{i} \in L_{j} \quad j=\{x, y, z\} \tag{4.5}
\end{equation*}
$$

where $\measuredangle$ indicates the angle between its two vector arguments. An example of lines classified is shown in (Figure 4.4b).

The horizon line is the line-image $l_{H}$ corresponding to the normal $\mathbf{n}_{l_{H}}=\mathbf{m}_{y}$ (drawn in dotted white line in Figure 4.4). Lines oriented in $\mathbf{m}_{x}$ and $\mathbf{m}_{z}$ are classified as upper lines $(\bar{L})$ when they are above horizon, and lower lines $(\underline{L})$ when they are below. Lines oriented in $\mathbf{m}_{y}\left(L_{y}\right)$ are classified as long lines when they have contour points above and below the horizon.

Some lines correspond to intersections of 3D planes extracted from the depth image. In order to detect such correspondences, we compute the 3D intersection lines of wall planes with the floor plane and between walls, that we call $L^{3 D}$. When there are two consecutive wall planes of the same orientation, the line of the border is computed instead. An example is shown in Figure 4.4a, where all $L^{3 D}$ have been drawn in yellow. Every 3D intersection line $l_{j}^{3 D}$ can be projected to the fisheye image and have its line normal computed ( $\mathbf{n}_{j}^{3 D}$ ). To perform the association, we evaluate the angular distance between their normals, and choose the closest if the angular distance is below a small threshold $\theta_{t h}$ :

$$
\begin{equation*}
\left|\measuredangle\left(\mathbf{n}_{l_{i}}, \mathbf{n}_{j}^{3 D}\right)\right|<\theta_{t h} \rightarrow l_{i} \in L^{3 D} \tag{4.6}
\end{equation*}
$$

Those lines supported by 3D evidence have more relevance when generating layout hypotheses. To refer to these lines we use the boolean function $\lambda\left(l_{i}\right)$ defined as:

$$
\lambda\left(l_{i}\right)= \begin{cases}1 & \text { if } l_{i} \in L^{3 D}  \tag{4.7}\\ 0 & \text { otherwise }\end{cases}
$$

### 4.4. LAYOUT ESTIMATION

To extend the depth information to the periphery, we look for features in the fisheye image that allow us to draw coherent layout hypotheses. We choose corners, i.e. points of intersection of three alternatively oriented structural planes in the 3D world, manifested in the image as intersections of lines. In Section 4.4.1 we describe how the corners are detected and scored for the next stage: the generation of layout hypotheses, explained in Section 4.4.2. Finally, we deal with the evaluation process in Section 4.4.3 and the final global scaling (which is to be applied when the floor has not been found) in Section 4.4.4.

### 4.4.1. CORNER EXTRACTION

We call corner $(C)$ in this context to the physical intersection of two walls and floor or ceiling. The junctions between these structural planes often produce detectable line segments in an image, whose intersection produces a corner detection. However, not all line intersections are actual corners, and not all actual corners have detectable line segments (e.g. occlusions or not enough contrast in the image).

To address this issue, first we translate the data from pixel space to 2D metric space by creating a floor plan projection of the lower lines (Section 4.4.1.1). Then we estimate the height of the ceiling and do a similar procedure with the upper lines (Section 4.4.1.2). We define the types of corners we detect in Section 4.4.1.3, covering all plausible cases with a minimum set of lines. At the end we describe how we score the corners in Section 4.4.1.4 to reward corners formed by more lines, longer lines, less distance from the lines to the intersection point and lines coming from 3D plane intersections.

### 4.4.1.1. FLOOR PLAN PROJECTION

The line segments from Section 4.3.5 represent just a projection, whose depth is unknown (except for those $l_{i} \in L^{3 D}$ ). From previous steps, we have the 3D location of at least one structural plane from the depth data (the floor plane $P_{\text {floor }}$ ). We use that plane to project all the lower lines and place them in a scaled 2D floor plan of the scene, we call XZ-plane. Notice that, in the cases the floor plane has not been found the algorithm continues with the virtual floor plane $P_{\text {floor }}^{*}$ normally. In those cases, the scale is lost in the process and it is recovered afterwards (Section 4.4.4).

We can get the ray emanating from the optical center to every contour point of every lower line $\underline{L}_{x}$ and $\underline{L}_{z}$ and intersect them with the $P_{f l o o r}$ in 3D (Figure 4.5). With the transformation ${ }^{M} \mathbf{T}_{F}$, we can transform these points from the camera reference frame $F$ to $M$, with the Manhattan directions and origin at the floor level. If we plot the transformed points in the axis $x-z$, we can get a 2D floor plan of the contours with scale (the $X Z$-plane in Figure 4.5). Notice also that we naively projected all lower lines, unaware if they actually belong to the wall-floor junction or to clutter, since it is impossible to know with the information we have. Further stages will choose the lines which most likely belong to the real junctions.

With the points now in this 2D projection, we define the angle of a point, $\alpha\left(\mathbf{X}_{i}\right)$, as the central angle of the arc between $-z$ and the radius from the origin to $\mathbf{X}_{i}$, as shown in Figure 4.5. Similarly, we compute the angle of a line, $\bar{\alpha}\left(l_{i}\right)$, as the central angle between its end points. In the Figure 4.5, $\bar{\alpha}\left(l_{1}\right)=\alpha\left(\mathbf{X}_{2}\right)-\alpha\left(\mathbf{X}_{1}\right)$. Note that these angles are defined not only by the value of the angle itself, but also by their starting and ending points. To extract the value of the angles we define the operation $\langle\bullet\rangle$ that returns a numeric value. For instance, $\left\langle\alpha\left(\mathbf{X}_{1}\right)\right\rangle=100^{\circ}$ and $\left\langle\bar{\alpha}\left(l_{1}\right)\right\rangle=40^{\circ}$ in Figure 4.5. Vertical lines $L_{y}$ are a special case only defined by a single angle $\alpha\left(L_{y}\right)$ and thus $\left\langle\bar{\alpha}\left(L_{y}\right)\right\rangle=0$. These definitions will be helpful in next stages.

(a)

Figure 4.5. Projection of the lower line segments of the image to the $P_{\text {floor }}$ in schematic 3D view and the resulting XZ-plane. Definition of angles of a point, $\alpha(\mathbf{X})$, and angle of a line, $\bar{\alpha}(l)$.

### 4.4.1.2. Ceiling plane projection

Similarly to the previous section, to get the ceiling plane projection, the rays traced from the optical center to the contour points of the upper lines must be intersected with the $P_{\text {ceil }}$. Since we consider floor and ceiling unique and symmetric, we know that the normal of the ceiling plane will be the same as the floor normal, but the distance to the origin is still unknown. To estimate the height of the ceiling ( $H_{\text {ceil }}$ ) we assume that, in the XZ-plane view, wall-floor $\left(l_{j}\right)$ and wall-ceiling $\left(\overline{l_{i}}\right)$ intersection segments of the same wall must be coincident. We can generate a $P_{\text {ceil }}$ at an arbitrary height, compute the 3D intersections of the projection rays and evaluate how well the contours from upper segments $c\left(\bar{l}_{i}\right)$ coincide with contours from lower segments $c\left(\underline{l}_{j}\right)$ in the XZ-plane. In Figure 4.6 there is an example with three different $H_{\text {ceil }} . H_{1}$ is too small and $H_{3}$ too big, so the segments of the floor do not match the segments of the ceiling in the XZplane. $H_{2}$ is the best one as contours from both planes match perfectly. Mathematically, $\forall l_{i} \in \bar{L}$ and $\forall l_{j} \in \underline{L}$, we express the overlap in two ways:

- Contour overlap. Denoted by $c\left(l_{i}\right) \cap c\left(l_{j}\right)$, determines the number of contours overlapping (i.e. in the 2D plane, contour points from ceiling and floor that are closer than a certain threshold).
- Angular overlap. Denoted by $\bar{\alpha}\left(l_{i}\right) \cap \bar{\alpha}\left(l_{j}\right)$, determines the shared angle of the two lines given the definitions from Section 4.4.1.1.

(a)

Figure 4.6. Projection of the upper line segments to three virtual ceiling planes ( $P_{\text {ceil }}$ ) at different $H_{\text {ceil }}$. The chosen $H_{\text {ceil }}$ is the one with highest angle coverage of overlapping line segments in the XZ-plane ( $\mathrm{H}_{2}$ in the example).

Then we propose the following optimization problem:

$$
\begin{equation*}
\underset{H_{c e i l}}{\arg \max }\left\langle\bar{A}\left(H_{\text {ceil }}\right)\right\rangle \tag{4.8}
\end{equation*}
$$

where $\bar{A}\left(H_{\text {ceil }}\right)$ is the Angular Coverage (AC) of the overlapping contours as a function of $H_{\text {ceil }}$. Here we introduce the concept of Angular Coverage (AC), denoted by the function $\bar{A}(\bullet)$, which returns the union of central angles around the origin that satisfy certain condition. In this case, the condition is having contour overlap of ceiling and floor line pairs, and $\bar{A}(\bullet)$ is a function of $H_{\text {ceil }}$. Mathematically:

$$
\begin{equation*}
\bar{A}\left(H_{\text {ceil }}\right)=\bigcup\left(\bar{\alpha}\left(l_{i}\right) \cap \bar{\alpha}\left(l_{j}\right)\right) \cdot\left(c\left(l_{i}\right) \cap c\left(l_{j}\right)>0\right) \tag{4.9}
\end{equation*}
$$

$\forall l_{i} \in \bar{L}, \forall l_{j} \in \underline{L}$, where $\left(c\left(l_{i}\right) \cap c\left(l_{j}\right)>0\right)$ returns 1 when there is contour overlap and zero otherwise. Higher $\left\langle\bar{A}\left(H_{\text {ceil }}\right)\right\rangle$ means that the lines whose contours are overlapping in the XZ-plane cover a greater angular area. In Figure 4.6, we can see the value of $\bar{A}$ of the three different $H_{\text {ceil }}$, where it can be visually appreciated why $H_{2}$ is the best result.

In [Perez-Yus et al., 2016b] we proposed an alternative method considering uniquely the number of contours overlapping. However, we found that the angular coverage method produces better results since they reward a consensus distributed in the scene instead of concentrated areas with many contours.


Figure 4.7. Real example of contour projection of lower lines (blue) and upper lines (red) to the XZ-plane (a) and 3D point cloud (b). The small circle represents the position of the camera system.

One of the advantages of working with scaled distances is that we can set reasonable valid ranges of heights to constrain the values of $H_{\text {ceil }}$. For example, we can set a default $H_{\text {ceil }}$ of 2.5 meters and a span of 2 to 3 meters to look for the $P_{\text {ceil }}$, which is very reasonable for indoor environments. If the problem has no solution between the valid range it could be due to clutter, undetected lines or absence of ceiling in the image. Then the algorithm goes on considering the default $H_{\text {ceil }}$. When the floor plane has not been found, the range of height values will not be constrained to the default values, which makes the system more prone to mismatches. In Figure 4.7a the XZ-plane with the contours of both lower and upper lines from the case from Figure 4.4b is shown. Those lines are plotted in 3D in Figure 4.7b over the initial point cloud, so we can see how the lines extend beyond the FOV of the RGB-D camera.

### 4.4.1.3. Corner definitions

Line segments are the main piece of information we use to create layout hypotheses. However, we do not know whether they come from actual wall-ceiling or wall-wall intersections, or from other elements of the scene. In the literature there are many approaches to tackle this problem. For instance, [Lee et al., 2009] defines a corner when a minimal set of three or four line segments in certain orientations are detected. This requires having uncluttered environments where most line segments can be perfectly detected. However, in the real world, occlusions or bad lighting conditions may cause some


Figure 4.8. (a) Graphical definition of a corner $C$ : the corner point $\mathbf{C}$, its line segments $\left(l_{x}, l_{y}, l_{z}\right)$ and the dist and leng functions used in our scoring method. (b) Four different types of corners we consider. Detected line segments as black thick lines and corresponding extracted corners as yellow circles.
contours to remain undetected. Other works such as [Lopez-Nicolas et al., 2014, Jia and $\mathrm{Li}, 2015]$ tend to give more emphasis to vertical lines and the extension of their segments in their corner definition, which may be problematic for the same reason as before. In a Manhattan World, two line segments are enough to define a corner.

An example of a corner is shown in Figure 4.8a, where the corner's 3D point, denoted as $\mathbf{C}$, is the intersection of the floor plane ( $P_{\text {floor }}$ ) and two walls ( $W_{x}, W_{z}$ ) with respective planes $P_{W_{x}}$ and $P_{W_{z}}$. The corner from Figure 4.8a represents a fully visible corner, since all three line segments and corner point can be extracted. However, there are some corners which are partially visible, meaning one of the walls and thus its intersection line segments are occluded by the other visible wall. There are also hidden corners, which have the corner point hidden by own occlusions of the walls from the room. All box-shape methods consider that all corners in the scene are fully visible corners (if they are inside the image). Our method aims to find all type of corners. We propose to use more relaxed requirements to define corners, using just one or two line segments, and then use a scoring function to select the most salient ones and favor their appearance in the layout hypotheses generation. In this section we detect the visible ones, and the hidden ones will be estimated while drawing hypotheses.

For the detection of visible corners, we consider four cases depending on the classification of the segments involved (Figure 4.8b):

1. Horizontal intersections ( $L_{x}-L_{z}$ ): These are by definition fully visible corners, formed by two lines in $x$ and $z$ respectively. If there is a $l_{y}$ passing through the intersection point $\mathbf{C}$, the contour points of $l_{y}$ are scaled by assuming they share the same wall as $l_{x}$ (i.e. 3D plane $P_{W_{x}}$ ) or $l_{z}\left(P_{W_{z}}\right)$. If the scaled 3D contours of $l_{y}$ have heights between 0 and $H_{\text {ceil }}$ the vertical is included to improve the score
of the corner.
2. Vertical intersections ( $L_{x}-L_{y}$ or $L_{y}-L_{z}$ ): These intersections could represent partially visible corners or fully visible corners with an undetected line. As with the previous case, the segment $l_{y}$ is scaled to verify plausibility.
3. Long vertical lines $\left(L_{y}\right)$ : These are added to include cases of misdetections of horizontal lines. Only the ones crossing the horizon are considered as they are more likely to be wall to wall intersections instead of clutter. The projection of their topmost contour point to the $P_{\text {ceil }}$ or the bottommost one to the $P_{\text {floor }}$ is considered, depending on which one makes the scaled $l_{y}$ not exceed the height of the ceiling.
4. Long horizontal lines ( $L_{x}$ or $L_{z}$ ): These are added since sometimes there is no visible or detected corner at the farther end of a corridor or a big room. We consider the possibility of long horizontal lines to intersect with the horizon. Horizontal lines are considered long if their length is over a threshold (we set 0.5 meters). To keep layouts of reasonable size we restrict the distance of intersection to a maximum of $D_{H}$ (in particular we set $D_{H}=10 \mathrm{~m}$ ).

These simple types of corner intersections include all necessary cases to build a layout of any shape. We call direction $(d)$ of the corner the position of their horizontal segments in the XZ-plane with respect to the corner point. For example, in the $x$ axis, a direction $d_{x}\left(C_{i}\right)=+1$ means that the corner $C_{i}$ has the $l_{x}$ defined from the corner point to the positive direction of the $x$ axis. A corner with no $l_{x}$ has $d_{x}\left(C_{i}\right)=0$. Similar definitions for $d_{z}\left(C_{i}\right)$. We also define the angle of a corner as the angle of the corner point $\alpha\left(\mathbf{C}_{i}\right)$ as well as the angle coverage of a corner $\bar{\alpha}\left(C_{i}\right)$ as the central angle of the arc between the minimum and the maximum angle of the contour points from $l_{x}$ or $l_{z}$, i.e. $\bar{\alpha}\left(C_{i}\right)=\max \left(\alpha\left(c\left(l_{x}, l_{z}\right)\right)-\min \left(\alpha\left(c\left(l_{x}, l_{z}\right)\right)\right.\right.$. Two simple examples of corners showing these parameter values are shown in Figure 4.9a. In the case of horizontal intersections, to evaluate if a line $l_{y}$ belongs to the corner, we check if the angle difference $\left|\left\langle\alpha\left(\mathbf{C}_{i}\right)\right\rangle-\left\langle\alpha\left(l_{y}\right)\right\rangle\right|<\theta_{t h}$.

### 4.4.1.4. CORNER SCORING

In a natural scene highly populated with line segments, the amount of line intersections and thus corner detections can be very high. Therefore, when generating hypotheses it will be difficult to find the best ones. To avoid excessive amount of corners, a solution is applying thresholds, but it is hard to tune the parameters correctly to make it work in all cases. Instead, we perform a scoring of corners to keep those with positive score and make high scored corners more relevant in the generation of hypotheses.

In particular we want to reward corners formed by line segments of great length and low distance from the segments to the corner point. To examine length and the proximity

$\mathrm{d}_{\mathrm{x}}\left(\mathrm{C}_{1}\right)=-1, \mathrm{~d}_{\mathrm{z}}\left(\mathrm{C}_{1}\right)=-1$


$$
\mathrm{d}_{\mathrm{x}}\left(\mathrm{C}_{2}\right)=0, \mathrm{~d}_{\mathrm{z}}\left(\mathrm{C}_{2}\right)=+1
$$

(a)

(b)

Figure 4.9. (a) XZ-plane view of two example corners showing their angles $\alpha\left(C_{i}\right)$, angle coverage $\bar{\alpha}\left(C_{i}\right)$ and directions $d_{x}$ and $d_{z}$ regarding the position of their line segments with respect to the plane axis. (b) Relevant corners in the scene plotted over the fisheye image as yellow circles with diameter proportional to their score.
between segment points, instead of using pixel distances, we reason in the 3D world with metric distances. Pixel distance is misleading, as it is affected by how far the points are from the camera, the perspective and the heavy distortion of the fisheye camera. Note that it is also difficult to deal with distances in the 3D world when there is no scale information available. With our system we integrate scale information in the process.

A corner $C$ is defined by a set of $N_{l}$ line segments, and its score $S_{C_{j}}$ depends on the number of lines and their respective score value $S_{l_{i}}$ :

$$
\begin{gather*}
S_{C_{j}}=N_{l_{j}} \cdot \sum_{i=1}^{N_{l_{j}}} S_{l_{i}}  \tag{4.10}\\
S_{l_{i}}=\left(\operatorname{leng}\left(l_{i}\right)-\operatorname{dist}\left(l_{i}, \mathbf{C}\right)\right) \cdot\left(1+\lambda\left(l_{i}\right)\right) \tag{4.11}
\end{gather*}
$$

where $\lambda\left(l_{i}\right)$ is defined in (4.7), leng $\left(l_{i}\right)$ measures the length of the line segment in meters, $\operatorname{dist}\left(l_{i}, \mathbf{C}\right)$ measures the distance of the closest point of the segment to the actual intersection point $\mathbf{C}$ in meters (Figure 4.8a). Note that $S_{C_{j}}$ computation includes a multiplication by $N_{l}$ to increase the score of corners supported by more lines. The line score for the corners in the horizon case is modified:

$$
\begin{equation*}
S_{l_{i}}=\operatorname{leng}\left(l_{i}\right) \cdot\left(1+\lambda\left(l_{i}\right)\right) \cdot\left(D_{H}>\operatorname{dist}\left(l_{i}, c\right)\right) \tag{4.12}
\end{equation*}
$$

After the extraction of corners we keep those with $S_{C}>0$. To avoid redundant corners, we merge those which are close to each other, have similar angle coverage and


Figure 4.10. Projection of the corners to the 3D point cloud as pink spheres with the line segments that form them, for each intersection case defined. Most relevant corners for the layout are outside the depth information range.
the same corner directions. When doing the merging we pick the maximum score among all corners involved. In [Perez-Yus et al., 2016b] we took the summation of the scores instead. However, we found that it rewarded too much specific areas crowded with line segments with no necessarily relevant corners. Finally, we assign a probability $\mathcal{P}$ to corners $C$ of occurring in the real world:

$$
\begin{equation*}
\mathcal{P}\left(C_{i}\right)=\frac{S_{C_{i}}}{\sum_{j=1}^{N_{C}} S_{C_{j}}} \tag{4.13}
\end{equation*}
$$

In Figure 4.9b there is an example of the 100 most probable corners represented as yellow circles with radius proportional to their probability. In Figure 4.10 there are corner results of a similar scene in a 3D point cloud, separated by the corner cases defined in Section 4.4.1.3.

