IMPERIAL COLLEGE LONDON

Department of Computing

Robot SLAM and Navigation with Multi-Camera Computer Vision

Gerardo Carrera Mendoza

Supervised by Dr. Andrew Davison

Submitted in part fulfilment of the requirements for the degree of PhD in Computing and the Diploma of Imperial College London. This thesis is entirely my own work, and, except where otherwise indicated, describes my own research.

March 29, 2012

Dedication

A mis papas y mi hermano, cuyo apoyo ha sido fundamental para poder completar mis estudios de doctorado.

Acknowledgements

My years in London as a student have given me the opportunity to realise that a PhD is not only about research and academia but a complete learning experience in every aspect of my life.

I am very grateful to my supervisor Dr. Andrew Davison for his timeless dedication, support, patience and exceptional guidance. His full commitment and endless passion to research were a fantastic source of inspiration and motivation through my life at Imperial College.

I would like to thank Dr. Adrien Angeli for his brilliant advice. I appreciate deeply the many conversations we had and the time and ideas we shared inside and outside the lab. To Ankur Handa for his support, his great ideas and for being a fantastic companion in the lab. To Hauke Strasdat for his valuable comments. To all my labmates and colleagues Margarita Chli, Klaus, Pablote, Steven Lovegrove, Richard Newcombe and Renato Salas.

I really appreciate the opportunity that the government of Mexico gave me for the completion of these PhD studies by awarding me a CONACYT scholarship.

Abstract

In this thesis we focus on computer vision capabilities suitable for practical mass-market mobile robots, with an emphasis on techniques using rigs of multiple standard cameras rather than more specialised sensors. We analyse the state of the art of service robotics, and attempt to distill the vision capabilities which will be required of mobile robots over the mid and long-term future to permit autonomous localisation, mapping and navigation while integrating with other task-based vision requirements.

The first main novel contribution of the work is to consider how an ad-hoc multi-camera rig can be used as the basis for metric navigation competences such as feature-based Simultaneous Localisation and Mapping (SLAM). The key requirement for the use of such techniques with multiple cameras is accurate calibration of the locations of the cameras as mounted on the robot. This is a challenging problem, since we consider the general case where the cameras might be mounted all around the robot with arbitrary 3D locations and orientations, and may have fields of view which do not intersect.

In the second main part of the thesis, we move away from the idea that all cameras should contribute in a uniform manner to a single consistent metric representation, inspired by recent work on SLAM systems which have demonstrated impressive performance by a combination of off-the-shelf or simple techniques which we generally categorise by the term 'lightweight'. We develop a multi-camera mobile robot vision system which goes beyond pure localisation and SLAM to permit fully autonomous mapping navigation within a cluttered room, requiring free-space mapping and obstacle-avoiding planning capabilities.

In the last part of the work we investigate the trade-offs involved in defining a camera rig suitable for this type of vision system and perform some experiments on camera placement.

Contents

1

Introduction

1.1 Mobile Robots for the Home Environment

Twenty years ago the idea of having robots moving around as humans do or doing tasks in normal human environments was a only a futuristic thought. Perhaps the time where an Asimov novel becomes reality is still some way off, but what is becoming real is people's acceptance of the integration of robots in daily life tasks. Nowadays, the need for practical mobile robotic applications is growing in importance, and the idea that future everyday environments could consist of smart objects such as robots moving around in houses performing otherwise time-consuming activities is acquiring a sense of reality.

In the last decade considerable progress has been made in the development of mobile robots to operate in indoor environments. There have been various well-known demonstration applications, such as tour guide robots [32] or hospital service robots [44] which help to transport medicine and medical instruments. There have also been several competitions aiming to increase activity in robotics research. *Robocup* [1] is one of the most successful, with the robot soccer competition being the most well-known component but also featuring the *Robocup@Home* contest focusing on the in-

troduction of autonomous robots to human society.

Autonomous robotics involves different areas of research which require special individual attention, but there are some capabilities which could be considered as general and fundamental whatever the particular task. These include localisation, navigation, obstacle avoidance and path planning. To perform any of these tasks, sensors must be used to perceive the environment surrounding the robot, sonar, laser range-finders and cameras have been the most prominent. Sonar [145] and laser [49] range finders have been subject to a vast amount of research and in general the fundamental tasks previously mentioned can be addressed with very good performance in indoor, single-level environments.

Computer vision on the other hand does not provide us with direct information about the depth of the surroundings. However, in principle this and much more can be estimated even with a single camera. There is growing agreement that vision should play a key role in autonomous robot systems and the past few years have proven that cameras are often a good choice as the primary outward-looking sensor. The amount of information that can be extracted from images is extremely rich, and a vision system gives us the possibility to aim at not only raw competences such as localisation but more general 3D spatial awareness and scene understanding. For instance, Simultaneous Localisation and Mapping (SLAM) can be achieved accurately with a single camera [27] and recently dense reconstruction of scenes has also been proved to be achievable in real-time with a single camera [97]. These are pieces of work at the robotics end of the vast research area of computer vision; but with the ever-increasing cheap processing capability we have available in robotics research we have every reason to believe that more and more of the advanced computer vision algorithms currently being developed for other applications (in image retrieval, object categorisation or computational photography for instance) will gradually become available for robotic use.

So cameras are potentially a very good choice as the main outward-looking sensor for a mobile robot, but an important factor is that a wide field of view is usually desirable to enable good performance in navigation tasks. This can potentially be provided by a single special omnidirectional device. Another possibility is to achieve all-around vision using multiple standard cameras mounted around the robot, with advantages including reconfigurability, flexibility and probably cost. With a camera rig we can combine the advantages of individual cameras such as different resolution, individual camera settings, different camera lenses could be attached, and we could also obtain omni-directional sensing. Another important aspect is that different roles could be assigned to the individual cameras, for instance, one camera can be fully devoted to object recognition while another one to obstacle avoidance and free space mapping.

Given this motivation we have chosen to focus on this option in in this thesis, developing novel methods for the use of multi-camera rigs for navigation.

We will concentrate our efforts on the core navigation capabilities a mobile robot needs irrespective of its intended use, and as we will explain more clearly later we have investigated two paradigms: a metric approach, where all information from multiple camera is integrated into a single map representation; and a very different 'lightweight' approach where individual cameras are assigned particular, modular tasks, and interact in a weakly-calibrated way to enable overall navigation behaviour.

1.2 Consumer Robotics: Current Status and Future Prospects

In contrast to pure software developments, the manipulation of the physical world by a robot, by definition makes it a more challenging problem. Before a consumer robotic application could be released to the market, a huge amount of research and development must be carried on. There are several technical and economical challenges as well. For instance, within the technical aspects a robot has to solve: autonomous localisation, path planning and area coverage (which by itself is a geometrical problem), power supply, human-robot interface and safety. In home and service robotics, there is the extremely important factor of safetiness in order to navigate around people or objects without creating a damage. Years have passed and some areas in robotics are getting more mature and gradually starting to move faster from a stage of pure research to a stage of deployment. People are starting to consider more seriously the potential practical applications. Several companies now have mobile robot products aimed to the consumer market; mainly in the floor cleaning domain. Others are beginning to look more deeply into the future of personal robotics and to design robots which might have the wide array of capabilities consumers would expect from a general purpose home robot. Probably the most representative example here is Willow Garage, a company with top robotics researchers that focus on developing integrated hardware and software for the future of personal robotics applications.

In this section we will review the products which currently exist, those which are likely to exist over perhaps the next five years and then look further forward at those which may be produced in the farther future. In each case, we will analyse the cognitive and sensing capabilities required, and in particular consider the role that computer vision should play.

1.2.1 Consumer Robots Already on the Market

The market for practical mobile robots is still small but is growing in importance. Since the year ~ 2000 we have perceived an increasing development of home-robotic applications, mostly in the floor cleaning area. In this section some of the most successful home-robotic products are analysed. However, for more a extensive discussion, the reader should refer to [109] .

In 2001 Electrolux lunched the first cleaning robot for mass production, the "Trilobite". This robot uses a ring of sonars covering 180 degrees to determine the perimeter of the room and to avoid obstacles. It also has the capability of recharging itself in a station base when the battery is about to deplete. In Figure 1.1 can be seen a comparison of some robotic vacuums.

In 2003, Karcher releases the Robocleaner RC3000. This device has a random operating mode which is helped by tactile sensors to gradually cover the area and to avoid obstacles. Its main advantage is the usage of a sensor that detects how polluted is the air being sucked by the vacuum, this makes the robot to spend more time on dirtier areas.

Probably the most successful practical application of home mobile robotics is the "Roomba" vacumm cleaner by iRobot Co. (see Figure 1.1), in sales it surpases by far its competitors. According to official statements it sold more than 2 million units by 2006. It is a small robot with a diameter of 34cm and 9cm high. Its main sensing components are a contact bumper at the front and several infrared sensors located in different places such as on the bottom to prevent it from falling down stairs or drops.

Its operation does not require a map of the area to be cleaned. It aims to cover an area using simple algorithms such as spiral patterns, random bounce and wall following. The new generation of Roombas can sense dirtier areas and spend more time there as well as calculate the time required to clean a room based on the amount of dirt detected and the longest straight line they can perform without bumping into an object. It has also the ability to recharge itself in a docking station.

In 2010, Neato Robotics started selling its vacuum cleaner that mechanically is meant to be better with a more powerful vacuum system, also its simplicity allows the user to press only one button to start operation and it has much more advanced features than others on the market. Its main capability is to use a SLAM approach using a specially designed low-cost laser range-finder to build a 2D map of its environment in real-time. This permits path planning to clean a room in an optimum manner. Also it charges itself and detects doors in order to clean a room before moving into another. This robot is very affordable and can be used as a research platform as well.

Also in 2010, Evolution Robotics released to the market its sweep and mop robot "Mint". It has similar features to other robots of its kind, such as easy of usage and

MODEL	COMPANY	SELF−CHARGING	SENSORS	CLEANING ALGORITHMS	PRICE (GBP)
Roomba	iRobot	Yes	cliff, bump, IR, dirt	Preprogrammed motions	100
Neato	Neato Robotics	Yes	cliff, 2D laser, bump	SLAM with path−planning	350
V−R4000	LG	Yes	cliff, bump, IR, gyro dirt	Preprogrammed motion (gyro gives more accuracy in robot's position)	800
Trilobite	Electrolux	Yes	cliff, bump, sonars	Preprogrammed motion (Perimeter determined by sonars)	900
RC−3000	Karcher	Yes	IR, bump, dirt	Preprogrammed motion	1200

Roomba Neato V−R4000 Trilobite RC−3000

Figure 1.1: Vacuum robots comparison

a small size. Its navigation system is based on the "NorthStar" technology. This is a photonic localisation system that projects a unique IR light spot onto the ceiling that is detected by a sensor in the robot and then triangulated to obtain an accurate position and heading. As it goes, the robot builds a map of the environment allowing it to have intelligent algorithms to cover the whole area in a minimum amount of time. One of the problems of this technology is the distance between the detector and the light plane which must not exceed 6m; so a network of projectors is needed to cover larger spaces. The way the robot works is by simply attaching clean pads to the front-lower part of the robot and by pressing start. In their first version the difference between mop and sweep modes was only a change in the robot's motion (mopping is by going forward and backwards). In the latest release "Mint plus" they add a liquid dispenser to fulfil partially the needs of a mopping task.

There are other commercialised robots with rather different practical applications, e.g:

- Mopping: "Scooba" by iRobot Co. mops the floor using similar operation modes as "roomba".

- Lawn mower: "Robomower" by Friendly robotics. It weights 22.5 Kg and it is equipped with bumper sensors to find obstacles, it also uses a wire that produces a signal as a virtual wall to delimit the area. It has some drawbacks as: the area has to be setup a priori, no obstacle avoidance is implemented and the obstacles detected by the bumper sensor have to be rigid and with certain dimensions. A more resent robot is "Robocut" by Brill, it was presented in 2006 claiming to have great cutting mechanisms and an optimised guidance system, it is designed

to operate in areas of $\sim 2500m^2$.

- Pool cleaner: "Polaris 145" is an automatic pool cleaner, which operates by wondering around at the bottom of the pool cleaning it until it covers the whole of the pool. Similar products are: "Aquabot" by Aqua Products and "Tiger Shark" by Aquavac.

- Ironing Robot: "Dressman" by Siemens. It is designed to iron shirts by fitting them on its inflatable torso. The ironing process consists of filling the robot's torso with hot air.

The autonomy level of the robots now on the market is improving, but generally still limited. There is a clear demand for products which can be trusted to perform complex tasks without human training, set-up or monitoring.

Having specialised sensors for every task could help in obtaining more data about the environment, but we would like to think also about the cost that the robot could be built for and we think that there will be a continued move towards commodity, standard sensing in robotics. In practical robotic applications we would like to increase performance while also decreasing price; the success of the Roomba vacuum cleaner compared to other products show the great difference in adoption that a cheap robot can achieve. So attention should primarily be focused on algorithms to make better real-time use of the information available from standard sensors.

None of the robots presented in the previous paragraphs has a camera as its main outward-looking navigation sensor; or do not have cameras at all. These is due to several reasons; but mainly because even though an image contains a vast amount of information, analysing it and extracting all the data needed is not as straightforward as obtaining depth measurements from a range sensor. Working with images requires much processing before obtaining the needed data and also is subject to illumination changes. Hence, for targeted applications such as a vacuum cleaner where only 2D information is needed to compute a map or to detect obstacles then using a range sensor might be much easier. However, by using a camera in a robot, we can aim at much more generality and a camera has almost limitless potential for more advanced capabilities such as recognising humans or characterising a scene with richer information that can be used in further tasks or applications. For instance, in a vacuum cleaner, a camera could potentially be used as the only sensor to perform SLAM, detect steps and doors, and depending on the colour and texture it might recognise the type of floor or carpet and use a different mode more suitable to it.

1.2.2 Mid-Term Potential

Over the next 5 years, we consider that the main developments in home robotics will be to consolidate current research capabilities such as:

- Robots with robust autonomous navigation capabilities, able to autonomously map complex real environments
- Robots with more advanced scene understanding: e.g. using wide angle vision to quickly estimate the type or main dimensions of a room.
- Elements of object recognition coming into play: door recognition, target object recognition, etc.
- Robot-computer interaction: the ability to interact with a person via gestures.

In this section some study cases will be analysed from the computer vision point of view to extract important aspects that will help as a starting point for a computer vision framework that could serve in future applications. We aim to conceive practical applications that could have a serious time-saving impact on people's lives. For the purpose of this research, we assume the most general case where the environment where the robot is moving in is not structured, and also that no beacons or landmarks are added.

- ***Autonomous vacuum cleaner***

  It can be seen from the last section that the vacuum cleaners now on the market are still limited in their capabilities, though with products like Trilobite and Neato this is changing. What would happen if we could make a smart vacuum cleaner that builds up a map automatically each time it works incrementing the efficiency only vacuuming the areas where it has not been yet? This could minimise battery use and time. Depending on the accuracy of the map, the robot could compare between a saved map and the actual map and give a report of the areas that were not cleaned due to different obstacles. It could also have the ability of interact with humans while it is working; a person could give it visual signals such as: stop, clean again or pause. It could find the best path to any room or the charging station and charge itself. These overall capabilities require a number of different modules to be developed and integrated, the main ones being the following:

  - *SLAM*

    Localisation and navigation could be said to be the base for an autonomous mobile robot. In vacuum cleaning, which requires precise and complete

coverage of an area rather than just 'A to B' planning, accurate and robust localisation is particularly critical. This module enables the robot to know which parts of a room have been cleaned and which not yet, and the relative locations of multiple rooms and its charging charging station.

Vacuum cleaning does require good accuracy and repeatability of localisation, though probably at the centimetre level rather than any more precise. A SLAM solution which is at least locally of good metric quality is therefore suitable. The robot will be making repeated and loopy movement in a restricted area, surrounded by furniture, so a wide clear of view for accurate mapping and relocalisation will be important.

- *Free space detection and obstacle avoidance*

  In a dynamic and complicated environment such as a house it is certainly necessary to have obstacle avoidance routines on a robot. A vacuum cleaner needs to identify all cleanable free space that needs to be cleaned in a comprehensive manner. It should also be able to approach obstacles very closely to clean along edges. A final possible capability, though possibly very difficult, would be actually to observe the floor itself to detect automatically how clean it was.

- *Human recognition and human interaction*

  Human interaction is possibly an important aspect in this application. The robot could be able to identify people in the environment and through visual signals it could take actions such as stop, clean again or pause. Another important aspect which could add more relevance to our work is having feedback from humans.

Different aspects have to be analysed before starting building the system. Problems such as illumination changes has to be taken in to account, illumination is a difficult problem when dealing with unstructured scenarios, the vacuum cleaner looking into a shadow could thing that it is dirty or when it is under the sun light from a window this could affect its perception of the surroundings, that is why having a reliable system invariable to illumination changes would be really important, if that can not be achieved with high accuracy then artificial intelligence algorithms could be applied in order to adapt itself to different times during the day.

Another aspect to think about, is the necessity or not about depth information, because the robot is only moving in 2D and different segmentation algorithms and occupancy grids could be used to detect objects and represent the map then I believe that for this task depth information could be not used.

In terms of the cameras to be used, an omnidirectional camera could be but the resolution is poor an problems could arise when recognising humans and human signs. Another option is the use of a ring of cameras covering must of the area, perhaps for this application one camera facing the ground and another one facing forward could make the work.

- ***Autonomous lawnmower***

  Mowing a lawn is clearly a similar task to vacuum cleaning, requiring localisation and regular coverage, though the outdoor environment changes some details of the challenge. The ground surface is less likely to be flat; safety is probably even more important; and the robot will have larger areas to cover under more varied lighting conditions. Hopefully gardens do not change fast so a map could be re-used for localisation and path planning. A camera facing forward could be used to detect moving obstacles, but since this machine requires a very safe performance other sensors might be needed in case abrupt illumination changes affect the vision system.

- ***Delivery robot***

  Another task that could be achieved with relatively little added to a core navigation competence is delivery, whether moving objects around a home or perhaps more usefully in a large building such as a hospital or factory. This task could have different levels of complication: it could operate minimally with a SLAM and path planning system. Objects could be given manually to the robot and an order such as "take it to the kitchen" will be given, then using its map, obstacle avoidance routines and path planning it could find its way to the destination. A more complicated task would be to make it completely autonomous and perform object recognition in such a way that a manipulator would be attached to the robot for object delivery and collection.

  - *SLAM*

    Having a 3D SLAM system would be ideal for this task. 3D information would allow the robot to have a better representation of the environment. If the robot had a manipulator then a 2D map of the floor such as in the previous study cases would be poor in terms of describing where objects could be located.

    It is also important to have a scalable approach to mapping because the map required in a large environment might have large processing and memory requirements. This could potentially consist of sub-maps automatically divided by characteristics places of the house such as rooms.

– *Object recognition*

This module would enable to recognise objects from their surroundings. A system robust to geometric and photometric changes would need to be built; and would need to be able not only to recognise individual objects but also to categorise objects into classes such as table, chair or spoon. Then general commands could be given such as 'bring me a small spoon' or 'bring me a glass'. Semantic context may also be used to identify objects in a particular environment.

Placing the objects in the map could give an advantage in future localisation of those objects. The object recognition module would be in communication with the SLAM module when an object is recognised.

Occlusions and general 3D transformations are two problems that have particular importance in the development of a good object recognition module.

– *Path planning*

This module should sit on top of the SLAM module and had the job of enabling to achieve autonomous navigation. It is important to be able to perform a trajectory with considerably high velocity in its motion because this could be a factor that makes the robot interesting to use; perhaps it could have a range of velocities based on the characteristics of the object to be delivered, e.g. a cup of coffee needs to be carried carefully. That is why trajectory estimation of the robot and moving objects is necessary to achieve high performance in this module.

So far we have described a brief analysis of some study cases in mid-term mobile robotics in order to deduce the common components of a computer vision system which could be useful in a wide range of low-cost consumer robotics over the coming few years. We can group these components into those that could serve as a base for achieving autonomy in a mobile robot and also higher level tasks can be developed on top these (Group 1); and those that are also important but can be developed depending on the targeted application (Group 2):

Within Group 1, the following components are identified:

- Fast and easy calibration and setup
- Simultaneous Localisation and Mapping (SLAM)
- Free space detection
- Obstacle avoidance
- Path planning

We can also identify some of the more specialised components forming group 2:

- Object recognition
- Human recognition
- Object and human tracking
- Human robot interaction

In these thesis we focus on developing and integrating algorithms within the first group using computer vision and specifically a rig of cameras. Hence, this work is oriented towards obtaining a general computer vision framework that can aid the autonomy of a mobile robot and that is not robot-specific.

1.2.3 Long-Term Prospects

If we look beyond the next five years, expectations for home and service robotics will advance significantly beyond tasks which require predominantly navigation, and we will be aiming at robots which have the capability to aid with a much wider range of human-like tasks. At this higher level of interaction with the environment a robot needs manipulation capabilities to perform complicated tasks. A robot with a manipulator can interact with and modify its environment in a general manner. These robots might require full, perhaps getting close to human level, scene understanding and manipulation capabilities such that they can undertake tasks such as general cleaning, table clearing, ironing and clothes handling.

We consider below some specific ideas for robots which might fit into this level of competence.

- *Cooking assistant*

  Cooking is a time-consuming activity which many people perform every day, more than once. The tasks for an assistive robot in this domain might eventually be perform all cooking tasks itself; but let us imagine a less advanced robot initially whose main job is to bring utensils and ingredients to a humans on command, and tidy things away once not needed, perhaps also involving some cleaning. This creates various technical challenges. For a mobile robot, it would certainly require SLAM, but also advanced object recognition, obstacle avoidance, voice recognition, planning and manipulation algorithms and a physical arm/gripper that can manipulate objects of different shapes. The task is simple: an order is given to the robot, e.g. bring me a spoon, the robot goes where the spoons are, if they are on a shelve then it opens it grab the spoon close the shelve and leaves the spoon besides the person.

Some assumptions can be made such as a-priori knowledge of where the objects are located in the kitchen environment might be. The robot would need to incorporate this prior information with accurate SLAM obstacle avoidance so as not to interfere with the person who is cooking.

Object recognition and manipulation are perhaps the most challenging tasks for this robot; e.g. how to recognise different cooking containers? If we think about the task of cleaning a table then how would it know when a dish is dirty or not? And how should they be put into the dishwasher? These are challenges currently on the boundary of what is possible in robot manipulation research in the laboratory.

- *Room clothes cleaner*

  In this application a robot would search a room for clothes thrown on the bed, floor or any furniture and put them in a basket. It should also classify clothes into different types. The technical challenges here are also large. The robot would need to have a specialised manipulator(s) that allows handling of clothes without damage. The sensing challenge is particularly tough in the case of clothes because they are highly deformable and difficult to recognise when folded or creased; there is the start of some research at the forefront of manipulation on handling simple cases of materials such as towels in a simple way but much more work on the interaction of computer vision and manipulation is needed here. The robot needs general SLAM and navigation capabilities to move around its workspace. Perhaps the position of the robot's cameras could be an issue because some pieces of clothes could be on the floor or in the top of a chair. A solution could be a rig of calibrated cameras facing in different directions.

- *Bed maker*

  This task I think is complicated in terms of hardware and software. The gripper for instance would need to have enough payload to carry and drag heavy blankets which sometimes weigh more than 2kg and have a long reach. It would need to make complicated moves to throw a blanket onto the bed. Another problem would be how to determine when the blanket is totally extended over the bed and it does not have a fold in it. It could be of great advantage to add markers or landmarks to the bed covers, whether temporary or permanent to help guidance.

- *Car mechanic assistant*

  The idea is to help a car mechanic in his/her daily work by in a manner somewhat similar to the cooking assistant. Clearly a robot with full object recognition

and manipulation capabilities could perform this task; but perhaps some simplifications could be made with assumptions. For instance, all the most-used tools could be systematically located in easy to find places for the robot to pick them up from and later return them to. Also possibly some extra infrastructure such as RFID tags could make the task easier.

1.2.4 Service Robotics Review: Summary

We have considered mobile consumer/service robots in three groups: robots already on the market; ideas with mid-term potential (perhaps 5 years); and longer term prospects. Those already on the markey and accepted by consumers are focused around floor cleaning and closely related applications. This domain is notably more easy to reason about than the general 3D world, and the robots required can be made small, safe and with some margin for error such that precise localisation and mapping is not necessarily required. Still, there is much that could be done to improve the reliability and efficiency of such robots.

There are several applications we can identify in the mid term (5 years) which are developments on these cleaning robots or concepts whose main requirements a essentially similar such as delivery robots. In this category, we should certainly expect robust and accurate autonomous navigation capabilities, and start to look towards additional functionality such as object recognition or human communication which could be used to augment capabilities. These are capabilities which are often today well proven in research labs but there is significant work to do to enable the robust and efficient performance needed for real-world robots.

The long-term of consumer robotics involves potentially a transfer of all of the current forward looking research in computer vision into robotics to permit more complex or full scene reconstruction and in general complete scene understanding needed for full interaction and manipulation. While long-term applications are more appealing, attractive and perhaps closer to the concept people have about mobile robotics, their realisation is going to take an extended period of work by various interactive research areas. There are still big difficulties in achieving even elementary manipulation of real, fragile or flexible objects in real environments.

Therefore we choose in this thesis to concentrate on the capabilities which will move consumer robotics on from current commercial systems towards mid-term ‘navigation++’ applications. There is clearly value and interest in these applications on their own merits. We also hope though that this would is a stepping stone towards the long-term applications discussed, which will certainly still need well-proven and robust autonomous navigation capabilities. We believe that our focus on computer vision is also forward-looking, because vision will surely only continue to increase in impor-

tance as the requirements from perception become more demanding. Our approach involves the use of only computer vision as the main sensor and in particular, the use multi-camera rigs.

Working with conventional cameras clearly adds a complication factor to robotics research due to the computation required to transform simple pixels into meaningful information. It is true that for example depth information could straightforwardly be obtained using a laser scanner or RGB-D camera such as Kinect (as we will discuss later), while this is a complicated task using vision. However, a camera is a more general sensor capable of giving a huge number of cues about the environment. An image can be analysed and interpreted in different ways to obtain a more complex and robust representation of the area where the robot is moving. Another factor in favour using only vision is that it is a cheap sensor, certainly compared to a laser scanner. While this argument changes somewhat with the recent arrival of Kinect at commodity prices, cameras are can still be made much smaller and with lower power reqirements than active sensors. Also, they have the easy potential for scaling of performance by changing resolution, frame-rate or dynamic range. Finally, there is the intuitive appeal of working with the same passive vision sensing capability of biological navigation systems where there are still often useful inspirations or parallels to be drawn.

1.3 Multi-Camera Rigs for Robot Navigation

A wide field of view is usually desirable for responsive and accurate navigation, SLAM and relocalisation. While this can potentially be provided by a single special omnidirectional camera (for example with a catadioptric optical system), it can also be flexibly achieved by multiple cameras with standard optics mounted around the robot.

Now that cameras and their supporting electronics are getting cheaper and smaller, mounting multiple cameras around a robot in a ring or other configuration is highly feasible, clearly offering the potential for both a wide field of view and high angular resolution. A multi-camera rig will in general not be fully equivalent to a single wide-angle camera in terms of imaging geometry since it will have multiple optical centres, but this is not a problem for applications such as navigation as long as the locations of those optical centres is known, and may even be an advantage in inferring depth information. Indeed, a multi-camera rig is fully reconfigurable and can be set up to provide the field of view needed, and if necessary regions where camera views overlap for stereo observation.

As previously mentioned, there are several advantages in using a multi-camera rig. However, there also exist disadvantages inherent to manipulating several cameras such as synchronisation or having different camera settings (exposure, white balance,

gain, etc.). Finally there is the larger problem of geometric calibration of the whole rig — where are the cameras with respect to one another? Nowadays, most of the cameras on the market are automatically synchronised if attached to the same bus and the individual camera setting can be easily adjusted manually or automatically depending on the application. From our perspective we consider the main disadvantage being the geometric calibration of the rig, and in that sense we have developed an automatic calibration technique which will be described in Chapter 4.

There have been various robots already which have used multi-camera rigs in different applications of multicamera rigs. For instance search and rescue robots utilise different camera rig configurations to be able to focus on different areas and to cover more space when performing tasks (see Figure 1.2).

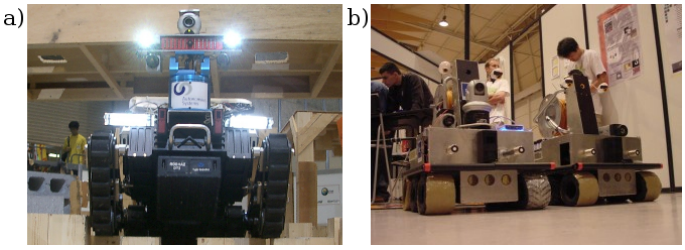


Figure 1.2: Search and Rescue robots with multiple cameras: a) Rescue robot CASTER from the Australian Research Council Centre of Excellence for Autonomous Systems. b) Jacobs University cooperating rescue robots.

In the literature we can find different efforts into building practical robots that can extract as much information as possible from the sensors on-board. From a visual perspective this can be achieved by the incorporation of multiple cameras in order to obtain either a wider field of view or an special role assigned to each camera, (see Figure 1.3).

Probably one of the most complete robots including hardware and software for research and development is the PR2 from Willow Garage (see Figure 1.4). This open platform allows the researcher start building specific robotic applications right out of the box and its design makes it suitable for tasks involving grasping, manipulation and navigation. Within its visual sensors we can find the following:

- Head:
  - Microsoft Kinect
  - 5-Megapixel Global Shutter Colour Gigabit Ethernet Camera.
  - Environment Stereo Camera: Wide-Angle Global Shutter Colour Stereo Ethernet Camera.

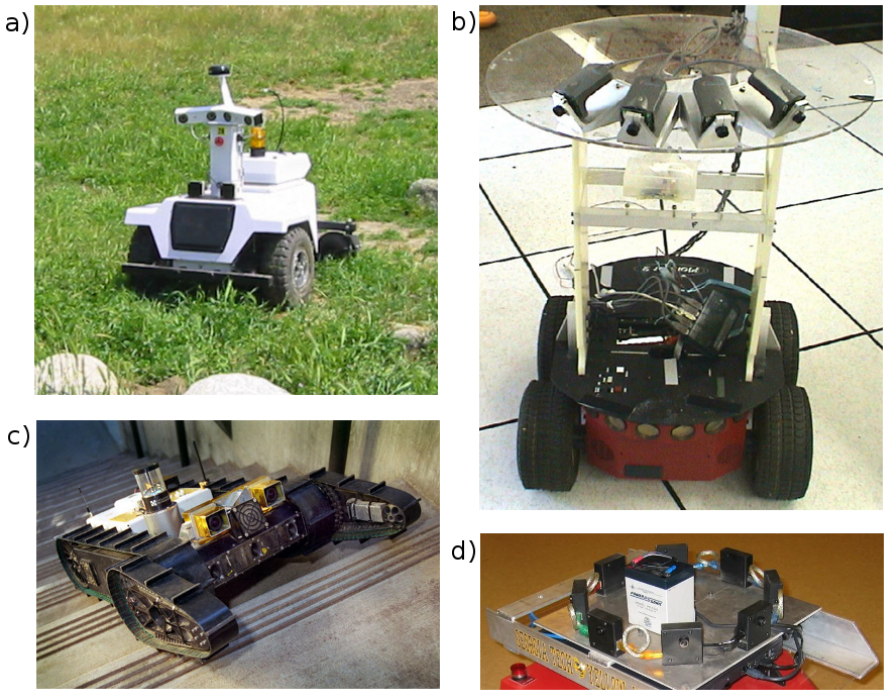


Figure 1.3: Different research robots with multiple cameras: Jet Propulsion Laboratory Robots: a) robot that includes two stereo camera pairs, b) robot supplemented with a vision 'turret' containing four USB cameras providing a 180° field of view, c) Robot equipped with an omni-directional camera and a binocular stereo camera pair, d) Georgia Tech robot with a custom-made 8-camera rig, mounted on top of an ATRV-Mini mobile robot platform. The FireWire cameras are distributed equally along a circle providing 360° field of view.

  - Manipulation Stereo Camera: Narrow-Angle Global Shutter Monochrome Stereo Ethernet Camera.
  - LED Texture Projector Triggered with Narrow-Angle Stereo Camera.

- Forearm:
  - Global Shutter Ethernet Camera.

Figure 1.4: PR2 robot from Willow Garage. The vision hardware consists of a Kinect device, two ethernet stereo camera pairs and a high resolution ethernet camera mounted on the head. On the forearm a colour ethernet camera is also mounted.

We can see that different robots have been subject to various choices with regard to the number, types and positions of the cameras mounted. A very interesting area of research involving multicamera scenarios is to investigate the best possible configurations or the best camera placements [59]. We take a serious look at this issue in Chapter 7 of this thesis.

1.4 Lightweight Computer Vision Techniques

We can think of two main paradigms for localisation and mapping in robotics: the first one consists of building a metric map which is constantly updated and is consistent with every measurement and position of the robot. This normally involves the use of a feature-based approach and a filtering technique and it has the advantage of producing accurate maps. These maps most of the time result in a sparse representation of the scene which sometimes cannot be used for the purpose of autonomous navigation. To deal with this issue a more complex 3D or 2D metric reconstruction is needed and in the case of a system with multiple cameras it requires the very accurate knowledge of

the position calibration of the cameras relative to each other. In Chapter 4 we describe a camera rig autocalibration approach which could allow the user to combine all camera views into a single panorama view and use this to obtain metrically precise maps. However, this comes with the cost of precise camera setups and more computational time involved.

The second paradigm evolves with the idea that in robot navigation there are several assumptions that can be used that facilitate the robot's task. The assumptions are normally geometrical properties such as the robot moving on a plane, or assuming some property of the scene and the trajectory. The resulting maps are fairly accurate and normally this type of approaches results in less computational time and fewer constraints on the configuration set up. Our main motivation is to advance this line of research to determine whether fully autonomous robot navigation could be achieved based on lightweight vision techniques (see Chapters 5, 6 and 7).

This section is dedicated to introducing the reader to the concept of algorithms that are easy to implement, robust, simple in concept and cheap to engineer in consumer robotic applications. These algorithms are catalogued as lightweight techniques. We would like to define our concept of 'lightweight' as a system component that incorporates some of the following characteristics:

- An easy to understand method which is easy to implement and modify.
- Little requirement for calibration.
- A 'black-box' method, which can be used without knowledge of internals.
- Computational efficiency.
- Modularity: can easily be combined with other techniques.

At the moment, the robotics research community is experiencing a better access to testing platforms mainly due to the standardisation of reliable research mobile robots which were not available 10 years ago. Nowadays, testing platforms such as "Create" from iRobot or any "Pioneer P3-DX" from MobileRobots make it possible to use them out of the box and to attach different sensors to them and start developing and testing algorithms without any big complication. Another important advantage in terms of software is the increasing amount of opensource libraries for development. For instance, OpenCV is a great library in computer vision which is constantly growing in quality and in the number of algorithms supported.

This make us think that there are a number of tools already available and accessible to scientists and students. However, sometimes state of the art approaches are difficult to implement and sometimes are not flexible; for instance, they only work in a particular platform or with specific hardware. Concentrating on lightweight techniques

would allow us to create robust algorithms which are flexible and does not require to much time to build and test.

Research on the usage of other visual clues such as colour has been less studied mostly due to the robustness provided by local features. However, different problems arise from the use of feature methods such as data association, adding more complexity to the problem.

Many of the most recent vision-based SLAM systems aim at maximum localisation accuracy through making a precise map of as many 3D point features as possible within computational limits. The lightweight approaches we study in this work have at their core a coarse metric-topological environment model, but they can still enable accurate and autonomous navigation.

The single most important function provided by vision within such a system boils down to image retrieval, where the most recent image is compared against all previous locations in the environment model. The use of colour and texture information encoded in global histograms for image representation has proved successful for topological robot localisation. For instance, Ulrich [141] developed an appearance-based place recognition using colour histograms to represent images in a topological map. This system used an omnidirectional camera that provides rotation invariance and also it needs an offline training stage where images are taken an labelled manually. Another work for localisation that uses histograms to globally represent an image is presented in [150], where different information such as colour, texture, gradient and edge density is obtained and represented in a multidimensional histogram.

An interesting lightweight system for Visual Odometry is presented by Milford [93]. The algorithm is part of a complete SLAM system called RatSLAM (see Section 5.1). This system obtains impressive results and the level of complexity of its visual system is very low.

In this thesis we aim to develop a lightweight framework that permits students and scientist to focus on developing consumer robotic applications using computer vision. It is true that there are different types of sensors and depending on the quality, a laser could cost the same as a camera. However, a laser does not provide flexibility in such a way that a camera does.

There is an increasing number of applications that require the 3D acquisition of a scene and it is known that this task is complicated and time consuming using sensors such as cameras or lasers. Depth cameras try to facilitate the 3D analysis of a scene by obtaining depth information per pixel, the range goes from 5m (Swiss Ranger4000) up to 60m in some cases (PMD - Photonic Mixer Device). This technology has two main ways of approaching: the first one is called time-of-flight, which measures the time delay between transmissions of a light pulse; the second one is based on projecting

a known infrared pattern and determining the amount of deformation. In previous years depth cameras had been used in the robotics community successfully but not extensively, this is mainly because of the cost of those devices. This issue has been addressed recently and as an example is the Kinect device by Microsoft. Kinect has a big number of qualities that makes it very suitable for robotics. Most importantly, it is cheap ($\sim$ £150), and easily accessible to a wide range or researchers. Its capturing speed is 30fps at a resolution of 640×480. It is true that these low-cost sensors can provide very accurate 3D reconstructions [98]. However, the resolution and field of view of the camera is low, the lens is fixed, and in some cases the camera range of depth does not cover big areas or contains gaps. Kinect for instance, only perceive objects which are not to close to the camera ($\sim 1m$). In this manner, cameras provide more flexibility, lenses can be used according to the needs, obtaining different features such as field of views or resolution, also the size of a normal camera is still smaller than depth cameras.

The future of consumer robotics is still uncertain and more research is needed before the world sees more robots on the market. We cannot be tell once and for all whether a lightweight or a complex system is better; this mostly depends on the application to be developed and the level of accuracy needed. Probably, in a home environment there is no need for millimetric accuracy in the SLAM system of a vacuum robot, but in an industrial application where precision is really important then a more complex system might be required. However, a lightweight framework would allow to develop and test practical applications faster.

1.5 Research Robot Platforms

During the work described in this thesis, two robots were used to test different algorithms (see Figure 1.5):

- *Segway Robotic Mobility Platform (RMP) 100*. This is a robot which has only two supporting wheels and stabilises itself dynamically making use of its internal sensors.

- *Pioneer 3-DX pioneer from mobile robotics*. This is one of the must general platforms used in several research labs due to its versatility, it can reach speeds of 1.6 meters per second and carry a payload of up to 23 kg. It uses 500 tick encoders and can run 18-24 hours on three fully charged batteries.

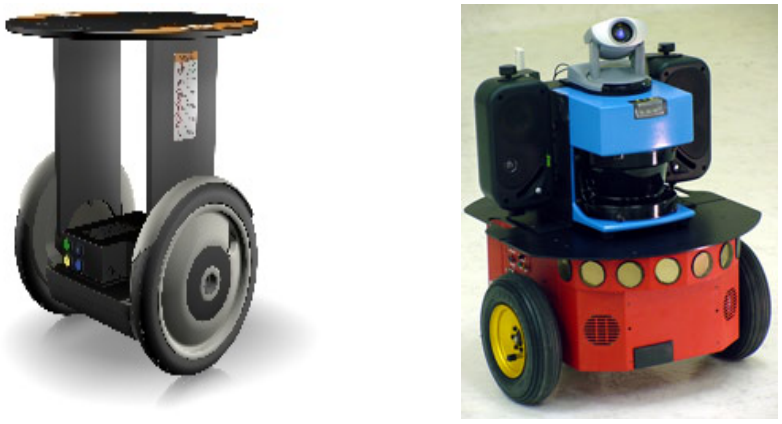
At the beginning of this research, the platform Segway RMP 100 was used and the algorithm presented in Chapter 5 was tested on it, this robot was already in the laboratory when I started my research and it was bought mainly because of the novelty on

its design and the self-balancing system. However, after finishing the experiments, we realised that this platform is not very suitable for small indoor spaces and because its dynamic stabilising system could potentially become dangerous if not operated carefully. For instance, if something is blocking the wheels when performing a motion operation the robot tries to keep balance by accelerating or decelerating, this resulted in a unsafe operation in small areas or in places where people are constantly moving. Those reasons were enough to try on another platform that was also reliable, but primarily safer to operate in indoor environments.

The platform we opted to used was the Pioneer 3D-X, This robot is very suitable in terms of reliability, durability and ease of operation, and it can be used outdoors and indoors. It is a more general platform which can be used on different surfaces, and its battery capacity is high, it has a good payload and also a wide area where devices can be mounted. For the remaining work this robot was the platform where the algorithms were tested. In Chapter 3, a more complete description of these platform is given.

1.6 This Thesis

The work in this thesis is motivated by the increasing demand for mobile robot applications in daily life. We focus on functional and practical robot applications. We aim to build novel frameworks that can use multiple cameras to enable different applications on a mobile robot. Our goal is to develop competences that are robust and lightweight, that could be potentially incorporated into a variety of different products, therefore enabling mid-term applications to be developed on top of our framework.



Segway RMP 100 Pioneer 3DX

Figure 1.5: The two robotic platforms used for experimentation: Segway RMP 100 and Pioneer 3D-X

As we metioned previously in 2.4, our 'lightweight' concept of a system is one which comprises modules each of which has some of the following characteristics:

- An easy to understand method which is easy to implement and modify.
- Little requirement for calibration.
- A 'black-box' method, which can be used without knowledge of internals.
- Computational efficiency.
- Modularity: can easily be combined with other techniques.

Many techniques in computer vision require significant expertise to use and put together into systems. Many others are brittle in their behaviour, or require precise calibration to use. We do not argue against the development of those algorithms here, and complicated methods are usually inevitable to explore new concepts and to push the boundaries of performance. But here we would like to show that a simpler, modular, lightweight approach comprising components each of which on its own can be easily understood by a relative non-expert can be surprisingly effective. We believe that this is of importance in particular in the expanding low-cost consumer robotics field where robots for the home or other mass-market domains are becoming a reality.

An important factor in the use of lightweight vision techniques is that a certain technique will often only be useful in a well-defined set of circumstances specified by the application domain. This brings up the incorporation of modularity into our lightweight concept.

We also focus on the ways that a multi-camera rig can be used both to enable SLAM and to go beyond this in leading towards fully autonomous navigation. We first take steps along the fully metric route of fully coordinated cameras with the development of a multi-camera extrinsic calibration method. Then, realising some of the weaknesses of this approach and noting the advantages in system building that a simpler, modular approach can offer, in the rest of the thesis we concentrate on methods built up from lightweight techniques and demonstrate a full autonomous navigation solution.

The thesis is organised as follows. This work contains algorithms from different areas in robotics such as SLAM, path planning, exploration and camera calibration. In Chapter 2, a review of the theory needed to understand the subsequent chapters is given as well as a look at the state of the art in each of the fields that we have tackled. Chapter 3 contains a description of the robot platform used, and the software development tools.

When working with multiple cameras in a metric setting, the first problem that has to be addressed is calibration, and in Chapter 4 we present an extrinsic autocalibration approach for the problem of non-overlapping views and without the need for calibration targets.

One of the main ideas we had in mind since the beginning of this PhD was to try to develop lightweight techniques that in essence deliver good results, the computational complexity is low and they are easy to implement. Chapter 5 introduces this methodology and presents a lightweight Visual Odometry approach inspired by the work of Milford and Wyeth [93]. In Chapter 6 we move onto the development of our full multi-camera navigation system based on lightweight techniques, capable of SLAM as well as free-space detection, path planning and basic autonomous exploration.

Chapter 7 rounds off this work with experiments and discussion of the issues of multi-camera rig design in the context of autonomous navigation using lightweight techniques, considering the trade-offs between the different places cameras can be mounted, and particularly considering cases where the same camera can perform multiple roles. Chapter 8 concludes the thesis.

The following articles were published based on the work contained in this thesis:

- G. Carrera, A. Angeli and A. Davison. Autonomous SLAM and Navigation with an ad-hoc Multi-Camera Rig. Proceedings of the European Conference on Mobile Robotics (ECMR), 2011 [15]. This paper describes the work presented in Chapter 6. This paper has been selected by the conference organisers as one of the best presented at ECMR 2011 and we have been invited to submit an extended version for a special issue of the Robotics and Autonomous Systems journal. We plan to include elements of the analysis in Chapter 7 in this expanded version.

- G. Carrera, A. Angeli and A. Davison. SLAM-Based Extrinsic Calibration of a Multi-Camera Rig. Proceedings of the International Conference on Robotics and Automation (ICRA), 2011 [16]. This paper describes the work presented in Chapter 4.

2

Background

The scope of this thesis is to employ computer vision techniques to achieve navigation and SLAM for a mobile robot equipped with a multi-camera rig. In particular, we present contributions in two areas which offer different perspectives on the ways to tackle the challenges of mobile robotics:

- Calibration of a general multi-camera rig (see Chapter 4), making it suitable for metric localisation and mapping, and achieved using feature-based SLAM and matching techniques.
- An integrated approach for SLAM, free space mapping and autonomous exploration using multiple cameras which is based on a collection of lightweight techniques and a weakly calibrated camera setup (see Chapters 5, 6 and 7).

In this chapter, a review of the main techniques and theory behind the algorithms developed on this thesis is given, as well as a discussion of the state of the art in the corresponding areas.

Autonomous robot navigation is a challenging task, and while there has been much progress in recent years we are still waiting for the presentation of algorithms and systems which will make it an everyday reality. Instead of tackling the full task, mobile

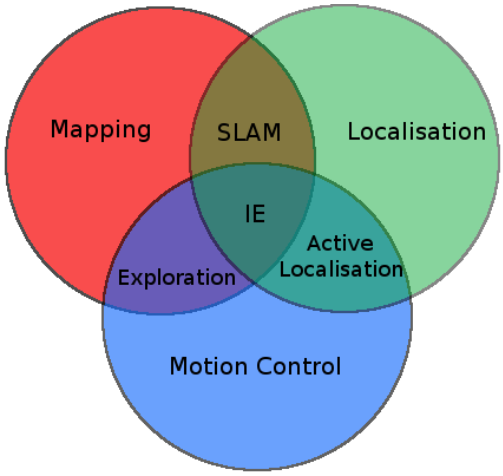


Figure 2.1: Diagram of the three major areas in mobile robotics and their interconnections [85]

robotics research often aims at solving limited sub-problems, and a useful place to start our review is to consider how to break this challenge down.

Figure 2.1 (originated by [85]) represents a standard view of the main tasks involved in mobile robot navigation. Three major areas serve as a base: mapping, localisation and path planning or motion control. It is possible to study each of these topics in isolation, and there are large amounts of work on each. However, they all interact and inter-depend closely and much research will be on topics which combine two or more of these main areas. For instance, localisation aims to position a robot in a map and in some situations localisation outputs are used to plan a path and to control the motion of the robot.

The tight relations between these areas open up different problems which can be represented as overlaps of the three major areas and which on their own represent complex research areas. Exploration algorithms often assume knowledge of the position of the robot and focus more on building maps while guiding a robot with motion commands. In order to improve the position estimate of a robot, active localisation approaches looks to move the robot towards previously mapped or other advantageous areas. The centre of the diagram represents approaches which try to address these three main areas together.

Before our review of engineering approaches to navigation, it is interesting to consider the biological point of view, since we know that many type of animal are able to robustly navigate around their environments. It is not the subject of this thesis to develop anything which resembles a biological navigation system. However, we could gain some insight about the different mechanisms and sensors used by insects of mammals and analyse if they also have a clear separation in modules or everything is more

mixed and connected in complicated manners.

Bees are subject to research due to their long distance navigation skills [115]. They mainly use 2 different sensors for navigation: compound eyes (detecting light and colour) and the head's antennae (for odour perception and speed measurement). Models of navigation and mapping involve different theories such as dead reckoning and landmark based navigation or cognitive maps. The difference in the theories is mainly because of the difficulty in reproducing the same experiments. However, there is some agreement on the spatial communication dancing-type method bees use between them [43].

Ants are also an important subject of study due to its navigation capabilities. Theory suggests that an ant perform a path integration method when they leave the nest [96]. This navigation system computes a home vector from the outbound trip based on averaging all directions in which the ant moves, weighted by the distance moved in that direction. On each outbound trip it corrects the home vector and if there is an error then the ant searches for the nest around the calculated point. It is also known that to adjust their navigation behaviour or in the case of losing track totally of their position, an ant uses visual landmarks [23].

Rodents create a cognitive map to have a spatial reference of an environment which is then used for navigation. There have been identified certain cells in a rodent's brain that respond to the spatial location (place cells) and orientation (head direction cells) of the animal. These cells respond to visual cues and the information can be retained for some time even in the absence of those cues. The first similarity with robots is that rodents have to update their perceived position, otherwise they can accumulate large errors if they rely only on self-motion information. There is relatively little information about the purpose of the navigation system or if it is goal or task oriented, so the research behind this tends to be speculative [89].

Clearly, humans main sensor for navigation is vision. The brain analyses accurately the information and with its level of sophistication and complexity, a fantastically effective navigation system is obtained. The navigation and mapping processes are relatively less studied than other mammals and there is little accurate information on how it actually works, mainly for ethical reasons due to the invasive nature that some experiments might require. Most experiments are related to path completion tasks [144]. Therefore, it is not know accurately how humans perform navigation and mapping. However, we can conclude that there are some similarities from the subjects studied and the navigation modules that are normally found in robotic systems.

With this overview in mind, our review will be organised as follows. First, we will provide a view of map representations used in mobile robotics in Section 2.1, and then review the literature on various approaches to SLAM (Section 2.2) and navigation

(Section 2.3). We will look at systems which have integrated approaches and achieve some form of fully autonomous operation (Section 2.3.4). Finally we will review approaches which focus on the lightweight techniques which are the purpose of the final part of our work (Section 2.4).

2.1 Map Representations

SLAM and localisation methods utilise different map representations to gather information about the environment. These maps representations have different characteristics and sometimes their use depends on the final application. However, they all share in common the idea of providing a reference for the information obtained from the robot sensors.

In most mobile robotic applications, the necessity of having or acquiring a map representation of the environment is fundamental for the task. In some cases, a priori maps exist in the form of a CAD (Computer-Aided Design) model; this in some cases is provided by the architects of a building for example. However, using hand made maps has the disadvantage of assuming a very rigid and static world and therefore is not very useful nowadays in real world applications.

There are many robotic tasks which require a map to operate successfully. These tasks normally involve traversing environments and sometimes require to operate in highly dynamic environments, e.g autonomous driving, delivery robots, assistive robots.

When a robot is able to learn the environment from scratch by putting it into any form of map representation, this increases the robot's autonomy and enables the robot to adapt to changes in the environment. This comes with the cost of dealing with challenging problems such as:

- High dimensionality in the space of all possible maps and observations.
- Uncertainty in the robot's position and observations.
- Map size and resolution.
- Perceptual aliasing (different places with very similar appearance)

This Section considers three types of maps that are most commonly used in current localisation systems: feature based, topological and occupancy grids. Sometimes hybrid maps involving some combination of these three types are also used.

2.1.1 Maps of Features

These maps are probably one of the most used in SLAM techniques. The map is a sparse representation of the scene formed by landmarks in the environment that are normally the most salient or characteristic features (see Figure 2.2), e.g points [27], lines [46] or planes [47]. Therefore, the map can be summarised as a structure that contains the 3D position and a unique description for every feature. Every time the robot obtains a measurement, it compares the features in the map with the features acquired at time t performing an action called data association that matches features in the map with its corresponding measurement. This step is critical to the whole system's performance because incorrect data association can lead to catastrophic consequences in the map.

Features in the map serve to localise a robot inside the map through measurements of the difference between the estimated feature positions and the corresponding features in the map. To recursively estimate the features' and robot's positions, filtering techniques are used. The most common method for localisation that allows one to obtain a consistent map along with representing uncertainty in the map and the robot's position is the the EKF (Extended Kalman Filter — see 2.2.1).

Feature point extraction is a major area in computer vision. In the literature, there are a number of feature point representations. The most used and representative of several SLAM approaches are described next.

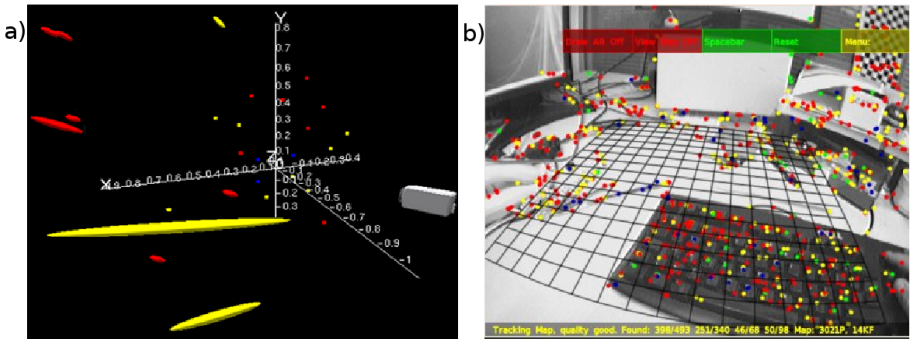


Figure 2.2: Map of features representation. a) Map built by MonoSLAM (see Section 2.2.1). b) PTAM snapshot.

- *Harris corner detector*. This feature point detector was developed by Harris and Stephens [51]. The detector is well known for detecting corners as locations in the image where the signal changes two-dimensionally. This is achieved by using the autocorrelation function defined by:

$$c(x,y) = [\Delta x, \Delta y] M [\Delta x, \Delta y]^T \ , \tag{2.1}$$

where Δx and Δy are shifts of small windows centred on (x,y). The matrix M denotes the intensity structure of the local neighbourhood. This 2×2 matrix is computed from image derivatives:

$$M = \begin{pmatrix} \frac{\partial^2 I}{\partial x^2}(i,j) & \frac{\partial^2 I}{\partial xy}(i,j) \\ \frac{\partial^2 I}{\partial xy}(i,j) & \frac{\partial^2 I}{\partial y^2}(i,j) \end{pmatrix} , \tag{2.2}$$

where (i,j) are the indices of the values in the window W over the image I. The location of the feature point is obtained by doing maximum suppression over a 3×3 region using the next function:

$$cornerness = det(M) \alpha trace(M)^2 . \tag{2.3}$$

- *Scale Invariant Feature Transform — SIFT*. Algorithm developed by Lowe [82] that has been very popular over the years given its repeatability and invariance to scale and rotation. To detect points, SIFT obtains 3D local maxima in the DoG (Difference of Gaussians) space, which can be obtained by the subtraction of successive scales of the original image. To improve the run-time of the method a pyramidal representation is built to represent the scale-space. To increase accuracy and stability, sub-pixel accuracy is performed by fitting a quadratic function on the detected points $x = (x,y,\sigma)$. By applying the Taylor series expansion on the image around the keypoint (see Equation 2.4) and then differentiating and equating to zero subpixel locations are obtained.

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^\mathbf{T}\frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x} . \tag{2.4}$$

To reject points that lie along edges, the trace and determinant of the Hessian matrix are used in such a manner that a point that is less than a certain threshold r is rejected.

$$\frac{Tr(\mathscr{H}^2)}{Det(\mathscr{H})} < \frac{(r+1)^2}{r} . \tag{2.5}$$

The next step is to assign an orientation to each keypoint. This is done by calculating the gradients inside a window around the keypoint and building a histogram of orientations. Then the histogram is used to detect the most prominent orientation.

To obtain the descriptor, a square window is divided into 4×4 subregions and rotated according to the dominant orientation of the keypoint. Over each subregion a histogram of 8 orientations is calculated. A vector of 128 dimensions is

obtained to describe the keypoint.

- *Speeded Up Robust Feature — SURF*. Algorithm developed in 2006 by Bay, Tuytelaars and Van Gool [9]. One of the characteristics of this method that speed ups the computation time is the use of integral images [143]. A point $p = (x, y)$ in an integral image $I_{integral}(x, y)$ represents the sum of all the pixels inside a rectangular region formed by the point p and the origin of the original image $I(x^{'}, y^{'})$.

$$I_{integral}(x, y) = \sum_{x'=0}^{x} \sum_{y'=0}^{y} I(x', y') \ . \tag{2.6}$$

Point detection is based on the use of the Hessian Matrix.

$$\mathscr{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial {x_1}^2} & \frac{\partial^2 f}{\partial x_1 x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 x_n} \\ \frac{\partial^2 f}{\partial x_2 x_1} & \frac{\partial^2 f}{\partial {x_2}^2} & \cdots & \frac{\partial^2 f}{\partial x_2 x_n} \\ \frac{\partial^2 f}{\partial x_n x_1} & \frac{\partial^2 f}{\partial x_n x_2} & \cdots & \frac{\partial^2 f}{\partial {x_n}^2} \end{bmatrix} \ . \tag{2.7}$$

Specifically, by calculating the determinant of the Hessian Matrix:

$$det(\mathscr{H}_{aprox}) = D_{xx} D_{yy} - (0.9 D_{xy})^2 \ . \tag{2.8}$$

Second order derivatives are approximated by using box filters and the scale space is obtained by applying filters of different sizes, given the nature of an integral image, the computational cost is independent of the filter's size.