### 4.4.2. LAYOUT HYPOTHESES GENERATION

We define completely a layout with their corners and the height of the ceiling, $\mathcal{L}=$ $\left\{C_{1} . . C_{N_{C}}, H_{\text {ceil }}\right\}$. The walls of the layout, $W$, are implicitly defined as the planes connecting two consecutive corners with a height $H_{\text {ceil }}$ (thus, there are $N_{C}$ walls). Each individual wall is noted as $w_{j} \in W$ and they are in practice used as virtual lines with similar properties. Since we have made a floor and ceiling projection, we reason with the position of the corners only in the 2D floor plan: corners are considered equally whether they have been detected on the floor or the ceiling. As in most indoor environments the level of clutter is higher in the lower part of the scene, having corners from the ceiling allows to provide results in difficult environments.

To get our layout solution, we generate a set of hypotheses from where a last evaluation process will select the better one. To generate a layout hypothesis, we already have $H_{\text {ceil }}$ and we just need to select the corners among the extracted set of corners with $S_{C}>0$. Once a hypothesis $\mathcal{L}$ has been generated, with the relative position of the camera to the scene ${ }^{M} \mathbf{T}_{F}$ and the calibration of the system we have enough information to build the complete 3D reconstruction of the scene.

In this section, first we present in Section 4.4.2.1 the conditions for a layout to be geometrically valid. Then, we generate the layouts as described in Section 4.4.2.2. Additionally, we introduce a pre-filtering procedure of corner connections to verify its plausible association before start generating layouts (Section 4.4.2.3).

### 4.4.2.1. CONDITIONS FOR A VALID LAYOUT

To generate hypotheses we do not impose any condition about the shape of the scene in order to provide valid solutions to any kind of indoor environment. However, we consider a layout valid when it satisfies the next conditions:


FIGURE 4.11. Examples of layouts that do not satisfy our conditions for a valid layout described in Section 4.4.2.1. In each case, corners are numbered blue crosses and the camera viewpoint is the green circled cross. In E the depth points and field of view in pink. In F1 and F2, line segments of the corners in red. In F2, the view of the segments of corner 3 in yellow.
A. The walls must follow the Manhattan World convention: a wall directed in $\mathbf{m}_{x}$, must be followed by a wall directed in $\mathbf{m}_{z}$, and vice versa.

A1. There must be no walls following other directions.
A2. Two consecutive walls must not have the same direction.
B. The layout must not have any non-consecutive wall intersecting another.
C. The layout must be closed, i.e. the wall sequence must end in the same point it begins.
D. The camera must be inside the layout.
E. The layout must not contradict the information from the depth camera, i.e. there cannot be a wall in front of the given depth map.
F. The layout must not contradict the information given by the line segments of the corners, since they are considered directly visible.

F1. Each wall connecting two corners must be on their corresponding line segments, if any.


Figure 4.12. Example of layout generation in the XZ-plane given pre-selected corners from our set (in blue, with their respective red horizontal contours and a green circle when there is a vertical line). The camera position is the green circled cross. Detailed explanation of the procedure is provided in the text.

F2. Walls are opaque, so they must not be in front of any line segment since that would mean it is visible through the wall.

Note that, in Figure 4.11 we show respective examples of layout proposals that do not satisfy each condition.

### 4.4.2.2. GENERATION OF LAYOUT HYPOTHESES

In the general case, our algorithm looks for a number of hypotheses by iterating following these steps (note that the explanation of this section can be followed using the graphical sample case from Figure 4.12):

1. Using the probability from (4.13), we randomly choose a number of corners from the set to generate a hypothesis. As the view of the scene is not complete and we do not impose any condition about the shape of the room, the number of corners to select cannot be fixed, and therefore it must be randomly chosen every time a hypothesis is generated. We believe a reasonable number of corners to draw is between 2 and 5 . In the example, we draw four corners.
2. The selected corners are ordered clockwise considering their angle $\alpha(C)$ as shown in the Figure 4.12 (1).
3. The walls from the layout are going to be created joining every corner with the following one. The angle $\beta$ between corners is observed to verify if the walls are oriented according to the Manhattan convention (Figure 4.12 (2)):

- If it is closer than an angular threshold to $0, \frac{\pi}{2},-\frac{\pi}{2}$ or $\pi$, then it is accepted as valid as it is (case between corners 1 and 2 and corners 2 and 3 in Figure 4.12).
- Else (angle between corners 3 and 4 in Figure 4.12), two additional corners ( $a$ and $b$ ) are created as shown in Figure 4.12 (3).

4. In case two additional corners are defined, the generation of layout goes on with consecutive corners in separate branches, as the cases (4a) and (4b) in Figure 4.12.
5. At any point the line segments composing a corner can invalidate a layout generation branch (condition F). For instance, in (4b) the wall from corner 3 to corner $b$ goes in direction $-x$, but there is a line segment that defines corner 3 in direction $-x$ as well that goes in opposite direction. Solution (4a) matches the line segments from corner 3 perfectly.
6. Continue in every branch until the layout is closed (Figure 4.12 (5a)) or the solution is invalid (Figure 4.12 (5b)). It can be seen how the layout is completed by defining an additional corner $c$ as performed before, and no line segments contradict the wall distribution.

A generated hypothesis has to satisfy the conditions mentioned in Section 4.4.2.1 or could be discarded. To verify conditions D and E we can treat the set of corners as a polygon and verify if the camera and depth points are inside it. For the depth points we set a threshold to determine the minimum percentage of points to be inside the polygon. To verify condition B we check the polygon does not self-intersect.

One of the keypoints of this method is that hidden corners can be estimated using the Manhattan assumption, even if there is no visible evidence of the presence of the corner in the image (e.g. in Figure 4.12 corners $a$ and $c$ were not detected but its definition provides a valid closed Manhattan layout). This means that the algorithm can handle heavy occlusions and still provide coherent results. Besides, whenever the information from behind the camera point is not enough to provide closing points, we can assume the walls extend beyond the field of view towards the rear vanishing point (following $-z$ ) in order to keep our layout closed (condition C). For this operation we need to place additional corners at the horizon, at a previously determined distance ( $D_{H}=10 \mathrm{~m}$ in our experiments), and watch if these solution do not break other rules. Note that when we introduce additional corners to perform the rear extension it is to keep the model consistent, but the final reconstruction only extends to where the field of view of the fisheye camera reaches.

In Figure 4.13 there is an example of a layout hypothesis from the scene from Figure 4.9, similar to the one from Figure 4.12. In Figure 4.13a the original corners (in


Figure 4.13. (a) Layout hypotheses example with its original corners in yellow with their line segments shown and the additional corners in light blue. (b) Colored wall-floor-ceiling distribution of the hypotheses. (c) XZ-plane with the layout overlaid. (d) Corresponding depth map of the hypotheses with scale in meters. (e) Different views of the corresponding 3D point cloud.
yellow) and the line segments that define them have been displayed in the image along with the additional corners (in light blue). The resulting wall distribution colored is shown in Figure 4.13b. In these labeled images each surface orientation is colored distinctly (Red-Green-Blue for surfaces with normals in $x-y-z$ ). In Figure 4.13c the solution has been plotted over the XZ-plane. As the XZ-plane is scaled and the $H_{\text {ceil }}$ have been estimated we can generate a 3D depth map of the scene (Figure 4.13d). The depth map can be used to recover the 3D point cloud of the complete layout, as it can be seen in Figure 4.13e.

### 4.4.2.3. PRE-FILTERING OF CORNER CONNECTIONS

Given the considerations from Section 4.4.2.1, before start drawing hypotheses we pre-filter the corners that can go with each other in the same layout. This process was not performed in [Perez-Yus et al., 2016b], but has proven to help reducing the computation time and providing better hypotheses. At the end of this process we obtain a matrix $\mathcal{M}$ of $N_{C} \times N_{C}$ which relates every corner to each other and indicates if they can be connected to each other with one wall, two walls (and thus introducing an additional corner) or cannot be connected by any mean. Besides, if they can be connected with two walls, we compute where should be the additional corner and if there is more than one option. Since we enforce closed layouts, we also verify if a closing strategy in the rear vanishing point can be created between two corners. Our filtering steps are the following:

- Considering the quadrant the corners are located on the XZ-plane, there are some detectable cases of impossible corners. For instance, a corner $C_{i}$ in the $\{+x,+z\}$ quadrant cannot have $d_{x}\left(C_{i}\right)=-1$ and $d_{z}\left(C_{i}\right)=+1$ since that would mean $l_{z}$ would be occluded by the wall of $l_{x}$, which makes impossible for the segment $l_{z}$ to be visible and therefore detected. Corners 1 and 3 in Figure 4.11 (F1) are impossible corners by this reasoning, and should not have been considered to generate hypotheses.
- Filter those corners close to each other, since we consider rooms with walls relatively large. To check this we compute distances between corner points and apply a threshold (we choose $H_{\text {ceil }} / 10$ ).
- Verify if they can be connected clockwise. This is the order we use to generate the hypotheses, so it is only necessary to check corner connections one way.
- Compute the orientation of the walls with respect to the Manhattan directions to verify if they can be connected with a single wall or two (condition 1).
- Single wall: if line segments are in the wall's direction, they must be facing each other (condition F1).
- Two walls: no shared angle coverage between corners (condition F2). Get the two positions of the additional corners and discard those cases whose walls do not satisfy conditions F1 and F2.
- Look if there are closing strategies in case they are the first and last corners once ordered with $\alpha$. Check if their corner points have different signs in the $x$ coordinate (otherwise it would not satisfy condition D).

After the pre-filtering process, there might be some corners that cannot be connected to any other, which are discarded. With matrix $\mathcal{M}$, right after performing step 1 from the generation of hypotheses we can quickly discard those cases of pseudo-randomly chosen corners and pick other ones, instead of going through all the process avoiding useless steps until finding that some condition is not met.

### 4.4.3. Evaluation of the hypotheses

To evaluate hypotheses we present three new methods and another adapted from the state of the art (Orientation Map [Lee et al., 2009]). Using the orientation map usually provides better results, but it requires the previous computation of the map itself, which can be very time consuming compared to the other alternatives. In this work we introduce an additional method not included in [Perez-Yus et al., 2016b] based on the Angle Coverage concept introduced in Section 4.4.1.2. It clearly outperforms our other two alternatives, being comparable to [Lee et al., 2009] while much faster. The following sections will describe each separate method.

### 4.4.3.1. Sum of Scores (SS)

We define the score of a hypothesis as the sum of scores of the corners that have been used to generate it. The additional corners defined to generate Manhattan layouts have a score of zero.

### 4.4.3.2. Sum of Edges (SE)

The polygon defined by the corners of the hypotheses as vertices can be drawn on the XZ-plane in order to choose the hypotheses which overlaps the most with the observed contours, i.e. the layout $\mathcal{L}_{k}$ with a set of walls $W^{k}$ such that the value of $c\left(l_{i}\right)_{i} \cap c\left(w_{j}\right)$ is maximum $\forall l_{i} \in L, \forall w_{j} \in W^{k}$.

### 4.4.3.3. Angle coverage (AC)

In this case we draw the polygon in the XZ-plane and we compute the angular coverage of the layout $(\bar{A}(\mathcal{L}))$ similar to the process of Section 4.4.1.2, but considering all line segments and wall lines instead of floor and ceiling lines. Mathematically:

$$
\begin{equation*}
\bar{A}\left(\mathcal{L}_{k}\right)=\bigcup\left(\bar{\alpha}\left(l_{i}\right) \cap \bar{\alpha}\left(w_{j}\right)\right) \cdot\left(c\left(l_{i}\right) \cap c\left(w_{j}\right)>0\right) \tag{4.14}
\end{equation*}
$$

$\forall l_{i} \in L, \forall w_{j} \in W^{k}$, where $w$ are the individual walls of the set $W^{k}$ of the layout $\mathcal{L}_{k}$. The hypothesis $\mathcal{L}_{k}$ with the highest angular coverage value $\left\langle\bar{A}\left(\mathcal{L}_{k}\right)\right\rangle$ is selected.

### 4.4.3.4. Orientation Map (OM)

It requires to build a reference image called orientation map [Lee et al., 2009], which is an image whose pixels encode the believed orientation given the line segments for perspective cameras. To build that image we create a set of overlapping perspective images from the fisheye image, apply the orientation map algorithm from [Lee et al., 2009] in each one of them and finally stitch them back together to form an omnidirectional orientation map. The evaluation consists in selecting the layout hypothesis that has better fitness between pixels with the same orientation. To compute that fitness, we generate for each hypothesis a labeled image such as the one Figure 4.13c.

### 4.4.4. SCALING OF HYPOTHESES

The layouts can be generated and evaluated as described above with no scale information. However, then the thresholds and parameters are harder to tune (e.g. no valid height of the room estimation), and depth cannot be used to discard incoherent layouts. Thus, the normal execution should include floor detection and scaling from the beginning in order to obtain better results. Nevertheless, in the cases the floor cannot be detected, we include a method to scale the layouts once generated, so that the height range and depth information conditions can be verified.

To perform layout scaling from a hypothesis, we should have the hypotheses defined ( $\mathcal{L}=\left\{C_{1} . . C_{N_{C}}, H_{\text {ceil }}\right\}$ ). Then we can get a labeled image as the one shown in Figure 4.13 c . If we compare that image to the labeled image with the areas of the planes obtained in the beginning, we should have an overlap between both labeled images in wall surfaces with equal orientation. The labeled wall/labeled plane pair with highest overlap would be used to provide the scale. The quotient between the distances to the origin of the corresponding planes will reveal the scale. Thus, the corner points or obtained point cloud can be simply multiplied to that scale in order to get the whole scaled 3D reconstruction.

### 4.5. EXPERIMENTS

In this work, we use a novel camera system with fisheye and depth image. Many datasets for indoor layout retrieval are usually based on conventional images, but not so many on omni-images, and none combining them with depth. For the experimental evaluation we have collected our own set of images with two different devices (Figure 4.14):


Figure 4.14. The two devices used to collect the data for our experiments. On the left, an RGB-D camera (ASUS Xtion Pro Live) with an adjacent fisheye camera. On the right, the Google Tango Development Kit.

- Conventional RGB-D system with fisheye camera: A hybrid camera system built and calibrated by ourselves [Perez-Yus et al., 2016a]. With this system we have a dataset with 70 image pairs from indoor scenarios, including 23 from corridors/entrances, 15 from four different bedrooms, 4 from a bathroom, 12 from two living rooms, 4 from a kitchen and 12 from two small cluttered rooms. We have manually labeled the 70 images of the dataset to provide a per pixel label of the three main classes (walls in $\mathbf{m}_{x}$ or $\mathbf{m}_{z}$ and floor/ceiling).
- Google Tango: A tablet for developers with built-in depth sensor and fisheye. The Tango technology is now available in commercial phones from well known brands (e.g. Lenovo, Asus). We have taken several images from similar environments as the previous device to test applicability of the method with commercial systems.

Quantitative detailed analysis is provided with the larger dataset from the first device. Unless noted, this is the dataset we use in our experiments. In the following sections, we analyze the performance of the corner extraction (Section 4.5.1) and the layout estimation (Section 4.5.2), which are the most important parts of our algorithm. Additionally, we provide some insight about using the proposed camera configuration in Section 4.5.3, and more results using the Google Tango dataset in Section 4.5.4.

### 4.5.1. CORNER EXTRACTION

In this section we analyze the capacity of the method to extract the relevant corners for the layout estimation, but also its limitations. Ultimately, the success of the system depends on the good extraction of corners. While our layout estimation process allows to


Figure 4.15. Examples of corner detections, represented by yellow circles, in four different scenes. In these examples all the important corners have been found, plus some outliers.
introduce additional undetected corners, they are placed between corners that had been extracted beforehand.

### 4.5.1.1. PERFORMANCE ANALYSIS OF CORNER EXTRACTION

In our method, we extract and rank the corners depending on their score, but for the layout extraction we only keep the 100 better ranked corners. The first experiment analyzes how well the corners found with our method correspond to the real world corners. To perform this experiment we have annotated the number of corners that should be found in the image in order to provide the best layout solution (206 corners in total). Then, we visually inspected if these corners are actually among the 100 best ranked corners. The ratio of number of corners found over number of corners to find is of $191 / 206 \rightarrow 92.7 \%$. All the important corners were found in 58 of the 70 images. This numbers are highly satisfactory, since some of the corners not found can also be estimated during layout hypotheses generation. In Figure 4.15 there are some cases where all the important corners have been detected. We can also see how very often several corner detections
appear around the same real world corner. This multiplicity is due to similar detections from close lines corresponding to different objects (e.g. embellishments). There are also some outliers (i.e. corner detections not corresponding to real world corners) that the layout estimation process has to overcome.

Next, we analyze the main difficulties and causes of failure that the corner extraction problem has, also showing some examples. Given that the basic elements to find corners are the lines, missing some of them may be critical. The line extraction method requires edges to be detected properly, which may not happen when there is low contrast in the image, e.g. because of non-existent color shift, bad lighting conditions or motion blur. Some of the reasons that harmed the results in some experiments are the following: in Figure 4.16 (a-c) most lines in the ceiling were not detected and thus, the height of the ceiling could not be obtained properly and some relevant corners are missing. Particularly, in Figure 4.16 c the illumination from the lamp itself casts a shadow that resembles a wall-ceiling intersection. The opposite problem arises sometimes as well, i.e. lines coming from textures (Figure 4.16d) or objects irrelevant for the task (Figure 4.16e) provoke accumulation of misleading corners.

The ceiling plane may be wrongly obtained when fortuitous scene configurations and line distributions occur. For example, in Figure 4.16 f the rectangular rug resembles the rectangular shape of the ceiling and thus the $H_{\text {ceil }}$ is such that makes the contour of the ceiling and the rug overlap in the XZ-plane. In Figure 4.16 g the wardrobe has parallel lines that deceive the ceiling plane extraction as well. This is a problematic issue, since it affects the layout proposals that combine corners from floor and ceiling (i.e. if the ceiling plane is not right the corner intersection in the ceiling will not be exactly on top of the intersection in the floor). Since we introduce depth information of the process, the ranges for a valid $H_{\text {ceil }}$ are restricted to common ones (e.g. from 2 to 3 meters). In Figure 4.16 h we can see the previous case by removing the input from the depth camera: the lines from the furniture and posters create a situation where the best ceiling plane solution is very inaccurate. Providing scale and restricting the measurements to natural ranges produces that even when the ceiling plane is not properly found, the value it takes is not very far off the real solution. In average, the ratio of success of finding a ceiling plane within a few centimeters error is about $80 \%$.