After computing the integral image, rotational invariance is achieved by calculating the Haar-wavelet responses in x and y directions with side length equal to 4s over a circular neighbourhood of radius 6s around the interest point, where s is the scale at which the interest point was detected. The wavelet responses are also weighted with a Gaussian ($\sigma = 2.5s$) centred at the interest point. Because of the use of the integral image, only six operations are needed to compute the response in the x or y direction at any scale.

The responses are represented as vectors. To get the dominant orientation, first it is needed to get the orientation in a sliding window covering an angle of $\pi/3$, by summing all the vectors that are within the window. The longest vector leads the dominant orientation of the feature point. To compute the descriptor, a square region of size 20s is defined centred around the feature point and oriented along the dominant orientation. This region is split into 4×4 square subregions. In each sub-region, regularly spaced sample points are taken and over these points the Haar wavelet responses in the x and y directions are calculated.

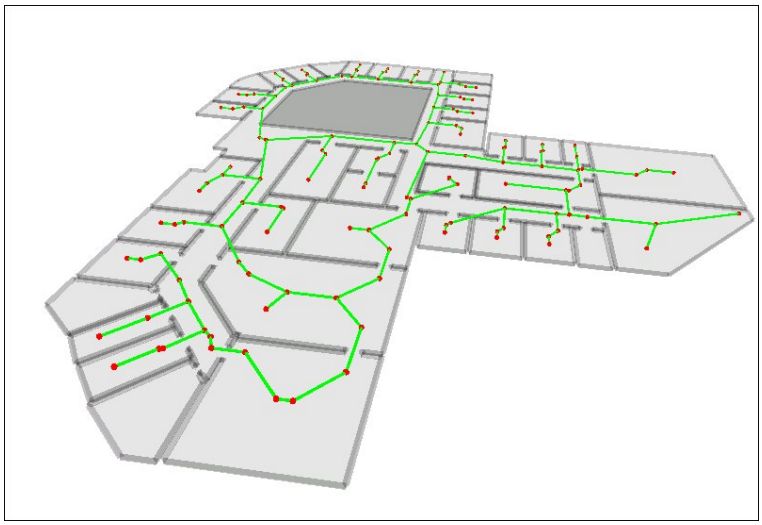


Figure 2.3: Topological map representation. Image courtesy of University of Bremen, Spatial Cognition R3-[Q-Shape] [103].

The size of the filter is 2s. These responses dx and dy are also wighted with a Gaussian ($\sigma = 3.3s$) centred at the feature point, this is done to achieve robustness towards geometric deformations and localisation errors. The responses over each sub-region are summed to obtain a vector over each region. These vectors and the sum of the absolute values of the responses over each sub-region give us the total entry of the descriptor. So each sub-region will contribute to the descriptor with 4 values. The structure of the descriptor is then $\mathbf{D} = (\sum d_{x_i}, \sum d_{y_i}, \sum |d_{x_i}|, \sum |d_{y_i}|, ...)$ where i = 1, ..., 16. The descriptor is turned into a unit vector to achieve invariance to contrast.

2.1.2 Topological Maps

A topological map can be represented as a graph of poses in which nodes store information about specific places in the environment and edges are usually a transformation that links one node to another (see Figure 2.3). This information is normally a transformation constraint. Usually, every node stores sensor measurements (e.g. images or laser readings) representing the environment in a specific location along with the position of that node in the map. Edges contain information that allows the robot to compute the cost of going from one node the another one. In this sense, algorithms for traversing graphs can be used in a topological map. For instance, algorithms such as A* or Dijkstra can be used to obtain the shortest path between two nodes.

One important issue involves the selection of nodes. A straightforward strategy would be to select nodes based on the distance, e.g. create a node every 1m. However,

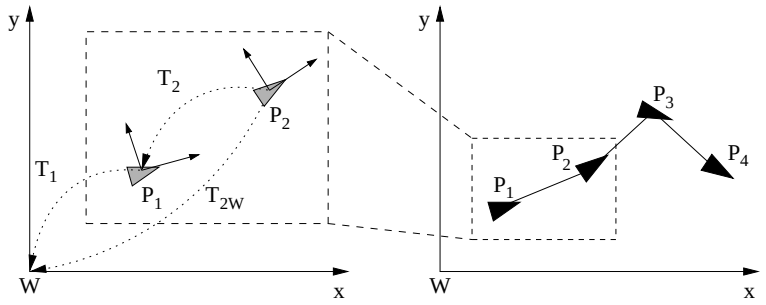


Figure 2.4: Robot poses represented in a topological map.

this could potentially create a graph with redundant information and with too many nodes. Another strategy would be to choose nodes relative to the difference in their appearance in such a way as to minimise the number of nodes and maximise the information contained in every node. To correct the drift in the map, place recognition plays a very important role. Therefore, we would like to have unique nodes to facilitate data association but at the same time we would also like to take into account long areas that appear the same, for instance corridors with repeatable colours and few textures. Then only a single node representing a big area would not be the best map representation.

When working with topological maps or graph of poses one should take care about maintaining a geometrical convention through the whole chain of transformations. In this sense, a transformation from one point to another is usually expressed in terms of a local coordinate frame relative to the previous pose. A 2D rigid transformation can be represented with a three degree of freedom parametrisation involving a translation (t_x, t_y) and a rotation θ. In homogeneous coordinates, a point $p = (x, y)$ can be transformed with the following equation:

$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{bmatrix} cos\theta & -sin\theta & t_x \\ sin\theta & cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} . \tag{2.9}$$

As can be seen in Figure 2.4 each robot pose has its own local coordinate frame. Hence, any position can be related to the global coordinate frame W by composing the corresponding chain of transformations. For instance, the global coordinates of pose P_3 can be obtained by applying the transformations $P_{3W} = T_1 T_2 P_3$.

2.1.3 Occupancy Grids

An occupancy grid partitions a region into evenly spaced cells. If the robot is moving on a flat surface, the map is commonly a 2D plane. As mentioned previously, each cell represents an area of the environment and the size of every cell depends on the

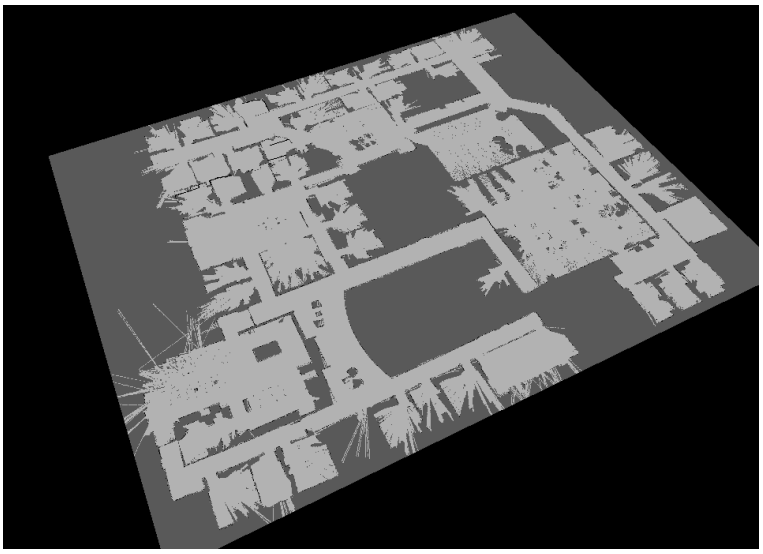


Figure 2.5: Occupancy grid representation; black indicates occupied, white represents free space and grey unexplored. Image courtesy of ROS [136].

resolution of the map and relates to the level of precision the user wants to obtain (see Figure 2.5).

These maps have the advantage of representing drivable areas or obstacles which is very useful for path planning and autonomous navigation. Difficulties are presented in the data association problem which is normally done by a cross-correlation search over the vehicle's pose region. Also, the resolution is very important to the computational complexity algorithms, as more precision leads to more computational expense. As in every map representation, the map drifts after a certain time and loop closing becomes very important to obtain consistent maps. In this thesis a method for correcting the drift over the whole occupancy grid map is presented (Chapter 6).

Occupancy grid algorithms where first introduced by Moravec and Elfes [95] and the goal is to calculate the posterior over maps given the set of measurements $z_{1:t}$ and the robot's positions $x_{1:t}$.

$$p(m|z_{1:t}, x_{1:t}) \ , \tag{2.10}$$

where the map m represents the collection of individual cells $m = c_i, i = 1, 2, ..., n$.

Each cell c_i represents a binary random variable which corresponds to the occupancy of the cell by an obstacle (0: non occupied, 1: occupied). It can be observed that this is a high dimensionality problem that arises from the number of maps that can be represented. For instance, a map of 10×10 meters with $1cm^2$ cells would give $2^{1000000}$ different maps. Calculating a posterior probability for each map would be highly expensive, therefore, the problem can be breakdown into computing the poste-

rior for all individual cells c_i with a static state. The posterior can be approximated as the products of its marginals.

$$p(m|z_{1:t}, x_{1:t}) = \prod_i p(c_i|z_{1:t}, x_{1:t}) \ . \tag{2.11}$$

This estimation can be elegantly represented by its log odds (see [138]),

$$l_{t,i} = log \frac{p(c_i|z_{1:t}, x_{1:t})}{1 - p(c_i|z_{1:t}, x_{1:t})} \ . \tag{2.12}$$

This avoids truncation problems from probabilities close to 0 or 1 and allows a simple additive update.

As an extension of occupancy grids, Stachniss and Burgard [128] introduced the concept of coverage maps, where each cell represents a posterior about the percentage of coverage of the cell occupied by an obstacle or object. In this sense, if an object is occupying 20% of a given cell, in the normal occupancy grid representation the cell would have a probability of 1, whereas in a coverage map the covering value would be 0.2. This is a richer representation but with higher memory requirements and higher computational cost. Instead of storing probability values, coverage maps store histograms that corresponds to the coverage posterior of the cells.

2.2 Simultaneous Localisation and Mapping

The ability of a mobile robot to navigate in an environment while localising itself in space is a fundamental issue in achieving autonomous navigation. When a robot knows about its location relative to an internal representation of the environment then different high level navigation tasks can be performed such as obstacle avoidance, path planning and task planning. We can think about the motivation behind this concept; it overcomes the need for a priori maps, and gives the advantage of handling more dynamic changes in environment by constructing maps from scratch and having a representation of the uncertainty inherent to the map and the robot pose.

In the robotics field, building a map incrementally while the robot is moving and at the same time using the same map to localise itself is known as SLAM.

Over the years great progress has been made and many ideas are continuously appearing addressing the SLAM problem with new improvements. Recent work has shown successful implementations of real-time single camera SLAM on a small scale [27], and more recently, large scale robotic SLAM with different sensors and platforms [11] [22].

The first SLAM algorithm that integrated uncertainty as a fundamental issue was called stochastic mapping [124]. It settled the basic formulation for subsequent SLAM

proposals using a landmark-based map framework. A vector with spatial variables such as robot and landmark positions was used to represent the state, and a covariance matrix was used to model the uncertainty, both updated using the Extended Kalman Filter (EKF). In this work they showed that landmark's location estimates are highly correlated and this correlation grows as more observations are added to the map implying that for a consistent SLAM solution it would be necessary to update the whole state vector following every landmark observation. As the dimension of the covariance matrix is the dimensionality of the state vector $\mathbf{x}$ squared, which depends on the number of landmarks stored in the map, for large scale SLAM this would result in a high computation demand making real time SLAM difficult to achieved.

Modern understanding of feature-based SLAM has developed and today the SLAM problem is often formulated in its general case as an inference graph. SLAM is performed by storing characteristic sensor measurements or landmarks in a map as they are observed, using the robot pose to estimate the location of previously stored landmarks while at the same time using each new measurement to refine the actual estimate of the robot's position. As we can see Figure 2.6, the variables invoked in the problem can be represented in a Bayesian network where the variables of interest are represented as: x_i: camera or robot positions, y_i: feature positions and z_i: feature measurements.

Previous and current algorithms for feature-based SLAM can then be understood of different ways to performing inference on this graph. There are different approaches to perform SLAM from a single moving camera. However, in this chapter we will briefly review two methodologies that we have used in some work developed in this thesis and that have proved to be prevalent along the SLAM community demonstrating excellent results: filtering approaches, and approaches that minimise the reprojection error (bundle adjustment) in an optimisation over selected images or keyframes. The

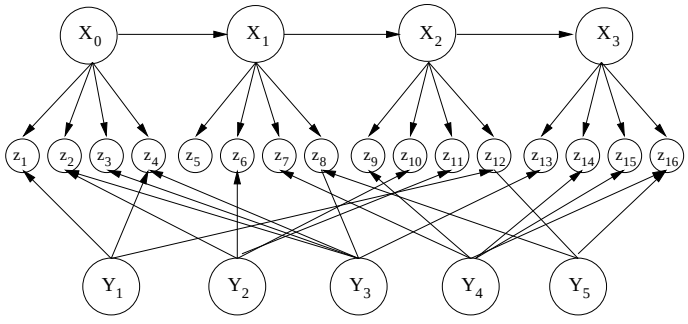


Figure 2.6: Bayesian Network representation for the SLAM problem. The variables involved are: x_i: position of the camera or robot, y_i: position of each individual feature in the map, z_i: feature measurements, depending on the features seen from a particular robot position.

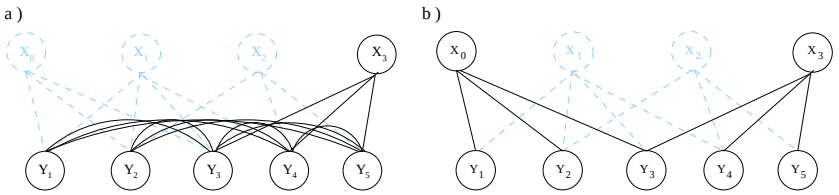


Figure 2.7: Inference graphs in filtering and Keyframe BA approaches. The image on the left shows how in filtering approaches all poses but the current one are marginalised out. The image on the right shows how in real-time BA optimisation approaches a set of carefully poses (keyframes) can be kept maintaining a graph which has many nodes but is sparsely interconnected.

latter provide the most accuracy per unit of computational time [132].

In the filtering approaches (see Figure 2.7(a)) the main idea is to propagate a joint probability distribution over the current pose and all the features seen so far. In every step, previous poses are marginalised out and propagated through time by means of a joint probability distribution represented by a single mean vector and a covariance matrix. Some of the drawbacks of this approaches are the linearization errors introduced by the filter and also that the computational cost scales poorly with the number of features . The most representative implementations are the ones that uses the EKF (Extended Kalman Filter), this methodology dating back to [125] and represented in visual SLAM by implementations such as [27].

The keyframe bundle adjustment style (see Figure 2.7(b) works instead via the idea of sparsifying the graph by keeping only some representative poses and all the observed features and solving the graph from scratch every time as needed. In these approaches there is no marginalisation over the poses not maintained; these and their correspondent measurements are simply discarded to obtain a sparsely interconnected graph. This approach has the advantage of been computationally less expensive than filtering for maps of large numbers of features. The system that brought this approach to the fore in monocular SLAM was Klein and Murray's PTAM [63].

I would like also to mention that there are other successful approaches to the problem of SLAM. For instance, those that use a particle filter. Montemerlo et al. [**?**] demonstrated scalability to handle large number of landmarks on its algorithm FastSLAM. It applies a version of particle filters knows as Rao-Backwellized relying on an observation that the SLAM problem exhibits conditional independence between any disjoint set of features in the map given the robot pose. A characteristic of Rao-Backwellized filters is that they use particles to represent the posterior over some variables and Gaussians or other PDF to represent other variables. In FastSLAM particles are used to estimate robots path and low dimensional EKFs to estimate feature locations, one for each feature.

Within the particle filter methods can mention the work of Eade and Drummond [33] that used the FastSLAM algorithm for real-time monocular SLAM, mapping a large numbers of landmarks. Pupilli and Calway [112] developed a real-time algorithm to handle erratic motions, the authors proposed a novel filtering approach that uses an auxiliary Unscented Kalman Filter coupled to the main particle filter.

Finally, there is a very recent branch of SLAM that takes advantage of the computational resources of nowadays systems. Newcombe et al. [99] developed DTAM, a dense real-time monocular SLAM, due to the large amount of data considered and a multi-scale alignment method, it achieves high accuracy managing fast camera motions and occlusions when compared with previous feature-based methods.

2.2.1 Feature-Based

EKF SLAM

Probably one of the best studied techniques for the process of filtering in SLAM is the EKF, which is a generalisation of the linear Kalman Filter (KF) to nonlinear systems. The KF has a fundamental assumption which is linearity, it means that observations must be linear functions of the state and the next state must be a linear function of the previous state. This assumption is basically due to the fact that the KF relies on Gaussianity and any linear transformation of a Gaussian random variable.

Linearity is rarely met in practice in both state transitions and measurements. For the EKF the state transition probability and the measurement probabilities are governed by general nonlinear functions f and h, respectively [138]:

$$\begin{aligned} x_t &= f(u_t, x_{t-1}) + \varepsilon_t \\ z_t &= h(x_t) + \delta_t \end{aligned}, \tag{2.13}$$

where ε_t and δ_t are zero mean uncorrelated Gaussian noise vectors with covariances $\mathbf{Q_t}$ and $\mathbf{R_t}$ for the transition and measurement processes respectively. The EKF linearises f and h and obtains the belief $\mathbf{x_t}$ by estimating the mean $\hat{\mathbf{x}}$ and the covariance $\mathbf{P}$ of a Gaussian approximation.

Basically, the EKF algorithm can be divided into two processes:

- **Prediction**. This step predicts the state $\hat{\mathbf{x}_t}$ and $\mathbf{P_t}$ before measurements are incorporated, taking as an input $\hat{\mathbf{x}_{t-1}}$ and $\mathbf{P_{t-1}}$. This prediction step also depends on the control u_t.

- **Update**. Here the state estimate improves and the uncertainty P is reduced by

the incorporation of new information (measurements).

When applying EKF to SLAM in addition to estimating the camera pose it also estimates every landmark's position in the map. The map is represented by a state vector $\mathbf{x}$ and a covariance matrix $\mathbf{P}$ given by:

$$\hat{\mathbf{x}} = \begin{pmatrix} \hat{x_v} \\ \hat{y_1} \\ \hat{y_2} \\ \vdots \\ \hat{y_N} \end{pmatrix} \quad , \quad \mathbf{P} = \begin{bmatrix} P_{xx} & P_{xy_1} & P_{xy_2} & \dots \\ P_{y_1x} & P_{y_1y_1} & P_{y_1y_2} & \dots \\ P_{y_2x} & P_{y_2y_1} & P_{y_2y_2} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} , \tag{2.14}$$

where x_v is the robot or camera pose and y_N are the positions of the features, $\hat{x_v}$ and $\hat{y_N}$ are its estimates. The diagonal elements of $\mathbf{P}$ represent the estimated error covariance of the camera position and features in the map, its off-diagonal elements represent the cross-covariance matrices between elements in the map.

The camera moves around a region according to a given motion model f and observes the world according to its measurement model h. When applying EKF to SLAM the prediction step is usually caused by the camera movement and the update phase occurs when the camera observe features.

By convention $\hat{\mathbf{x}}_{[t|t]}$ represents the state estimate at time t with the information acquired up to t and $\hat{\mathbf{x}}_{[t|t-1]}$ represents the state estimate at time t with the information given up to $t-1$. This suggests $\hat{\mathbf{x}}_{[t|t-1]}$ is the estimate acquired in the prediction phase and $\hat{\mathbf{x}}_{[t|t]}$ in the update phase. Therefore, more formally, the prediction phase is given by,

$$\begin{aligned} \hat{\mathbf{x}}_{[t|t-1]} &= f(\hat{\mathbf{x}}_{[t-1|t-1]}, u_t) \\ \mathbf{P}_{xx,[t|t-1]} &= \nabla f \mathbf{P}_{xx,[t-1|t-1]} \nabla f^T + \mathbf{Q}_t \end{aligned} , \tag{2.15}$$

where ∇f is the Jacobian of f evaluated at $\hat{x}_{[t-1|t-1]}$. The update phase is given by,

$$\begin{aligned} \hat{\mathbf{x}}_{[t|t]} &= \hat{\mathbf{x}}_{[t|t-1]} + \mathbf{W}_t \nu_t \\ \mathbf{P}_{[t|t]} &= \mathbf{P}_{[t|t-1]} - \mathbf{W}_t \mathbf{S}_t \mathbf{W}_t^T \end{aligned} , \tag{2.16}$$

where the innovation, which is the difference between the actual measurement z and the prediction h, is defined as ν, the innovation covariance as S and the Kalman Gain which specifies the degree in which the measurement is incorporated into the new state estimate is represented as W,

$$\begin{aligned} \nu_t &= \mathbf{z_t} - \mathbf{h}(\hat{\mathbf{x}}_{[t|t-1]}) \\ \mathbf{S}_t &= \nabla \mathbf{h} \mathbf{P}_{[t|t-1]} \nabla \mathbf{h}^T + \mathbf{R}_t \\ \mathbf{W}_t &= \mathbf{P}_{[t|t-1]} \nabla h^T \mathbf{S}_t^{-1} \end{aligned} .$$

MonoSLAM

The first practical real-time 3D monocular SLAM based on the probabilistic framework proposed by [124] was Davison's MonoSLAM [27], building on earlier single camera filtering work such as [50, 19]. Since we will use MonoSLAM in Chapter 4.

- Map Representation

  The map is represented as a set of sparse but sufficient features. Each time a new frame is grabbed the Shi and Tomasi [118] operator is applied to detect stable long-term image regions using relatively large patches [31]. The state of the system **x** is represented by a vector composed of the state of the robot or the camera $\hat{x_v}$ and the states $\hat{y_N} = (x_N\, y_N\, z_N)^T$ which correspond to feature locations in the map. A single covariance matrix is used to represent the uncertainty in all the quantities of the state vector (see Equation 2.14).

  Having a covariance matrix is important not only to have a consistent map at every timestep, but also because features are matched in an 'active search' process according to the uncertainty predicted for each feature in the matrix. The state vector and the covariance matrix grow dynamically each time a new feature is observed.

- Initialisation

  One of the main difficulties of monocular SLAM is depth recovery, because the 3D position of a feature can not be initialised from a single observation. In the original MonoSLAM algorithm a pattern with known geometry was necessary to start mapping, providing a metric scale. Usually this reference frame comprised of features with known position and appearance (see Figure 2.8).

  Later, Montiel *et al.*[94] proposed an inverse-depth feature representation for undelayed initialisation. Here the feature state is represented as $\mathbf{y}_i = (x_i, y_i, z_i, \theta_i, \phi_i, \rho_i)^T$, where θ_i and ϕ_i are the azimuth and elevation of the ray from the camera to the feature point, and the depth along this ray is represented by its inverse $\rho_i = 1/d_i$.

- Filtering Process

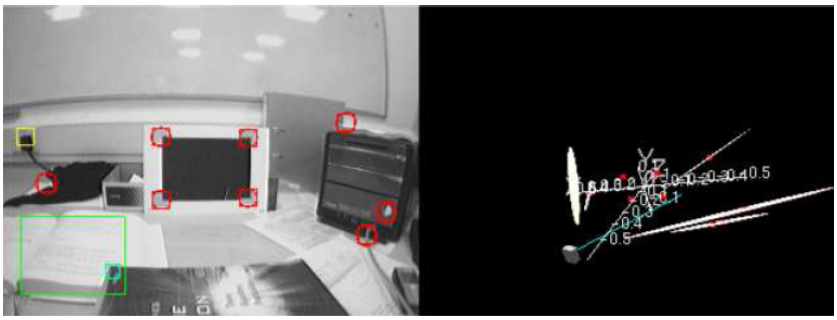


Figure 2.8: MonoSLAM: Initialisation with a known pattern.

The EKF proceeds by repeated and interleaved prediction and update steps. There are two corresponding models, the motion model which describes the expected dynamics of how the camera moves in the blind interval in between measurements, and the measurement model that describes how measurements affect the map and camera uncertainty.

The motion of the camera follows a "constant velocity, constant angular velocity model". This means that on average undetermined accelerations are expected to occur with Gaussian profile. Two reference frame are defined; W fixed in the world and R attached to the moving camera. State vector $r^W = (x, y, z)$ represent the position of the camera and $q^W = (q_0, q_x, q_y, q_z)$ the orientation. Then the state of the camera can be modelled as:

$$\mathbf{x}_v = \begin{pmatrix} r^W \\ q^{WR} \\ v^W \\ \omega^W \end{pmatrix},$$

The state vector also includes v^W representing linear velocity and ω^W the angular velocity. At each time step it is assumed that an impulse in velocity $V^W = a^w \Delta t$ and angular velocity $\Omega^W = \alpha^w \Delta t$ occurs as noise in the model with unknown acceleration a^W and angular acceleration α^W processes of zero mean and Gaussian distribution. The state update is given by:

$$\mathbf{f}_v = \begin{pmatrix} r^W_{new} \\ q^{WR}_{new} \\ v^W_{new} \\ \omega^W_{new} \end{pmatrix} = \begin{pmatrix} r^W + (v^W + V^W)\Delta t \\ q^{WR} xq((\omega^W + \Omega^W)\Delta t) \\ v^W + V^W \\ \omega^W + \Omega^W \end{pmatrix}. \tag{2.17}$$

Once the camera state has been predicted via this motion model, the measure-

ment model allows prediction of the values of the measurement. The means that it predicts the position (u,v) at which each feature is expected to be found. The this follows the pinhole camera model based on current estimates of the positions of camera and scene features.:

$$h_i = \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} u_0 - fk_u \frac{h^R_{L_x}}{h^R_{L_z}} \\ v_0 - fk_v \frac{h^R_{L_y}}{h^R_{L_z}} \end{pmatrix}, \tag{2.18}$$

where h^R_L is the position of a point feature relative to the camera. The MonoSLAM method takes advantage of the innovation covariance matrix $\mathbf{S}_i$ to restrict the search for each feature to an elliptical region around the predicted feature location (u,v), the size of this region is defined by the innovation matrix and a chosen number of standard deviations (normally three), this is called active search.

Active search is performed by correlating a template exhaustively within the boundaries of this elliptical region, then the top scoring template match is taken as correct if it normalised sum of squares difference score passes a threshold

Each feature's innovation is defined to be the discrepancy between the predicted feature location and the measurement.

$$\mathbf{S}_i = \frac{\partial h_i}{\partial x_v}\mathbf{P}_{xx}\frac{\partial h_i}{\partial x_v}^T + \frac{\partial h_i}{\partial x_v}\mathbf{P}_{xy_i}\frac{\partial h_i}{\partial y_i}^T + \frac{\partial h_i}{\partial y_i}\mathbf{P}_{y_i x}\frac{\partial h_i}{\partial x_v}^T + \frac{\partial h_i}{\partial y_i}\mathbf{P}_{y_i y_i}\frac{\partial h_i}{\partial y_i}^T + \mathbf{R}\,. \tag{2.19}$$

At any given frame if the number of features being tracked drops bellow a threshold, new features are initialised by randomly selecting new positions in the image.

MonoSLAM can maintain a small map of up to 100 features in real-time, this performance is achieved mainly by modelling the motion of the camera, maintaining a sparse map of features and by using small regions guided by uncertainty to search for measurements.

In large mapped areas the performance of MonoSLAM is degraded due to its quadratic computational complexity in the total number of features in the map. However, this issue can be addressed by using a hierarchical submapping approach that maintains a set of submaps and keeps the corresponding links in a global map [22].

Visual Odometry

MonoSLAM and related algorithms are optimised towards enabling repeatable localisation within a small operating workspace. A strongly related but slightly different

stream of research in real-time geometrical vision has instead concentrated on accurate relative motion estimation, usually called Visual Odometry (VO). VO is the incremental, online estimation of camera motion from a video sequence. It is closely related to SLAM and only one or two cameras are needed for the task, making it suitable for low cost applications.

The main difference between probabilistic approaches such as EKF-SLAM and VO is that while EKF-SLAM maintains an estimate of all feature positions relative to the robot pose from the beginning until time t, VO only uses a few images or one image before time t to estimate motion. This has the advantages of bounded computational cost. While EKF-SLAM updates its main vector and covariance matrix each time any measurement is made, in VO all processing is temporally local independent of the time the system as been in operation or how far the camera has moved. This leads the fact that while VO estimation can be locally very accurate, it will lead to drifting motion estimates over long trajectories unless some other component is used to explicitly resolve this.