Despite the aforementioned cases, most important corners are generally well extracted in our experiments. The majority of these problems are derived from the line extraction method of our current implementation and not the method itself. More sophisticated approaches of line detection in the vein of [Von Gioi et al., 2010] could be used to improve the results. Additionally, some filtering methods could be used to remove textures and highlight borders (e.g. [Zhang et al., 2014a]), even deep learning methods have been used to detect only structural edges and ignore those from other objects or clutter [Mallya and Lazebnik, 2015]. However, this line of research was out of scope for this work, and instead we focus on developing a layout estimation method robust enough to overcome the fact that not all corners are always detected.


Figure 4.16. Examples of corner detections damaged by scene conditions. The pink rectangle points at the region where failure happens.

### 4.5.1.2. TIME ANALYSIS OF CORNER EXTRACTION

In this work, the experiments were performed offline, in single-image and without major effort in optimizing the implementation to make it able to run in real-time. Consequently, we used the images at full resolution: fisheye image of $2560 \times 1920$ and depth image of $640 \times 480$. We annotated the time spent to perform all the operations including reading the images and the processes that ended up with a set of corners with the pre-filtering of connections between them already computed. The times range from 8.17 to 24.58 seconds depending on the complexity of the scene (i.e. amount of lines and planes) with an average of 15.45 seconds. However, testing the behavior on the Google Tango dataset (fisheye image of $640 \times 480$ and depth image of $320 \times 180$ ) the times range from 1.25 to 2.4 seconds, with an average of 1.63 seconds. We believe our system could be implemented in real time by using smaller images and a more efficient programming language for the task (e.g. C++ instead of Matlab).

### 4.5.2. LAYOUT ESTIMATION

In this section, we analyze quantitatively the results of the system regarding layout estimation: from hypotheses generation to the evaluation and the final result. For this we use the dataset of 70 images for which we have labeled the ground truth. Our ground truth is the labeled images such as the one from Figure 4.13c, where each color represents a layout surface of different orientation. Since only the structural information of the scene is to be extracted, in the tagging we ignore all the objects unless they cover entire walls (e.g. wardrobes or bookshelves). We only extract single room layouts, meaning that open doors are ignored during tagging phase as well. The measure employed is the percentage of pixels correctly tagged over the totality of pixels from the ground truth, which we call Pixel Accuracy (PA). With that metric, we analyze the quality of our solution depending on the number of hypotheses drawn and the evaluation method.

### 4.5.2.1. NUMBER OF HYPOTHESES

This experiment analyzes how the PA changes depending on the number of hypotheses to draw with the four evaluation methods presented. The objective of this experiment is to observe the behavior and determine how many hypotheses we need to have the best results. We have registered the mean PA obtained from 5 to 50 hypotheses by intervals of 5 and from 50 to 200 by intervals of 10. The resulting graph is shown in Figure 4.17. At a glance we can see two distinct trends: The Sum of Scores (SS) and Sum of Edges (SE) evaluation methods present lower score and a small decline through iterations, whereas Angular Coverage (AC) and Orientation Map (OM) rise briefly at the beginning until they reach a steady maximum.


Figure 4.17. Pixel accuracy over the number of hypotheses generated.

However, in all four cases we can note that the variation of PA through iterations is negligible. This is a consequence of the good performance of the detection and scoring of the corners, which makes higher scored corners more likely to appear in layout hypotheses. In many cases the highest scored corners are the real world corners that we are looking for. Thus, a small number of hypotheses is likely to already provide a good result. Other cases may have a more complex corner distribution, which leads to more variety of layout hypotheses among which the SS and SE evaluation methods may find one that fits better their criterion. On the other hand, AC and OM prove to be better for the task, since they look for the best distributed consensus in the scene. Therefore, these methods tend to improve with a larger variety of hypotheses.

The best number of hypotheses to draw will depend on the method. For SS and SE, lower number of hypotheses improves the results, but at least a few should be required (otherwise there is a risk of getting oddly-shaped layouts). Thus, 5-10 hypotheses seem reasonable. For AC and OM, on the other hand, the PA rises until 30-40 hypotheses, and the improvements afterwards are marginal. In all cases, we choose a very reduced number of hypotheses, substantially less than other similar works [Zhang et al., 2014b]. In the following experiment we continue discussing the evaluation methods.


Figure 4.18. Boxplot of the results using the four evaluation methods with the set of 70 images. The graph is divided in results using depth (on the left, in green) and without depth (on the right, in red). For all cases the black line marks the mean value, the black dotted line the median value, the dark rectangles are the standard error of the mean (SEM) with $95 \%$ of confidence and the bright rectangles the standard deviation (SD). The individual values per image are also scattered over each column.

### 4.5.2.2. COMPARISON OF EVALUATION METHODS

To provide a more meaningful discussion about the evaluation methods, the mean PA displayed in Figure 4.17 is not enough. In Figure 4.18, there is another boxplottype graph showing the distribution of pixel accuracy in the 70 images with the four evaluation methods at 50 hypotheses. For each column we show the mean (black line), median (black dotted line), standard error of the mean (SEM) at $95 \%$ of confidence (dark rectangle), and standard deviation (SD) (bright rectangle). The values of the mean and median are shown in the Table 4.1. Additionally, each individual result of the 70 images is also scattered on the graph for visualization purposes. Analyzing only the left part of the graph (the general case, with depth information), we can see how the SS and SE evaluation methods are able to tag correctly a median of $86 \%$ and $85.7 \%$ of the pixels in the image respectively. Both the AC and OM reach over $90 \%$, particularly $90.3 \%$ and $91.5 \%$ of PA. While all methods perform well, the AC and OM are clearly the best. The OM has the better scoring overall, but the AC has smaller standard deviation and less outliers.

However, accuracy is not the only factor to compare methods, so we extend the

Table 4.1. Mean pixel accuracy of the system with and without depth information (\%).

| Method | With depth |  | No depth |  |
| :--- | :---: | :---: | :---: | :---: |
|  | Mean | Median | Mean | Median |
| SS | 84.96 | 86.05 | 81.62 | 85.13 |
| SE | 84.59 | 85.73 | 81.00 | 81.70 |
| AC | 89.63 | 90.34 | 85.90 | 87.43 |
| OM | 90.38 | 91.53 | 87.97 | 89.38 |

Table 4.2. Comparison of computation time of each stage of the evaluation for each method in our current implementation (in milliseconds).

| Stage of evaluation | SS | SE | AC | OM |
| :--- | :---: | :---: | :---: | :---: |
| Generate orientation map | - | - | - | 18000 |
| Generate 1 labeled image | - | - | - | 50 |
| Evaluate 1 hypotheses | 0.05 | 0.4 | 1.8 | 2.5 |
| Total (1 hypotheses) | 0.05 | 0.4 | 1.8 | 18052.5 |
| Total (50 hypotheses) | 2 | 20 | 90 | 20625 |

experiments to test efficiency in terms of computation time. In the Table 4.2 there is a breakdown of the mean times in our current implementation. The first three methods (SS, SE and AC) only require simple operations, and thus, are extremely fast (less than 2 milliseconds). On the other hand, the OM is very slow in comparison. Just to generate the map, assuming we have the lines and vanishing points extracted, it takes around 18 seconds. Then it needs to compute the corresponding labeled image per hypothesis, in order to find the best fitting one. To save time we resize the orientation map and labeled images by a scale of 0.25 . Then, generating labeled images takes about 0.05 second/hypothesis and selecting the better one takes 2.5 millisecond/hypothesis. Thus, for example, evaluating 50 hypotheses would take about 0.002 seconds to the SS, 0.02 seconds to the SE and 0.09 seconds the AC. The OM method would take $18+0.05 \times$ $50+0.0025 \times 50=20.625$ seconds. The difference between OM and the other three methods is of several levels of magnitude. Thus, considering the small PA value shift between AC and OM , when time is a requirement, AC is much better for the task.

### 4.5.2.3. PERFORMANCE UNDER DIFFERENT TYPES OF SCENES

A breakdown of the results depending on the type of room is provided in Table 4.3. In general we have experienced better performance in environments where structural lines can be easily seen. For example, corridors have often less objects occluding the important lines. On the other hand, corridors have often more complex shapes. Our method is able to overcome complex shapes in most cases as the high scores in corridors

Table 4.3. Mean pixel accuracy depending on the type of scene tested (\%).

| Room | SS | SE | AC | OM |
| :--- | :---: | :---: | :---: | :---: |
| Corridor | 89.52 | 85.92 | 93.53 | 91.8 |
| Bedroom | 84.63 | 85.57 | 87.64 | 90.74 |
| Bathroom | 80.10 | 83.87 | 86.84 | 86.42 |
| Living Room | 85.03 | 85.70 | 89.03 | 90.18 |
| Kitchen | 73.02 | 79.77 | 82.87 | 87.61 |
| Other | 82.52 | 82.13 | 88.29 | 89.78 |

show.
In contrast, bathrooms and kitchens are the most problematic. They are usually crowded with objects and cabinets, even mirrors in the case of the bathroom, which are often problematic in any computer vision algorithm. The rest of the rooms are very scene dependent, and it is harder to establish any correlation in type of room and results. As mentioned in Section 4.5.1.1, the results depend on the specifics of each scene, including parameters such as illumination.

### 4.5.2.4. Scaled reconstruction of scenes

In Figure 4.19 there are some examples of 3D reconstructions obtained with our method. We show the fisheye with the depth information that we use as input of the system to visualize how much the depth has been extended. It can be seen that the system is able to reconstruct not only 'box-shaped' rooms, looking at the corridor or bedroom scenes. These results are scaled with the depth information provided, so in a single shot our system is able to get the whole scene at once. We believe this information could be valuable for many tasks.

We have to note that in all cases this is an estimation of the layout, but the only information that is fully reliable all the time is the one that comes from the depth information. Our layout solution can be merged with the initial depth so we can actually use both sources of data at the same time to our advantage. The depth image provides a safe zone where we know for certain what is in front of the camera, but we also have spatial context of the room we are in, enabling many possibilities of higher level reasoning that extends what a conventional depth camera can do, without diminishing its advantages. We additionally provide a video ${ }^{2}$ which, besides a brief description of the method, shows more examples of scaled room reconstructions and visual comparison with the original point clouds.

[^5]

Figure 4.19. Pair of images of fisheye images with the depth information from the depth camera overlaid and the 3D layouts we are able to retrieve corresponding to each case.

### 4.5.3. AdVANTAGES OF THE CAMERA PAIRING

We explore the benefits of using our camera system compared to merely using a fisheye camera. The main one is that the scale of the scene would be lost without the depth camera. Additionally, we repeated the experiments removing all depth information throughout the system in order to numerically observe how the results are affected. The absence of depth affects the computation of the VPs, the scoring of lines, the retrieval of the $H_{\text {ceil }}$ and the elimination of contradictory hypotheses. In particular, in Figure 4.16h there is an example of corner extraction without depth failing at getting the ceiling plane.

A comparison of results with 50 hypotheses with and without depth information is shown in the Figure 4.18 and Table 4.1. It can be observed that the standard deviation of PA increases since many cases result in much lower scores. Consequently, the mean pixel accuracy decreases about $4 \%$. However, observing the per-image values on the Figure 4.18, in many cases the results are not affected by the lack of depth, but in some others the results are so bad that it is very noticeable in the mean value. This is also observable on the median value, which does not experiment so much decrease in comparison. Thus, the depth not only provides scale, but it also helps in many individual cases.

### 4.5.4. Results with Google Tango

We want to show the applicability of the method with commercial devices, such as the Google Tango (Figure 4.14). Even though the device at our disposal is a Development Kit, there are several phones with the same technology already in the market. Among other sensors, this device includes: a depth camera which is similar to the other camera system but with less resolution ( $320 \times 180$ instead of $640 \times 480$ ), and a motion-tracking camera which basically is a fisheye of about $170^{\circ}$ of field of view and resolution of $640 \times 480$. The simultaneous depth and fisheye image pairs have been captured using Tango ROS Streamer ${ }^{3}$.

The decrease in the resolution does not affect the quality of the results notably. Note that one of the first operations with the depth information consists on downsizing the point cloud. In the fisheye camera the loss of information is not significant and it actually helps speeding up the algorithm. However, the decrease in field of view is quite relevant for the task, since the lesser spatial view of the environment we have, the lesser likelihood of finding the most useful corners. In order to increase the amount of lines from the ceiling (which usually belong to less cluttered areas) the camera could be pointing slightly upwards than the previous device. This may result in loss of floor plane view, and thus scaled layout recovery. However, we can proceed with the non-scaled layout recovery and apply the scaling procedure presented in Section 4.4.4 to recover the scaled layout.

[^6]In Figure 4.20 we show several examples with images from the Tango device in similar type of views to the first dataset, where the non-scaled layout recovery with Angle Coverage method and final scaling have been applied. This shows that our method is able to estimate the scaled layout when the floor is not in the image, and thus with no restrictions about how the camera is posed in the scene. With the previous dataset most images had views of the floor and we could not show this feature. While this alternative approach to solve the problem works for most scenes, it is still useful to have the floor and thus the scale since the beginning. For example, the second example include depth points out of the room through the open door, which breaks condition E (Section 4.4.2.1). However, since when generating hypotheses we have no scale, we cannot use that condition to discard the hypotheses. Apart from these types of exceptional cases, our method is able to extend the depth information by extracting the correct layout.

### 4.6. DISCUSSION

In this chapter, we have presented a new method to extend the 3D information of a depth camera to a field of view of over 180 degrees. In particular, we propose a novel spatial layout retrieval algorithm, whose main novelty is combining a fisheye and a depth camera. The large field of view helps to use information from both the ceiling and the floor, which is helpful when there is clutter in the scene. The depth information helps by providing scale, necessary for the final 3D reconstruction, and by enhancing the performance of the method. Experimental evaluation with real images of indoor environments shows good results in terms of accuracy, improving the state of the art in functionality: our method has less layout shape restrictions, needs fewer hypotheses and provides full-scaled 3D models of the scene in a single shot. One of the advantages of returning a full-scaled reconstruction is that it complements the information coming from the depth camera: besides the small part of the scene reliably captured by the depth camera, now it is possible to have a good estimation of the surroundings to over 180 degrees. This kind of information could be useful in fields such as robotics, augmented reality and assistive computer vision. Additionally, the method has been tested with data from a portable consumer device successfully, showing great potential for the future, especially regarding the possibility of using it in a wearable configuration.


Figure 4.20. Twelve examples of application of our method with Google Tango device. On the left of each case the fisheye image with the depth points projected in colors (variable color with depth). On the right, the resulting 3D point cloud. For visualization purposes, the initial 3D point cloud is also displayed in color to show how the resulting cloud has been scaled and fits accordingly.

## 5

## ICONIC REPRESENTATION FOR NAVIGATION WITH PROSTHETIC VISION

In the last part of this thesis we address the communication of perceived information to the visually impaired people. Recent research demonstrates that visual prostheses are able to provide visual perception to people with some kind of blindness. In visual prostheses, image information from the scene is transformed to a phosphene pattern to be sent to the implant. This is a complex problem where the main challenge is the very limited spatial and intensity resolution of the phosphene patterns. Moreover, depth perception, which is relevant to perform agile navigation, is lost, and codifying the semantic information to phosphene patterns remains an open problem. In this chapter, we consider the framework of perception for navigation where aspects such as obstacle avoidance are critical. We propose using a head-mounted RGB-D camera to detect free-space, obstacles and scene direction in front of the user. The main contribution is a new approach to represent depth information and provide motion cues by using particular phosphene patterns. The effectiveness of this approach is tested in simulation with real data from indoor environments.


### 5.1. Introduction

The ability to navigate and move around complex or unfamiliar environments is essential for people, and this is a non-trivial task to be automated. People solve these tasks primarily through vision, combined with their ability to memorize and learn. These tasks are even more critical for visually impaired people since additional personal safety issues appear. While mobility aids such as the white cane are helpful in short-range navigation, the usage of cameras enable the recovery of mid- and long-range information from the environment and thus, a more effective navigation. A key issue in Navigation Assistance for Visually Impaired (NAVI) is obstacle avoidance. Different approaches for NAVI have been developed based on vision sensors such as in [Wong et al., 2003, Peasley and Birchfield, 2013, Schafer et al., 2008, Aladren et al., 2016], or with other types of sensors [Dakopoulos and Bourbakis, 2010, Oktem et al., 2008, Guimaraes et al., 2013]. In the context of prosthetic vision, different visual processing techniques were proposed for obstacle avoidance [Stacey et al., 2011, Weiland et al., 2012,McCarthy et al., 2011,Li et al., 2012].

In the following sections we provide some background on the topic of prosthetic vision. Then, we describe the main aspects of phosphene mapping techniques and how they are usually tested with users. Finally, we describe the problem of depth and motion perception considered and the proposed contributions.

### 5.1.1. BACKGROUND ON PROSTHETIC VISION

Since 1968, different research works have found that electrical stimulation of the visual cortex or other parts of the visual pathway (such as retina) caused patients to perceive bright dots of light called phosphenes [Brindley and Lewin, 1968]. Thus, visual prostheses generally consist of retinal or cortical implants that apply electrical stimulation using an electrode array to generate a grid of phosphenes similar to a low resolution dot image [Dagnelie, 2006].

The typical components of this technology are as follows: A small camera mounted on the eyeglasses is used for image acquisition. The images are then processed by a portable computer to convert the image data into an electronic coded signal. This signal is transferred to the implant via wireless communication and the signal finally reaches the microelectrode array causing the grid of phosphenes. Experimental results demonstrate that patients with this kind of devices can detect phosphenes at individual electrodes and they were able to develop coordination using their visual prosthetic device [Ahuja et al., 2011].

### 5.1.2. MODELS OF PHOSPHENE PATTERNS

Unfortunately, the resolution of the phosphene grid produced by visual prostheses is constrained by biology, technology and safety [Meffin, 2013]. Current devices provide a few dozens of phosphenes, like the Argus II system from Second Sight, which achieves a 60 phosphene array (Figure 1.6). Therefore, implanted visual prostheses provide bionic vision with very limited spatial and intensity resolution when compared against healthy vision. According to [Cha et al., 1992] a pattern of $25 \times 25$ phosphenes allows to recognize text in a reading speed of 100 words per minute for stationary text and 170 words per minute for text moving automatically. Other related works also study performance in the task of reading [Fornos et al., 2011], or finding text [Denis et al., 2014]. However, other tasks like face recognition require hundreds of phosphenes [Thompson et al., 2003, Wang et al., 2014a]. Nevertheless, the technology is in constant evolution and experimental systems with hundreds of phosphenes already exist, and the perspective for the future promises to keep increasing that number [Ha et al., 2016]. However, such advanced technologies are still in trials, and for the moment we consider a moderate phosphene map resolutions, ranging from around 200 to 2000 phosphenes.

Moreover, traditional works generally assume regular phosphene patterns to be created with the prosthetic vision device. However, there is clinical and biological proofs that phosphene patterns are irregular and patient-specific [Srivastava, 2011, Li, 2015]. Still, regular patterns are usually assumed, and works that cope with irregular phosphene maps generally consider close to regular patterns where small spatial shifts in phosphene locations and electrode dropouts are modeled [van Rheede et al., 2010], or only irregular phosphene shapes are considered over a regular grid [Kiral-Kornek et al., 2013]. Regarding the particular shape of the visual phosphenes, there is a large variety of profiles described in the literature. For simplicity most works choose either perfectly circular or square shaped phosphenes for their simulation studies [Chen et al., 2009a]. However, neither perfectly circular nor square phosphenes capture the exact shape of real phosphenes.

Given the highly limited resolution, important efforts have been performed on the application of vision algorithms to improve the phosphene patterns for prosthetic vision [Barnes, 2013]. For example, vision processing can make better use of the limited resolution by highlighting salient features such as edges [Lui et al., 2012, McCarthy et al., 2013, Feng and McCarthy, 2013]. Currently, the way to process and code the image information to the low resolution device to be useful and meaningful is still an open issue.

### 5.1.3. Simulated Prosthetic Vision

In order to avoid complex and costly trials on patients, a non-invasive method to evaluate the efficacy of visual prostheses is by means of Simulated Prosthetic Vision


Figure 5.1. Examples of Simulated Prosthetic Vision setups from the literature. Both consist of an RGB-D camera (Kinect and Kinect 2 respectively) and a head mounted display (e.g. Oculus Rift in (b)).
(SPV). Most SPV systems make use of a head mounted display with a forward facing camera, which allows fast testing of a great variety of methods while constraining the user to a particular model of bionic vision such as visual angle or resolution. A thorough discussion about different SPV is provided in [Chen et al., 2009a, Chen et al., 2009b]. Two examples of experimental setups for SPV are shown in Figure 5.1.

Most of the current approaches used in prosthetic vision and SPV are based on basic image processing techniques [Ayton et al., 2013, Barnes, 2013]. However, this visionbased configuration allows exploring more advanced computer vision techniques to enhance the semantics and the relevance of the information displayed to the patient. Many researchers, inspired by a line of reasoning similar to the one we follow in this thesis, have use RGB-D instead of conventional cameras, since the rich structural information they provide has a lot of advantages compared to intensity-based representations. For example, as shown in Figure 5.2a, [Lui et al., 2012] compare the visual representation of a common scene by using a simple phosphene representation based on the intensity of the image with a representation of the depth edges from an RGB-D camera, being the latter the one where the information provided is much more useful and informative. There are many examples of depth processing in the literature whose output can be relevant for mobility. In Figure 5.2b there is an example from [Lui et al., 2012], where ground segmentation is applied to detect obstacle-free areas where the user can walk, outperforming traditional vision methods. In Figure 5.2c, the approach from [McCarthy et al., 2014] shows an augmented depth representation, where the intensity of the phosphenes maps the distance to the objects and the segmented floor is dimmed down to make obstacles more salient.

More complex computer vision algorithms can also be used, e.g. visual recognition can be used for enhancing the saliency of meaningful objects [Jung et al., 2015]. Besides, clarity of symbolic information can be improved with image segmentation techniques [Horne et al., 2012]. A relevant example for mobility is the semantic segmentation


Figure 5.2. (a) Comparison of a visual representation based on intensity and traditional image processing (top) and an edge-based representation from a range image (bottom) from [Lui et al., 2012]. (b) Same comparison between traditional vision processing (top) and depth processing after floor segmentation (bottom), from [Lui et al., 2012]. (c) Proposal of an augmented depth representation from [McCarthy et al., 2014], that combines floor segmentation with depth representation, compared to simple intensity based representation and depth based representation.
based on conditional random fields from [Horne et al., 2016], where with the input of just a color image, it produces a segmentation among road, sidewalk and obstacles, that is used to generate a phosphenic representation (Figure 5.3a). With depth information, a neural network is trained in [Feng et al., 2017] to detect structural edge information that is appropriately mapped to SPV representation (Figure 5.3b). Face and body detection can be used for human interaction [Lui et al., 2012, Bermudez-Cameo et al., 2017], as shown in Figure 5.3 (c) and (d). Other works take a different approach and use virtual-reality-based environments to evaluate the user response with different models of visual representation [Josh et al., 2013, Vergnieux et al., 2014,Zapf et al., 2016]. This procedure allows to try new representations and perform extensive tests with people in a realistic manner but at the same time reducing the complexity of the experiment.


Figure 5.3. (a) Semantic segmentation of a color image for SPV representation, from [Horne et al., 2016]. (b) Structure retrieval using Convolutional Neural Networks (CNN) for relevant structure representation in SPV, from [Feng et al., 2017]. (c) Face and body detection with depth compared to traditional image processing, from [Lui et al., 2012]. (d) Face expression recognition and body skeleton extraction from [BermudezCameo et al., 2017].

### 5.1.4. PROBLEM DEFINITION

As previously said, in prosthetic vision a visual scene is composed of relatively large and isolated spots of light called phosphenes. However, very low resolution images are frequently meaningless to the user. Moreover, representing depth in phosphene maps is very relevant to achieve adequate navigation, but its implementation is particularly challenging. Notice that depth perception cannot be transmitted using stereo effect because of intrinsic technical limitations of prosthetic vision. Thus, it requires alternative strategies to transmit depth such as using an iconic representation.

Our proposal consists of a perception system module (Section 5.2) and the iconic representation module (Section 5.3) for the camera-computer configuration in prosthetic vision. The goal of our perception module is to retrieve:

- The relative movement of the user in the scene.
- The orientation of the scene.
- A collision-free walkable path.

The type of camera we choose for information acquisition is an RGB-D camera, carried by the user mounted in the head. In our framework, this type of information is particularly useful to reliably detect obstacles and, for example, warn of other potentially dangerous situations such as the presence of curbs or stairs [Perez-Yus et al., 2017b], or detect the location of an empty chair [Wang et al., 2017]. Usually, it is assumed that man-made environments are essentially composed of three main directions orthogonal to each other. Taking this assumption into account, denoted as Manhattan world assumption, some works have been proposed for recovering the scene layout [Lee et al., 2009, Hedau et al., 2009,Flint et al., 2011], or just the orientation of the scene [Coughlan and Yuille, 1999] as we do in this work.

In the proposed iconic representation module we code the information perceived to the phosphene map. This task is challenging, since our approach tries to accommodate to the current state of technology of prosthetic visual devices. Despite the recent progress in the field, the resolution and dynamic range are still low. Moreover, related works focus on 2D information neglecting the three-dimensional nature of the world, and depth perception is lost to the user. Systems displaying depth and contrast edges in a phosphene-based display are described in [Li, 2013, Lui et al., 2012] and more recently in [McCarthy et al., 2014]. In [Horne et al., 2016], a semantic labeling of the image provides a representation for obstacle avoidance. Here we aim to the ambitious goal of providing depth information by designing appropriate processing algorithms to be used on the vision-based input information.

In this chapter, we present a novel phosphene map coding for navigation tasks based on a ground representation of the obstacle-free space as a polygon and a ceiling representation based on vanishing lines pointing towards a previously determined moving direction. The ground polygon is codified with a chess pattern to provide the effect of
displacement over the ground with the relative pose obtained with the odometry. This pattern additionally produces visual cues about the distance of the objects and the orientation of the scene. The effectiveness of the proposed representation is illustrated in a simulated environment and with real data from indoor scenes in Section 5.4.

### 5.1.5. GEOMETRY AND NOTATION DETAILS

Consider a set of points, planes and lines in a given reference. We denote $\mathbf{X} \in \mathbb{P}^{3}$ a 3D point in homogeneous coordinates. We denote $\mathbf{U}=\left(\mathbf{u}^{\top}, u_{0}\right)^{\top}$ a plane in homogeneous coordinates. We denote $\mathbf{L} \in \mathbb{P}^{5}$ a 3D line in Plücker coordinates composed by two vectors $\mathbf{L}=\left(\mathbf{l}^{\top}, \overline{\mathbf{l}}^{\top}\right)^{\top}$ being $\mathbf{l} \in \mathbb{R}^{3}$ a vector describing the direction of the line, $\overline{\mathbf{l}} \in \mathbb{R}^{3}$ is a vector representing the normal to a plane passing through the 3 D line and the origin of the reference system $O$, and the ratio between its norms $d_{l}=\frac{\|\overline{1}\|}{\|1\|}$ is the minimum distance from the line to the origin of the reference system (see Fig. 5.4). To allow $\mathbf{L}$ being a 3 D line $\mathrm{l}^{\top} \overline{\mathbf{l}}=0$. Rays are also codified as Plücker coordinates but denoted with $\boldsymbol{\Xi}=\left(\boldsymbol{\xi}^{\top}, \overline{\boldsymbol{\xi}}^{\top}\right)^{\top}$.

Consider a reference system composed of a rotation $\mathbf{R} \in S O(3)$ and a translation $\mathbf{t} \in \mathbb{R}^{3}$. A change of reference of points is performed by using the linear transformation $\mathbf{T} \in S E(3)$ such that:

$$
\mathbf{T}=\left(\begin{array}{cc}
\mathbf{R} & \mathbf{t}  \tag{5.1}\\
\mathbf{0}^{\top} & 1
\end{array}\right)
$$

A change of reference of a plane is done through $\mathbf{T}^{-\top}$. Finally, a change of reference of a line or a ray is described by the linear transformation:

$$
\mathbf{G}=\left(\begin{array}{cc}
\mathbf{R} & \mathbf{0}  \tag{5.2}\\
{[\mathbf{t}]_{\times} \mathbf{R}} & \mathbf{R}
\end{array}\right)
$$

where $[\mathbf{t}]_{\times}$denotes the construction of an antisymmetric matrix from the vector $\mathbf{t}$.

### 5.2. PERCEPTION OF FREE SPACE AND SCENE POSE

The proposed system includes an RGB-D camera for the perception part. In this section we describe the main sub-tasks as defined in the introduction: to obtain the relative movement of the user in the scene (Section 5.2.1), to get the orientation of the scene (Section 5.2.2), and to retrieve the zone of free moving space (Section 5.2.3).


Figure 5.4. Components of a Plücker description of a 3D line. The direction vector $\mathbf{l}$ and the moment vector $\overline{\mathbf{l}}$. Under Manhattan World assumption a main direction $\mathbf{v}_{v p}$ is coincident with the direction $\mathbf{l}_{i}$ of lines following this direction and orthogonal with their moment vectors $\overline{1}_{i}$.

### 5.2.1. RELATIVE MOVEMENT OF THE USER IN THE SCENE

In robotics, the estimation of the position of the robot with respect to the starting location is called odometry. When the information to compute the odometry comes from a camera, it is called visual odometry. This is a classic topic in computer vision, which recently has been enhanced with the advent of RGB-D cameras. We use the algorithm from [Gutierrez-Gomez et al., 2015], which is a method for dense visual odometry estimation performed by minimizing photometric (in the RGB image) and geometric (in the inverse depth map) errors, and therefore takes advantage of the RGB-D camera. More extended explanation of the algorithm is provided in Section 2.4.3.

With this method, for each frame we compute the pose ${ }^{0} \mathbf{T}_{k} \in S E(3)$ that transforms the reference frame from $k$ to 0 , being 0 the initial reference frame. This transformation ${ }^{0} \mathbf{T}_{k}$ consists of a rotation matrix ${ }^{0} \mathbf{R}_{k} \in S O(3)$ and a translation vector ${ }^{0} \mathbf{t}_{k}$. These transformation is necessary for our method to provide sense of movement in the environment.

### 5.2.2. ORIENTATION OF THE SCENE

In our work we assume scenes satisfy the Manhattan World assumption [Coughlan and Yuille, 1999], meaning the world is organized according to three orthogonal directions, we call Manhattan directions or main directions. In order to get these directions, we perform a vanishing point extraction, since all lines directed in one of the Manhattan directions intersect in one of the three main vanishing points.

First, from the distribution of the normals of the point cloud we obtain a set of rough candidates for being the main three directions. This can be performed following the approach from Section 4.3.3 for depth information. However, since we do not have omnidirectional view of the scene, the refinement with lines in the color image needs
to be performed differently. In particular, lines are extracted from the RGB-image and clustered in main directions following a Random Sample Consensus (RANSAC) approach [Fischler and Bolles, 1981]. Assuming Manhattan directions, we can assemble the direction vectors to create the rotation matrix ${ }^{A b s} \mathbf{R}_{k} \in S O(3)$, being $A b s$ the reference of the system with the axis aligned with the Manhattan directions, called absolute reference. To enforce the obtained directions to be orthogonal we optimize ${ }^{A b s} \boldsymbol{\omega}_{k} \in \mathfrak{s o}(3)$ such that ${ }^{A b s} \mathbf{R}_{k}=\exp \left(\left[{ }^{A b s} \boldsymbol{\omega}_{k}\right]_{\times}\right)$. The distance of the minimization $d_{o p t}=\mathbf{v}_{v p}^{\top} \overline{\mathbf{l}}_{i}$ exploits the constraint that the direction vector of a 3D line $\mathbf{L}_{i}$ must be orthogonal to its corresponding projection plane (see Fig. 5.4). For considering that the original clusters could contain some misclassified lines we use a $L 1$-norm as loss function. Finally, the result is fine-tuned with a $L 2$-norm using only a selected collection of well conditioned lines.

In practice, this procedure needs to be performed only once and then carried over by the odometry (although periodic computations of the orientation may be performed if there is accumulation of drift on the pose estimation). For example, let us consider we obtain ${ }^{A b s} \mathbf{R}_{0}$ at first frame. At frame $k$ we can compute the pose ${ }^{A b s} \mathbf{T}_{k}$ with ${ }^{A b s} \mathbf{t}_{k}={ }^{0} \mathbf{t}_{k}$ and ${ }^{A b s} \mathbf{R}_{k}={ }^{A b s} \mathbf{R}_{0}{ }^{0} \mathbf{R}_{k}$. We choose the axis in Abs to be as follows: $z_{A b s}$ pointing upwards (to where the ceiling should be), $x_{A b s}$ to the front of the user in that moment and $y_{A b s}$ to its left.

### 5.2.3. Perception of free space

The free space around the user is retrieved using the information from the depth camera, specifically the point cloud data. A point cloud is a set of 3D points $\mathbf{X}_{i}=$ $\left(x_{i}, y_{i}, z_{i}, 1\right)^{\top}$, each one corresponding to a pixel in the depth camera. To speed up the algorithm, instead of making operations to the whole cloud we perform downsampling via voxel grid filter. For example, applying a voxel size of 0.10 meters could reduce the cloud approximately 100 times (it depends on the scene) without major loss of relevant data for this task. The point cloud can be transformed to the absolute reference frame by $\mathbf{X}^{A b s}={ }^{A b s} \mathbf{T}_{k} \cdot \mathbf{X}^{k}$.

Once we have our data pre-processed and in the absolute reference frame, we compute the floor plane. To do so, we have a tentative orientation of the normal of the floor plane, since it should align with the main direction corresponding to $z_{A b s}$. Thus, we proceed by computing the normals of the points via principal component analysis, and selecting the points whose normal is near $z_{A b s}$. Then, a RANSAC procedure for planes is applied to that subset of points, and among the resulting plane candidates computes their distance to the origin $u_{0}$ and chooses as solution that with highest value of $u_{0}$. Note that we are assuming that the floor is visible and that there is no other horizontal plane below it. Again, like with the main directions, the floor plane does not need to be retrieved every frame, and it can be carried over by the odometry. This has an important
advantage, since then the floor does not need to be in the image all the time and the user can be looking elsewhere.

The points of the cloud that are not classified as floor points are considered obstacles unless they are considerably higher than the person (e.g. ceiling). Since the camera is designed to be on the head, we choose $z=0.5$ meters over the head as a safe maximum threshold to consider points out of reach. The points are then grouped in planes and clusters combining a RANSAC approach with Euclidean cluster extraction (i.e. points that are close to each other under certain threshold are grouped in clusters). The segmentation process resembles what was described in Section 2.3.1. Each plane or cluster is considered obstacle, meaning that no further classification or semantic reasoning has been performed.

To determine the area of free space we project the points to the floor plane to reason in 2D (i.e. ignore the component in $z$ ). The area of free space will be the polygon on the floor plane whose edges are given by the bounding boxes of the obstacles and the rays from the camera. The procedure to build the floor polygon goes as follows (see Fig. 5.5 for graphical explanation):
(a) We look for visible vertices in the corners of the bounding boxes of the floor projections of the obstacles. Visibility is checked by the intersection of the segments from the origin to the vertices and the segments of the bounding boxes. Also, intersection points between bounding boxes are considered.
(b) We project rays from the origin to the previous vertices to verify if they intersect with their bounding boxes in other point than the vertices themselves or not. Those who do not intersect are extended until intersection with other bounding box or to a maximum distance (e.g. 10 meters).
(c) We remove those vertices outside the field of view (FOV) of the camera, and include the origin as vertex.
(d) We sort the vertices clockwise with the angle $\alpha$, sorting accordingly vertices with same angle (i.e. coming from (b)).

The output of the proposed procedure is a 2 D polygon in the floor plane that we can transform in 3D since we know the plane equation (see Fig. 5.5 (d)). Notice that, for simplicity, bounding boxes of the obstacles have been represented so that they are oriented with the main directions of the scene. This is the fastest approximation for getting an enclosing rectangle of an obstacle, but in some cases may not be a good approximation (e.g. circular shape or diagonal wall). The pipeline of the method holds for any other computed enclosing polygon, such as the convex hull, whose retrieval is widely implemented in many programming libraries. The convex hull produces more accurate obstacle representation at the expense of introducing more vertices in the floor polygon and, in general, more computation time. Therefore, if the algorithm is able to perform fast enough with convex hull, that is usually preferable.


FIGURE 5.5. Four basic steps for the construction of the floor polygon following the explanation from Section 5.2.3. Four obstacles are drawn with the projection to the floor in different colors and bounding boxes. Valid vertices are colored in green while invalid vertices are dark red. In (d) the floor polygon is drawn in green.

### 5.3. ICONIC REPRESENTATION OF FREE SPACE

Even when the environment is known, the lack of dynamic range and resolution in prosthetic visual devices complicates the perception of depth. On the one hand, the quantification given by the low resolution hinders the possibility of stereographic vision. On the other hand, the low dynamic range dilutes the texture of landmarks that humans use for locating themselves.

In order to tackle with these perception problems, we consider using an iconic representation of the scene capable of giving support for navigation tasks. Our proposal consists of a polygonal representation of the ground describing the free space and a simplified representation of the ceiling suggesting the motion direction to be planned in a higher level. The displacement with respect the ground is evoked by using a chess pattern in the ground map. This representation, inspired by old low-resolution 3D games,


Figure 5.6. Screenshot from Sonic the Hedgehog video game, where the floor was represented as a checkerboard to provide sense of movement and perspective.
is useful as an iconic description of perspective projection. An example of a classic video game where this technique was utilized is shown in Figure 5.6. The motivation for using this representation is threefold: it helps to transmit depth and sense of movement when walking, but also provides an estimate of the distance to an obstacle since the size of the tiles is fixed. This also provides cues of the orientation of the scene, since the checkerboard is oriented according to the main directions.