Over the past decade there has been a great improvement in mapping large areas using VO. One of the most important pieces of work on VO in large areas was by Nister [101], presenting a real-time system tested in video taken from aerial, automotive and hand-held platforms showing results of camera trajectories of over 600 meters. Harris features [51] are extracted and tracked over a certain number of frames. Pose estimation is an iterative pose refinement implemented in two different schemes: monocular and stereo motion estimation. In the monocular scheme first the 5-point algorithm [100] is used to triangulate the observed feature tracks into 3D points, then the 3-point algorithm [113] is used to compute the pose of the camera. In the stereo scheme after matching left and right images and triangulating the observed points into 3D points, the 3-point algorithm is used as the hypothesis generator to compute the stereo rig pose.

In [69] a real-time system for VO outdoors using stereo was presented. Interest points are used to find correspondences between images as well as sparse bundle adjustment to correct estimate poses. An Inertial Measurement Unit (IMU) is also used to refine positions using its angle and gravity information.

In [70] a system called FrameSLAM is presented as a full real-time visual mapping method. The authors perform SLAM by using a stereo camera mounted on a robot. The map representation is a set of keyframes which are saved and joined together by nonlinear constrains, this is called a skeleton (a pose graph). The goal is to obtain the maximum likelihood (ML) map by optimising a cost function that takes into account the non-linear constrains between poses or nodes and the desired graph configuration. To estimate local motion, features are extracted from left and right images of the stereo

pair and matched with those of the previous frame, then a RANSAC three-point algorithm [38] is used to obtain the relative pose between frames given the calibrated camera set up.

Another state of the art method in VO is the work of Mei et.al. [87]. The authors also address the problem of obtaining precise positioning of a stereo camera. As FrameSLAM, the environment is represented in a graph, the difference is the use of a relative pose representation. A bundle adjustment method incrementally computes the ML map in its relative formulation in constant time [119]. Tracking is done frame to frame using FAST corners in different pyramid levels. This approach also performs relocalisation when frame to frame data association fails by attaching SIFT descriptors to the previously computed FAST corners. Loop closure detection is done by a bag of words method.

Pose-Based Bundle Adjustment SLAM

One of the pioneers of Pose-Based graphical formulation in SLAM is the work of Thurn and Montemerlo [139] coined GraphSLAM. This algorithm solves the full SLAM problem, its goal is to estimate a solution for all robot poses and features in the map. The name of this algorithm comes from the idea that SLAM can be thought or visualised as a graph in which nodes are usually robot positions and edges are relations between each node. In GraphSLAM nodes are robot and feature positions connected through motion and measurement edges. Motion edges link any two consecutive robot poses and measurement edges link poses to features that were measure there. Each edge represents a nonlinear constrain encapsulating the motion model in the motion constrains and the measurement model in the measurement constrains.

GraphSLAM linearises these constrains obtaining an information matrix in which after applying an elimination algorithm a smaller graph is obtained defined only over robot poses, after this process standard inference techniques are used to compute the posterior map.

To understand the key concepts of this algorithm some differences between EKF SLAM and GraphSLAM could be emphasise. EKF represents information in a covariance matrix and a mean vector, GraphSLAM does it with constrains over a graph of robot and feature poses. EKF is an online algorithm which means that is active in every step while GraphSLAM accumulates information into the graph not solving it until the end. GraphSLAM calculate posteriors for robots paths and EKF calculates posterior over a momentary pose of the robot.

A final class of real-time visual motion estimation methods we will consider in detail take ideas from both the EKF-SLAM approaches and the VO methods. Like MonoSLAM, they aim at repeatable localisation within a small area, but rather than

achieving this by filtering they use a carefully interleaved combination of methods very similar to those used in the best VO approaches. The key factor that enables repeatable localisation rather than exploratory motion estimation is the automatic definition of a set of keyframes which are spatially, rather than temporally selected.

Probably the most representative approach on real-time keyframe-bundle adjustment is PTAM (Parallel Tracking and Mapping) [63] [64].

PTAM is a real-time SLAM method which is very well known for its accuracy and its excellent performance. It tracks thousands of features but is able to perform in real-time over a small workspace with modest computing hardware. The system is based on the idea of splitting tracking and mapping into two separate tasks and processing them in parallel threads. This implementation as well as the use of only selected keyframes for tracking, compensate the cost of operation of a batch optimisation technique in the back end — classical bundle adjustment.

The map contains a set of features along with estimates of their patch normals and a reference to the patch source pixels. It also contains the keyframes used along the sequence and a four-level grey-scale pyramid representation per keyframe. The map initialisation involves user interaction by indicating to save the first keyframe and then moving the camera smoothly and after some motion saving a second keyframe. While this happens the features are tracked and then initialised by using the five-point stereo algorithm [100].

The tracking method maintains a camera pose estimate in real time in two coarse to fine stages: first it updates the camera pose by searching only for a small number of features at the coarsest scale. Then, a final pose estimate is obtained by reprojecting a large number of points (~ 1000) and searching over the whole image space.

Keyframe insertion is done based on time since the last keyframe and also taking into account the minimum distance from the nearest keyframe already in the map. Finally, BA adjusts the map in a separate thread everytime a new keyframe is added. Running BA over all keyframes is computationally expensive even with a dedicated thread, and for this reason it is also allowed to perform local BA over a fixed set of neighbouring keyframes.

PTAM avoids the computation expense of filtering approaches that try to build optimal maps at each frame. Instead, it is able to obtain more accuracy and to manage faster motions by robustly tracking hundreds of features and correcting the error in a BA scheme. The recent work of Strasdat et al. [132] confirms PTAM engineering framework by concluding that the SLAM problem is computationally more efficient when keeping minimum number of keyframes (except when tracking cost is to high) and a maximum number of correspondent features.

Appearance Based SLAM, FabMap

In recent years, there has been several attempts to determine the position of a robot based on comparing previously visited places based on the appearance. In these methods the world is represented as a set of discrete locations. In computer vision, the appearance can be extracted by representing images in a global descriptor with tools such as histograms that contain colour, texture, intensities or gradient information, or with more recently approaches that make use of quantised features — bag or words — that allow a more local description.

One disadvantage of these methods is that they provide information about how certain the robot is to be around a particular place but they do not generally provide with geometric information about the scene or the robot's position. However, these can be extracted later with other methods.

FabMap (Fast Appearance Based Mapping) [24] is one of the most successful appearance based approaches inspired by the bag of words systems developed in previous years by the computer vision community [122] [102]. A bag of words refers to an image as part of a visual vocabulary where words are quantised features in the descriptor space.

FabMap computes the probability distribution over its location based on appearance without using any metric information. It learns a generative model of appearance. This model captures and learns the fact that certain combinations of appearance words tend to co-occur because they are generated by the same object. Creating a generative model of the environment based on appearance, involves having a high dimensional space that quickly becomes intractable. This algorithm makes use of Chow Liu Trees [20] which approximates a discrete distribution by the closest tree-structure Bayesian network. It is based on the mutual information graph which measures the degree to which knowledge of the value of one variable predicts the value of another. In that sense, it obtains the maximum-weight spanning tree of the mutual information graph.

An observation of a local scene appearance is represented by a vector $Z_k = z_1, ..., z_{|v|}$, where z_i is a binary variable indicating the presence of the ith word of the vocabulary v, Z_k is the set of all observations up to time k. Also, at time k, the map has a collection of $L_k = l_1, ..., l_{n_k}$ poses and each location is modelled as a set $p(e_1 = 1|l_i), ..., p(e_{|v|} = 1|l_i)$ where e_i are feature generating objects (words). Hence, the method looks to obtain the probability of being in a location given all observations up to time k:

$$p(L_i|Z_k) = \frac{p(z_k|L_i, Z_{k-1})p(L_i|Z_{k-1})}{p(z_k|Z_{k-1})} \ . \tag{2.20}$$

For detail equation derivations please refer to [24].

2.3 Mobile Robot Navigation

We have reviewed approaches to localisation and mapping, but these are not the only capabilities required by a mobile robot aiming to be fully autonomous. In fact, the majority of research on SLAM has not involved the control of a real robot but the processing of datasets obtained during manually controlled run. Autonomous navigation presents significant further challenges alongside localisation and mapping, including path planning, exploration and free space detection.

2.3.1 Path Planning

Path planning is very important in different areas such as robotics, control theory, artificial intelligence or video games. In robotics, the task of deciding and executing a sequence of motions that takes a robot from one place to another following a collision-free path is called motion planning. The computation of the path, usually in a feasible and optimal way is what we call path planning.

In this Section we will give a brief overview of the general concept in robotics and specifically in a discrete and finite state space. This simplifies the formulation because the world can be defined as a set of finite and countable states. Hence, we can model the world as a 2D grid composed of individual cells, the robot then can take discrete steps on the individual cells (see Figure 2.9).

More formally, in this discrete scenario, the world can be modelled using the following formulation :

- A finite and non empty state space $\mathbf{X} = x_1, ...x_n$.
- For each state $x \in \mathbf{X}$ a finite action space $U(x)$.
- A state transition function that when applied produces a new state $x' = f(x,u)$.
- An initial state x_I and a goal state(s) x_G.

The state usually contains the position and orientation of the robot, as we can notice, a state is analogous with a node in pose-based SLAM approaches.

Given this discrete representation of the world, different graph search algorithms can be directly applied from a planning perspective with the requirement that the algorithm has to keep track of the states already visited to ensure no redundant exploration. In the next Section we will cover some of the most used search algorithms for path planning.

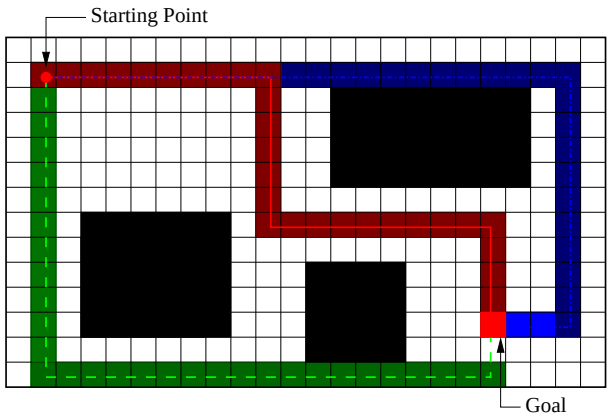


Figure 2.9: World represented as a 2D Grid. Three different paths are shown. Those paths take the robot from a starting point to a Goal, path planning involves computing such a path that is feasible and somehow optimal.

Graph Search Algorithms

- *Breadth and Depth First Search*. These two algorithms are two of the simplest methods for searching a graph and producing a shortest path. Both work on directed and undirected graphs and have complexity of $\Theta(V+E)$, where V is a set of vertex or nodes and E is the set of edges in the graph . The main difference is in the way they traverse the graph which reflects the data structure they use.

  Given a graph $G=(V,E)$ and a root node s, breadth first search discovers all vertices at distance k from s before discovering any vertices at distance $k+1$. It expands the frontier between discovered an undiscovered vertices across the breath. Its data structure is a First-In, First-Out (FIFO) queue.

  The strategy in depth first search is to go deeper in the graph by exploring the edges discovered by the most recent vertex v that still have unexplored edges. It basically stores nodes in a Last-In, First-Out (LIFO) stack in order to backtrack and explore the edges from which v was discovered.

- *Dijkstra's algorithm*. This method obtains the shortest path from a starting node N_s to a goal node N_g on a weighted, directed graph $G=(V,E)$ with the assumption that all weights w are positive. The weight can be seen as the cost to get from one node to another, for instance, it could be the 2D distance. This method uses a min-priority queue to find the vertex with the less weight outside a given set. The complexity of the algorithm is $\Theta((V+E)logV)$. However, by using a Fibonacci heap as a min-priority queue, the complexity is reduced to $\Theta(VlogV+E)$.

- *A* algorithm*. This algorithm is one of the most used methods to find a shortest

path in a graph with the minimum steps. It works very similarly to Dijkstra's algorithm. The main difference is a heuristic $H(N)$ which makes the algorithm run in general faster than Dijkstra. The heuristic is an estimated distance from any given node to the destination or goal node N_g. The closer the heuristic is to the actual distance to the goal, the better the path and the running time will be. A possible heuristic would be to use for example just an absolute or L1 'Manhattan' distance d from $N_1(x_1, y_1)$ to $N_2(x_2, y_2)$, where $d = |x_2 - x_1| + |y_2 - y_1|$.

This method can be implemented in the same way as Dijkstra but with a change in the manner the priority queue Q is sorted. The distance obtained from the heuristic is now added to the vector of distances d implying that the priority queue contains estimates to the optimal cost from N_s to N_g. If $H(N)$ is underestimated from the true optimal distance then this method is guaranteed to find optimal paths. If $H(N) = 0(\forall N \in G)$, then the method converges to Dijkstra's solution.

Dynamic Window Approach — DWA

In the previous Section we reviewed different graph searching algorithms that are commonly used in path planning. Those methods are normally employed in situations where the map is already well defined and static, i.e. no dynamic or moving objects are present, obtaining a more global path planning strategy.

In a more local path planning strategy within a dynamic environment and when the map is not known in advance, the previous methods could be used by defining local goals. However, because the map is not well defined and obstacles can be moving around the robot, on each time step, the robot would have to compute a new path and because the velocities are not modified the robot could collide with an object. There are another methods that work in the velocity space incorporating vehicle dynamics, this methods reduce the velocity when the robot find an obstacle near its surroundings and increase it when there is an open space ahead of it, making then more suitable for local path planning. One of the most common used algorithm in this domain is the Dynamic Window Approach (DWA) .

The Dynamic Window Approach [40] explores the idea of computing smooth circular trajectories over a short time interval, dealing with the constraints imposed by limited velocities and accelerations. The set of possible circular *arc* trajectories is represented by a 2D space of translational and rotational velocities (v, w). These trajectories are based on the Curvature Velocity Method [121] with the difference that in DWA, the velocities are computed only in a local window centred on the current velocity of the robot.

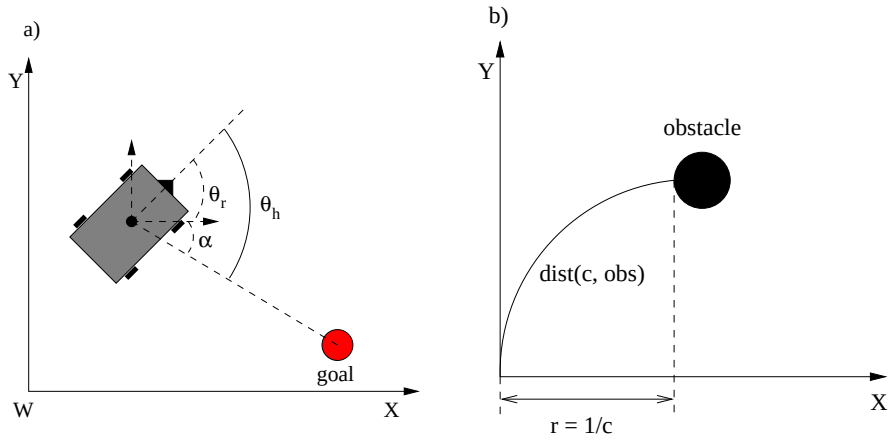


Figure 2.10: Robot's heading (a) and curvature (b) diagrams. Diagram a) shows in global coordinates the angles computed from the robot's position to the goal: θ_h - Angle from the robot to the goal, θ_r - Robot's heading position, α - Angle to the goal. Diagram b) indicates the distance from the robot's centre to the obstacle obtained by computing the curvature $c = w/v$.

The best trajectory is obtained by maximising the following function:

$$G(v,w) = \sigma(\alpha * heading(v,w) + \beta * dist(v,w) + \gamma * vel(v,w)) \ , \tag{2.21}$$

where $heading(v,w)$ is a measure of the direction of the robot with respect to the goal. The purpose of minimising this measure is so that the robot is aligned towards the goal. For a given v and w, the predicted position (see Equation 2.23) is obtained and from there, the angle θ_h of the target point relative to the robot's orientation is computed (see Figure 2.10(a)).

$$\theta_h = \alpha - \theta_r \ . \tag{2.22}$$

$$\begin{pmatrix} x' \\ y' \\ \theta_r' \end{pmatrix} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} -\frac{v}{w}\sin\theta_r + \frac{v}{w}\sin(\theta_r + w\Delta t) \\ \frac{v}{w}\cos\theta_r - \frac{v}{w}\cos(\theta_r + w\Delta t) \\ w\Delta t \end{pmatrix} \ . \tag{2.23}$$

The term $vel(v,w)$ is a measure that prefers higher velocities. In this manner, the robot always tries to keep the maximum admissible velocity in order to minimise the time spent to reach the target. An admissible velocity V_a is defined as a pair (v,w) such that when applied, the robot is able to stop before reaching the object.

$$V_a = (v,w) | v \leq \sqrt{2 * dist(v,w) * \dot{v}_b} \ \ \wedge w \leq \sqrt{2 * dist(v,w) * \dot{w}_b} \ , \tag{2.24}$$

where $(\dot{v}_b, \dot{w}_b)$ are the accelerations for breakage.

The measure $dist(v,w)$ represents the distance to the closest obstacle that intersects with the curvature $c = w/v$ (see Figure 2.10(b)).

Partially Observable Markov Decision Processes - POMDP

In this Section we will review a probabilistic approach to motion and robot control, this technique has not been used in the theses. However, we would like to make the reader aware of this other type of methods.

As we can observe from the previous classical approaches to path planning, they are deterministic and no uncertainty involved, the planning algorithms are often combined with reactive methods to adjust the plan and to avoid collisions. A paradigm that can be used to incorporate uncertainty in the robot's motion are the Markov Decision Processes (MDP). MDP is a stochastic framework that assumes a fully observable state, meaning that the state x can be fully sensed at all time. In this situation, a planner has to generate actions for a whole range of situations (belief space) and to choose the best one using a policy for action selection. A policy is basically a function that maps states into controls $\pi : x \rightarrow u$ and the goal is to find the policy that generates actions so as to maximises the immediate and future expected payoff. A policy is optimal if it maximises the sum of all future cumulative payoffs. In MDP the optimal policy involves the calculation of a value function which measures the expected cumulative payoff R_T and which defines a policy for selecting the control that maximises value [138].

$$R_T = E[\sum_{\tau}^{T} \gamma^T r_{t+\tau}] \,, \tag{2.25}$$

where $E[]$ is the expectation operator, $r_{t+\tau}$ is the payoff value at time $t+\tau$, γ is a discount factor that favours exponentially earlier payoffs and its range is $0 < \gamma < 1$ and T is the planning horizon, such that if $T = 1$ then the algorithm only seeks immediate payoff in what is called greedy case.

One of the most used algorithms for finding control policies is *value iteration*. The main idea of this method is to recursively calculate the utility of each action relative to a payoff function. The policy π is obtained by maximising the expected payoff over all controls u, the payoff is defined as $r(x,u)$ which is a function of the state x and the control u. To every policy there is also an associated value function V which meassures the expected value of the specific policy. In stochastic systems, the algorithm successively updates an approximate value function V by a recursive rule. Each update propagates information back in time through the value function.

For instance, starting with a value $T = 1$, we have π_1 and V_1 given by:

$$\begin{aligned} \pi_1(x) &= argmax_u r(x,u) \\ V_1(x) &= \gamma \ max_u r(x,u) \end{aligned} , \tag{2.26}$$

then for time $T = 2$ the policy takes into account V_1 and posterior $p(x'|x,u)$,

$$\begin{aligned} \pi_2(x) &= argmax_u [r(x,u) + \int V_1(x') p(x'|x,u) dx'] \\ V_2(x) &= \gamma \ max_u [r(x,u) + \int V_1(x') p(x'|x,u) dx'] \end{aligned} , \tag{2.27}$$

finally, for any T, the optimal policy and the value function can be obtained recursively from time step $T-1$

$$\begin{aligned} \pi_T(x) &= argmax_u [r(x,u) + \int V_{T-1}(x') p(x'|x,u) dx'] \\ V_T(x) &= \gamma \ max_u [r(x,u) + \int V_{T-1}(x') p(x'|x,u) dx'] \end{aligned} , \tag{2.28}$$

In a more realistic case, where the state is not fully observable due to incomplete measurements or noise then the MDP framework is called Partially Observable Markov Decision Process (POMDP). Partially observability implies that the robot has to estimate a posterior distribution over all possible states. This means that the same idea behind Equation 2.28 can be applied but the robot has to make its decision in the belief space.

Complexity problems arise here because the belief is a probability distribution. Finding solutions involves the computation of the space of all distributions over the state space. A solution to this problem is to represent value functions by piecewise linear functions over the belief space [138] [17].

2.3.2 Free Space Detection

Fully coupled SLAM and free-space mapping suitable for autonomous navigation has been a standard capability in 2D SLAM using laser range-finders for several years, since early work such as [137]. 2D occupancy maps 'jump out' relatively straightforwardly from aligned laser scans, which directly sweep out free space. We should expect to see very similar methods successfully and widely applied soon in 3D to the data from 3D depth cameras which are now becoming widely available at low cost.

On the assumption that our robot moves on a ground plane, 2D free space mapping is critical to allow obstacle avoidance and path planning. Methods extracting feature

points will always be limited when used in free space identification: what is really needed is a dense analysis which aims to label every pixel in an image. Recent work in advanced stereo in computer vision is now starting to achieve real-time performance in reconstructing dense surface models (e.g. [108] in urban environments, or [97] indoors on a desktop).

Another variant would be to obtain 3D occupancy information from a CAD model [66] and from there correlate features extracted from the model to features in the actual image. This may produce good results but assumes that manual maps are available. A more complete approach is the work by [86] where a texturized 3D occupancy grid is constructed from 3D data obtained from range sensors and a single camera. This textured map may further being used for localisation and navigation.

Free space detection can be aided in applications where specific assumptions can be made. One example is in road detection for vehicles to aid autonomous driving. Most approaches to the road detection problem make use of the vanishing point as the main indicator of the road geometry (e.g [114, 68]). Some approaches attempt to define the drivable area either using off-line machine learning techniques such as Support Vector Machines [146] or by a combination of geometry information and offline learning [4]. Such techniques cannot however be directly transposed to the indoor 2D free-space detection problem, because in this context there is no such thing as a general geometric pattern for the boundary of the drivable area that can be retrieved in each image based on (potentially learnt) a priori information.

Whereas road detection is still a challenging problem, in most of the cases the vanishing point or the geometry that delineates the boundary of the road is very well defined. When working in indoor environments, the free space or drivable area is geometrically challenging and difficulties can arise when using outdoor methods.

However, a wheeled robot can take advantage of the known structure of the environment to make progress in a simpler and computationally cheaper way. Some successful methods for obstacle avoidance and free-space mapping for mobile robot navigation rely solely on the use of colour information [53], or infer a planar homography mapping image pixels to 2D floor coordinates under a ground plane assumption [151, 61]. The authors of [79] propose to make use of multiple cues to calculate horizontal lines defining the boundary between floor and walls, which is well suited to corridors, but presumably not adapted to the case of more complicated structure.

2.3.3 Exploration

Exploration can be defined as the task of controlling a robot through an unknown environment by maximising the information obtained by its sensors and keeping a record of the explored areas and the positions where the robot has been. The complexity re-

lies on the idea of covering the whole area minimising the required time or the number of steps of the robot. Therefore, if the map representation is a grid, the exploration problem maximises the cumulative information for each cell [138].

In this Section we will review two of the most used methodologies for robot exploration: those that are based in information gain and the frontier-based approach.

In the literature a vast number of exploration algorithms are based on the idea of maximising the information obtained while minimising the number of control steps. These techniques use the concept of information gain employing greedy policies to limit the computation complexity due to the high dimensional nature of the exploration problem. For instance, in [129] a robot equipped with a laser range finder is used for exploration, and mapping and localisation algorithms are deployed. The method computes the expected information gain taking into account the uncertainty in the map and the pose of the robot in a particle filter framework. In [13] a solution to SLAM and exploration for an indoor environment is presented. An EKF filter is used for the SLAM problem with a laser range finder providing feature measurements. This is combined with an occupancy grid representation maximising the information gain and minimising the uncertainty in the robot's pose and the map of features.

When talking about information gain, the first concept that comes to mind is entropy. Entropy quantifies the expected value of the information contained in a message. The entropy of a probability distribution is given by the following equations:

$$H_p(x) = E[-log_2 p(x)] \,, \tag{2.29}$$

which for the discrete and continuous cases resolves as:

$$H_p(x) = -\sum_x p(x) log_2 p(x) or H_p(x) = -\int p(x) log_2 p(x) dx \,. \tag{2.30}$$

When exploring we aim to find the entropy of the belief $B(b,z,u)$ after executing a control action u and observing the measurements z. Hence, the conditional entropy is given by:

$$H_b(x'|z,u) = -\int B(b,z,u)(x') log B(b,z,u)(x') dx' \,. \tag{2.31}$$

The expected information gain can be conceptualised as the expected change of entropy given that the robot obtains a measurement at a certain location in the map [127]. Therefore, for a cell c the information gain is defined as:

$$I_b(u) = H_p(x) - E_z[H_b(x'|z,u)] \,. \tag{2.32}$$

Greedy techniques look ahead one step from the current robot position. Hence, the they look to minimise the expected cost of any control action and maximise the

expected information gain. The optimal greedy exploration maximises the difference between the information obtained and the cost $r(x,u)$ associated to it.

$$\pi(b) = argmax_u \ \ \alpha(I_b(u)) + \int r(x,u)b(u)dx \ . \tag{2.33}$$

The second exploration algorithm to review is the well known Frontier-Based Approach [148]. The main idea behind this method is to move in the boundaries between open space and unexplored areas, pushing back the frontiers between unexplored and explored areas. In an occupancy grid, the algorithm first assigns a prior probability to each cell and then each cell has 3 possible scenarios: *open* (occupancy probability $<$ prior probability), *unknown* (occupancy probability = prior probability), *occupied* (occupancy probability $>$ prior probability). Then, frontier regions are detected as an aggregation of frontier edges which are any open cells adjacent to an unknown cell. Once the first frontiers are detected, the algorithm iterates in a depth first search manner trying to reach the nearest accessible unvisited frontier from the robot's current cell.

2.3.4 Autonomous Navigation

This Section presents work that has gone further than pure localisation and mapping. For a robot, it is certainly important to build a map and to localise within that map. However, for a robot to be useful it has to utilise its own built map. The systems presented in this section therefore look to answer questions such as: once a map is built then what can the robot do with that map? Or in which manner it is useful for other tasks? And how can the robot autonomously explore the environment and build a map at the same time? We need to have in mind that in most SLAM methods, maps are not built autonomously but by a human controlling the sensor motion. The task of exploration very difficult on its own.

In some areas, efforts are focused on trying to build systems that resemble human intelligence. However, we still don't have a clear understanding on the whole methodology of how humans perceive the world. Some others aim to build systems focusing in modularity and building successive layers that solve individual activities. An interesting example and a pioneer of this methodology is the work of Brooks [14], who proposed the subsumption architecture. The main characteristic of this work is the idea of a decentralised system in which layers of intelligence or competence are incrementally added and which operate in parallel. Each layer is composed of finite state machines that interact between each other without central command. This early work opened ups new paradigms where researchers instead of trying to build complex systems trying to understand whole human behaviour, they aim to build systems that gradually increment the level of intelligence by adding new modules.

We are particularly interested in Brook's idea to build complete systems based on modularity and incremental intelligence. In this sense, we aim to review those works that aim to build integral systems focused in robot navigation that can be modular, we are not interested in those that are very centralised and try to build complete systems all in once or in a black box style (e.g. by employing a Neural Networks or an Expert System).

People normally associate mobile robotics with autonomous devices which have full capability to move around an area, interacting with humans and going from one place to another without crashing. So autonomous navigation is a base capability for a robot which involves different tasks all interacting together. Navigation in indoor scenarios can be broken down into several levels of difficulty. First it can be assume a known environment (where the robot usually starts with a previously built map) with only static objects. In this scenario, several global path planning algorithms can be applied in a straightforward manner, only concerned about correcting the drift in the robots position by recognising known regions. A second level could be the incorporation of dynamic obstacles where obstacle avoidance techniques have to be applied. In a final more complex level the robot has to perform the following activities:

- Perform SLAM continuously finding loop closures and correcting the map error.
- Local path planning and obstacle avoidance.
- Exploration techniques.