As introduced in Section 5.2, we estimate the obstacle-free ground and represent it with a polygon. This polygon is defined in a global reference we have obtained from the main directions of the scene and the odometry we get from an RGB-D SLAM system [Gutierrez-Gomez et al., 2015]. The phosphene map has an associated projective ray for each phosphene. For representing the ground, we first estimate which rays intersect with the floor polygon. For this we use the Plücker polygon-ray intersection approach which works with convex polygons. Since the floor polygon is in general non-convex, we estimate the rays intersecting the convex hull of the polygon, which quickly provides information about the phosphenes that are likely to be turned on. Then, we evaluate if the points that intersect in the convex hull actually lie inside the non-convex polygon or they are outliers by performing a point in polygon test.


Figure 5.7. The sign of the distances from ray $\mathrm{A}\left(\boldsymbol{\xi}_{A}, \overline{\boldsymbol{\xi}}_{A}\right)$ to the sides ( $\left.\mathbf{s}_{i}, \overline{\mathbf{s}}_{i}\right)$ are $(-,-,-,-,-)$. By contrast, the sign of the distances from ray $\mathrm{B}\left(\boldsymbol{\xi}_{B}, \overline{\boldsymbol{\xi}}_{B}\right)$ to the sides $\left(\mathbf{s}_{i}, \overline{\mathbf{s}}_{i}\right)$ are $(-,+,-,-,-)$.

### 5.3.1. PLUCKER POLYGON-RAY INTERSECTION METHOD

Given a line $\mathbf{L}=\left(\mathbf{l}^{\top}, \overline{\mathbf{l}}^{\top}\right)^{\top}$ and a ray $\boldsymbol{\Xi}=\left(\boldsymbol{\xi}^{\top}, \overline{\boldsymbol{\xi}}^{\top}\right)^{\top}$ represented in Plücker coordinates, the side operator

$$
\begin{equation*}
\operatorname{side}(\mathbf{L}, \boldsymbol{\Xi})=\mathbf{1}^{\top} \overline{\boldsymbol{\xi}}+\overline{\mathbf{l}}^{\top} \boldsymbol{\xi} \tag{5.3}
\end{equation*}
$$

returns a signed distance. The sign of this distance depends on the side where a line is located with respect to the other (clockwise or counterclockwise) (see Fig. 5.7). If we define the sides of a convex polygon with their corresponding Plücker coordinates and we follow a clockwise sense in this definition we can determine if a ray intersects the interior of the polygon or not by using the following rule.
(a) Estimate the sign of the side between the ray and each side.
(b) If all the side distances have the same sign the ray intersects the interior of the polygon.
(c) If this sign is positive we are looking to the front of the polygon, if negative we are looking to the back.

Once we know the rays intersecting the convex hull of the polygon we compute the corresponding projected points. Then, we collect the projected points which are inside the polygon and mark them with the chess pattern. This texture is parametrically defined by quantizing X and Y coordinates. Since the points are computed in the global reference (which is aligned with the vanishing points) the chess pattern follows the main directions of the scene.

### 5.4. EXPERIMENTS OF THE ICONIC REPRESENTATION

What we have proposed in this chapter is an algorithm that extracts information from the environment and translates it to a limited representation that will be interpreted by a human afterwards. Therefore, the experiments need to evaluate both parts: the extraction of the information from the environment, and the quality of the iconic representation. We have tested our method in two experimental frameworks: simulation in a realistic virtual environment and with real images taken with a head mounted RGB-D system. With the simulation framework, we have built a platform to test different iconic representations and to easily perform numerous experiments with people, being able to collect all the data and thus evaluate numerically the quality of the representations. On the other hand, the experiments with real images show that our method is applicable to the real world. The following sections detail the experiments from each framework, providing qualitative insight about the method, its limitations and possibilities.

### 5.4.1. EXPERIMENTS IN SIMULATED ENVIRONMENTS

Performing experiments in simulation or in virtual experiments is a common way to proceed in computer vision and robotics. Usually it is used as the previous step before experiments in real situations, since it should indicate whether the method works abstracting the performance of the method itself from other problems that may come up in the real world. However, the more realistic the simulation is, the closer it will be to translate the implementation to a real system. In the next section we introduce our simulation framework, and then we provide some qualitative results of our proposal.

### 5.4.1.1. SIMULATION FRAMEWORK

For our framework, we need a realistic physics engine to simulate a real world environment and the means of capturing information from it via virtual data acquisition devices. Besides, the software needs to allow us to develop an intuitive user interface and to manage all communications of the system. Particularly, we have used ROS (Robot Operating System ${ }^{1}$ ) and Gazebo ${ }^{2}$. ROS is a set of software libraries especially convenient to manage communication in robotic tasks. The information flows via messages, such as the commands to move a robot, or the perceptual information retrieved by it. Gazebo is a robotics simulator that includes realistic robot and sensor models that was designed to test algorithm implementations. The combination of ROS with Gazebo is perfectly suitable for our needs: we can use a simulated RGB-D camera in Gazebo

[^7]

Figure 5.8. (a) Example of virtual map used in our experiments. (b) Close up of the robot navigating inside the map. The camera sensor is over the robotic platform, virtually attached to provide a point of view similar to humans.
(e.g. an Asus Xtion Pro Live), and transmit the information captured in the virtual environment to our perception module using the message system from ROS. The same implementation would be compatible straight away when a real camera is used instead of a Gazebo simulation, since they use the same type of messages.

Using simulated environments has other advantages, such as being able to quickly create many different types of environments and evaluate the performance of the method under specific situations without needing to find or create particular scenarios in the real world. It also provides ground truth, allowing to analyze results more accurately, but also to decouple the separate problems of the full system. For our application, we can retrieve the pose of the robot at any time with respect to the absolute reference of the virtual environment, removing the need of a visual odometry module and the computation of the orientation of the scene, and thus allow us to focus on the perception of free space.

In our simulation framework, a simulated robot with a mounted RGB-D camera can be intuitively moved with a game controller by an external subject. We consider simple robot movements: left-right rotation, forward-backward movement, and upwarddownward camera tilt. In the experiments with simulation, the subject that controls the robot can only see the phosphene map representation, and needs to move around without hitting any obstacle. Having a working simulation framework allow us to test different representations, with a considerable amount of experimental subjects, without having to create complex experimental setups (just a computer is needed). An advantage of using simulated environments is that we can collect all the data from the experiments to analyze the results quantitatively, including trajectories, velocities or number of collisions. The latter can be performed reading the information of the bumpers that many robots usually have. Last but not least, simulation test are free of safety issues, since the user can evaluate different configurations without the risk of crashing with a wall or falling down the stairs.


Figure 5.9. Two frames extracted from our free space perception algorithm running in a Gazebo simulation. RGB and depth images are shown at the left and the center respectively, and the resulting map of free space is at the right. The depth image has been scaled so that warmer colors mean farther distances, and dark blue means no depth measurement (in this case, because points are very far). The free space polygon is depicted in green in the map, confined between the two lines that enclose the FOV of the system.

### 5.4.1.2. EVALUATION OF THE METHOD IN SIMULATION

For the experiments we have created several maps such as the one shown in Figure 5.8a, that represents a small house. The movement inside the map is achieved by adding a Turtlebot robot, that includes basic mobility, an Asus Xtion Pro Live camera, and bumper measurements. It is completely modeled in Gazebo, although we have modified it to have the camera at a height that would resemble human vision, also with the addition of a tilting movement to the camera so the view can be modulated to watch out long distances or closer ranges (Figure 5.8b).

In the Figure 5.9 there are two examples of how our free space perception method performs in a virtual environment. In particular, we can observe how the images from the depth camera are very accurate and, therefore, it makes these kind of data the best to test our algorithm in initial stages of development. The free space polygon obtained, as seen at the right in Figure 5.9, adapts well to the environment, being able to show the area which is free of obstacles. A simple floor plane segmentation would have returned some space below the table, where obviously the user cannot access by just walking. Some of the obstacle shown are a person, walls or a table, being successful at detecting all of them.

It is also interesting to observe the phosphene map iconic representation of the free space polygon. Since we are working in the framework of SPV, we can choose the phosphene representation parameters. In order to pick some, our goal is to remain as faithful as possible to the current state of technology. Nevertheless, the continuous advances


Figure 5.10. Four phosphene representations considered in the virtual experiments, in this case showing an open space free of obstacles. (a-c) are the proposed checkerboard-based representation at three resolutions, and (d) is the simple representation of the walking polygon without the chess pattern.
in the field made us consider a flexible approach and test several amount of phosphenes ( $N_{p}$ ), ranging from low resolution representation ( $N_{p}=216$ phosphenes), medium resolution representation ( $N_{p}=484$ phosphenes), and high resolution ( $N_{p}=1862$ phosphenes). The intensity values of the phosphenes have been restricted to a minimum, to ensure the representation would be functional in reality, and thus we consider at maximum three levels of intensity: black (turned off phosphene), gray (phosphene turned on at intermediate intensity) and white (phosphene turned on at maximum intensity). We can also choose the field of view of the representation. Large field of view would imply more information to fit in an already small display, whereas small field of view would provide more detail. In our simulation experiments we choose a focal length $f$ so that the field of view of the representation is similar to the one of the camera ( $f=525$ pixel).

Our proposal of iconic representation of the free space in phosphene maps is to use a chess pattern to provide depth and motion cues for navigation. The pattern is displayed by using gray and white levels on the phosphenes whose direction points towards the free space polygon, whereas turned off phosphenes mean no free space, therefore implicitly suggesting where the obstacles are. The ceiling, which would be empty in this floor-based representation, shows two lines pointing to the vanishing point at the front, which was selected as a direction to follow. Note that more advanced algorithms could have been used to determine the suggested direction to follow, but we consider these possibilities out of scope in this work.

In Figure 5.10 there are four possible representations of the floor space when there are no obstacles: the first three with our proposed representation at three resolutions, and the last one with no chess pattern displayed. As it could be intuitively expected, the higher the resolution, the better representation of the chess pattern. With low resolution, the black and white tiles are not so clearly observed like in the other two. However, medium resolution produces quite usable representation, thus showing it is not necessary to have very high resolution to benefit from this iconic representation. Notice that the
size of the black and white tiles could be increased to improve visualization in low and medium resolution (here it is fixed at 0.5 meters). In an open space such as this one, we can observe how the chess pattern displaces as the user moves, thus providing comfortable depth and perspective cues. Removing the pattern like in Figure 5.10d removes all that intuitive information.

We show several results of our iconic representation in the presence of obstacles in Figure 5.11. For each result, apart from our iconic representation proposal at the three levels of resolution, we include the highest resolution without gray level, and thus, using only one level of intensity. The first row shows how an structure that resembles an open door appears in this representation, where the path seems to continue in the middle but blocked at the sides. Second and third row shows examples where complexshaped rooms with several objects appear. We can see how the representation without gray level is useless, since it is hard to tell when the phosphenes are turned off because of an obstacle or because the tile is black. If the system was limited to only one level of intensity, we could overcome this limitation by drawing the borders of the floor polygon. However, this representation would be more jarring and less intuitive when complex polygons are detected. Regarding the resolution, we can see how the lowest resolution does not clearly shows a path through the door at the left in row 2 , whereas medium and high resolution show it perfectly. The fourth row show the behavior of the system with thin objects. In this case, the leg of a table is only observable in high resolution. The fifth row shows the view at the same position when the user tilts the head up and sees the table: the region of floor below the table is removed from the free space. In reality, moving the head around helps notably to understand the scene, since one single view is often not enough to see all the hazards.

It is interesting to observe the algorithm performing in a sequence in order to see how the checkerboard representation works compared to the plain floor view in just single level of intensity. We choose a corridor sequence, shown in Figure 5.12. At the bottom, we can observe how adding a chess texture to the floor helps providing that effect of being actually moving. In contrast, at the middle there is a mere floor representation without any texture, where it is more difficult to tell the differences between consecutive frames. Therefore, the sense of depth and movement is loss by removing the chess texture. In both cases, the presence of doors along the corridor can be perceived by looking at the ramifications at the sides that indicate that the floor extends further in that direction.

### 5.4.2. EXPERIMENTS WITH REAL IMAGES

Apart from simulated environments, we evaluate the performance of the method with real sequences. For these experiments, we used an Asus Xtion Pro Live RGB-D camera mounted in a helmet to capture video sequences. We used a laptop to record the sequences, but no direct feedback to the user (e.g. via virtual reality glasses) was tested


Figure 5.11. Five examples of phosphene representations when obstacles appear in virtual environments. Each example includes the RGB image, our proposed iconic representation at three resolutions and the high resolution without gray level.


Figure 5.12. Corridor sequence (top), where the effect of progressing through the user movement is provided by the chess pattern in the floor (bottom). In the middle the same results where the chess pattern is not display, and thus, the feedback provided to the user is less comfortable and intuitive.


Figure 5.13. Each row shows an example of the perception of free moving space. The first two columns show the RGB and depth image, which has been scaled so warmer colors mean farther distances. Column 3 shows the floor plan view of the moving polygon and in column 4 the chess pattern has been overlaid to the RGB image for visualization.
yet at this early stage. However, this approach allows us try different ways of encoding the data in phosphenic representation and analyze the problems that may emerge in a real scenario.

### 5.4.2.1. Evaluation of Free space perception

First, it is important to have a reliable perception module to create a safe and useful assistive aid. The recovery of the main orientation of the scene works well considering the Manhattan World assumption holds for a vast majority of indoor environments. On the other hand, the visual odometry has a slight drift that is more noticeable the longer the algorithm is running. However, by re-computing the vanishing points and floor plane from time to time the effect of the drift can be minimized.

The most important part about the perception module is the obstacle detection, since this is what warns the user when he is close to have an accident. Its proper functioning depends mostly on the sensor and its limitations. Conventional RGB-D cameras are well known for not working well at direct sunlight, which should not be problematic when
staying indoors. In this situation most obstacles are detectable, with the exception of certain materials that absorb or reflect the infrared light emitted by the camera [PerezYus et al., 2018c]. In our experiments, the only type of surface that remains always undetected was glass, which is also undetectable with conventional cameras.

We show some examples in Figure 5.13, including RGB and depth captures, along with a 2D floor projection of the scene, where the obstacles are drawn as blue points and the free moving space is the green polygon. It is also depicted the overlay of the corresponding chess pattern to the color image. The first two rows correspond to a corridor scenario: a successful case and an unsuccessful one. The former not only is able to recover the main path to follow the corridor, but it also shows the beginning of new paths on the left, that the user may like to know to explore the environment. The second example shows the opposite: a misdetection on the left wall and the glass surface on the right, showing some misleading free space, which could be problematic. The third and fourth row in Figure 5.13 show another environment, in this case an office place full of tables and other obstacles. In both cases the algorithm is able to show the main path to follow and some additional sidetracks. The last row finds a traversable path through the door. However, it leaves some actual free space undetected at the left, next to the door.

### 5.4.2.2. Evaluation of phosphenic representation

Regarding the phosphenic representation, we show some examples in Figure 5.14. We have tested different parameters of the codification, including the number of phosphenes and the field of view of the representation. About the first, we can observe in Figure 5.14 the differences among columns (1) and (2), where different values of number of phosphenes $N_{p}=1862$ and $N_{p}=484$ phosphenes. While still useful, it is less intuitive the representation with fewer phosphenes: the chess pattern is not so easily observed, and the bigger discretization in the representation produces sudden jumps in the representation in the borders of the obstacles. When the amount of phosphenes increases, the changes in the phosphene map are usually less aggressive and the chess pattern can be clearly observed. We have to note that this is a simulation to show how our approach works for several levels of detail. In a real-world prosthetic system the number of phosphenes would be limited by the current state of technology.

The field of view of the representation is another parameter we can tune since we are turning 3D information into a 2D representation. Column (2) shows a representation considering a focal length similar to the RGB-D camera ( $f=525 \mathrm{~mm}$ ). This has the advantage of representing only what is actually viewed by the camera, and provides more accurate delimitation of the obstacles and therefore how to avoid them. However, the field of view of conventional RGB-D cameras is limited, less than normal human vision. An alternative is shown in column (4), where the field of view selected is very high ( $f=200$ pixels). This has the advantage of showing the information in shorter



RGB Image

(1)

$$
\begin{aligned}
& f=525 p x \\
& N_{p}=484
\end{aligned}
$$


(2)
$f=525 p x$
$N_{p}=1862$

(3)
(4)
$f=400 p x$
$f=200 p x$
$N_{p}=1862$
$N_{p}=1862$

Figure 5.14. Four real examples of our phosphene-based representation including different parameters. In particular, we wanted to show the difference of using different number of phosphenes $\left(N_{p}\right)$ and field of view of simulated phosphene camera (given by its focal length $f$ ). By columns, RGB Image and four alternative phosphenic representations ( $1-4$ ) with different parameters.
ranges. For example, when looking at a wall, a low-FOV representation turns all the phosphenes to black (since no floor is seen). With a high-FOV representation, a portion of the floor still appears in the image showing the user that he still has some space to move (see third row in Figure 5.14). The high-FOV representation has an important drawback, since it needs to encode more information in an already limited display. Note that, in the fourth phosphene map column in Figure 5.14, the path seems narrower than with larger values of $f$. This is because we limit the extension of the free space polygon to the field of view of the sensor (since it is our only cue to detect obstacles), and it represents less extension relative to the large field of view of the phosphene camera ( $f=200$ pixels). In column (3) we show a middle ground ( $f=400$ pixels), where the information at mid-distance is informative enough and also gives more information about the free space in short distances. In general, having the FOV closer to the camera represents better the information captured, with more detail and less unused space. In order to see free space at closer range in front, tilting the head down should suffice the issue.

### 5.4.2.3. VIDEO DEMONSTRATION

We have included two videos to show the algorithm running in real case scenarios, that can be found online ${ }^{3}$. Two screenshots from the videos are shown in Figure 5.15. In particular, we show a video where the user moves in a corridor, and another where the user moves inside an office. In these sequences we use $N_{p}=1862$ and $f=525 p x$. We can see how the phosphenic representation shows clearly the moving area over which the user can walk safely.

Unlike other works, our representation includes a checkerboard floor which shows the movement of the user in the scene providing a comfortable sense of depth. This is more noticeable in the corridor sequence. In that sequence, the windows at the right part of the corridor sometimes show absence of obstacles since glass remains undetected, returning erroneous floor polygons. Note that the floor gives few valid depth points. However, we can maintain its position with respect to the user with the odometry.

The office sequence, on the other hand, presents a cluttered environment with many obstacles. Our method shows the free space in front of the user, removing the tables and other objects from the floor polygon. This scenario is particularly challenging, and thus some frames show undetected portions of obstacles, producing inaccurate polygons. However, these situations occur mostly in isolated frames, producing an effect similar to flickering. Ongoing work solves and improves the algorithm for obstacle detection. In fact, an updated implementation that solves these issues and that also works close to real time was used in the simulation framework. Since the software we use for simulation uses the same type of information that the real camera does, the developments of the algorithm in virtual environments should translate seamlessly to real world images.

[^8]

Figure 5.15. Two screenshots from the video demonstration, where the RGB and depth images are shown on the left, the odometry map with free space in green at the center, and the chess pattern overlaid in the image and the corresponding phosphene map on the right.

### 5.5. DISCUSSION

Visual prostheses are able to evoke visual perception in blind people by using electrical stimuli. However, these prototypes suffer from a lack of spatial and intensity resolution that in practice prevents from transmitting depth perception. In this chapter we present an approach to represent depth and motion cues with phosphene patterns in the context of safe navigation of blind people in complex or unfamiliar environments. This approach takes advantage of computer vision algorithms for evoking phosphenes-based stimuli with semantic meaning. In particular, we propose a free-space and obstacles detection algorithm for depicting an iconic representation of a safe navigation area. The effectiveness of this approach is tested in simulation with real data from indoor environments.

## 6

## CONCLUSIONS AND FUTURE WORK

In this thesis, we presented some contributions to the state of the art regarding the recovery of relevant information from the scene in the context of assistive computer vision. We have proposed several methods that attempt to extract useful cues from the environment that may be utilized to enhance the navigation in man-made scenarios among other tasks. For this, we have developed computer vision algorithms that work with non-conventional camera systems such as RGB-D cameras, omnidirectional cameras, or combination of both. To summarize our contributions, we have divided our research in four big blocks:

- We developed a perception system that is able to segment and orient the scene, find the floor and obstacles, and detect stairs and curbs. The latter is the main contribution of this block, considering this is one of the first methods to perform stair detection with RGB-D cameras, improving existing methods in results and functionality. The main advantage of our approach is the recovery of the staircase model, with full measurements and orientation with respect to the user, which can be used for validation of the detection and to provide navigational cues. Furthermore, with a visual odometry algorithm running in parallel, we also dealt with the problem of stair traversal, achieving complete models of the staircases, continuous recovery of the user's pose, and detection of the end of the stair. Additionally, online detections of the staircase during traversal were used to reduce the drift of the visual odometry.
- In order to reason about the whole scene and not only the small part in front of the user, we explored the possibility of enhance current RGB-D cameras by extending their field of view. In this line, our two major contributions were the introduction of a novel hybrid camera system with depth and fisheye cameras and its calibration, and the proposal of another calibration technique based on line observations. This novel technique is a more general alternative, able to calibrate multiple combinations of camera systems. An important advantage is that it does not require that cameras have overlapping field of view, which makes the approach
perfectly suitable to expand fields of view. Besides, since it works by extracting and matching lines that can be easily found in man-made environments, there is no need to build any calibration device as in many alternatives.
- Following this idea of taking advantage of unconventional camera systems to capture large field of view, we propose the first application that uses our novel hybrid camera system with fisheye and depth. We used it to develop an algorithm to estimate the 3D scaled reconstruction of the room the user is in, being able to recover complex-shaped spaces and not merely box-shaped rooms. The advantage of combining both cameras is clear, since the fisheye views larger portion of the room that allows to see even the usually less cluttered ceiling, and the depth provides scale and enhances the performance of the method. Our layout estimation method is even able to overcome the hidden and occluded corners that may be invisible for any camera.
- Given that our contributions were focused on the framework of visually impaired assistance, we also deal with the communication of the information perceived to the user. We center our attention on the current state of visual prostheses, which allow the users of such technologies to perceive certain visual information in the form of dots of light, called phosphenes. The low spatial and dynamic range resolution of these kinds of systems make the problem of conveying the information with such systems to the people quite challenging. We worked in simulation, encoding the information in a type of image that emulates what patients with visual prostheses can see. To tackle the problem with the resolution, we proposed an iconic representation of the environment informative enough to navigate comfortably avoiding obstacles and with some notions about the orientation of the scene. In particular, our chess-pattern ground plane representation provides, with minimal number of levels of intensities of the phosphenes, a sense of depth that is usually misrepresented in other attempts of iconic representations for prosthetic vision.

All this research has been proposed in the framework of assistive computer vision, especially considering the problem of visually impaired assistance. However, most of the work presented here is general enough to be useful for other applications. For example, the stairs detection algorithm from Chapter 2 has obvious connections with robotics, where RGB-D cameras are often used to perceive the environment in a similar way. Not only that, but stairs are a standard construction whose detection and modeling could be used to add some semantic information to 3D mapping operations of entire buildings, where only geometrical relations between points and shapes are often used. The Chapter 2 itself mentions how it can use the shape of the staircases to reduce the drift of visual odometry algorithms by using the known shapes of stairs and continuous detections during traversal. The calibration approaches presented in Chapter 3 are also not specific to visually impaired aids. Extrinsic calibration of multiple cameras is often dealt with in areas like robotics, autonomous driving or manufacturing processes. Our method based on line observations is especially useful for these problems, since not requiring cali-
bration patterns enables a potential automatic re-calibration, often necessary in systems where the mechanical links between cameras may deteriorate over time. The 3D fullscaled layouts estimated in Chapter 4 can be of interest in mapping, scene recognition, and augmented reality, among many others.

On the other hand, the work presented in Chapter 5 is clearly tied to the visually impaired assistance problem. The determination of free space could be applied to navigation or robotics, but everything else is completely related to visual prostheses. Moreover, it is in this block where we directly focus on the communication interface with the user. This lead us to one of the main problems that we wish to address in the future: the lack of experiments with real people with visual impairment. This is, however, a very difficult task, for many reasons. Particularly, in the case of visual prostheses, the access to people with such systems is very complicated due to medical and economic reasons. The process of installing an implant entails surgical procedures, user specific customizations, and, afterwards, supervision and training. The current state of development is still quite experimental and controlled, and the cost is still considerably high. Even with commercial systems such as the Argus II Retinal Prostheses System, from Second Sight, which is probably the most popular visual prostheses commercially available, the amount of users of the system is still rather reduced. To throw some numbers, in 2017, only 75 implants were installed worldwide. This is, however, a big improvement with respect to the previous year, when there were only 42 people implanted, which shows that the interest of such systems is increasing. The price at this time is about $\$ 150,000$ US dollars, excluding the cost of the implantation surgery and training to learn the use of the device. Nevertheless, we believe it is still important to keep researching on how to codify the environment to the type of information that those prostheses are able to provide, since medical enhancements are happening fast and the number of people using such aids could be much higher in the following years. Therefore, the technology to extract that information from the real world should be ready when the time comes and both lines of research converge. In our case, given the difficulty of working with real patients, we are currently working with SPV (Simulated Prosthetic Vision) [BermudezCameo et al., 2017]. The idea is to try different iconic representations and let people with normal vision make some trials only watching phosphene images to evaluate how well a representation is helpful to understand the environment. For this, immersive virtual reality glasses could be used to get more realistic results. In what concerns this thesis, we have started to develop virtual environments to study the mobility in indoor environments without colliding with walls or any obstacles.

Nevertheless, visual prostheses are not the only way to communicate with the user. Another line of future work is to explore other possibilities of communication of information. For example, in many works, audio signals and haptic feedback is typically used. Our group has worked in the past with audio signals [Aladren et al., 2016], with voice messages and a continuous audio signal that intensifies as the obstacle approaches, on the left or right headphone depending on from where the hazard comes. But while
this approach is functional and can be tested with blindfolded people, again, the collaboration of actual visually impaired would be desirable, since they are the best to express how the communication is more effective. For its daily life, it has been expressed that the constant audio signals could be overwhelming and uncomfortable if they are not implemented properly. Thus, it is necessary to include them in the process when developing an assistive device in order to not commit common mistakes. They could provide additional valuable input, such as other features they would want to have detected that we may not have think of. In general, a closer collaboration with visually impaired collectives is something we will seek in the future.

As announced in the introduction, we have predominantly used RGB-D cameras in this thesis. The motivations of this selection were clear, and the results deliver the expectations we had on these systems. However, this extensive usage of RGB-D cameras causes that our experimental evaluation has the limitations of the sensor. The most notable one is the poor performance under the influence of the sun, mostly outdoors. Our response to this problem has been to restrict our experimental evaluation to indoor environments. In the future, we would like to extend our study to outdoor environments as well. That would require using appropriate sensors. Most of the existing alternatives pose some other problems, such as poorer performance and accuracy on certain textures of stereo cameras, or heavy weight and power consumption of laser scanners. Recently, due to the increase interest on vision-based systems in areas such as autonomous vehicles or even optical developments in smartphones and similar devices, we think in the near future the state of technology would allow us to retrieve accurate 3D information very robustly. Nevertheless, switching most of our methods here presented to outdoor operation should be almost seamless for any sensor able to provide range images, at least in urban environments. In such environments, assumptions made such as Manhattan World [Coughlan and Yuille, 1999] and that most surfaces are planes should usually hold. However, the likelihood of facing irregularities is certainly higher than indoors, and if we consider non-urban environments even basic assumptions could not hold (e.g. the ground is a plane). Thus, some methods would require being adapted to such circumstances.

In the future we would like to explore novel techniques that are lately almost taking over the world of computer vision, namely deep learning. In the last few years, deep learning has exploded, beating old traditional methods of the state of the art in many tasks, such as object detection, segmentation or human pose tracking, among many others. Part of the reason why this is happening is due to technological advancements that arise from the exploitation of new and powerful GPUs for parallel computation. Before that, training a neural network with many layers was very costly, and thus these methods remained stagnant. The new and huge datasets that have been published also help to develop these approaches, considering deep learning methods are mainly datadriven.

For assistive computer vision, the applications that deep learning techniques could
have are endless. For example, object detection and recognition could be extremely useful for blind people, as well as facial recognition. Applications here presented could benefit from deep learning too, such as layout estimation or stair detection. It could be used for tasks not treated in this thesis but also relevant, such as door detection. The depth estimation from monocular images of works such as [Eigen and Fergus, 2015] could potentially be used to extract range images without a depth camera in the future. However, while deep learning produces good results in many tasks with a higher level of abstraction compared to what is achievable with traditional methods, it has still some limitations. For example, the performance depends on the amount and variability of the data used to train, and it may produce inaccurate results if it is the first time certain instance is viewed. Moreover, methods that work on 2D images usually lose the scale and 3D pose of the objects with respect to the user, which is crucial when giving indications to a person. Nowadays, deep learning is a great tool that solves specific tasks producing astonishing results, but in order to be functional in a complex and multi-tasking system moving in the 3D world it still needs the usage of geometric methods and reasoning as a part of it to integrate all the information. Besides, multi-layered neural networks often need powerful machines with large GPUs in order to run in real-time, which is a pre-requisite still hard to meet in a wearable system.

To conclude, we could say that the development of assistive devices it is still an open problem with many considerations to take into account, and many possible lines of research to move forward. This thesis proposes a collection of new methods that address some of the most important tasks related to mobility, but there are certainly others. The current climate of profound interest in computer vision along with the progressive enhancements in technology to expect in the near future are very promising and encouraging to keep working in this important topic.
6. CONCLUSIONS AND FUTURE WORK