In the literature, most of the work on systems that perform incorporate exploration, localisation, navigation, path planning and obstacle avoidance has been done by either using range sensors or a combination of multiple sensors. Range sensors provide an easy way to obtain an out of the box metric distance to objects. With a camera this is not straightforward and this is one of the reasons why many researchers prefer range sensors over cameras to achieve more reactive tasks such as obstacle avoidance. Our main focus in this thesis is a computer vision based approach to see whether these capabilities can be achieved while only relying on cameras, with all the potential forward-looking advantages they have over more specialised sensors.

Autonomous car driving has been an area where people has put a lot of attention on it. The Defence Advanced Research Projects Agency (DARPA) Grand Challenge was a competition in the U.S. for driveless vehicles, where they had to complete long desertic trajectories. The last one occurred in 2007 and it was performed on an urban scenario. The vehicles were equipped with latest technology, such as GPS, IMU, lasers and cameras.

Other applications have been seen in museum-tour robots and nowadays an increasing number of commercial floor cleaning robots have this capability. The aim of

this thesis is to focus on computer vision for autonomous navigation. Therefore, in this section we will only review some of the main works whose sensing is based on cameras.

In robotics, the work of Makarenko [85] is very representative of this section. It ts an integrated approach (see Figure 2.1) that develops an exploration strategy that looks for motion actions to obtain a balance between maximising map coverage and accuracy and minimising exploration time.

The method aims to solve first the basic task of exploration which is tightly coupled with localisation, mapping and motion control. The method breaks up the exploration mission into a series of discrete subtasks by the use of payoff functions. To do that, it first obtains sensor data that maximises information gain. A set of possible destinations is obtained and evaluated in those multiple utility functions in order to obtain a score. Once the robot selects a single destination it plans a path to reach it. Destinations are based on the frontier method [148] which proposes those that are on the edge between explored and unexplored areas.

The method combines three utility functions: the first one obtains the information gain of every cell and is based on the entropy around a destination point. The idea is to select those whose entropy is high, which are more suitable to explore; the second function favours those paths that reduce navigation time; the third one evaluates robot localisation based on the uncertainty around the current position. The algorithm maintains two map representations: and occupancy grid for information gain an navigation tasks , and map of features for localisation. For localisation, the method uses a feature-based EKF-SLAM approach.

The above system performs exploration, motion control, and SLAM in an integral system. However, it is only tested in simulations and not in real robots.

Stachniss *et al.* [130] presented an integrated approach that combines autonomous exploration and SLAM by using a FastSLAM grid-based approach [49] using a laser range sensor. The key idea of this work is to use a utility function that identifies opportunities for closing loops while performing FastSLAM during terrain acquisition. This approach also contains two map representations, an occupancy grid and a topological map, that are associated individually with each particle. Once a loop closure is found, the robot follows the trajectory associated with the particle in order to reduce the uncertainty in the robot's pose.

In other work, Fraundorfer *et al.* [41] developed a complete system for mapping, localisation and navigation. This approach is an appearance-based system that represents the environment as a topological map (see Section 2.1.2) where images represent each node in the map. For each image, features are extracted and described, then a vector quantisation algorithm is applied to cluster similar features into visual words. For

image retrieval, after finding a set of image correspondences, a RANSAC procedure checks for geometry consistency by estimating a homography between feature correspondences. To enforce geometry information in the corresponding nodes the essential matrix is computed and rotation and translation information is obtained from it. Navigation and path planning is done in this work by traversing the graph of poses.

There have only been few visual SLAM systems using standard cameras which enable fully autonomous navigation. Davison and Murray's early visual SLAM system based on fixating active stereo [30] was used for real-time position-based navigation, but the feature map generated was too sparse to permit reliable reasoning about free and occupied areas of space. With the advent of feature-based robot SLAM systems with much increased feature density, there have recently been some attempts at performing free-space mapping based on semi-dense point clouds. Notably, systems like [120] offer good possibilities for free-space detection (at least in highly textured areas with many feature points), and enable autonomous navigation and exploration.

The latest feature-based robot SLAM systems have much increased feature density. This has mainly been motivated by the increase in localisation accuracy enabled by using more features, but there have been some attempts at performing free space mapping based on these semi-dense point clouds. Sim and Little [120] presented an approach using trinocular stereo which mapped SIFT features with high enough density to permit the determination of free space in an indoor environment and enable autonomous navigation and exploration. Konolige and Argrawal's binocular more recent stereo FrameSLAM system [70] represents close to the state of the art in vehicle-based feature-based SLAM, and generates hundreds of point correspondences per frame and a substantial 3D point cloud and seems to offer good possibilities for free-space detection, at least in highly textured areas which offer many feature points.

2.4 Computer Vision Based Lightweight Systems

There is an interesting branch of computer vision-based approaches in robotics that is characterised for being simple, robust and lightweight when compared to the more established and complex geometrical methods. We would like to define our concept of 'lightweight' as a system that incorporates some of the following characteristics:

- An easy to understand method which is easy to implement and modify.
- Little requirement for calibration.
- A 'black-box' method, which can be used without knowledge of internals.
- Computational efficiency.

- Modularity: can easily be combined with other techniques.

Typically, a lightweight system comprise mechanical odometry or simple visual odometry for local motion estimation; appearance-based loop closure detection using either whole image statistics or invariant feature matching; and some type of efficient pose graph relaxation. An interesting example is the work developed by Andreasson, etal [5], its 'minimalistic' concept is very related to what we look in a lightweight system. It is a visual SLAM approach that uses omnidirectional images to represent nodes in a graph, and which edges contain odometry and visual relation between nodes. To compute the visual relations, SIFT features are extracted and a similarity measure is computed. Because there is no multiple view geometry involved, an estimate of the displacement between correspondent nodes is computed by matching features between the omnidirectional images. To estimate the relative position between two frames, geometric information is obtained from the image similarity of the surrounding images encoded in a covariance matrix. Even though features are extracted every frame, their position is not estimated which contrast with other SLAM approaches, this which allows the algorithm to be more computationally tractable. To optimise the SLAM map, a multilevel relaxation algorithm [42] is used.

A recently lightweight VO approach is presented in [93], a single camera is used mounted in the top of a car travelling at speeds of up to 60 km/hr. The system is termed RatSLAM in analogy to how rodents performs localisation. In section 5.1 we will provide detailed information about this method.

Autonomous driving and exploration in a mobile robot is a task that has been achieved by few using computer vision as the main sensor. Santosh [117] presented an approach for localisation, navigation and image based exploration uses a single pan-til camera. The method assumes a calibrated camera and the knowledge of some extrinsic parameters of the camera relative to the robot. The basic idea is to segment the scene between floor and obstacle, the segmentation is done by obtaining a model of the floor region by computing a HSV histogram. The model is initialised assuming that a small region in front of the robot is free of obstacles. The region is further extended by comparing the texture of the boundary regions. By knowing the camera's internal parameters, the height from the camera centre to the ground plane and the horizontal angle, the distance to the obstacles can be computed from the segmentation already available. A topological map is built by saving images and its corresponding SIFT features on different poses along the trajectory. Epipolar geometry is used to obtain the extrinsics of two different views up to scale, the solutions are disambiguated by making use of the robot's odometry. SIFT features are used also for visual servoing. Path planning is done by traversing the shortest trajectory in the graph of poses.

One of the problems of having a map of features is its further usability to its sparse representation. An interesting work is presented by Valencia [142] where they use a computed map for path planning using the uncertainty estimated during the mapping execution. A graph of poses is built by an approximation coined Pose SLAM [56], here only the robot trajectory is estimated and features are used as constrains between robot poses. The assumption is that all the trajectory built by the robot is free of obstacles at any point in time and therefore it can be traversed. The idea is to search for a path that has the minimum accumulated uncertainty to reach a destination node. The search is done also in the neighbouring nodes. If a neighbouring node has less uncertainty in the pose than some node in the current computed trajectory then a kinematic consistency check is done to see if it is feasible to reach. It is true that reaching a neighbour in certain cases will not be collision-free. However, authors claimed that those cases are easily detectable.

In this Chapter we have tried to provide the most usefull and general information related to the background of this thesis. This information includes theory, relative work and state of the art methods which we believe are important for a good understanding of our work. However, on each of the subsequent Chapters when needed, a more detailed information is also provided.

3

Experimental Mobile Robot System and Tools

The aim of this thesis is to explore and develop computer vision algorithms for mobile robots with multiple cameras. Experimental validation using a real robot platform has always been at the core of our aims. However, since this is a thesis predominantly about algorithms and software rather than mechanical details, it was desirable to make use of robot platforms for experimentation where the 'hardware overhead' of dealing with a real robot was minimised and efforts could always be focused on the core vision issues. In this Chapter we describe the hardware, interfacing and software platform which was put together specifically to support the research in this thesis. The key design aims to be:

- Ease of development and programming.
- Reliability for extended and repeated experimentation.
- Reconfigurability, particularly with regard to flexible camera placement.

For these reasons, the hardware platforms considered consisted of off the shelf components, and as far as possible widely used software components.

3.1 System Requirements

The approach followed by this thesis was to keep the system as simple and general as possible, making it practical and reliable by using the robot as a plug and play device avoiding robot-hardware programming and giving full responsibility for decision making to our processing system.

The experimental set up can be broken into three main areas:

- Robot vehicle platform,
- System coordination and processing,
- Vision acquisition.

3.1.1 Robot vehicle platform

During the first stage of this research (see Chapter 5), the robotic platform *Segway Robotic Mobility Platform (RMP) 100* (see Figure 3.1) was used. This is a robot which has only two supporting wheels and stabilises itself dynamically making use of its internal sensors. During the early stage, the first work was to develop a visual odometry system (see Chapter 5) using this platform, the robot's motion was controlled completely with a wireless joystick.

Figure 3.1: Segway RMP 100.

The Segway RMP is based on the self-balancing Seaway human Transporter (HT). It is capable of high speed ($\sim 13 km/hr$) indoor and outdoor operation. One of its advantages is that it can rotate on the spot (Z axis) making a turn of zero radius. Its self-balancing feature can introduce a rotation in the Y axis of $\pm 45 degrees$ (see Figure 3.2). It has two interfaces for input/output communication (USB and CANbus) and there is no need to download anything into the robot's hardware, the software interface is via a high level API.

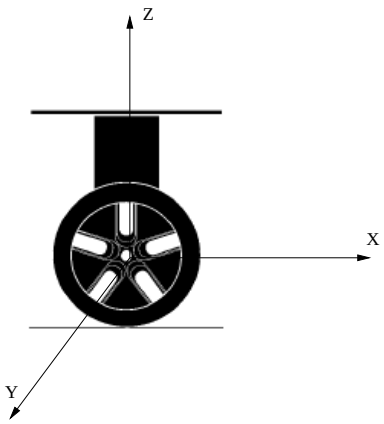


Figure 3.2: Segway coordinate frame: X (roll), Y (pitch), Z(yaw).

The Segway is a complex robot which weights around 60 kg and is dangerous to operate in small places, making it suitable for long or outdoor areas. For that reason we have chosen to have an experimental platform which is reliable and broadly used in the robotics community. We chose the Pioneer P3-DX from Adept MobileRobots (see Figure 3.3).

Figure 3.3: Experimental Mobile Robotic Platform Pioneer P3-DX.

The P3-DX arrives with a ring of 8 forward sonars and with an optional 8 rear sonars. It can reach speeds of 1.6 meters per second and carry a payload of up to 23kg. In order to maintain accurate dead reckoning data at these speeds, the Pioneer uses 500 tick encoders, which for our navigation purposes are fairly accurate. It offers the option of an embedded computer and different sensors such as bumpers, vision, laser range finders and compass. We decided to use the most basic platform and therefore use our own computer and vision sensors, with the idea of making the algorithms developed in this work more adaptable to a range of platforms.

Efforts have been made to develop algorithms which are not robot-dependent and that in practise could be incorporated into any robot system. The algorithms developed so far only need feedback from the robot coming from odometry and no other sensor is used but computer vision.

One of the advantages of the P3-DX platform is that it comes with software which

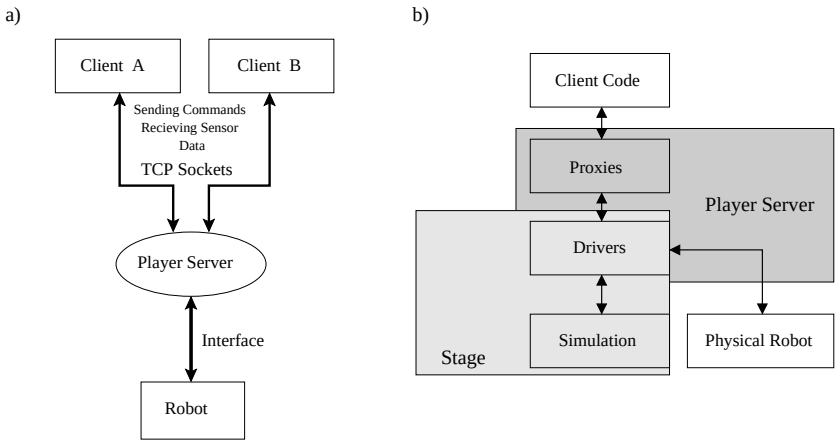


Figure 3.4: Player/Stage Architecture: a) Player high-level diagram. b) Player/Stage interconnection.

can be used right out of the box, allowing the user to spend time writing software for a specific application rather than writing code for the robot itself. If this was not the case, then there are also different opensource tools such as *Player and Stage* [111] or ROS (Robot Operating System) [135] that can also be used.

Player is a set of free software tools that provides a straightforward interface for the manipulation of a wide variety of robots and sensors. It is probably the most popular open source robot interface. It uses a client/server model with TCP sockets allowing developers to write code in several programming languages. Its several layers of abstraction allow the user to develop modular code by using set of pre-defined interfaces which are basically low-level hardware-specific implementations (see Figure 3.4(a)).

Because manipulating robots in a real environment is not always possible and sometimes debugging can be risky when doing it on the physical robot, *Player/Stage* provides the tools to simulate the code written as it was tested on the physical robot. An image illustrating the architecture is shown in Figure 3.4(b). The code that works on the simulator should work on the real robot without any modifications. A world can be created manually in a separate file with a specific set of instructions and then loaded and visualised on Stage (see Figure 3.5).

Player is designed to be language and platform independent. The client program can run on any machine that has a network connection to the robot and the code can be written in any language that supports TCP sockets. It allows the connection of multiple clients making possible a distributed sensing control.

More recently, Willow Garage and the University of Stanford developed an open-source platform for robotic applications called ROS. It is structured in such a manner that provides an Operating System (OS) for Robots. In this context normal services that are present in an OS are also present here. Within its capabilities, it provides hard-

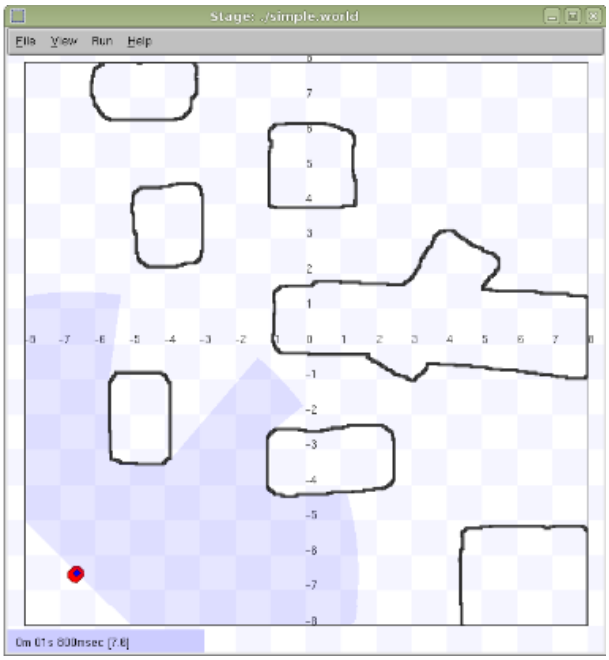


Figure 3.5: Stage's simulation display interface (Simple.cfg player example).

ware abstraction, device drivers, libraries, visualisers, message-passing and package management. The main idea behind ROS being also to allow a platform for reusable code, it supports a federated system of code repositories that enable collaboration to be distributed as well. It contains a wide variety of algorithm implementations. One of the disadvantages at the moment is that it only runs on Unix-based platforms. ROS is a massive project that support interfaces for the incorporation of other robotic platforms such as Player and Stage. Another advantage is that OpenCV and ROS are supported by the same company so it is expected to see a vast usage of OpenCV within ROS.

Have considered both of these two main robot development platforms, some differing advantages can be noticed between the two. ROS is probably a more powerful and complete package but also it is more complex and because it has been released recently, there is a steeper learning curve.

Sometimes the use of these platforms facilitates interaction with robots and allows the development of applications faster, and during the work of this thesis we started trying and developing some algorithms using Player. After some time however it was noted that the only external software needed for the Pioneer robot was the driver. Since the main objective of this work is to develop computer vision algorithms then the only data exchanged with the robot from the main computer is as input, motion commands and as output, odometry information. In this sense, this computer vision framework is more generalisable and is not dependant on any software robotic platform. The main

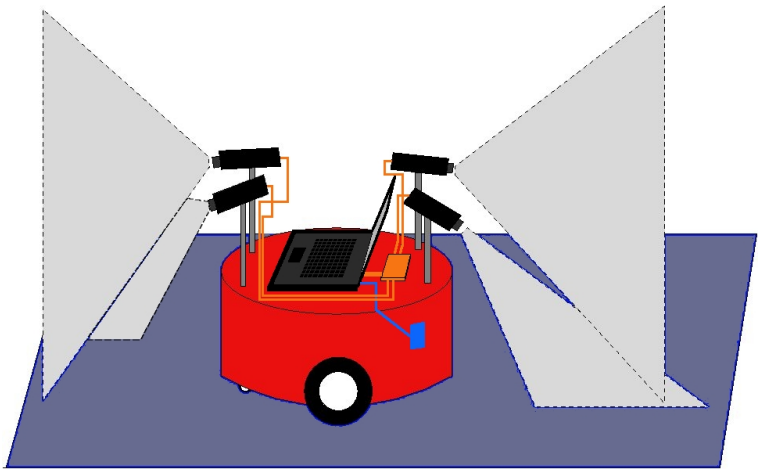


Figure 3.6: Robot platform: cameras are connected to a hub which is connected to a PC using a single bus. Algorithms and control are computed on the PC in a manner such that the only information sent to the robot is motion commands and the only information the laptop receives is wheel odometry.

development in this thesis therefore used custom software and did not rely on either the main two platforms. We would agree that this comes with some disadvantages such as the inability to use an integrated simulator but because we are working with multiple cameras it is also complicated to simulate this data.

3.1.2 System coordination and processing

We have adopted a centralised control framework where all the algorithms were computed in a single laptop mounted on top of the robot. With data acquired from the cameras, decisions must be made about the future behaviour of the robot's motion. The laptop then transmits a series of motion commands to the robot and also acquires odometry information every time it is needed (see Figure 3.6).

All algorithms were developed in Linux (Ubuntu) due to the advantages it presents in terms of easy adaptation or modification of open source code when required and the increasing amount of open source tools that exists nowadays in the robotics community.

The algorithms implemented on this thesis were written in C++ using libraries such as: OpenCV [45], libCVD and TooN [104], and our own library robotvision [2].

OpenCV is a cross-platform library initially from Intel Research and now fully supported by Willow Garage. It focuses on image processing and computer vision. Its extensive collection of methods, ease of use and constant support make it an excellent tool to use. OpenCV is divided into different categories that include state of the art methods. These categories include: image processing, feature detection, camera

calibration and machine learning.

CVD is a very portable and high performance C++ library for computer vision, image and video processing. Particularly this library is used in this thesis to provide simple and efficient image and video handling. The video grabbing module provides a simple, uniform interface for videos from a variety of sources (live and recorded) and allows easy access to the raw pixel data.

TooN is a C++ numeric library specifically designed to operate with matrices. Is a library that handles large and small matrices very efficiently, and includes static type safety features for matrix operations. It sits on top of very powerful libraries such as Lapack and Blas. Another important feature is that it integrates very well with CVD.

Librobotvision is the internal library developed by the members of the Robot Vision Research Group at Imperial College which relies on CVD, TooN and other libraries such as Boost and Lapack. Is a very user friendly library that provides different interfaces to interact with those libraries and high level methods.

For display purposes, different GUI's (Graphical User Interfaces) were developed using Qt. Qt has different qualities which makes it adequate and is also a standard in the Robot Vision Research Group at Imperial College. Qt is a cross-platform application framework with API's for C++ that allows fast UI creation and debugging. Another advantage is its own IDE (Integrated Development Environment) QtCreator, which is very lightweight and given that it is tailored for Qt, it speeds up debugging and facilitates the creation of projects.

3.1.3 Vision acquisition

At the beginning of this research an important decision was the selection of suitable cameras and lenses to fit our purposes. When selecting a camera for computer vision or robotics there are different features that have to be taken into account. Some of them are:

- Image Scanning. Broadly speaking a classification that is commonly used is: *Interlaced Camera*, a camera that reads the image in two separate fields: odd and even lines separated by a 40ms time interval. This causes splitting in the final fused image. *Progressive Area Scan*, where the camera reads the image as a whole reading all lines with the same scan resulting in an image with no splitting. *Line Scan Cameras*, these cameras have a line image sensor (a row of pixels) that increases the image acquisition speed, make them more suitable for fast moving objects.

- Image Colour. There are two types of sensor: Monochrome and Colour. A monochrome sensor has low power consumption with the highest resolution.

There are two types of colour camera: one is a monochrome sensor with a colour filter; this causes a degradation of the resolution of the image due to the interpolation occurring inside the filter; the second one is a colour system that has three dedicated sensors for the primary colours (RGB) obtaining a resolution that rivals the monochrome one but at the cost of more power consumption and a more costly device.

- Image Sensor. *Charge Coupled Device (CCD)* cameras have 100% active sensor usage, meaning that all of the pixels can be devoted to light capture obtaining a more uniform image; therefore it obtains a more sensitive, less noisy image at the cost of more power consumption. For low contrast images these sensors are ideal. In *Complementary Metal Oxide Semiconductor (CMOS)*, each pixel has its own charge to voltage conversion (less uniformity in the whole image), and the sensor includes other circuits such as: amplifiers, noise suppressors and digitalises. This reduces the area available for light capturing which is normally 70%. In the early days, the idea was that CCDs have better image quality but more power consumption and with CMOS we obtain the opposite. However, nowadays that manufacturers are trying to fill the gaps between the two types, great quality images with low power consumption can be obtained from both.

- Output type. *USB* is a standard for peripheral devices which is convenient and compatible between multiple computers. USB 2.0 has a maximum transfer rate of 480 Mbit/s. However, until recently most cameras available have less flexibility in the optics, most of them having non-changeable lenses. *Firewire IEEE 1394*. There are 2 standards: IEEE 1394a with a maximum transfer speed of around 400Mbps, and IEEE 1394b with a maximum transfer speed of around 800Mbps. These ports are not as common as USB, but in general most personal computers at the moment have both ports. IEEE 1394b provides more theoretical speed than USB 2.0 and therefore is more suitable for high bandwidth devices such as video cameras. Most firewire cameras are fitted with a lens based on the C-mount or CS-mount standard providing a wide range of lenses. Therefore, these cameras can be used for different applications regarding the optics needed. *Gig-E*. Gigabit Ethernet is relatively new. It is increasing in importance mainly due to its fast transmission speed (1000 Mbit/s) and flexibility, they also offer CCD and CMOS sensors. However, a Gig-E card is needed, and in many computers it is not provided by default.

Considering these factors, the decision on the camera acquisition type was based on the idea of having more general cameras that can be adapted to different needs. These are: more resolution and quality in the images, different field of views as well,

Figure 3.7: Point Grey Flea2 Camera.

speed in the acquisition and an output connection that can be used in most computer devices without the need for extra hardware. A rig of four Point Grey Flea2 IEEE 1394 cameras (see Figure 3.7) were therefore used with the following characteristics:

- *Resolution*: 1296x964 Colour, C-mount,
- *Maximum Frame Rate*: 30FPS at 1288x964
- *Dimensions*: 29mm x 29mm x 30mm (excluding lens holder, without optics)
- *Image Data Formats*: Y8, Y16 (all models), RGB, YUV411, YUV422, YUV444, 8-bit and 16-bit raw Bayer data (colour models).
- *Synchronisation*: Multiple Flea2 cameras networked on the same IEEE-1394 bus are automatically synchronised to within $125\mu s$ (maximum deviation) of each other.

These cameras have the advantage of acquiring high resolution images which could be useful when a high detailed analysis of the scene is needed. However, in our work we try to keep the resolution low (320 or 640) because of the lightweight nature of our approaches.

When working with multiple cameras, synchronisation plays an important role. This can be achieved by external triggering hardware or in some cases they automatically synchronise when connected to the same bus.

Assuming the cameras are synchronised, it was important to make an analysis of how many cameras could be used at the same time when connected on the same bus.

Based on the IEEE 1394b bandwidth (800 Mbps), a theoretical comparison can be obtained on the framerate and image resolution on a camera rig (see Table 3.1).

One of the ideas about using a multicamera rig is also to have the ability to obtain different field of views. In our application we decided to focus on trying to cover as much as the scene as possible. Therefore an important task was the one of choosing the

		Image Resolution				
Frame Rate	Cameras	320x240	640x480	800x600	1024x768	1280x960
3.75	2	13.824	55.296	86.4	141.558	221.184
	3	20.736	82.944	129.6	212.337	331.776
	4	27.648	110.592	172.8	283.116	442.368
7.5	2	27.648	110.592	172.8	283.116	442.368
	3	41.472	165.888	259.2	424.673	663.552
	4	55.296	221.184	345.6	566.231	884.736
15	2	55.296	221.184	345.6	566.231	884.736
	3	82.944	331.776	518.4	849.347	1327.104
	4	110.592	442.368	691.2	1132.462	1769.472
30	2	110.592	442.368	691.2	1132.462	1769.472
	3	165.888	663.552	1036.8	1698.693	2654.208
	4	221.184	884.736	1382.4	2264.924	3538.944

Table 3.1: Evaluation of Camera Framerate vs Image Resolution, on colour images. Red indicates that the bus bandwidth has been exceeded

most suitable wide angle lenses. the characteristics that have to be taken into account are [57]:

- *Focal Length*. This is a measure of how strongly the system converges or diverges light. It is the distance between the lens and the principal focal point (point where all the ray lights converge). In a few words, it determines the angle of view; wide angle lenses have a short focal length (see Figure 3.8a).
- *Aperture*. This refers to the amount of the lens that can be open up or close down to let light passing trough and it is measured in f-numbers. An f-number is a ratio between the diameter of the aperture and the focal length. For instance, on a 50mm lens, f/4 indicates that the diameter of the aperture is 12.5mm (see Figure 3.8b).
- *Depth of Field (DOF)*. This is the distance between the nearest and farthest objects in a scene that still appear sharp in an image.
- *Other features*. There are other important features such as: manual or fixed iris and focus and zoom.

A broad variety of lenses are commercially available. However, most of the high quality wide angle lenses are expensive and developed mostly for security applications. Another factor was that most of the wide angle lenses are developed for CS-mount given that the lenses are positioned 5mm closer to the CMOS sensor or CCD sensor (for a wide angle lens a short focal length is required, see Figure 3.8).

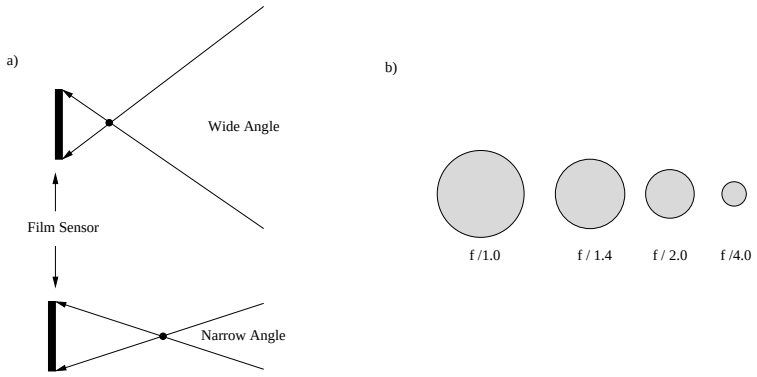


Figure 3.8: Diagram of some lens features: a)Focal length and angle of view, b) Aperture. [57].

Figure 3.9: Lens used in our cameras, Model L-SV-03514: 3.5mm, F1.4 CCTV, 1/2in format, C- mount, HAV of c.80deg for 1/3" sensor.