## Bibliography

[Abdulatif et al., 2017] Abdulatif, S., Kleiner, B., Aziz, F., Riehs, C., Cooper, R., and Schneider, U. (2017). Stairs detection for enhancing wheelchair capabilities based on radar sensors. arXiv preprint arXiv:1711.09206.
[Ahmetovic et al., 2016] Ahmetovic, D., Gleason, C., Ruan, C., Kitani, K., Takagi, H., and Asakawa, C. (2016). NavCog: a navigational cognitive assistant for the blind. In Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services, pages 90-99. ACM.
[Ahuja et al., 2011] Ahuja, A., Dorn, J., Caspi, A., McMahon, M., Dagnelie, G., DaCruz, L., Stanga, P., Humayun, M., and Greenberg, R. (2011). Blind subjects implanted with the Argus II retinal prosthesis are able to improve performance in a spatial-motor task. British Journal of Ophthalmology, 95(4):539-543.
[Aladren et al., 2016] Aladren, A., Lopez-Nicolas, G., Puig, L., and Guerrero, J. J. (2016). Navigation assistance for the visually impaired using RGB-D sensor with range expansion. IEEE Systems Journal, Special Issue on Robotics \& Automation for Human Health, 10(3):922-932.
[Albert et al., 2001] Albert, A., Suppa, M., and Gerth, W. (2001). Detection of stair dimensions for the path planning of a bipedal robot. In IEEE/ASME International Conference on Advanced Intelligent Mechatronics, volume 2, pages 1291-1296.
[Armeni et al., 2017] Armeni, I., Sax, S., Zamir, A. R., and Savarese, S. (2017). Joint 2D-3D-Semantic data for indoor scene understanding. arXiv preprint arXiv:1702.01105.
[Ayton et al., 2013] Ayton, L. N., Luu, C. D., Bentley, S. A., Allen, P. J., and Guymer, R. H. (2013). Image processing for visual prostheses: A clinical perspective. In IEEE International Conference on Image Processing (ICIP), pages 1540-1544.
[Baltzakis et al., 2003] Baltzakis, H., Argyros, A., and Trahanias, P. (2003). Fusion of laser and visual data for robot motion planning and collision avoidance. Machine Vision and Applications, 15(2):92-100.
[Bansal et al., 2011] Bansal, M., Matei, B., Southall, B., Eledath, J., and Sawhney, H. (2011). A LIDAR streaming architecture for mobile robotics with application to 3D structure characterization. In IEEE International Conference on Robotics and Automation (ICRA), pages 1803-1810.
[Barnes, 2013] Barnes, N. (2013). An overview of vision processing in implantable prosthetic vision. In IEEE International Conference on Image Processing (ICIP), pages 1532-1535.
[Bazin et al., 2008] Bazin, J.-C., Kweon, I., Demonceaux, C., and Vasseur, P. (2008). A robust top-down approach for rotation estimation and vanishing points extraction by catadioptric vision in urban environment. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 346-353.
[Bermudez-Cameo et al., 2017] Bermudez-Cameo, J., Badias-Herbera, A., GuerreroViu, M., Lopez-Nicolas, G., and Guerrero, J. J. (2017). RGB-D computer vision techniques for simulated prosthetic vision. In Iberian Conference on Pattern Recognition and Image Analysis, pages 427-436.
[Bermudez-Cameo et al., 2014] Bermudez-Cameo, J., Lopez-Nicolas, G., and Guerrero, J. (2014). Line-images in cone mirror catadioptric systems. In IEEE International Conference on Pattern Recognition (ICPR), pages 2083-2088.
[Bermudez-Cameo et al., 2015] Bermudez-Cameo, J., Lopez-Nicolas, G., and Guerrero, J. J. (2015). Automatic line extraction in uncalibrated omnidirectional cameras with revolution symmetry. International Journal of Computer Vision, 114(1):16-37.
[Bernabei et al., 2011] Bernabei, D., Ganovelli, F., Benedetto, M., Dellepiane, M., and Scopigno, R. (2011). A low-cost time-critical obstacle avoidance system for the visually impaired. In International Conference on Indoor Positioning and Indoor Navigation.
[Borenstein and Ulrich, 1997] Borenstein, J. and Ulrich, I. (1997). The GuideCane - A computerized travel aid for the active guidance of blind pedestrians. In IEEE International Conference on Robotics and Automation (ICRA), volume 2, pages 1283-1288.
[Bostelman et al., 2006] Bostelman, R., Russo, P., Albus, J., Hong, T., and Madhavan, R. (2006). Applications of a 3D range camera towards healthcare mobility aids. In IEEE International Conference on Networking, Sensing and Control (ICNSC), pages 416-421.
[Bourne et al., 2017] Bourne, R. R., Flaxman, S. R., Braithwaite, T., Cicinelli, M. V., Das, A., Jonas, J. B., Keeffe, J., Kempen, J. H., Leasher, J., Limburg, H., et al. (2017). Magnitude, temporal trends, and projections of the global prevalence of blindness
and distance and near vision impairment: a systematic review and meta-analysis. The Lancet Global Health, 5(9):e888-e897.
[Briales and Gonzalez-Jimenez, 2015] Briales, J. and Gonzalez-Jimenez, J. (2015). A minimal solution for the calibration of a 2d laser-rangefinder and a camera based on scene corners. In Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, pages 1891-1896. IEEE.
[Brindley and Lewin, 1968] Brindley, G. S. and Lewin, W. S. (1968). The sensations produced by electrical stimulation of the visual cortex. The Journal of Physiology, 196:479-493.
[Brookshire and Teller, 2013] Brookshire, J. and Teller, S. (2013). Extrinsic calibration from per-sensor egomotion. Robotics: Science and Systems VIII, pages 504-512.
[Cabral and Furukawa, 2014] Cabral, R. and Furukawa, Y. (2014). Piecewise planar and compact floorplan reconstruction from images. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 628-635.
[Canny, 1986] Canny, J. (1986). A computational approach to edge detection. IEEE Transactions on pattern analysis and machine intelligence, (6):679-698.
[Cantón Toro, 2018] Cantón Toro, D. (Expected Sept. 2018). Virtual reality for simulated prosthetic vision with phosphene patterns. Bachelor thesis, Universidad de Zaragoza. Supervised by G. Lopez-Nicolas and A. Perez-Yus.
[Capi and Toda, 2011] Capi, G. and Toda, H. (2011). A new robotic system to assist visually impaired people. In IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), pages 259-263.
[Carbonara and Guaragnella, 2014] Carbonara, S. and Guaragnella, C. (2014). Efficient stairs detection algorithm assisted navigation for vision impaired people. In IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA), pages 313-318.
[Cha et al., 1992] Cha, K., Boman, D. K., Horch, K. W., and Normann, R. A. (1992). Reading speed with a pixelized vision system. Journal of the Optical Society of America A (JOSA A), 9(5):673-677.
[Chan et al., 2017] Chan, D. S., Silva, R. K., Monteiro, J. C., and Lizarralde, F. (2017). Efficient stairway detection and modeling for autonomous robot climbing. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 5916-5921.
[Chang et al., 2015] Chang, H.-C., Huang, S.-H., and Lai, S.-H. (2015). Using line consistency to estimate 3D indoor Manhattan scene layout from a single image. In IEEE International Conference on Image Processing (ICIP), pages 4723-4727.
[Chen et al., 2009a] Chen, S. C., Suaning, G. J., Morley, J. W., and Lovell, N. H. (2009a). Simulating prosthetic vision: I. visual models of phosphenes. Vision Research, 49(12):1493-1506.
[Chen et al., 2009b] Chen, S. C., Suaning, G. J., Morley, J. W., and Lovell, N. H. (2009b). Simulating prosthetic vision: II. measuring functional capacity. Vision research, 49(19):2329-2343.
[Choi et al., 2013] Choi, W., Chao, Y.-W., Pantofaru, C., and Savarese, S. (2013). Understanding indoor scenes using 3D geometric phrases. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
[Ciobanu et al., 2017] Ciobanu, A., Morar, A., Moldoveanu, F., Petrescu, L., Ferche, O., and Moldoveanu, A. (2017). Real-time indoor staircase detection on mobile devices. In IEEE International Conference on Control Systems and Computer Science (CSCS), pages 287-293.
[Coughlan and Yuille, 1999] Coughlan, J. M. and Yuille, A. L. (1999). Manhattan world: Compass direction from a single image by bayesian inference. In IEEE International Conference on Computer Vision (ICCV), volume 2, pages 941-947.
[Dagnelie, 2006] Dagnelie, G. (2006). Visual prosthetics 2006: assessment and expectations. Expert review of medical devices, 3(3):315-325.
[Dakopoulos and Bourbakis, 2010] Dakopoulos, D. and Bourbakis, N. G. (2010). Wearable obstacle avoidance electronic travel aids for blind: a survey. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 40(1):25-35.
[Dasgupta et al., 2016] Dasgupta, S., Fang, K., Chen, K., and Savarese, S. (2016). Delay: Robust spatial layout estimation for cluttered indoor scenes. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 616-624.
[Del Pero et al., 2012] Del Pero, L., Bowdish, J., Fried, D., Kermgard, B., Hartley, E., and Barnard, K. (2012). Bayesian geometric modeling of indoor scenes. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2719-2726.
[Del Pero et al., 2011] Del Pero, L., Guan, J., Brau, E., Schlecht, J., and Barnard, K. (2011). Sampling bedrooms. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2009-2016.
[Delage et al., 2006] Delage, E., Lee, H., and Ng, A. Y. (2006). A dynamic bayesian network model for autonomous 3D reconstruction from a single indoor image. In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), volume 2, pages 2418-2428.
[Delmerico et al., 2013] Delmerico, J. A., Baran, D., David, P., Ryde, J., and Corso, J. J. (2013). Ascending stairway modeling from dense depth imagery for traversability analysis. In IEEE International Conference on Robotics and Automation (ICRA), pages 2283-2290.
[Denis et al., 2014] Denis, G., Jouffrais, C., Mailhes, C., and Macé, M. J. (2014). Simulated prosthetic vision: Improving text accessibility with retinal prostheses. In Annual International Conference of the IEEE Engineering in Medicine and Biology Society, pages 1719-1722.
[Dong and Isler, 2017] Dong, W. and Isler, V. (2017). A novel method for the extrinsic calibration of a 2-d laser-rangefinder \& a camera. In Robotics and Automation (ICRA), 2017 IEEE International Conference on, pages 5104-5109. IEEE.
[Eigen and Fergus, 2015] Eigen, D. and Fergus, R. (2015). Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In IEEE International Conference on Computer Vision, pages 2650-2658.
[Endres et al., 2014] Endres, F., Sprunk, C., Kummerle, R., and Burgard, W. (2014). A catadioptric extension for RGB-D cameras. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 466-471.
[Faria et al., 2010] Faria, J., Lopes, S., Fernandes, H., Martins, P., and Barroso, J. (2010). Electronic white cane for blind people navigation assistance. In IEEE World Automation Congress (WAC).
[Feng et al., 2017] Feng, D., Barnes, N., and You, S. (2017). Dsd: Depth structural descriptor for edge-based assistive navigation. In IEEE International Conference on Computer Vision Workshops (ICCVW), pages 1536-1544.
[Feng and McCarthy, 2013] Feng, D. and McCarthy, C. (2013). Enhancing scene structure in prosthetic vision using iso-disparity contour perturbance maps. In International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), pages 5283-5286.
[Fernández Labrador, 2017] Fernández Labrador, C. (2017). Estimation of the 3D layout of indoor environments from panoramic images. Master thesis, Universidad de Zaragoza. Supervised by J.J. Guerrero and A. Perez-Yus.
[Fernandez-Labrador et al., 2018] Fernandez-Labrador, C., Perez-Yus, A., LopezNicolas, G., and Guerrero, J. J. (2018). Layouts from panoramic images with geometry and deep learning. Submitted to a journal.
[Fernandez-Madrigal and Claraco, 2013] Fernandez-Madrigal, J.-A. and Claraco, J. L. B. (2013). Simultaneous Localization and Mapping for Mobile Robots: Introduction and Methods. Information Science Reference.
[Fernandez-Moral et al., 2015] Fernandez-Moral, E., Gonzalez-Jimenez, J., and Arevalo, V. (2015). Extrinsic calibration of 2D laser rangefinders from perpendicular plane observations. The International Journal of Robotics Research, 34(11):14011417.
[Fernandez-Moral et al., 2014] Fernandez-Moral, E., Gonzalez-Jimenez, J., Rives, P., and Arevalo, V. (2014). Extrinsic calibration of a set of range cameras in 5 seconds without pattern. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 429-435.
[Filipe et al., 2012] Filipe, V., Fernandes, F., Fernandes, H., Sousa, A., Paredes, H., and Barroso, J. (2012). Blind navigation support system based on Microsoft Kinect. Procedia Computer Science, 14:94-101.
[Fischler and Bolles, 1981] Fischler, M. A. and Bolles, R. C. (1981). Random Sample Consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM, 24(6):381-395.
[Flint et al., 2011] Flint, A., Murray, D., and Reid, I. (2011). Manhattan scene understanding using monocular, stereo, and 3D features. In IEEE International Conference on Computer Vision (ICCV), pages 2228-2235.
[Fornos et al., 2011] Fornos, A. P., Sommerhalder, J., and Pelizzone, M. (2011). Reading with a simulated 60 -channel implant. Frontiers in neuroscience, 5.
[Fukano et al., 2016] Fukano, K., Mochizuki, Y., Iizuka, S., Simo-Serra, E., Sugimoto, A., and Ishikawa, H. (2016). Room reconstruction from a single spherical image by higher-order energy minimization. In IAPR International Conference on Pattern Recognition (ICPR), pages 1768-1773.
[Furlan et al., 2013] Furlan, A., Miller, S. D., Sorrenti, D. G., Li, F.-F., and Savarese, S. (2013). Free your camera: 3D indoor scene understanding from arbitrary camera motion. In British Machine Vision Conference (BMVC).
[Geiger et al., 2012] Geiger, A., Moosmann, F., Car, O., and Schuster, B. (2012). Automatic camera and range sensor calibration using a single shot. In IEEE International Conference on Robotics and Automation (ICRA), pages 3936-3943.
[Gomez-Ojeda et al., 2015] Gomez-Ojeda, R., Briales, J., Fernandez-Moral, E., and Gonzalez-Jimenez, J. (2015). Extrinsic calibration of a 2D laser-rangefinder and a camera based on scene corners. In IEEE International Conference on Robotics and Automation (ICRA), pages 3611-3616.
[Guerrero et al., 2015] Guerrero, J. J., Perez-Yus, A., Gutierrez-Gomez, D., Rituerto, A., and Lopez-Nicolas, G. (2015). Human navigation assistance with a RGB-D sensor. In VI Congreso Internacional de Diseño, Redes de Investigación y Tecnología para todos (DRT4ALL), pages 285-312.
[Guimaraes et al., 2013] Guimaraes, C. S. S., Henriques, R. V. B., and Pereira, C. E. (2013). Analysis and design of an embedded system to aid the navigation of the visually impaired. In ISSNIP Biosignals and Biorobotics Conference: Biosignals and Robotics for Better and Safer Living (BRC), pages 1-6.
[Gutierrez-Gomez et al., 2015] Gutierrez-Gomez, D., Mayol-Cuevas, W., and Guerrero, J. J. (2015). Inverse depth for accurate photometric and geometric error minimisation in RGB-D dense visual odometry. In IEEE International Conference on Robotics and Automation (ICRA), pages 83-89.
[Gutmann et al., 2004] Gutmann, J.-S., Fukuchi, M., and Fujita, M. (2004). Stair climbing for humanoid robots using stereo vision. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), volume 2, pages 1407-1413.
[Ha, 2012] Ha, J.-E. (2012). Extrinsic calibration of a camera and laser range finder using a new calibration structure of a plane with a triangular hole. International Journal of Control, Automation and Systems, 10(6):1240-1244.
[Ha et al., 2016] Ha, S., Khraiche, M. L., Akinin, A., Jing, Y., Damle, S., Kuang, Y., Bauchner, S., Lo, Y.-H., Freeman, W. R., Silva, G. A., et al. (2016). Towards highresolution retinal prostheses with direct optical addressing and inductive telemetry. Journal of neural engineering, 13(5):056008.
[Habib et al., 2002] Habib, A. F., Morgan, M., and Lee, Y.-R. (2002). Bundle adjustment with self-calibration using straight lines. The Photogrammetric Record, 17(100):635-650.
[Harms et al., 2015] Harms, H., Rehder, E., Schwarze, T., and Lauer, M. (2015). Detection of ascending stairs using stereo vision. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 2496-2502.
[Hartley, 1993] Hartley, R. I. (1993). Camera calibration using line correspondences. In Proc. DARPA Image Understanding Workshop, pages 361-366.
[Hedau et al., 2009] Hedau, V., Hoiem, D., and Forsyth, D. (2009). Recovering the spatial layout of cluttered rooms. In IEEE International Conference on Computer Vision (ICCV), pages 1849-1856.
[Hedau et al., 2010] Hedau, V., Hoiem, D., and Forsyth, D. (2010). Thinking inside the box: Using appearance models and context based on room geometry. In European Conference on Computer Vision (ECCV), pages 224-237. Springer.
[Heikkilä, 2000] Heikkilä, J. (2000). Geometric camera calibration using circular control points. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(10):1066-1077.
[Heng et al., 2013] Heng, L., Li, B., and Pollefeys, M. (2013). CamOdoCal: Automatic intrinsic and extrinsic calibration of a rig with multiple generic cameras and odometry. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1793-1800.
[Hernandez and Jo, 2010] Hernandez, D. C. and Jo, K.-H. (2010). Outdoor stairway segmentation using vertical vanishing point and directional filter. In IEEE International Forum on Strategic Technology (IFOST), pages 82-86.
[Herrera et al., 2012] Herrera, D., Kannala, J., and Heikkilä, J. (2012). Joint depth and color camera calibration with distortion correction. IEEE Transactions on Pattern Analysis and Machine Intelligence, 34(10):2058-2064.
[Hesch et al., 2010] Hesch, J. A., Mariottini, G. L., and Roumeliotis, S. I. (2010). Descending-stair detection, approach, and traversal with an autonomous tracked vehicle. In IEEE International Conference on Intelligent Robots and Systems (IROS), pages 5525-5531.
[Hoiem et al., 2007] Hoiem, D., Efros, A. A., and Hebert, M. (2007). Recovering surface layout from an image. International Journal of Computer Vision, 75(1):151.
[Holland and Welsch, 1977] Holland, P. W. and Welsch, R. E. (1977). Robust regression using iteratively reweighted least-squares. Communications in Statistics-theory and Methods, 6(9):813-827.
[Hoover et al., 1996] Hoover, A., Jean-Baptiste, G., Jiang, X., Flynn, P. J., Bunke, H., Goldgof, D. B., Bowyer, K., Eggert, D. W., Fitzgibbon, A., and Fisher, R. B. (1996). An experimental comparison of range image segmentation algorithms. IEEE Transactions on Pattern Analysis and Machine Intelligence, 18(7):673-689.
[Horne et al., 2016] Horne, L., Alvarez, J., McCarthy, C., Salzmann, M., and Barnes, N. (2016). Semantic labeling for prosthetic vision. Computer Vision and Image Understanding, 149:113-125.
[Horne et al., 2012] Horne, L., Barnes, N., McCarthy, C., and He, X. (2012). Image segmentation for enhancing symbol recognition in prosthetic vision. In Annual International Conference of the IEEE Engineering in Medicine and Biology Society, pages 2792-2795.
[Hu et al., 2016] Hu, Z., Li, Y., Li, N., and Zhao, B. (2016). Extrinsic calibration of 2-D laser rangefinder and camera from single shot based on minimal solution. IEEE Transactions on Instrumentation and Measurement, 65(4):915-929.
[Iglesias et al., 2016] Iglesias, J., Mirado, P., and Ventura, R. (2016). Towards an omnidirectional catadioptric RGB-D camera. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 2506-2513.
[Ishiwata et al., 2013] Ishiwata, K., Sekiguchi, M., Fuchida, M., and Nakamura, A. (2013). Basic study on step detection system for the visually impaired. In IEEE International Conference on Mechatronics and Automation (ICMA), pages 1332-1337.
[Jia and Li, 2013] Jia, H. and Li, S. (2013). Estimating the structure of rooms from a single fisheye image. In IAPR Asian Conference on Pattern Recognition (ACPR), pages 818-822. IEEE.
[Jia and Li, 2015] Jia, H. and Li, S. (2015). Estimating structure of indoor scene from a single full-view image. In IEEE International Conference on Robotics and Automation (ICRA), pages 4851-4858.
[Josh et al., 2013] Josh, H., Mann, C., Kleeman, L., and Lui, W. L. D. (2013). Psychophysics testing of bionic vision image processing algorithms using an FPGA hatpack. In IEEE International Conference on Image Processing (ICIP), pages 1550-1554.
[Jung et al., 2015] Jung, J.-H., Aloni, D., Yitzhaky, Y., and Peli, E. (2015). Active confocal imaging for visual prostheses. Vision research, 111:182-196.
[Kerl et al., 2013] Kerl, C., Sturm, J., and Cremers, D. (2013). Robust odometry estimation for rgb-d cameras. In IEEE International Conference on Robotics and Automation (ICRA), pages 3748-3754.
[Kiral-Kornek et al., 2013] Kiral-Kornek, F. I., Savage, C. O., O’Sullivan-Greene, E., Burkitt, A. N., and Grayden, D. B. (2013). Embracing the irregular: A patient-specific image processing strategy for visual prostheses. In Annual International Conference of the IEEE Engineering in Medicine and Biology Society, pages 3563-3566.
[Konolige and Mihelich, 2015] Konolige, K. and Mihelich, P. (2010, [Online; accessed 14-April-2015]). Technical description of kinect calibration. http://wiki.ros.org/kinect_calibration/technical.
[Kulyukin et al., 2004] Kulyukin, V., Gharpure, C., Nicholson, J., and Pavithran, S. (2004). RFID in robot-assisted indoor navigation for the visually impaired. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), volume 2, pages 1979-1984.
[Lee et al., 2012] Lee, C.-H., Su, Y.-C., and Chen, L.-G. (2012). An intelligent depthbased obstacle detection system for visually-impaired aid applications. In IEEE International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS).
[Lee et al., 2017] Lee, C.-Y., Badrinarayanan, V., Malisiewicz, T., and Rabinovich, A. (2017). Roomnet: End-to-end room layout estimation. pages 4875-4884.
[Lee et al., 2010] Lee, D. C., Gupta, A., Hebert, M., and Kanade, T. (2010). Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces. In Advances in Neural Information Processing Systems 23, pages 1288-1296. Curran Associates, Inc.
[Lee et al., 2009] Lee, D. C., Hebert, M., and Kanade, T. (2009). Geometric reasoning for single image structure recovery. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2136-2143.
[Lee, 2016] Lee, G. H. (2016). A minimal solution for non-perspective pose estimation from line correspondences. In European Conference on Computer Vision (ECCV), pages 170-185. Springer.
[Lee and Medioni, 2011] Lee, Y. H. and Medioni, G. (2011). A RGB-D camera based navigation for the visually impaired. In RSS Workshop on RGBD: Advanced Reasoning with Depth Camera.