Taking into account our requirement of a non expensive robot framework but also trying to obtain high quality optics (see Figure 3.9), the characteristics of the lenses chosen for our cameras are the following:

- 3.5mm / F1.4 CCTV.
- Locking focus and iris.
- 1/2in format, C-mount.
- Horizontal FOV of 80° for 1/3" sensor.

In this chapter we have reviewed the overall structure of the hardware and software used for the development of the work presented in this thesis. In the subsequent chapters the reader can assume that these tools were used for the implementation and experimentation of the systems developed.

4

Automatic Extrinsic Calibration of a Multi-Camera Rig

4.1 Introduction

This thesis is intended to address some of the problems related to autonomous navigation with a robot equipped with multiple cameras. In this sense, we investigate the posibilities and capabilities that a robot can obtain by having such a system and we aim to investigate methods in two different paradigms. The first approach is the more standard route of exploiting image geometry as much as possible in order to obtain metrically consistent maps. To do this, we need to know precisely the position of the cameras related to each other and therefore obtain the calibration parameters for the whole rig. The second approach is to relax the constraints of the whole system in a more lightweight methodology.

In this Chapter we present a general method for fully automatic extrinsic auto-calibration of a fixed multi camera rig, with no requirement for calibration patterns or other infrastructure, which works even in the case where the cameras have completely non-overlapping views. The robot makes a set of programmed movements including

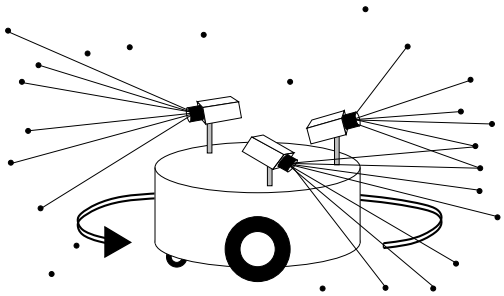


Figure 4.1: Our goal is automatic extrinsic calibration of an arbitrary set of cameras mounted rigidly on a mobile robot and without external infrastructure. The cameras may or may not have overlapping fields of view, but will observe common parts of the surrounding scene as the robot makes a loopy motion including 360° rotation. We make monocular SLAM maps of the invariant feature points observed individually by each camera, match features robustly, perform 3D alignment to fuse these into a single joint scene map, and globally optimise enforcing rig rigidity.

a full rotation in an arbitrary environment and captures an image sequence from each camera. These sequences are processed individually with a monocular visual SLAM algorithm. The resulting maps are matched and fused robustly based on corresponding invariant features, and then all estimates are optimised in a full joint bundle adjustment, where we constrain the relative poses of the cameras to be fixed. We present results showing accurate performance of the method for various two and four camera configurations.

It is now widely accepted that cameras are highly suitable sensors for providing mobile robots with the situational awareness they need to navigate and operate autonomously in the real world. Although the geometrical shape of the scene around a robot does not immediately pop out of cameras in the same way as from a depth sensor such as a laser range-finder, it has been convincingly shown that with appropriate processing they can be used both to estimate motion accurately and recover detailed 3D geometric information about a scene (e.g. [70, 87]). This seems to be only the beginning of the role cameras can play in robotics, however, since ongoing progress in computer vision indicates that the detailed photometric information they offer has almost unlimited potential for problems such as dense volumetric reconstruction and object recognition.

A clearly desirable characteristic of a sensing system for situational awareness is a wide field of view, and the advantages of wide-angle vision have been proven for instance both in geometric SLAM [28] and appearance-based place recognition [25]. Wide angle vision can be achieved by single cameras with special optics such as fish-eye lenses or catadioptric systems, which can sometimes be elegant and practical. However, all such cameras will suffer from relatively poor angular resolution, since

the pixels available must be spread thinly. Another alternative is an active camera system, with a mechatronic platform providing pan/tilt rotation of a narrow field of view camera; but this option suffers from the serious disadvantage that views of highly separated angles cannot be simultaneous and the robot loses the 'eyes in the back of its head' true omnidirectional vision provides.

A multi-camera rig will in general not be fully equivalent to a single wide-angle camera in terms of imaging geometry since it will have multiple optical centres, but this is not a problem for applications such as navigation as long as the locations of those optical centres is known, and may even be an advantage in inferring depth information. Indeed, a multi-camera rig is fully reconfigurable and can be set up to provide the field of view needed, and if necessary regions where camera views overlap for stereo observation.

For a multi-camera rig to be useful to a mobile robot, it is critical to know the relative poses of the cameras. The aim of this work is therefore to propose an automatic procedure to achieve this extrinsic calibration of the rig, by using SLAM techniques as a robot makes a small set of programmed motions through an unstructured scene. Multi-camera rigs, particularly those involving cheap cameras and mounts, some of which may be located in extruding locations on the robot, are prone to moving about as the result of vibrations or knocks. While it would be ultimately desirable to detect and correct this online during the robot's normal operation, the next best this is to have an automatic calibration procedure which can easily be run in a few minutes whenever needed. Note that commercial multi-camera systems such as the Point Grey Ladybug, which are factory-calibrated, feature expensive, heavy and bulky rigid casings to ensure that the calibrated configuration does not change once the rig is out in the real world.

Our method (outlined briefly in Figure 4.1):

- Is fully automatic, needing no human intervention between starting the robot motion, image capture and when the calibration result is returned.
- Requires no calibration patterns or *a priori* known scene structure.
- Solves primarily for the relative poses (up to scale) of the cameras, which can be arbitrarily located and oriented in 3D on the robot's structure.
- Currently assumes that individual camera intrinsic parameters are known *a priori*.
- Assumes that the multiple cameras capture synchronised images, but can otherwise have the same or varying types and optics.

Before explaining the details of our method, we will review camera calibration theory and methods which are the background to the first main contribution of our work.

4.2 Camera Calibration Background and Related Work

When working with computer vision and mobile robotics, the task of camera calibration is necessary to obtain a geometric relationship between the world and the camera plane. This is important in several applications that require a scene to be measured, such as architecture, art or the video game industry. In robotics, we believe it is important to analyse and understand the advantages of using calibrated or uncalibrated cameras. Given that this thesis is about computer vision techniques using a rig of cameras, the first task to solve is the one of calibration.

4.2.1 Camera Calibration

In robotics, computer vision not only provides a photometric description of the world but also it can provide geometric information about the scene. For instance, 3D reconstruction is an interesting area of research that has many interesting applications such as building virtual environments and in general it provides the richest information possible about the world. It was been proven that by using photogrametry tools and knowing the internal parameters of the camera (intrinsic calibration), metric information can be obtained [123]. In the absence of the internal parameters, reconstruction can be achieved up to a projective transformation [36, 52]. Another interesting area in computer vision where geometry plays an important role is Augmented Reality (AR) [7] where the real world is augmented or superimposed with 3D virtual objects. This can be achieved by knowing the camera position and the structure of the scene.

Camera calibration can be defined as finding the camera parameters that affect the imaging process and that allow one to obtain relationships between the image plane and the three dimensional world. In this Section, will give a description of the necessary tools to have an understanding of the methodology behind camera calibration as well as a description of classical and state of the art methods to achieve this.

Intrinsic calibration involves computing the internal parameters of the camera which are normally (see Figure 4.2):

- *Image centre or principal point* $p = (p_x, p_y)$. Typically this point is not at the centre (width/2, height/2) of the image
- *Focal length* f. This is the distance from the centre of the lens to the principal point.

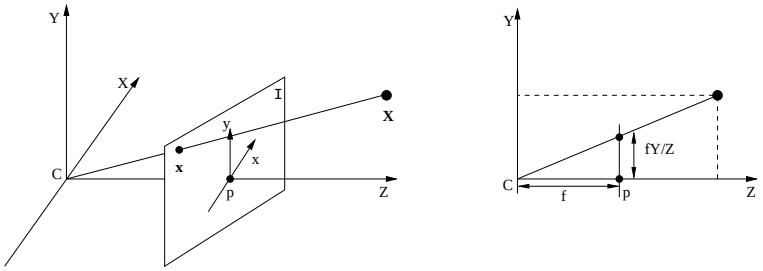


Figure 4.2: Pinhole camera model. *I* represents the image plane

- *Aspect ratio r*. Refers to the relative vertical and horizontal scaling of the image.
- *Skew k*. Is a factor that depends on the angle between the vertical and horizontal axis.
- *Lens distortion*. This is usually modelled by radial distortion (which is due to the spherical shape of the lens) and sometimes an additional tangential distortion component (normally due to bad positioning of the lens).

While extrinsic calibration involves the computation of a rotation matrix **R** and a translation vector **t** that relates the camera frame with the world coordinate frame.

The pinhole camera model

This model represents the projection of a 3D point **X** in the world coordinate frame to a point **x** in the image plane. This is achieved by obtaining a 3×4 projection matrix **P** which is defined by the internal camera parameters **K** and the position of the camera relative to the world described by a rotation matrix **R** and a translation vector **t**

$$\mathbf{x} = \mathbf{P}\mathbf{X} = \mathbf{K}(\mathbf{R}|\mathbf{t})\mathbf{X} \, . \tag{4.1}$$

The matrix **K** is defined as:

$$\mathbf{K} = \begin{pmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{pmatrix} , \tag{4.2}$$

where $\mathbf{f} = (f_x, f_y)$ is the focal length, $\mathbf{p} = (p_x, p_y)$ is the principal point and the parameter s is the skew which in usually zero for most normal cameras.

As we can observe in image 4.2, the principal point p is the intersection of image plane I and the principal axis which is perpendicular to the image plane, in the Figure is located in the Z axis. A point in space $\mathbf{X} = (X, Y, Z)$ can be mapped to the image

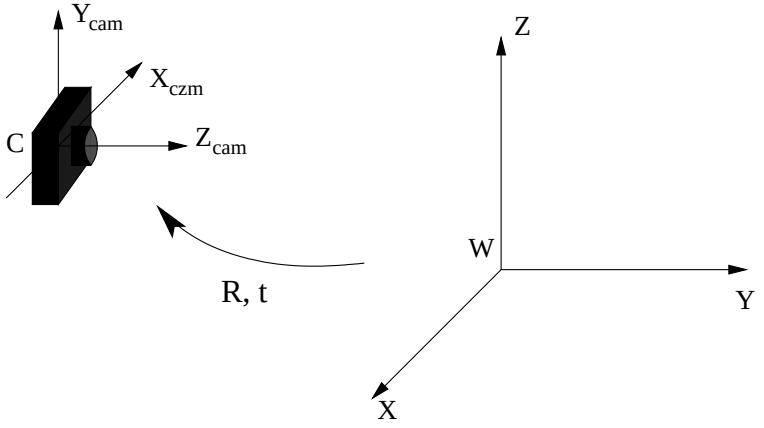


Figure 4.3: Euclidean transformation between camera and world frames

plane by the line that joins the centre of projection C and the image plane. From the image it can be observed that $(X,Y,Z)^T \rightarrow (f_x X/Z, f_y Y/Z)^T$.

As mentioned previously, the rotation matrix **R** and the translation vector **t** is a transformation that relates the camera frame with the world coordinate frame (see Figure 4.3) and are normally known as the extrinsic parameters.

4.2.2 Related Work on Intrinsic and Extrinsic Calibration

Intrinsic calibration methods aim to obtain the matrix of internal parameters **K** which defines the internal geometry of the camera and defines the relationship between camera-centric coordinates and the image coordinates. Extrinsic parameters refer to the position of a camera in the world; the relationship between the world frame and the camera frame and consist of a six degree of freedom transformation: three for rotation and three for translation (see Figure 4.3). Therefore, *intrinsic* calibration of a single camera is the task of estimating the internal parameters specifying its operation, normally consisting of focal length, aspect ratio, principal point and radial distortion coefficients, while *extrinsic* calibration means determining the relative poses (3D translation and rotation) of a set of rigidly connected cameras.

Classic methods for intrinsic calibration make use of calibration patterns to provide external metric information [140, 133, 149] and are well-tested and accurate. However, it is also possible to *auto-calibrate* camera intrinsics without a pattern if additional assumptions are made, most usually that several images of a rigid scene taken by a camera with unchanging parameters are available and between which correspondences can be obtained. Under such conditions, enough constraints are available from image feature measurements to uniquely determine the calibration constants, and there has been much work on this topic in the computer vision community.

For instance, Pollefeys *et al.*[107] solved for intrinsic camera parameters as part of a complete off-line structure from motion algorithm. A more recent approach using SLAM-based probabilistic filtering allowed sequential auto-calibration estimates of camera internal parameters to be refined sequentially [21].

A distinction can be made between the different approaches to solve calibration problems. In this Section a revision of two major branches is given: classical and auto-calibration methods. Inside the auto-calibration methods we can also find those approaches that assume fixed internal parameters and others that relax this assumption with varying internal parameters.

Classical methods involve the use of known points in objects or calibration patterns to infer external and internal parameters. Perhaps the first methods that uses a set of control points whose coordinates are known and fixed to a rigid frame is the Direct Linear Transformation first introduced by Abdel-Aziz [3]. A calibration pattern provides a relationship between points in the image and points in the world, this idea was successfully developed by Tsai [140]. Tsai's method recovers the intrinsic parameters (including distortion coefficients) and the extrinsic parameters. The method works first by obtaining the correspondence between points in a calibration target (usually a chessboard-type pattern) and the image points, then it looks to estimate as many parameters as possible running a linear least-squares minimisation, the parameters estimates found on this step are further used as an initial guess on a final non-linear optimisation.

Before introducing auto-calibration methods it is important to talk briefly about some terminology in epipolar geometry. Epipolar geometry describes the intrinsic projective geometry between two views and encapsulates it in a 3×3 matrix $\mathbf{F}$ called the fundamental matrix satisfying the relation $x'^T \mathbf{F} x = 0$ where x is a point in the first image and x' is the same point in a second image. The most important concepts are [52]:

- *The Epipole* is the point of intersection of the line joining the camera centres with the image plane.

- *An Epipolar Plane* is any plane containing the baseline.

- *An Epipolar line* is the intersection of the epipolar plane with the image plane. Therefore, all the epipolar lines intersect at the epipole. Specifically $l_e^{'} = \mathbf{F}x$ and $l_e = \mathbf{F}^T x^{'}$.

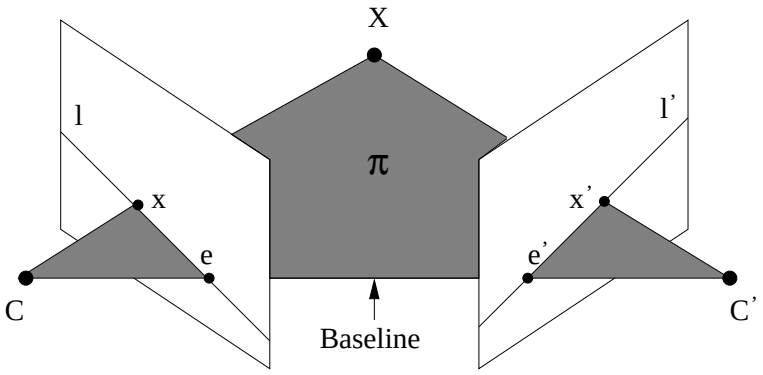


Figure 4.4: Epipolar Geometry Diagram .

4.2.3 Related Work on Multi-Camera Calibration Methods

Now that cameras and their supporting electronics are getting cheaper and smaller, mounting multiple cameras around a robot in a ring or other configuration is highly feasible, clearly offering the potential for both a wide field of view and high angular resolution.

Multi-camera systems have increased in popularity in different areas such as robotics, human tracking or virtual and augmented reality. There are different works using multi-camera environments to create 3-D environments. For instance, a room equipped with fifteen cameras to capture 3D content is presented in [110], this room is used to detect a user inside the environment and create augmented reality scenarios by detecting a head-mounted camera and trackers. Calibration is done by traditional approaches were a chessboard pattern is presented to the cameras and the extrinsics are recovered with OpenCV or Matlab calibration libraries. Another approach is to manually create a very distinguished feature that is detected by all cameras in a number of frames during a video sequence, in [134], a bright spot is created by a laser pointer which is waved on a working volume and detected by all cameras.

To recover the extrinsic parameters of a multi-camera system, an interesting solution is presented in [76]. The authors propose to match and fit tracked objects centroids to a planar model by obtaining an homography between camera pairs with respect to the ground plane, then the 3D relative positions are recovered from the homographies. A ground plane and a partially overlapping setup assumptions are made.

Work on extrinsic calibration has also focused on the case of stereo rigs, designed such that two or sometimes more cameras have highly overlapping views and simultaneously observe mostly the same parts of a scene. Calibration methods based on external metric targets are relatively straightforward to formulate for this problem, allowing both intrinsic and extrinsic calibration to be unified (e.g. [37]), and are there-

fore prevalent practically.

The goal we target is to obtain the extrinsic parameters of a camera rig in the general case of non-overlapping views.

Relevant to the goal of our work on extrinsic calibration are methods which can perform extrinsic auto-calibration of stereo rigs presented with general, unknown scenes. For instance, extending the approach of [140], Luong and Faugeras [84] proposed to calibrate a stereo rig using point correspondences between the two cameras when the system undergoes a series of displacements in an *a priori* unknown rigid scene. The resulting estimated extrinsic parameters are determined up to a scale factor. Similar work on solving the stereo-camera calibration problem using correspondences information includes [54, 65]. Another interesting approach is also presented in [126] where a central EKF-SLAM filter fuses the information coming from a stereo camera rig to obtain the extrinsic parameters.

There has been relatively little work on auto-calibration in the case we are considering in this work, multi-camera rigs designed primarily to provide a wide field of view and which may therefore have little or even no image overlap. In fact, even using metric patterns to calibrate such rigs is challenging because the patterns must accurately surround the whole rig in the form of a 'calibration room' or similar. Different solutions have been proposed to achieve such behaviour. For instance, in order to accurately calibrate an eight camera rig for SLAM purposes, Kaess and Dellaert [58] designed a semi-automatic procedure which involved placing the robot in a special scene with calibration targets on three walls and capturing a number of images manually before feeding these to automatic target matching and optimisation procedures. A complicated semi-automatic procedure is also proposed in [55], where a calibration grid is presented in front of each camera individually, at several positions, while a laser measurement system is used to determine the 3D coordinates of the grid's corners. From this geometrical information, and taking advantage of point correspondences over images of the grid in the different cameras, calibration parameters can be determined easily.

The two above approaches require significant user intervention to obtain the calibration parameters, thus degrading their autonomy. A simpler yet still semi-automatic solution is the one proposed by Li *et al.* [78], where calibration can be achieved in unprepared natural or man-made scenes. A three-step procedure sequentially determines the centre of distortion of the setup (assuming a single unique optic centre for all the cameras), the individual intrinsic parameters of each camera, and the relative camera orientations (under the unique optic centre assumption, relative camera poses differ only by a rotation). The procedure however still requires user intervention, with the manual specification of point correspondences between the views of two neighboring

cameras.

In one more recent solution [72] a planar mirror is used to make a single calibration object visible to all the cameras. However, even though the method allows for both intrinsic and extrinsic calibration, the approach would appear inconvenient in practise, especially in the experimental setup we are considering here where cameras could be mounted all around the robot, making it difficult to observe the same object in all the cameras at the same time.

A closely related approach to our work is presented in [73], the relative orientation of a set of images taken with a calibrated camera is obtained by first extracting and matching SIFT features, then a RANSAC procedure estimates relative orientations between pairs of images which are later integrated into a bundle ajustment refinement to optimally estimate all unknown orientation parameters.

Another work we have found which is similar to our current approach is that by Esquivel *et al.* [35], who explicitly tackle auto-calibration of a non-overlapping rig and present some limited results obtained using real image sequences. They perform individual structure from motion computation for each camera separately, determining a trajectory for each as the rig makes a movement. These trajectories are then locally aligned in 3D and the transformation estimate of the relative camera poses is obtained. A similar method that tackles the problem by tracking a moving object and matching trajectories is presented in [6]. Also, in [105] a system with two wearable forward and backward facing stereo is developed. The individual stereo rigs are calibrated in a standard way using a calibration pattern. The relative calibration between the rigs is achieved automatically via trajectory matching.

These techniques are more limited since they rely only on trajectory matching rather than aiming to build a globally consistent feature map within which the multi-camera rig is located as our method does. Our method, via its final global optimisation, implicitly takes advantage of the same trajectory matching behaviour, but is able to fully digest all of the other information available.

Another relevant paper is the recent work of Koch and Teller [67], who automatically learn about how feature motion in the images of a multi-camera rig relates to the rig's motion as it is carried by a person, but stop short of aiming to determine full geometric extrinsic calibration.

Finally, the work which we found the closest approach to our methodology, is that by Lebraly [74] [75]. This work was done at the same time as ours. Their approach exploits the rigidity assumtion of the camera rig and presents a framework based on the algorithm proposed by Esquivel [35] where first they obtain the camera trajetories using a standard Structure and Motion algorithm and then those trajectories are used to obtain estimates of the extrinsic parameters. Those estimates are then integrated into

a bundle adjustment scheme that optimises the scene structure, the camera-rig motion and the extrinsic parameters. The points used as the structure of the scene are manually constructed features which are easily detected. This, I could say is the big limitation of this system, the approach is dependant on the use this features. Our method uses natural features.

4.3 Method

Our method in detail consists of the following steps:

1. Cameras are attached to a mobile robot in arbitrary positions and orientations, and locked into place. After this, all the following operations are fully automatic.

2. The robot makes a pre-programmed movement, of duration around 1–2 minutes and involving loopy motion and 360° rotation, capturing synchronised video streams from the cameras as it moves through an unprepared environment.

3. The video sequence from each camera is fed to a modified version of the MonoSLAM algorithm [29]. Individually for each camera, MonoSLAM estimates camera motion and builds a 3D map of visual feature locations up to scale. Each feature is characterized by a SURF descriptor [9].

4. Each full video sequence is decimated regularly into evenly spaced keyframes. Each camera's map and motion estimates are then refined individually using bundle adjustment over these keyframes.

5. Candidate feature correspondences between each pair of individual maps are obtained via thresholded matching between their SURF descriptors.

6. Initial alignment of the maps' relative 3D pose and scale is achieved using 3D similarity alignment and RANSAC [38], with sets of three correspondences between maps derived from the SURF matches used to generate hypotheses. A reliable set of correspondences between each pair of maps, satisfying both descriptor matching and geometrical correspondence, is deduced, and the monocular maps are fused into a single joint map each of whose features has been mapped by one or more cameras.

7. This initial joint map is used as the starting point for full bundle adjustment optimisation to estimate the relative poses of the cameras, 3D positions of scene points and motion of the robot, taking advantage of the synchronisation of the keyframes image streams to impose the rigidity of the multi-camera rig.

We will now explain each of these steps in detail.

4.3.1 Robot Motion and Sequence Capture

The set of cameras whose extrinsic calibration is required are fixed to the robot in the desired configuration, and it commences a set of pre-programed movements, controlled by odometry, of duration around 1–2 minutes. The robot captures a synchronised sequence of images from each camera, typically at a fixed frame-rate.

When the cameras used are of the same type, it is often possible to synchronise capture using capabilities provided by the manufacturer — many IEEE 1394 cameras for instance will sychronise by default when attached to the same bus. In the case that cameras of different types or otherwise unsynchronisable cameras were required to be used, a somewhat tedious but still fully automatic procedure to achieve the same effect would be to program the robot to move in a stop-start fashion, making many small motions instead of a single continuous one and stopping to request one image from each of the cameras every time is stopped.

There are no absolute requirements for the type of robot motion used; there are many different types of robot (wheeled, legged, flying...) with different type of movement, and our method is in principle applicable to any of these. However, there are certain characteristics which are desirable. To be practical and straightforward the motions used should be simple and short, but from an estimation standpoint there is a clear advantage to 'loop closing' motions where the robot makes a full rotation while moving. As is well understood in SLAM research, this will remove drift and permit tightly converged individual camera maps. Also, it maximises the chances of many feature correspondences between the multiple cameras since cameras pointing at different horizontal angles will come to view all of the same scene elements (occlusion notwithstanding). Even if there is no overlap between the images simultaneously captured from the different cameras, there will be overlap between the monocular *maps* built. Or put in another way, many of the parts of the scene observed by one camera will also be observed by a second camera, but just not at the same time (see Figure 4.5). In our case the best motion was to perform a forward-backward motion for 1m long each, then turning the robot to the left on the spot for 15 degrees, then turning it right on the spot for 30 degrees, then going back to the left for 45 degrees, then following that loopy sequence of increments of 15 degrees for each side until a complete loop closure is archived.

A more formal evaluation about specific camera motions is studied in [60] [74]. Based on their study they conclude that the set of degenerate or singular motions are:

- Rotations and screw motions (rotation plus translation on the same axis) about an axis, when the axes of all camera positions are all the same
- Pure translations, screw motions and planar motions with parallel axes when the

Figure 4.5: Images of the robot performing a pre-programmed motion for the camera rig calibration procedure.

axes of all camera positions are different.

Our algorithm places no specific requirements on the scene around the robot such as the presence of calibration objects, but there are some favourable characteristics which will lead to easier and more accurate operation. A highly textured scene which offers many trackable features across the whole field of view of the cameras is best, and it will also be helpful if this texture is not overly repetitive to aid unambiguous matching. Many typical indoor scenes are suitable. The depth of the main elements of the scene should ideally also be relatively small compared to the size of the robot's motion in order that parallax can be observed and 3D locations of the features determined. Note that a distant scene would permit only the relative orientations of the cameras to be determined accurately, not their translational positions.

4.3.2 Single Camera Mapping Using MonoSLAM

The first processing step of our method consists of estimating the structure of the scene and the robot's motion from each camera individually. For this we use a slightly modified version of MonoSLAM [29], a sequential 3D visual SLAM algorithm for monocular cameras based on the EKF. MonoSLAM is able to routinely build maps within room-sized domains, including the capability to automatically close loops induced by rotation (Figure 4.6 (top row)).

The first modification is the use of SURF features [9] rather than the Shi-Tomasi image patches MonoSLAM normally relies on. The motivation of this modification is to allow for invariant feature description, which will be crucial for the later step of getting map-to-map correspondences. We choose SURF because of its proven recognition performances and reasonable computational demand.

The second modification is to MonoSLAM's map management policy, also with the aim of improving inter-map matching performance. Instead of the random feature initialisation strategy originally used in MonoSLAM, the image is divided into a regular grid as shown in Figure 4.6. At each frame of the sequence, if one of the cells of

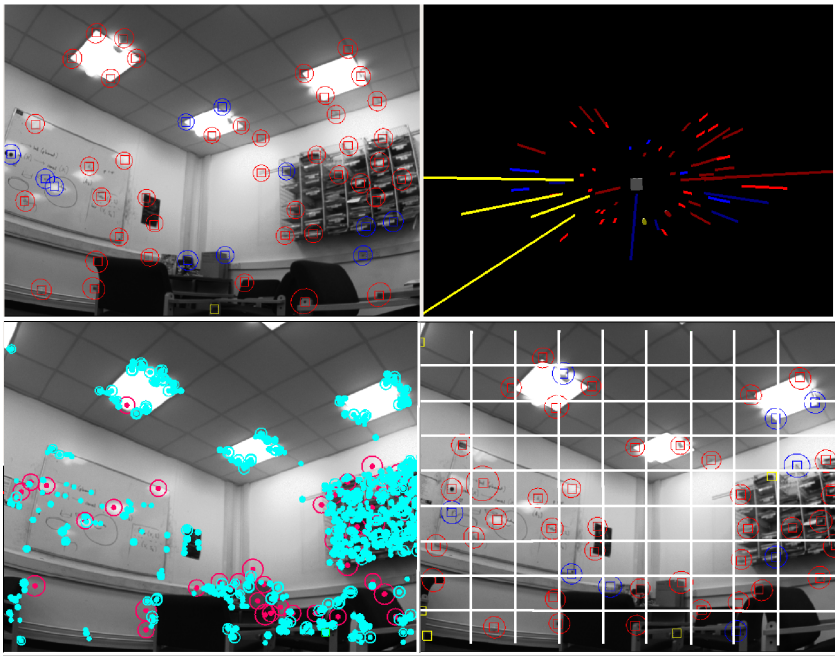


Figure 4.6: Top: MonoSLAM in operation, showing processing of an image stream from a single camera on a robot which has made nearly a full 360° rotation, highlighting feature tracking and 3D map creation. Bottom: adding SURF features to a MonoSLAM map: from all the SURF features detected in a frame (left), up to one new feature from each cell of a regular grid is added to the map (right).

this grid is unoccupied by a feature in the existing map, the most salient SURF feature detected in that region is initialised into the map (i.e., the one with the highest Hessian score, see [9] for details). By ensuring that features are initialised right across the field of view of each camera, we maximise the chances of detecting a decent number of corresponding features between the individual maps. In order to ensure that the feature scale do not affect significantly in the measurement accuracy, we only use SURF features from the first 2 octaves of the image pyramid, which are therefore well located in the images.