[Li et al., 2016] Li, B., Munoz, J. P., Rong, X., Xiao, J., Tian, Y., and Arditi, A. (2016). ISANA: wearable context-aware indoor assistive navigation with obstacle avoidance for the blind. In European Conference on Computer Vision Workshops (ECCVW), pages 448-462. Springer.
[Li, 2013] Li, W. H. (2013). Wearable computer vision systems for a cortical visual prosthesis. In Workshop on Assistive Computer Vision and Robotics (ACVR) - Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCVW), pages 428-435.
[Li, 2015] Li, W. H. (2015). A Fast and Flexible Computer Vision System for Implanted Visual Prostheses, pages 686-701. Springer International Publishing, Cham.
[Li et al., 2012] Li, Y., McCarthy, C., and Barnes, N. (2012). On just noticeable difference for bionic eye. In Annual International Conference of the IEEE Engineering in Medicine and Biology Society, pages 2961-2964.
[Lopez-Nicolas et al., 2014] Lopez-Nicolas, G., Omedes, J., and Guerrero, J. J. (2014). Spatial layout recovery from a single omnidirectional image and its matching-free sequential propagation. Robotics and Autonomous Systems, 62(9):1271-1281.
[Lu and Manduchi, 2005] Lu, X. and Manduchi, R. (2005). Detection and localization of curbs and stairways using stereo vision. In IEEE International Conference on Robotics and Automation (ICRA), volume 4, pages 4648-4654.
[Lui et al., 2012] Lui, W. L. D., Browne, D., Kleeman, L., Drummond, T., and Li, W. H. (2012). Transformative reality: improving bionic vision with robotic sensing. In Annual International Conference of the IEEE Engineering in Medicine and Biology Society, pages 304-307.
[Mallya and Lazebnik, 2015] Mallya, A. and Lazebnik, S. (2015). Learning informative edge maps for indoor scene layout prediction. In IEEE International Conference on Computer Vision (ICCV), pages 936-944.
[Manduchi and Kurniawan, 2011] Manduchi, R. and Kurniawan, S. (2011). Mobilityrelated accidents experienced by people with visual impairment. Research and Practice in Visual Impairment and Blindness, 4(2):44-54.
[Mann et al., 2011] Mann, S., Huang, J., Janzen, R., Lo, R., Rampersad, V., Chen, A., and Doha, T. (2011). Blind navigation with a wearable range camera and vibrotactile helmet. In Proceedings of the 19th ACM international conference on Multimedia, pages 1325-1328.
[Martins et al., 2015] Martins, R., Fernandez-Moral, E., and Rives, P. (2015). Dense accurate urban mapping from spherical RGB-D images. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 6259-6264.
[Mayol-Cuevas et al., 2009] Mayol-Cuevas, W. W., Tordoff, B. J., and Murray, D. W. (2009). On the choice and placement of wearable vision sensors. Systems, Man and Cybernetics, Part A: Systems and Humans, 39(2):414-425.
[McCarthy et al., 2011] McCarthy, C., Barnes, N., and Lieby, P. (2011). Ground surface segmentation for navigation with a low resolution visual prosthesis. In Annual International Conference of the IEEE Engineering in Medicine and Biology Society, pages 4457-4460.
[McCarthy et al., 2013] McCarthy, C., Feng, D., and Barnes, N. (2013). Augmenting intensity to enhance scene structure in prosthetic vision. In IEEE International Conference on Multimedia and Expo Workshops (ICMEW), pages 1-6.
[McCarthy et al., 2014] McCarthy, C., Walker, J. G., Lieby, P., Scott, A., and Barnes, N. (2014). Mobility and low contrast trip hazard avoidance using augmented depth. Journal of neural engineering, 12(1):016003.
[Meffin, 2013] Meffin, H. (2013). What limits spatial perception with retinal implants? In IEEE International Conference on Image Processing, pages 1545-1549.
[Mihankhah et al., 2009] Mihankhah, E., Kalantari, A., Aboosaeedan, E., Taghirad, H. D., Ali, S., and Moosavian, A. (2009). Autonomous staircase detection and stair climbing for a tracked mobile robot using fuzzy controller. In IEEE International Conference on Robotics and Biomimetics (ROBIO), pages 1980-1985.
[Mikhelson et al., 2014] Mikhelson, I. V., Lee, P. G., Sahakian, A. V., Wu, Y., and Katsaggelos, A. K. (2014). Automatic, fast, online calibration between depth and color cameras. Journal of Visual Communication and Image Representation, 25(1):218226.
[Molton et al., 1998] Molton, N., Se, S., Brady, J., Lee, D., and Probert, P. (1998). A stereo vision-based aid for the visually impaired. Image and vision computing, 16(4):251-263.
[Munoz et al., 2016] Munoz, R., Rong, X., and Tian, Y. (2016). Depth-aware indoor staircase detection and recognition for the visually impaired. In IEEE International Conference on Multimedia \& Expo Workshops (ICMEW), pages 1-6.
[Oktem et al., 2008] Oktem, R., Aydin, E., and Cagiltay, N. (2008). An indoor navigation aid designed for visually impaired people. IEEE Conference on Industrial Electronics (IECON), pages 2982-2987.
[Oßwald et al., 2011a] Oßwald, S., Gorog, A., Hornung, A., and Bennewitz, M. (2011a). Autonomous climbing of spiral staircases with humanoids. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 4844-4849.
[Oßwald et al., 2011b] Oßwald, S., Gutmann, J.-S., Hornung, A., and Bennewitz, M. (2011b). From 3d point clouds to climbing stairs: A comparison of plane segmentation approaches for humanoids. In IEEE-RAS International Conference on Humanoid Robots, pages 93-98.
[Oßwald et al., 2012] Oßwald, S., Hornung, A., and Bennewitz, M. (2012). Improved proposals for highly accurate localization using range and vision data. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 18091814.
[Otero Moliner, 2017] Otero Moliner, E. (2017). Development of simulated prosthetic vision in virtual environments. Bachelor thesis, Universidad de Zaragoza. Supervised by G. Lopez-Nicolas and A. Perez-Yus.
[Park et al., 2011] Park, C.-S., Seo, E.-H., Kim, D., You, B.-J., and Oh, S.-R. (2011). Stair boundary extraction using the 2d laser scanner. In IEEE International Conference on Mechatronics and Automation (ICMA), pages 1538-1543.
[Peasley and Birchfield, 2013] Peasley, B. and Birchfield, S. (2013). Real-time obstacle detection and avoidance in the presence of specular surfaces using an active 3D sensor. In IEEE Workshop on Robot Vision (WORV), pages 197-202.
[Perez-Yus et al., 2017a] Perez-Yus, A., Bermudez-Cameo, J., Lopez-Nicolas, G., and Guerrero, J. J. (2017a). Depth and motion cues with phosphene patterns for prosthetic vision. In IEEE International Conference on Computer Vision Workshops (ICCVW), pages 1516-1525.
[Perez-Yus et al., 2018a] Perez-Yus, A., Fernandez-Moral, E., Lopez-Nicolas, G., Guerrero, J. J., and Rives, P. (2018a). Extrinsic calibration of multiple RGB-D cameras from line observations. IEEE Robotics and Automation Letters, 3(1):273-280.
[Perez-Yus et al., 2017b] Perez-Yus, A., Gutierrez-Gomez, D., Lopez-Nicolas, G., and Guerrero, J. J. (2017b). Stairs detection with odometry-aided traversal from a wearable RGB-D camera. Computer Vision and Image Understanding, 154:192-205.
[Perez-Yus et al., 2015] Perez-Yus, A., Lopez-Nicolas, G., and Guerrero, J. J. (2015). Detection and modelling of staircases using a wearable depth sensor. ECCV 2014 Workshops, Part III, LNCS 8927(3):449-463.
[Perez-Yus et al., 2016a] Perez-Yus, A., Lopez-Nicolas, G., and Guerrero, J. J. (2016a). A novel hybrid camera system with depth and fisheye cameras. In IAPR International Conference on Pattern Recognition (ICPR), pages 2789-2794.
[Perez-Yus et al., 2016b] Perez-Yus, A., Lopez-Nicolas, G., and Guerrero, J. J. (2016b). Peripheral expansion of depth information via layout estimation with fisheye camera. In European Conference on Computer Vision (ECCV), pages 396-412. Springer.
[Perez-Yus et al., 2018b] Perez-Yus, A., Lopez-Nicolas, G., and Guerrero, J. J. (2018b). Scaled layout recovery with wide field of view RGB-D. Submitted to a journal.
[Perez-Yus et al., 2018c] Perez-Yus, A., Puig, L., Lopez-Nicolas, G., Guerrero, J. J., and Fox, D. (2018c). RGB-D based tracking of complex objects. Understanding Human Activities through 3D Sensors Workshop (UHA3DS), in conjunction with ICPR 2016, LNCS 10188.
[Pradeep et al., 2010] Pradeep, V., Medioni, G., and Weiland, J. (2010). Robot vision for the visually impaired. In IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pages 15-22.
[Pradeep et al., 2008] Pradeep, V., Medioni, G., Weiland, J., et al. (2008). Piecewise planar modeling for step detection using stereo vision. In Workshop on Computer Vision Applications for the Visually Impaired.
[Přibyl et al., 2017] Přibyl, B., Zemčík, P., and Čadík, M. (2017). Absolute pose estimation from line correspondences using direct linear transformation. Computer Vision and Image Understanding, 161:130-144.
[Puig et al., 2012] Puig, L., Bermudez-Cameo, J., Sturm, P., and Guerrero, J. J. (2012). Calibration of omnidirectional cameras in practice: A comparison of methods. Computer Vision and Image Understanding, 116(1):120-137.
[Qian and Ye, 2013] Qian, X. and Ye, C. (2013). NCC-RANSAC: A fast plane extraction method for navigating a smart cane for the visually impaired. In IEEE International Conference on Automation Science and Engineering (CASE), pages 261267.
[Rabbani et al., 2006] Rabbani, T., van den Heuvel, F., and Vosselmann, G. (2006). Segmentation of point clouds using smoothness constraint. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, 36(5):248-253.
[Ramalingam et al., 2013] Ramalingam, S., Pillai, J., Jain, A., and Taguchi, Y. (2013) Manhattan junction catalogue for spatial reasoning of indoor scenes. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3065-3072.
[Raposo et al., 2013] Raposo, C., Lourenço, M., Antunes, M., and Barreto, J. P. (2013). Plane-based odometry using an RGB-D camera. In British Machine Vision Conference.
[Rituerto et al., 2010] Rituerto, A., Puig, L., and Guerrero, J. (2010). Visual slam with an omnidirectional camera. In 20th IEEE International Conference on Pattern Recognition (ICPR), pages 348-351.
[Rodriguez et al., 2012] Rodriguez, A., Yebes, J. J., Alcantarilla, P. F., Bergasa, L. M., Almazan, J., and Cela, A. (2012). Assisting the visually impaired: obstacle detection and warning system by acoustic feedback. Sensors, 12(12):17476-17496.
[Romić et al., 2017] Romić, K., Galić, I., and Galba, T. (2017). Technology assisting the blind-routing on the staircases using wide-angle camera. In IEEE International Symposium ELMAR, pages 43-46.
[Rusu and Cousins, 2011] Rusu, R. B. and Cousins, S. (2011). 3D is here: Point cloud library (PCL). In IEEE International Conference on Robotics and Automation (ICRA), pages 1-4.
[Scaramuzza et al., 2007] Scaramuzza, D., Harati, A., and Siegwart, R. (2007). Extrinsic self calibration of a camera and a 3D laser range finder from natural scenes. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 4164-4169.
[Scaramuzza et al., 2006] Scaramuzza, D., Martinelli, A., and Siegwart, R. (2006). A toolbox for easily calibrating omnidirectional cameras. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 5695-5701.
[Schafer et al., 2008] Schafer, H., Hach, A., Proetzsch, M., and Berns, K. (2008). 3D obstacle detection and avoidance in vegetated off-road terrain. In IEEE International Conference on Robotics and Automation (ICRA), pages 923-928.
[Schneider et al., 2013] Schneider, S., Luettel, T., and Wuensche, H.-J. (2013). Odometry-based online extrinsic sensor calibration. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1287-1292.
[Schwarze and Zhong, 2015] Schwarze, T. and Zhong, Z. (2015). Stair detection and tracking from egocentric stereo vision. In IEEE International Conference onImage Processing (ICIP), pages 2690-2694.
[Schwing et al., 2013] Schwing, A. G., Fidler, S., Pollefeys, M., and Urtasun, R. (2013). Box in the box: Joint 3D layout and object reasoning from single images. In IEEE International Conference on Computer Vision (ICCV), pages 353-360.
[Schwing et al., 2012] Schwing, A. G., Hazan, T., Pollefeys, M., and Urtasun, R. (2012). Efficient structured prediction for 3D indoor scene understanding. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2815-2822.
[Se and Brady, 2000] Se, S. and Brady, M. (2000). Vision-based detection of staircases. In Asian Conference on Computer Vision (ACCV), volume 1, pages 535-540.
[Shoval et al., 1998] Shoval, S., Borenstein, J., and Koren, Y. (1998). The Navbelt - A computerized travel aid for the blind based on mobile robotics technology. Transactions on Biomedical Engineering, 45(11):1376-1386.
[Shoval et al., 2000] Shoval, S., Ulrich, I., Borenstein, J., et al. (2000). Computerized obstacle avoidance systems for the blind and visually impaired. Intelligent Systems and Technologies in Rehabilitation Engineering, pages 414-448.
[Silberman et al., 2012] Silberman, N., Hoiem, D., Kohli, P., and Fergus, R. (2012). Indoor segmentation and support inference from RGBD images. In European Conference on Computer Vision (ECCV), pages 746-760.
[Smisek et al., 2013] Smisek, J., Jancosek, M., and Pajdla, T. (2013). 3d with kinect. In Consumer Depth Cameras for Computer Vision, pages 3-25. Springer.
[Soheilian et al., 2013] Soheilian, B., Tournaire, O., Paparoditis, N., Vallet, B., and Papelard, J.-P. (2013). Generation of an integrated 3D city model with visual landmarks for autonomous navigation in dense urban areas. In IEEE Intelligent Vehicles Symposium (IV), pages 304-309.
[Srivastava, 2011] Srivastava, N. R. (2011). Simulations of cortical prosthetic vision. In Visual Prosthetics, pages 355-365. Springer.
[Stacey et al., 2011] Stacey, A., Li, Y., and Barnes, N. (2011). A salient information processing system for bionic eye with application to obstacle avoidance. In Annual International Conference of the IEEE Engineering in Medicine and Biology Society, pages 5116-5119.
[Stahlschmidt et al., 2015a] Stahlschmidt, C., Gavriilidis, A., and Kummert, A. (2015a). Classification of ascending steps and stairs using time-of-flight sensor data. In IEEE 9th International Workshop on Multidimensional (nD) Systems ( $n D S$ ), pages 1-6.
[Stahlschmidt et al., 2015b] Stahlschmidt, C., Gavriilidis, A., and Kummert, A. (2015b). Posture independent stair parameter estimation. In IEEE International Symposium on Intelligent Control (ISIC), pages 65-70.
[Stahlschmidt et al., 2015c] Stahlschmidt, C., von Camen, S., Gavriilidis, A., and Kummert, A. (2015c). Descending step classification using time-of-flight sensor data. In IEEE Intelligent Vehicles Symposium (IV), pages 362-367.
[Szeliski and Shum, 1997] Szeliski, R. and Shum, H.-Y. (1997). Creating full view panoramic image mosaics and environment maps. In International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), pages 251-258. ACM Press/Addison-Wesley Publishing Co.
[Taguchi et al., 2013] Taguchi, Y., Jian, Y.-D., Ramalingam, S., and Feng, C. (2013). Point-plane SLAM for hand-held 3D sensors. In IEEE International Conference on Robotics and Automation (ICRA), pages 5182-5189.
[Tang et al., 2012] Tang, T. J. J., Lui, W. L. D., and Li, W. H. (2012). Plane-based detection of staircases using inverse depth. Australasian Conference on Robotics and Automation (ACRA).
[Thompson et al., 2003] Thompson, R. W., Barnett, G. D., Humayun, M. S., and Dagnelie, G. (2003). Facial recognition using simulated prosthetic pixelized vision. Investigative ophthalmology \& visual science, 44(11):5035-5042.
[Tian et al., 2013] Tian, Y., Yang, X., Yi, C., and Arditi, A. (2013). Toward a computer vision-based wayfinding aid for blind persons to access unfamiliar indoor environments. Machine vision and applications, 24(3):521-535.
[Tomari et al., 2012] Tomari, R., Kobayashi, Y., and Kuno, Y. (2012). Wide field of view Kinect undistortion for social navigation implementation. In Advances in Visual Computing, pages 526-535. Springer.
[Ueda et al., 2006] Ueda, T., Kawata, H., Tomizawa, T., Ohya, A., and Yuta, S. (2006). Visual information assist system using 3D SOKUIKI sensor for blind people, system concept and object detecting experiments. In IEEE Conference on Industrial Electronics (IECON), pages 3058-3063.
[van Rheede et al., 2010] van Rheede, J. J., Kennard, C., and Hicks, S. L. (2010). Simulating prosthetic vision: Optimizing the information content of a limited visual display. Journal of vision, 10(14):32-32.
[Vasconcelos et al., 2012] Vasconcelos, F., Barreto, J. P., and Nunes, U. (2012). A minimal solution for the extrinsic calibration of a camera and a laser-rangefinder. IEEE Transactions on Pattern Analysis and Machine Intelligence, 34(11):2097-2107.
[Vergnieux et al., 2014] Vergnieux, V., Macé, M. J. M., and Jouffrais, C. (2014). Wayfinding with simulated prosthetic vision: Performance comparison with regular and structure-enhanced renderings. In International Conference of the IEEE Engineering in Medicine and Biology Society, pages 2585-2588.
[Vlaminck et al., 2013] Vlaminck, M., Jovanov, L., Van Hese, P., Goossens, B., Philips, W., and Pizurica, A. (2013). Obstacle detection for pedestrians with a visual impairment based on 3D imaging. In IEEE International Conference on 3D Imaging (IC3D).
[Von Gioi et al., 2010] Von Gioi, R. G., Jakubowicz, J., Morel, J.-M., and Randall, G. (2010). Lsd: A fast line segment detector with a false detection control. IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(4):722-732.
[Wang et al., 2017] Wang, H.-C., Katzschmann, R. K., Teng, S., Araki, B., Giarre, L., and Rus, D. (2017). Enabling independent navigation for visually impaired people through a wearable vision-based feedback system. In IEEE International Conference on Robotics and Automation (ICRA), pages 6533-6540.
[Wang et al., 2014a] Wang, J., Wu, X., Lu, Y., Wu, H., Kan, H., and Chai, X. (2014a). Face recognition in simulated prosthetic vision: face detection-based image processing strategies. Journal of neural engineering, 11(4):046009.
[Wang and Tian, 2012] Wang, S. and Tian, Y. (2012). Detecting stairs and pedestrian crosswalks for the blind by RGBD camera. In IEEE International Conference on Bioinformatics and Biomedicine Workshops (BIBMW), pages 732-739.
[Wang and Wang, 2009] Wang, S. and Wang, H. (2009). 2D staircase detection using real adaboost. In IEEE International Conference on Information, Communications and Signal Processing (ICICS), pages 1-5.
[Wang et al., 2014b] Wang, Z., Liu, H., Wang, X., and Qian, Y. (2014b). Segment and label indoor scene based on RGB-D for the visually impaired. In MultiMedia Modeling, pages 449-460. Springer.
[Weiland et al., 2012] Weiland, J. D., Parikh, N., Pradeep, V., and Medioni, G. (2012). Smart image processing system for retinal prosthesis. In Annual International Conference of the IEEE Engineering in Medicine and Biology Society, pages 300-303.
[Westfechtel et al., 2016] Westfechtel, T., Ohno, K., Mertsching, B., Nickchen, D., Kojima, S., and Tadokoro, S. (2016). 3D graph based stairway detection and localization for mobile robots. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 473-479.
[Wong et al., 2003] Wong, F., Nagarajan, R., and Yaacob, S. (2003). Application of stereovision in a navigation aid for blind people. Proceedings of the 2003 Joint Conference of the Fourth International Conference on Information, Communications and Signal Processing, 2003 and Fourth Pacific Rim Conference on Multimedia, 2:734737 vol.2.
[Xu et al., 2017] Xu, J., Stenger, B., Kerola, T., and Tung, T. (2017). Pano2CAD: Room layout from a single panorama image. In IEEE Winter Conference on Applications of Computer Vision (WACV), pages 354-362.
[Yang and Zhang, 2016] Yang, H. and Zhang, H. (2016). Efficient 3D room shape recovery from a single panorama. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 5422-5430.
[Yoshida et al., 2011] Yoshida, T., Kitani, K. M., Koike, H., Belongie, S., and Schlei, K. (2011). EdgeSonic: image feature sonification for the visually impaired. In Proceedings of the 2nd Augmented Human International Conference, page 11. ACM.
[Zapf et al., 2016] Zapf, M. P. H., Boon, M.-Y., Lovell, N. H., and Suaning, G. J. (2016). Assistive peripheral phosphene arrays deliver advantages in obstacle avoidance in simulated end-stage retinitis pigmentosa: a virtual-reality study. Journal of neural engineering, 13(2):026022.
[Zhang and Zhang, 2011] Zhang, C. and Zhang, Z. (2011). Calibration between depth and color sensors for commodity depth cameras. In International Workshop on Hot Topics in 3D, in conjunction with ICME. IEEE.
[Zhang and Pless, 2004] Zhang, Q. and Pless, R. (2004). Extrinsic calibration of a camera and laser range finder (improves camera calibration). In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), volume 3, pages 2301-2306.
[Zhang et al., 2014a] Zhang, Q., Shen, X., Xu, L., and Jia, J. (2014a). Rolling guidance filter. In European Conference on Computer Vision (ECCV), pages 815-830. Springer.
[Zhang et al., 2017] Zhang, W., Zhang, W., Liu, K., and Gu, J. (2017). Learning to predict high-quality edge maps for room layout estimation. IEEE Transactions on Multimedia, 19(5):935-943.
[Zhang et al., 2014b] Zhang, Y., Song, S., Tan, P., and Xiao, J. (2014b). PanoContext: A whole-room 3D context model for panoramic scene understanding. In European Conference on Computer Vision (ECCV), pages 668-686. Springer.
[Zhang et al., 2016] Zhang, Y., Zhou, L., Liu, H., and Shang, Y. (2016). A flexible online camera calibration using line segments. Journal of Sensors.
[Zhang, 1999] Zhang, Z. (1999). Flexible camera calibration by viewing a plane from unknown orientations. In IEEE International Conference on Computer Vision (ICCV), volume 1, pages 666-673.
[Zhou, 2014] Zhou, L. (2014). A new minimal solution for the extrinsic calibration of a 2d lidar and a camera using three plane-line correspondences. Sensors Journal, 14(2):442-454.
[Zöllner et al., 2011] Zöllner, M., Huber, S., Jetter, H.-C., and Reiterer, H. (2011). NAVI-A proof-of-concept of a mobile navigational aid for visually impaired based on the Microsoft Kinect. Springer.


[^0]:    ${ }^{1}$ https://www.ultracane.com/

[^1]:    2https://www.orcam.com/en/

[^2]:    ${ }^{3}$ http://www.secondsight.com/

[^3]:    ${ }^{1}$ Código Técnico de la Edificación (2006). CTE, DB-SUA, section 1.4.2., http://www. codigotecnico.org/

[^4]:    ${ }^{1}$ http://lsun.cs.princeton.edu/

[^5]:    ${ }^{2}$ http://webdiis.unizar.es/~jguerrer/Publicaciones_archivos/2016_ECCV_ video_PeripheralExpansion.mp4

[^6]:    ${ }^{3}$ http://wiki.ros.org/tango_ros_streamer

[^7]:    ${ }^{1}$ http://www.ros.org/
    ${ }^{2}$ http://gazebosim.org/

[^8]:    ${ }^{3}$ http://webdiis.unizar.es/\%7Eglopez/spv.html