In our current implementation, multiple instances of MonoSLAM, one for each camera, are run simultaneously, frame by frame, from within a single program. These instances do not interact currently, but there is potential to improve performance in the future by considering this option. For instance, each instance may aim to initialise features into its map which have descriptors similar to those already in the other maps if available, to maximise the probability of obtaining a large number of correspondences. Note also that MonoSLAM, a real-time algorithm, could potentially run on-line on board the robot, though since the later processing steps are off-line there would be little to gain by doing this.

4.3.3 Bundle Adjustment of Single Camera Maps

MonoSLAM uses EKF processing and outputs filtered maps which may include linearisation errors due to the linear propagation of uncertainty used. Therefore, we look to refine the maps for each camera individualy to get as much accuracy as we can in the structure and camera poses of each map before attempting 3D alignment. We make use of the free software library SBA for this purpose [80].

To save on computational load at the expense of a relatively small loss in accuracy, at this stage we decimate the full image sequences processed by MonoSLAM into *keyframes* regularly spaced in time, and in the rest of the method we use only these. By controlling the robot's motion at a constant speed we ensure that the keyframes are spatially well distributed. It has been well proven in small scale structure from motion that a set of well chosen keyframes is quite sufficient for accurate camera motion estimation and scene reconstruction so we are confident that regularly decimating the sequence is a very sound procedure with regard to accuracy. This has most recently been confirmed in [132].

Each keyframe image is saved along with the 2D feature measurements made in that image by MonoSLAM and the camera position estimate from MonoSLAM at that time instant. After the robot completes its trajectory, the vector of 3D features position estimates $\hat{Y}$, the estimated camera poses $\hat{C}$, and the vector of 2D keyframe measured feature positions X are used in a bundle adjustment (BA) procedure for map

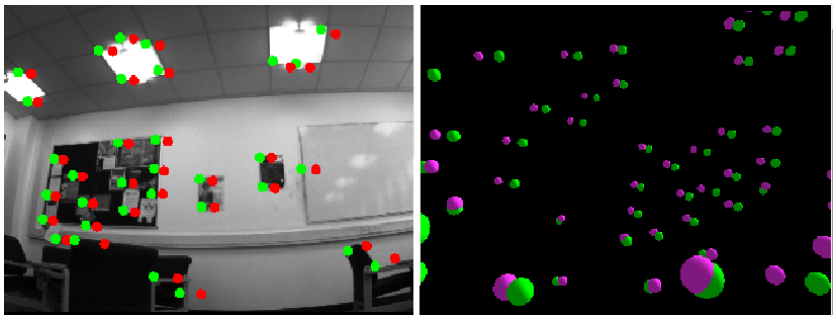


Figure 4.7: Bundle adjustment of an individual camera map generated by MonoSLAM. Here the 3D map generated by MonoSLAM and that after bundle adjustment is shown reprojected into one of the keyframes (left) and in 3D (right). In each view, the green points are after bundle adjustment and red/purple are before. Reprojection accuracy across the full keyframe set clearly improves after bundle adjustment.

refinement. The above quantities, which are input to the BA procedure, can be written in expanded vector form as follows:

$$\begin{aligned} \hat{C} &= [\hat{c_0}, \ldots, \hat{c_m}] \\ \hat{Y} &= [\hat{y_0}, \ldots, \hat{y_n}] \\ X &= [x_{00}, \ldots, x_{0m}, \ldots, x_{10}, \ldots, x_{1m}, \ldots, x_{nm}] \end{aligned} \quad , \tag{4.3}$$

where $\hat{c_i}$ is the estimated 6DoF camera pose of keyframe i, $\hat{y_j}$ is the estimated 3D position of feature j, and x_{ij} is the measurement corresponding to feature j in the image of keyframe i. The prediction $\hat{x_{ij}}$ of this measurement can be obtained through a standard camera projection function $h(\hat{c_i}, \hat{y_j})$, while the noise associated with this measurement, modeled by the Gaussian distribution $N(0, \sigma_x^2)$, is represented by the 2×2 matrix $\Sigma_{x_{ij}} = \text{diag}(\sigma_x^2, \sigma_x^2)$. Concatenating all predicted measurements in a single vector leads to the predicted measurement vector $\hat{X}$, with corresponding measurement noise encoded by the diagonal matrix Σ_X formed by concatenating all individual $\Sigma_{x_{ij}}$ matrices. Then, the vector of parameters to be estimated is given by $\hat{P} = [\hat{C}, \hat{Y}]$. In BA we look to minimize the Mahalanobis distance $\varepsilon^T \Sigma_X^{-1} \varepsilon$ (i.e., the weighted re-projection error), where $\varepsilon = X - \hat{X}$ [80].

The results of individual map bundle adjustment can be confirmed if necessary from a console within our application whereby the keyframe sequences from each camera are easily browsable with a slider and the reprojections of the optimised point locations can be checked for stability (see Figure 4.7).

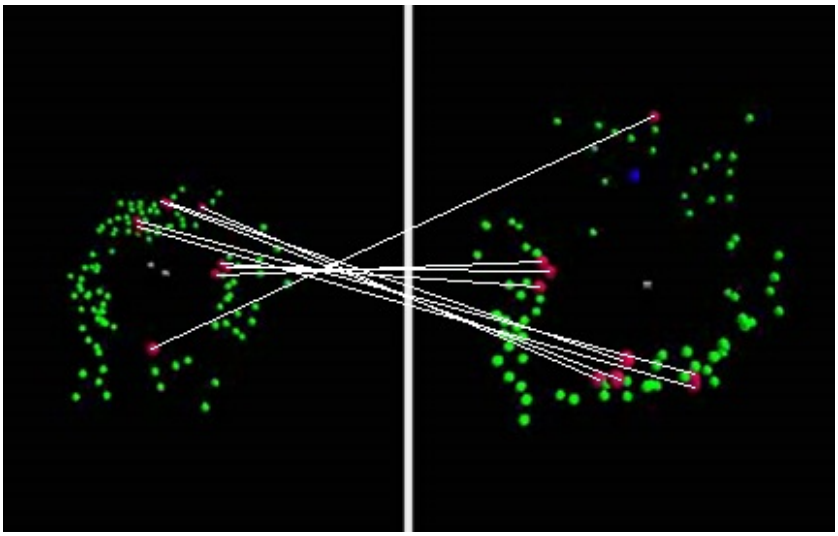


Figure 4.8: Correspondences (white lines) obtained from matching SURF descriptors attached to the 3D features of the maps constructed by two cameras. The red features represent those that are used for the final transformation estimate after a RANSAC procedure.

4.3.4 Inter-Map Correspondences from SURF Descriptors

Now that we have built individual feature maps for each camera, we must start to align them in order to produce a single joint map of the environment around the robot. First we find candidate correspondences between the features in each pair of individual maps. Exhaustively, the SURF descriptors of all features f_a of map A are compared with those of all features f_b of map B. During this procedure, the closest match in map B for a given feature f_a in map A is determined by the *closest-to-second-closest* ratio proposed in [82], by comparing (using the L2 norm) the SURF descriptors associated to the features. As a first step towards outlier rejection, the distance to the closest match is thresholded via an empirically determined value to retain only candidate matches which are highly similar in appearance, but there are likely to still be many outliers in the correspondence set (see Figure 4.8).

4.3.5 Confirming Correspondences and Finding Initial 3D Map Alignment using RANSAC

The goal of this step is to determine which of the inter-map correspondences between two maps suggested by SURF matching are geometrically consistent and therefore highly likely to be correct, and to compute an initial estimate of the cameras' relative poses. From a set of at least three feature correspondences between any two maps, the rigid body similarity transformation between those maps (and therefore between

the cameras which made them) can be hypothesized. We therefore run a RANSAC procedure where at each iteration three candidate SURF correspondences are chosen, a similarity transformation is calculated, and then the rest of the candidate correspondences are checked for consistency with this (via a Euclidean distance threshold) when the transformation is applied. Note that the potential transformation between the maps is a similarity transformation (we must solve for scale as well as rotation and translation) since they have been built by monocular cameras.

While there are various techniques for aligning 3D point sets, here we use generic non-linear optimisation via the Levenberg-Marquardt algorithm as in [39] to minimise the distance between transformed points, as computed by the following similarity transformation:

$$(y_i)_A = s\mathrm{R}_A^B(y_i)_B + t_A^B \ , \tag{4.4}$$

relating a feature's coordinates y_i in maps A and B via rotation matrix R_A^B, translation vector t_A^B and scale change s.

This alignment calculation is carried out for each random choice of three correspondences, and inlier scores for each candidate correspondence are counted up until a maximum number of RANSAC iterations. The final set of correspondences with the highest inlier counts are then used to make a final similarity estimate between the pair of maps. In experiments we normally obtain between 10 and 20 accurate correspondences between each pair of maps. The similarity transformations estimated for each pair of maps from the multiple cameras are then finally chained together, nearest neighbour to nearest neighbour, to form an initial estimate of the full set of relative camera poses.

4.3.6 Global Bundle Adjustment

Now that the individual maps from each camera have been aligned in 3D, an initial estimate is available of the relative pose of the cameras on the robot. However, this estimate relies heavily on the correspondences which have been obtained between the maps using SURF and RANSAC, and the number of these correspondences may be quite low. As a consequence, the resulting transformation may not be very accurate. There is still potential for improvement, by applying global bundle adjustment of robot motion, scene structure and relative camera poses together. Importantly, in this optimisation we enforce the constraint that the relative camera poses are constant, this being enabled since we know that our keyframes were captured in a synchronised manner.

After alignment and matching of the individual camera maps, we can work with the concept of a single joint map which is the union of the individual maps. Many of the features in this map will have only ever been observed by one of the cameras (fea-

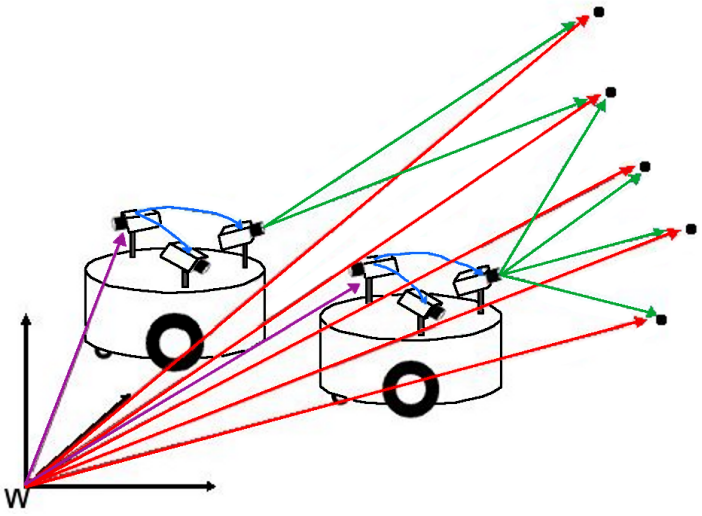


Figure 4.9: Diagram of the inputs used for the final Global Optimisation. The vector of parameters to be optimised is given by $\hat{P} = [\hat{C}(Purple), \hat{T}(Blue), \hat{Y}(Red)]$.

tures with no inter-map correspondences), while others (those successfully matched in the previous map alignment step) will have been observed by two or more cameras. Nevertheless, *all* of the features and *all* of the measurements in keyframes are useful in joint optimisation.

We formulate global bundle adjustment with the following state and measurement vector elements (see Figure 4.9):

1. The set of estimated camera poses $\hat{C} = [\hat{c_0}, \ldots, \hat{c_m}]$ associated to the keyframes of one camera, called the *reference camera*[1]. In particular, the pose $\hat{c_0}$ will define the origin of the absolute reference frame for all the estimated quantities.

2. The set of estimated relative camera poses $\hat{T} = \{\hat{T}_k\}$, where $\hat{T}_k$ is the estimated rigid body transformation from the reference camera to the k^{th} camera on the robot. Note that each transformation is a 6DoF transformation made up of rotation $\hat{R}_k$ and a translation $\hat{t_k}$ components. Note also that the estimated transformations do not depend on time, modeling the fact that cameras are rigidly fixed on the rig.

3. The set of 3D coordinates estimates for the features of the joint map $\hat{Y} = [\hat{y_0}, \ldots, \hat{y_n}]$.

4. The set of measurements $X = \{(x_{ij})_k\}$, where $(x_{ij})_k$ is the measurement corresponding to feature j in the i^{th} keyframe of camera k. The prediction $(\hat{x_{ij}})_k$ of a

[1] Note that any of the cameras mounted on the robot can be selected here.

Figure 4.10: The three experimental configurations, with a pair of cameras mounted at relative horizontal angles of 0°, 90° and 180°.

measurement can be obtained through the composition of a similarity transformation and a standard camera projection $h_k(\hat{c}_i, \hat{y}_j, \hat{T}_k)$: the feature $\hat{y}_j$ must first be transformed into the coordinate frame of the i^{th} keyframe of camera k, i.e. by the means of $\hat{c}_i$ and T_k, before being projected into the corresponding image plane.

The vector of parameters to be optimised is given by $\hat{P} = [\hat{C}, \hat{T}, \hat{Y}]$. Again, we look to minimise the Mahalanobis distance $\varepsilon^T \Sigma_X^{-1} \varepsilon$ (i.e., the weighted re-projection error), where $\varepsilon = X - \hat{X}$. Correctly formulating bundle adjustment in this way means that all of the useful information can be sucked out of the data available. The relative poses of the robot's cameras are estimated not just based on the structure of the scene, but also based on the motion of the robot, by enforcing the rigidity of the relative camera transformations over the trajectory of the robot.

4.4 Experiments

As our method is valid for any number of cameras mounted rigidly on a robot, we have currently been able to test it with different configurations of two cameras, and a four camera set up.

We ran the experiments in a normal university room with no modification and of size around 5×5 metres.

The robot's pre-programmed motion was as a turn on the spot controlled in a saw-tooth manner, such that the robot would rotate by left 90°, right 45°, left 90° again and so on until it had completed somewhat more than a full rotation.

In most experiments, this motion was completed in around 1 minute and image capture was at 15Hz. After execution of MonoSLAM, the sequences were decimated by a factor of 40 to leave only keyframes to be used in the rest of the algorithm. We used a set of Point Grey Flea2 cameras, which are able to synchronise automatically across the IEEE 1394b bus. The cameras were fitted with identical wide angle lenses giving a field of view of around 80°, having had their intrinsic parameters (includ-

ing substantial radial distortion) calibrated individually using a standard calibration pattern-based method.

To perform verification of our experimental calibrations against ground truth, we used the commercial photogrammetry software Photomodeler (Eos Systems Inc. Image-based photogrametry software used for accurate measurement and 3D-Modelling) to manually make a surveyed model of the rooms in which experiments were carried out from a set of high resolution digital photographs, with additional scale information provided by manual measurements. Images taken from our camera rig were then matched into this model to obtain estimates of the camera positions within the room and therefore their relative pose.

4.4.1 Different Configurations of Two Cameras

The camera calibrations used in our three experiments with two cameras are shown in Figure 4.10, and we present results from each of these in the following sections.

Parallel Cameras Facing Forwards

The first experiment, to confirm operation in a standard binocular stereo camera configuration, featured approximately parallel cameras facing forwards mounted on the left and right of the robot. All steps of the algorithm were carried out as explained in the previous sections, and we concentrate on the results of the final global bundle adjustment.

Figure 4.11 demonstrates the effects of the final BA procedure. The first row of the Figure shows a 3D view of the camera rig before (left) and after (right) the final BA. As it can be seen, the transformation between the two cameras is improved after the final BA: in the right image, the cameras look coplanar, whereas in the left image, one camera is slightly in front of the other. The improvement is proven quantitatively in the second row of the Figure, which shows the re-projection errors of features into the maps, taking into account all of the estimated transformations, in one video frame before and after the final global BA. After BA (right), the projected map points (circles) match more accurately the image positions of the measurements (crosses) than in the left image. The mean squared reprojection error achieved with this configuration was 0.90 square pixels.

Non-Overlapping Cameras Separated by 90°.

In the second experiment, the cameras faced horizontally but at around 45° to the left and right of the forward direction of the robot respectively. Figure 4.12 shows the

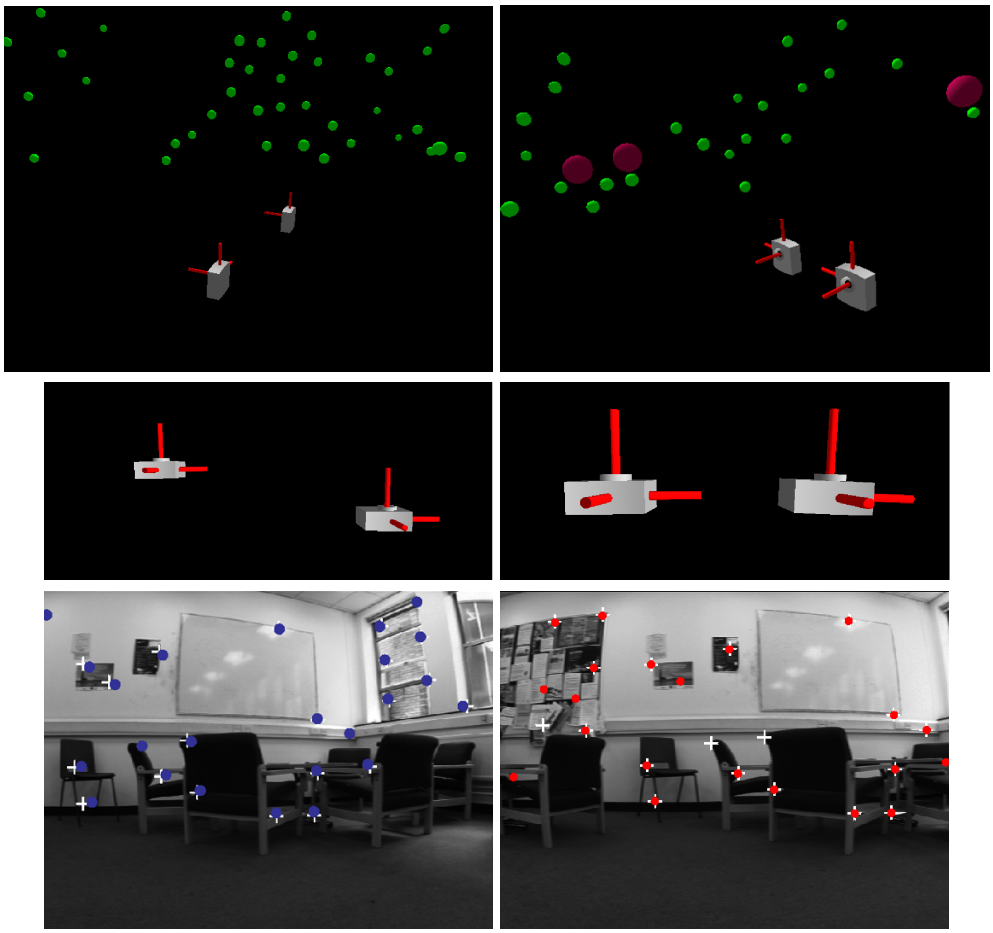


Figure 4.11: Effects of the final BA procedure, parallel cameras sequence: the images of the first and second row show the improvement in the estimated transformation between the two cameras (left column: before BA; right column: after BA), while the images of the second row show the reduction in reprojection error (see text for details).

improvement in reprojection error during global BA in this experiment, achieving an mean squared error of 1.4 square pixels.

Non-Overlapping Cameras Separated by 180°.

In the third experiment the cameras were mounted horizontally on the left and right of the robot, but with the right camera pointing approximately forwards and the left camera backwards. In Figure 4.13 we can see that after global BA the estimate of the cameras' relative pose was accurate, with a mean squared reprojection error of 0.807 square pixels.

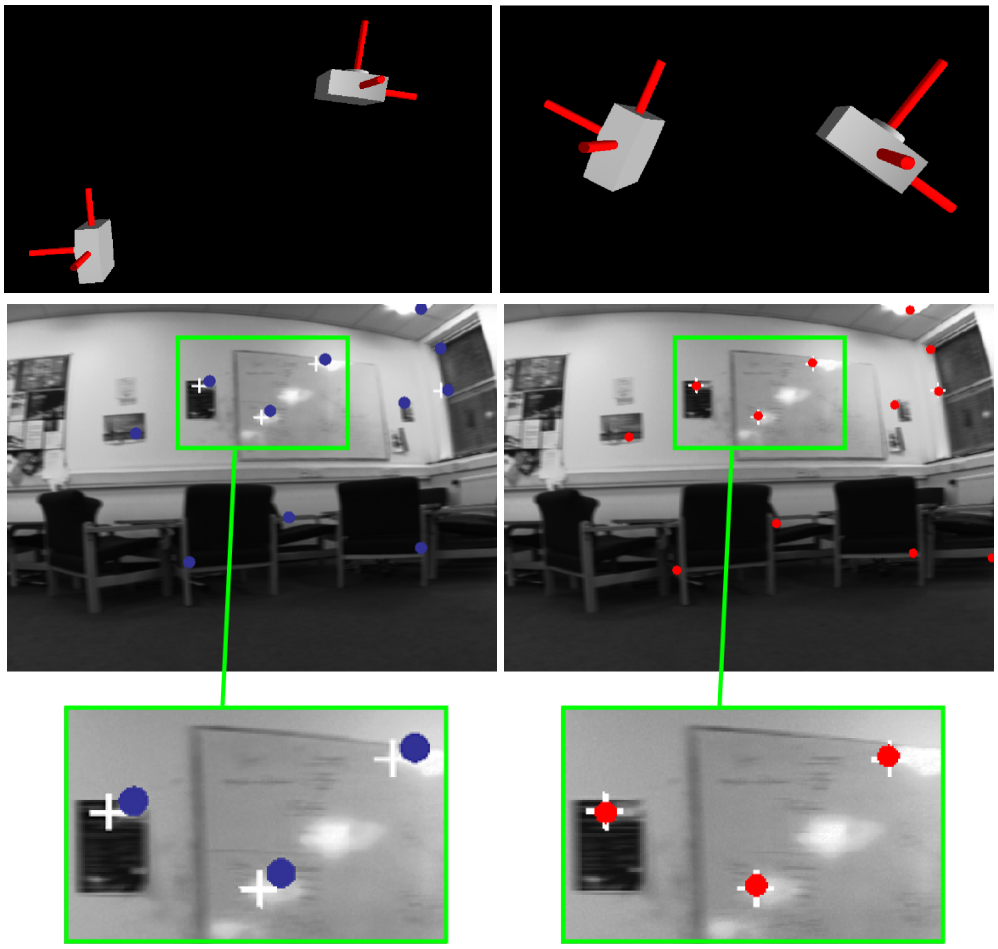


Figure 4.12: Effects of the final BA procedure, 90° cameras sequence in terms of camera pose and reprojection error (left: before BA; right: after BA.)

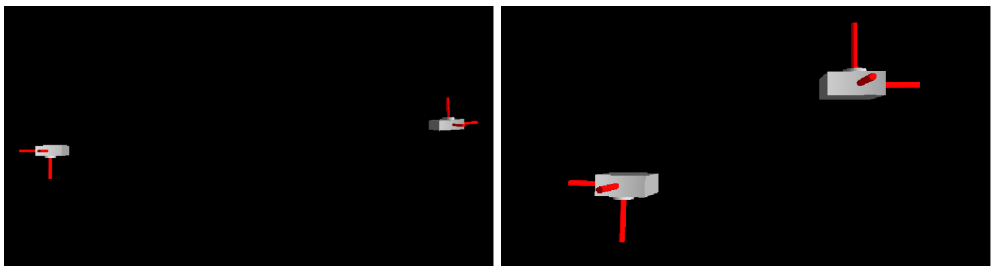


Figure 4.13: Effects of the final BA procedure, 180° camera sequence in terms of camera pose (left: before BA; right: after BA.)

4.4.2 Omnidirectional Four Camera Configuration

In this experiment we show that our method is applicable to any number of cameras rigidly attached to the robot. We set up two slightly divergent stereo pairs with no over-

	Rig Configuration with Two Cameras		
	Parallel	**90°**	**180°**
Photomod	$2.88° \pm 0.5°$	$94.72 ° \pm 0.5°$	$176.14° \pm 0.5 °$
Our System	$1.38° \pm 0.22°$	$94.10° \pm 0.83°$	$174.69° \pm 0.43°$

Table 4.1: Angles between cameras estimated by our approach and photomodeler for the three two-camera experiments.

lap between the front pair and the back pair, as illustrated in Figure 4.14. As in our previous experiments, we performed all the stages mentioned in Section 4.3 including a full optimisation in which the global map, the motion and the transformations between each camera and the reference frame are globally optimised.

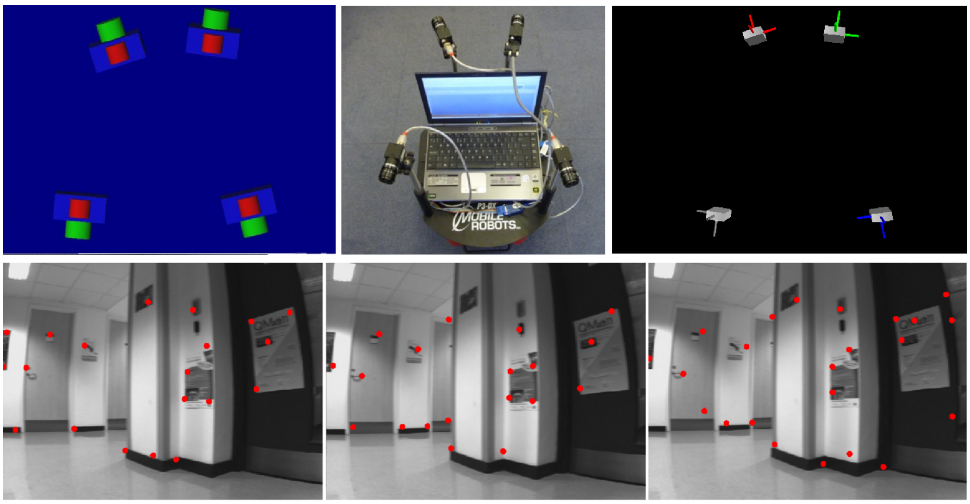


Figure 4.14: Experiment with an omnidirectional four camera configuration (top-left: Camera configuration obtained by photomodeler (the green cylinders represent the front of the cameras), top-middle: photograph of the robot, top-right: automatic result from our system for a camera configuration). The bottom images show the reprojection of features in the maps of cameras 2, 3 and 4 respectively from the global map into keyframes of camera 1. Note the high quality alignment between these points and scene corners, indicating excellent registration accuracy.

4.4.3 Ground Truth Comparison

Our system computes the full relative 3D pose (up to scale) between the cameras in each experiment, but for ground truth comparison the most straightforward parameter to analyse is the total relative rotation angle between the two camera frames. For each of the experimental scenarios presented above, we computed this angle and compared

	Rig Configuration with Four Cameras		
	Camera_{1_2}	Camera_{1_3}	Camera_{1_4}
Photomodeler	25.20°	173.76°	156.96°
Our System	25.30°	174.71°	157.20°

Table 4.2: Estimated angles between camera 1 (reference camera) and the other three cameras for the four camera experiment.

it with the relative angle determined by registering several pairs of camera images within the 3D model of the room obtained using Photomodeler. Table 4.1 presents for comparison the angles obtained with our system and Photomodeler for the two camera configurations, including an assessment of the standard deviation we observed over multiple runs of our whole system on the same image data.

We certainly see gross agreement, but we are inclined to believe that the not insignificant difference is due to limitations of our use of Photomodeler rather than a weakness of our method. The Photomodeler estimates were achieved from matching in just a few image pairs from our robot against a relatively sparse 3D point model, whereas the estimates from our method were achieved from global optimisation of long sequences with all the correct constraints applied. We are therefore inclined to put much more trust in the reprojection measures we obtained in experiments, all of which were in the range 0.8–1.5 pixels RMS. Considered as an angle in our 640×480 resolution images over an 80° field of view, this corresponds to an angular error of around 0.1° and we believe that this is much more indicative of the accuracy of our method.

For the four camera configuration we present results comparing angles between our reference frame (camera_1) and the other cameras (see Table 4.2). We can see that the differences between the angles obtained by Photomodeler and our system are smaller than for the two camera configuration, indicating as expected that using more cameras adds more constraints to the final global optimisation.

4.5 Summary

In this Chapter we have presented a fully automatic algorithm for general multi-camera rig calibration which does not require calibration targets or other infrastructure. Well known SLAM techniques are used to build monocular feature maps as the robot makes controlled movements. These maps are matched and aligned in 3D using invariant descriptors and RANSAC to determine the correct correspondences. Final joint bundle adjustment is then used to refine estimates and take account of all feature data. We

have demonstrated the accurate ability of the method to recover the configuration of a camera rig with two and four cameras in a variety of configurations.

One improvement of the system could be the use of the motion information, by knowing the motion of the robot we could incorporate that information into the filter to obtain more accurate estimates. Also, the features detected in one camera could be used to search for the same features in the other cameras by matching correspondent descriptors. This could increase the number of correspondences between maps which would help in the RANSAC procedure in order to obtain a transformation estimate that is more consistent and accurate.

It would be interesting and beneficial to determine the intrinsic parameters of the individual cameras as part of the full calibration procedure, rather than requiring them to be known *a priori*. It would be straightforward to include and optimise these parameters in the final bundle adjustment step, but the problem is that the construction of individual camera maps using MonoSLAM would be inaccurate without well known intrinsics for each camera. This could be tackled using the approach recently proposed by Civera *et al.* [21] for sequential auto-calibration of a single camera.

5

A Pose-Based Approach to Visual SLAM Using 2D Visual Odometry

Our main motivation in this thesis is to investigate the area of lightweight systems and to determine whether fully autonomous robot navigation could be achieved based on vision techniques which are easy to understand and implement, require little calibration and can be use in a modular way.

In this chapter, we propose a lightweight visual odometry (VO) approach that could serve for robot navigation purposes. We have decided to base the method on an easy VO algorithm that solves the task in an intuitive manner with just a few calculations, this algorithm is called RatSLAM [93]. We are not interested in the whole system but only in the VO section. Our approach is a small novel contribution that finds camera motion using optical flow with the RatSLAM VO framework as a base. In the next sections a review on RatSLAM and optical flow will be provided first and followed by a description of our lightweight VO system.

5.1 RatSLAM

RatSLAM is a biologically inspired approach for achieving SLAM and its impressive results are compared with the leading probabilistic approaches. Its SLAM framework is based on computational models of the rodent hippocampus along with a lightweight vision system that provides odometry estimates and appearance based loop closures.

This system involves the understanding of the spatial encoding of rodents and tries to relate it with an artificial system. The spatial representation of the rodents does not provide a detailed geometrical representation of the environment, it rather provides a pose estimate which is continuously updated, this is represented as pose cells. Several experiments in rodents have shown the presence of pose cells called: place, head and grid cells. Place cells are active when the rat is in a particular location, head cells fire depending on the rat's head orientation and grid cells show both properties. These cells are modelled normally using Continuous Attractor Networks (CANs) which consist of a number of neural units with weighted connections between them. Pose cells form the core of RatSLAM representing the conjunction of head direction and grid location using a three dimensional CAN.

For the purpose of our work, the emphasis will be given on the computer vision framework rather than the biological approximation, more information can be found in [88] [91] [92].

The vision system makes use of VO for obtaining odometry information, and for image matching in recognition of familiar visual scenes. Only one camera is used as a sensor capturing images in gray-scale. The algorithm runs in the following manner: images are captured from a small window inside the image, from this window, pixel intensity values are summed over columns forming a vector with a dimension according to the window width.

This vector is used to estimate rotation and forward speed for VO for every frame. It is also used as a global representation of an image, which is used to compare previously seen images for loop closure detection. These vectors are stored regularly representing a pose appearance in the robot's trajectory

Rotation information is estimated by comparing two consecutive arrays, the comparison is performed by calculating the average absolute intensity difference between the two of them as they are shifted relative to each other:

$$f(s) = \frac{1}{w-|s|}\left(\sum_{n=1}^{w-|s|} |I^{k+1}_{n+max(s,0)} - I^{k}_{n-min(s,0)}|\right), \tag{5.1}$$

where I is the image array of the k^{th} time stamp, s is the array shift and w the image width (see Figure 5.1). Rotation is then equal to the minimum value of $f(s)$ multiplied by an empirical constant λ to obtain an approximate angular shift $\Delta\theta$:

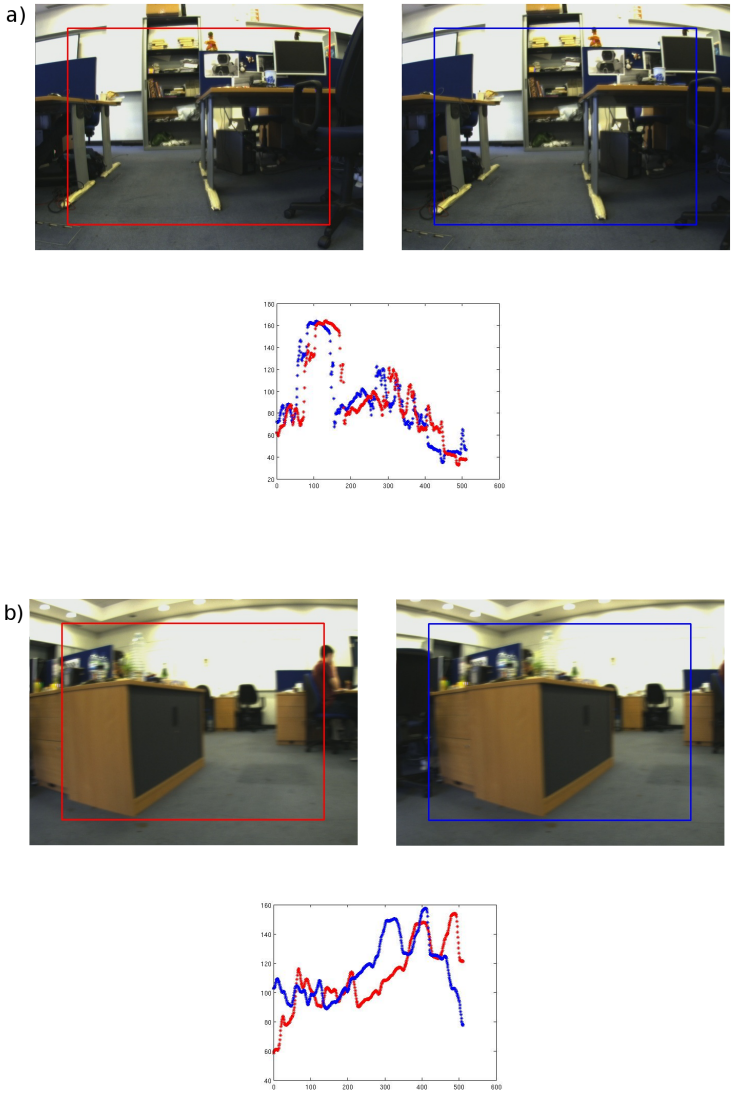


Figure 5.1: RatSLAM 1D vector representation for Visual Odometry. The images on the top of the graphs show: left - image at time $t-1$, right - image at time t. The graphs the 1D vector representation between two frames, by shifting both vectors we can obtain a rotation estimate, a) small rotation, b) larger rotation.

$$\begin{aligned} s_m &= argmin f(s) \\ \Delta\theta &= \lambda s_m \end{aligned} . \tag{5.2}$$

Velocity v is estimated by considering the amount of image change. This simple approximation to calculate rotation and velocity suggests that an assumption is made on the image input during the whole trajectory. This assumes that the image appearance from one frame to the next does not change in intensity too much. For instance, if the robot is navigating in an indoor scenario with narrow spaces driving very close to obstacles which are in front of the filed of view of the camera then the estimate of velocity is very different than when it is far from them even if the displacement is the same.

$$v = f(s_m) . \tag{5.3}$$

In order to recognise places or images already seen, an experience map is built. This map is composed by individual experiences which in this case are images along the path. The first experience is created from the starting position or from another arbitrary position, then each time an image is sufficiently different from a stored experience, a new experience is created.

A really important process in SLAM is loop closure in which the drift of the robot pose is corrected. In RatSLAM, loop closure is done by matching previously stored experiences which are represented as the vector previously computed to estimate rotation and translation. This representation is basically a graph of poses that is optimised every time the robot has found a loop closure. Therefore, once a good match is found, the positions of all experiences are updated using the following expression:

$$\Delta p^i = \alpha \left[\sum_{j=1}^{N_f} (p^j - p^i - \Delta p^{ij} + \sum_{k=1}^{N_t} (p^k - p^i - \Delta p^{kj}) \right] , \tag{5.4}$$

where Δp^{ij} is the change from position p^i to p^j, N_f is the number of links form experience i to other experiences, N_t is the number of links from other experiences to experience i, and α is a correction rate constant. This equation suggests that the error in the robot's position is distributed along the whole graph of poses.

The VO approach used here, runs in real-time and captures with good accuracy the essence of the map, fixing the errors in the loop closure step. However, it is prone to errors and noisy measurements limiting the construction of a geometrically consistent map.

In [90], Milford presents a system that performs persistent navigation along with

exploration, mock deliveries and recharging tasks in an indoor environment travelling 40km over 37 hours of active operation. The system sits on top of RatSLAM which is the main module. It maintains two separate map representations: the global semi-metric topological map for SLAM that uses odometry and computer vision for loop closure detection; and a local obstacle map built with sonar and laser readings which is centred in the robot's position and provides local navigable regions based on a short history of sensor readings.

5.2 Optical Flow

The objective in optical flow is to find the 2D motion field in an image using only local information. One way of doing it is by modelling the motion of an image with a continuous variation of image intensity as a function of position and time. In our notation, $I(\mathbf{X},t)$ represents the image intensity of position $\mathbf{X}$ at time t. After a change in time Δt the purpose is to find the displacement vector d that fulfils the following criteria:

$$I(\mathbf{X},t) = I(\mathbf{X}+d, t+\Delta t)\ . \tag{5.5}$$

There are different methods to calculate optical flow, e.g. differential, correlation based and frequency based methods. Probably one of the must studied algorithms to calculate optical flow in a differential manner is the seminal work done by Lukas-Kanade [83].

5.2.1 Lucas-Kanade

The aim of the algorithm is to find the optical flow between two images I and J by minimising the error function:

$$\varepsilon(\Delta x, \Delta y) = \sum_{x=x-w}^{x+w} \sum_{x=y-w}^{y+w} (I(x,y) - J(x+\Delta x, y+\Delta y))^2\ , \tag{5.6}$$

w represents the size of neighbourhood region around a pixel (x,y) in which the match is expected to lie. Denoting $d = (\Delta x, \Delta y)$ and taking the derivative of ε with respect to d, we obtain,

$$\frac{\partial \varepsilon(d)}{\partial d} = -2 \sum_{x=x-w}^{x+w} \sum_{y=y-w}^{y+w} (I(x,y) - J(x+\Delta x, y+\Delta y)) \begin{bmatrix} \frac{\partial J}{\partial x} & \frac{\partial J}{\partial y} \end{bmatrix}^T\ . \tag{5.7}$$

Substituting the displaced image $J(x+\Delta x, y+\Delta y)$ with Taylor series expansion:

$$\frac{\partial \varepsilon(d)}{\partial d} \approx -2 \sum_{x=x-w}^{x+w} \sum_{y=y-w}^{y+w} \left(I(x,y) - J(x,y) - \begin{bmatrix} \frac{\partial J}{\partial x} & \frac{\partial J}{\partial y} \end{bmatrix} d \right) \begin{bmatrix} \frac{\partial J}{\partial x} & \frac{\partial J}{\partial y} \end{bmatrix}^T , \quad (5.8)$$

where $I(x,y) - J(x,y)$ can be interpreted as the time-derivative of $I(x,y)$. $\begin{bmatrix} \frac{\partial J}{\partial x} & \frac{\partial J}{\partial y} \end{bmatrix}^T$ is the image gradient and can be denoted as $\nabla I = [I_x I_y]$, then we have:

$$\frac{1}{2} \frac{\partial \varepsilon(d)}{\partial d} \approx \sum_{x=x-w}^{x+w} \sum_{y=y-w}^{y+w} ((\nabla I)^T d - \delta I) \nabla I , \quad (5.9)$$

where,

$$G = \sum_{x=x-w}^{x+w} \sum_{y=y-w}^{y+w} ((\nabla I)^T \nabla I = \sum_{x=x-w}^{x+w} \sum_{y=y-w}^{y+w} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} ,$$

thus

$$\frac{1}{2} \frac{\partial \varepsilon(d)}{\partial d} \approx Gd - b . \quad (5.10)$$

At the optimum it is wanted that $\frac{\partial E(d)}{\partial d} = 0$ then d can be obtained as:

$$d = G^{-1} b . \quad (5.11)$$

In practise this has to be done in an iterative manner and it works for small pixel displacements. To be able to handle larger pixel displacements a pyramidal implementation can be used [8]. A Gaussian Pyramid is built from images I and J, then at each level of the pyramid starting from the coarser level, the Lucas-Kanade algorithm is applied for several iterations before handing off to the next level. The output vector d from a lower level is used as the input to the next finer level.

5.3 LK-RatSLAM

In this section, we explain our VO implementation which is inspired by ideas of RatSLAM [93]. As we can analyse from the previous section, the RatSLAM approach assumes the robot is moving on the same plane and it only provides estimates of displacements on the horizontal axis and not in the vertical one. We aim to build an approach capable to deal with both estimates. This is mainly because if the robot has small heading movements when going forward or backwards then this approach would allow us to deal with this situation much better. For instance, in the case of the Segway robot (see Chapter 3) this kind of movement is present most of the time.

Our algorithm works in the following manner. Images of size 320×240 are taken from a single camera and processed using gray intensity values (see Figure 5.2).

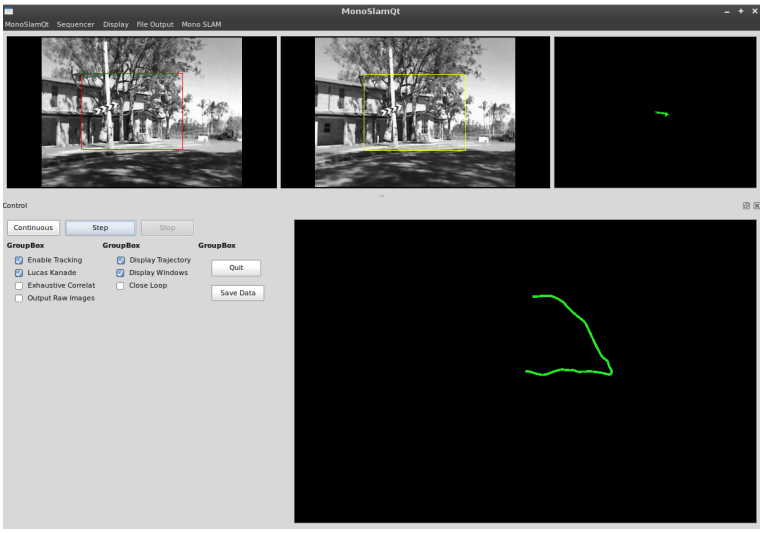


Figure 5.2: Graphical Interface of our system mapping an 18km trajectory. The left image represents the current frame and the image on the centre, the previous frame. The top left image indicates the mean estimate of the displacement vector. At the bottom is the computed trajectory.

We have decided to use optical flow as the motion estimate framework because of the facility to obtain 2D displacements on the image plane. The pyramidal implementation of Lucas-Kanade [12] is used to find the vector displacement d of two subsequent images (see Figure 5.3). Optical flow is only computed for a single pixel which is the centre of a window of size 160×120. The whole window is used as the pixel neighbourhood in which Eq. 5.11 is applied.

As we mentioned previously, with the camera facing horizontally RatSLAM assumes that rotation is only in the vertical axis considering that most robots and mobile vehicles rotate only along this axis (yaw). Our method allows us to estimate horizontal and vertical rotation of the camera (yaw and pitch), this provides to the system the ability to handle more complicated motions. The system can address the types of rotation occurring in our robotic platform (see Figure 3.2).

After the vector d has been found, the horizontal displacement Δx is multiplied by an empirically determined constant λ to convert it into an approximate angular shift $\Delta\theta$. This horizontal displacement is used as the rotation estimate.

Once both windows w_1 and w_2 are matched, normalised sum of squares is used over the two regions to correlate them,

$$N = \frac{E(w_1 w_2) - E(w_1)E(w_2)}{\sigma(w_1)\sigma(w_2)} \,, \tag{5.12}$$

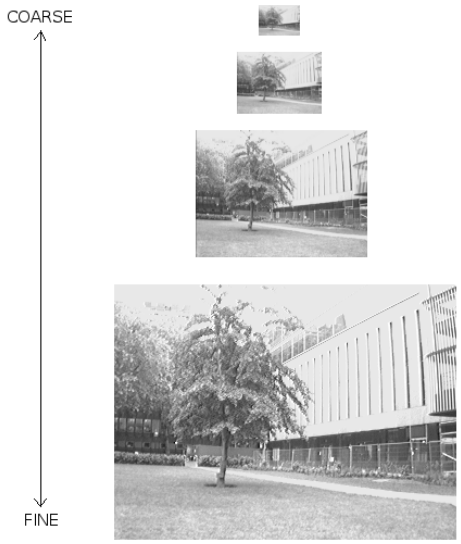


Figure 5.3: Pyramidal representation used for optical flow.

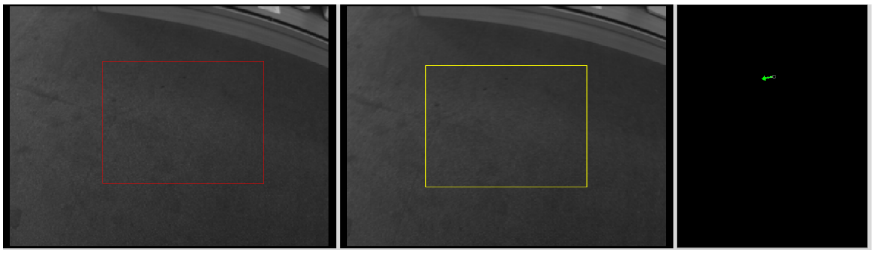


Figure 5.4: Displacement Estimate found with our VO approach. Image on the left represents the visual input at time t, image on the centre shows the visual input at time $t-1$. The Figure on the right shows the displacement estimate vector.

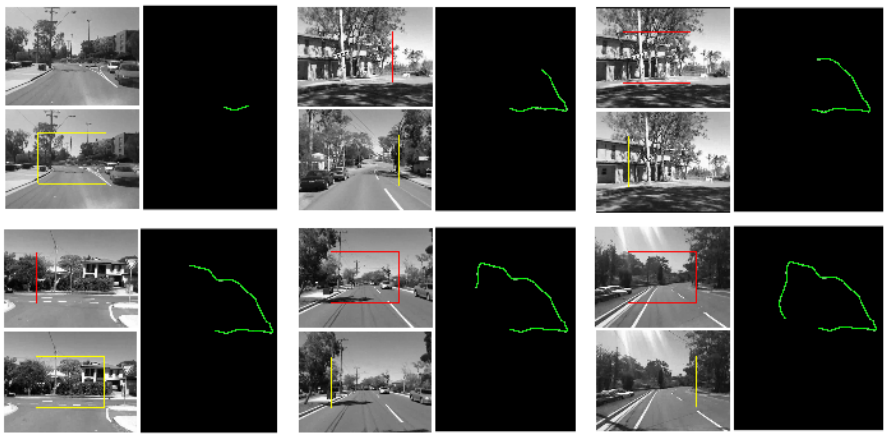


Figure 5.5: Frames taken from an 18km sequence. Each sub-image consist of two images which indicates the two frames from where motion estimates have been taken, the black area on the right shows the trajectory.

where $E()$ is the expectation operator, N is used as the translational velocity estimation. Therefore, once we obtained translational velocity and the rotation estimate, the 2D trajectory can be obtained.

In the next section we provide experimentation to demonstrate that this lightweight technique allows the computation of 2D trajectories in a simple manner and by comparing it with RatSLAM we can see that we obtain better results.

The trajectories taken are also assuming that the scene does not change too much from one frame to the next one. Experimentation on an indoor scenario is provided in Section 7 .

5.3.1 Testing and Results

This work was part of the first year work during my PhD and at that stage, loop closure detection and graph relaxation were still not implemented. However, the purpose of this chapter is to show the performance of our VO approach and to show that when compared against RatSLAM on the sequence reported in [93], it provides better results.

Our system was tested with the sequence used in [93]. The sequence was taken from a normal webcam mounted on the top of a car. The car was driven along an 18km path with different loops (see Figure 5.5). We used the same sequence for both algorithms until the first loop closure occurred. The sequence was provided by the author of RatSLAM Michel Milford. In Figure 5.6, a graphical comparison is presented, the sub-image on the left represents the output from RatSLAM and the one on the right represents our system.

As we previously mentioned, the sequence is stopped a frame before RatSLAM

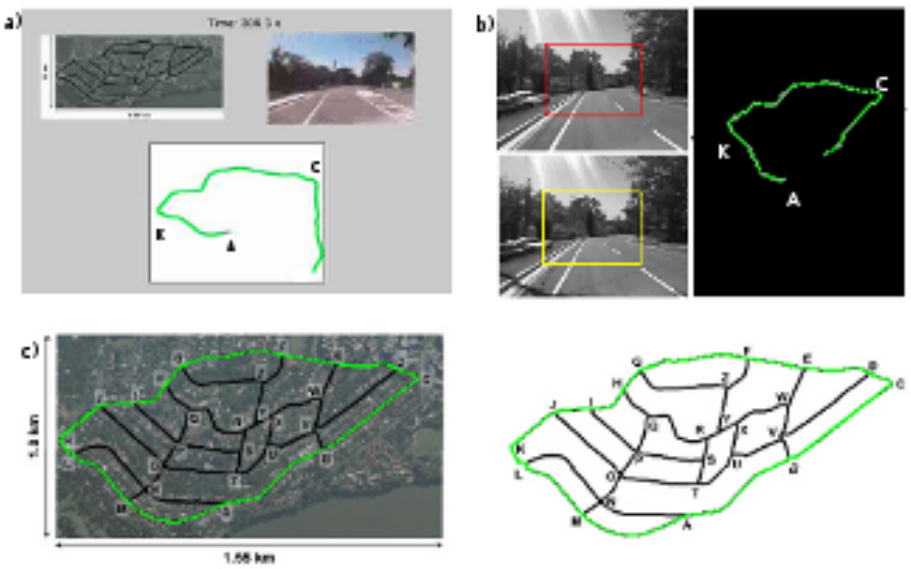


Figure 5.6: Comparison between RatSLAM and our system. Figure a) indicates the RatSLAM output, Figure b) indicates the output of our system. Figure c) shows the ground truth on a map of the whole trajectory mapped in the RatSLAM experiments (image obtained from [93]). We have tried the sequence until the first loop closure which is represented by the green line.

detects a loop. In the image it can be seen from the trajectory outputs from both systems that ours has a better motion estimation, the path is more similar to the ground truth (see Figure 5.6.c).

To test our algorithm against hand-held motions we grabbed another sequence at Imperial College around The Queen's lawn, the purpose of this experiment was to walk around a square to observe the amount of drift obtained. It can be seen from Figure 5.7 that the trajectory follows a square figure although a drift is presented at the moment of closing the loop. In the bottom middle sub-image it can be seen that the camera was facing against the sun and the brightness in the image introduced noise on the motion estimate. In bottom image of Figure 5.7, we can observe that the estimated trajectory is fairly accurate when visually compared with ground truth.

This Chapter gave us an overview of a simple method that can deliver good results under fair assumptions. The main motivation was to provide to the reader a better understanding of the algorithms we are aiming at this thesis. In the next Section, a integral system for autonomous navigation is developed based on our lightweight concept.

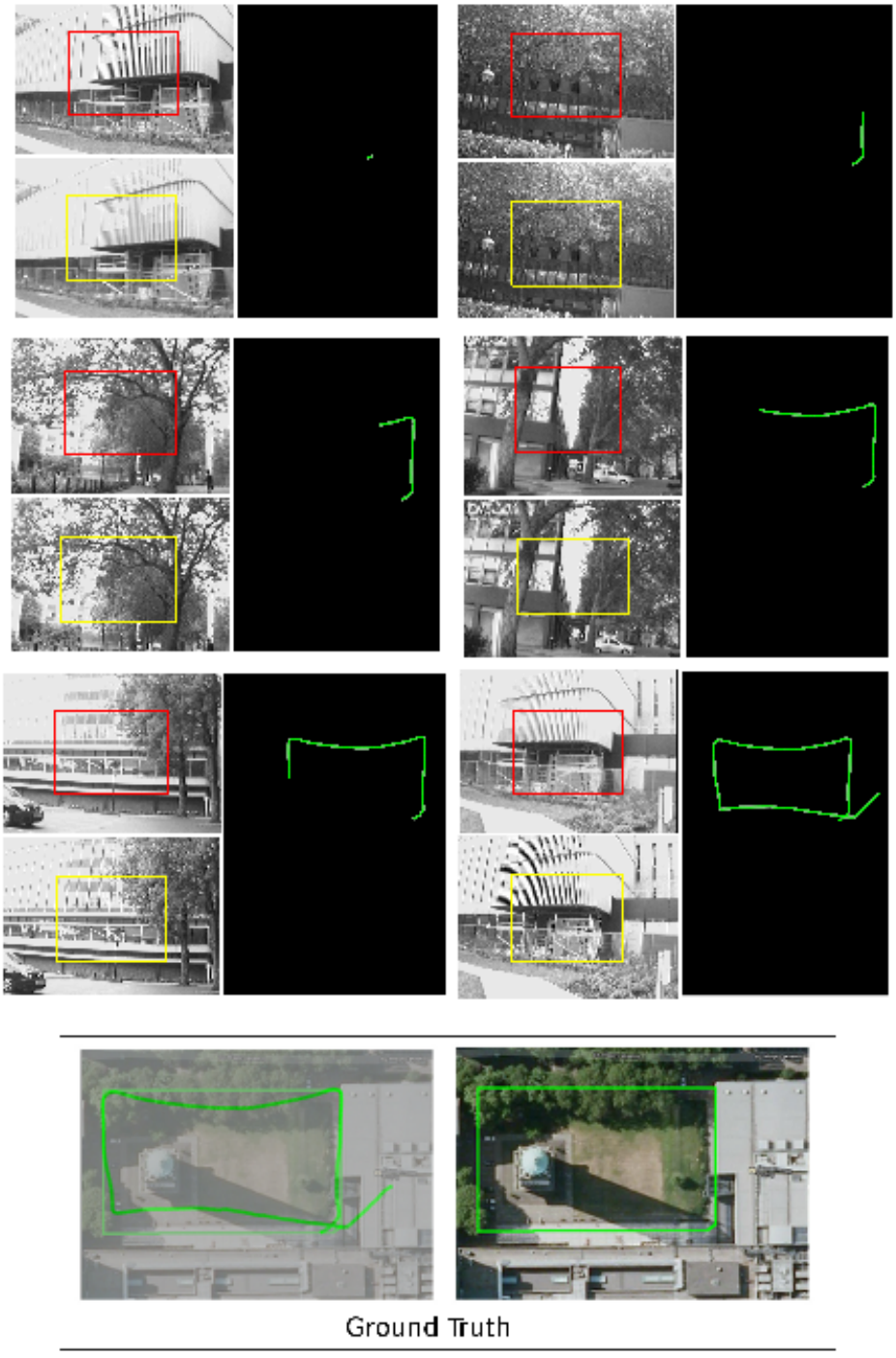


Figure 5.7: Computed trajectory of a our LK-RatSLAM approach on sequence taken at Imperial College.

6

Lightweight SLAM and Navigation with a Multi-Camera Rig

As we have seen in the last chapter, an interesting recent branch of SLAM algorithms using vision has taken an appealing approach which can be characterised as simple, robust and lightweight when compared to the more established and complex geometrical methods. These lightweight approaches typically comprise mechanical odometry or simple visual odometry for local motion estimation; appearance-based loop closure detection using either whole image statistics or invariant feature matching; and some type of efficient pose graph relaxation. However, such algorithms have so far been proven only as localisation systems, since they have not offered the semantic demarcation of free space and obstacles necessary to guide fully autonomous navigation and exploration. In this chapter we investigate how to apply and augment such approaches with other lightweight techniques to permit fully autonomous navigation and exploration around a large and complicated room environment. Our approach uses only odometry and visual sensing, the latter being provided by a rig of multiple standard cameras without the need for precise inter-camera calibration.

6.1 Introduction

Computer vision is an increasingly appealing sensing modality for mobile robotics, and a number of successful SLAM systems involving vision as the primary outward-looking sensor have been recently presented. In particular, besides the more standard feature-based reconstruction approaches, there have been several systems which have solved the visual SLAM problem using much 'simpler' lightweight techniques which combine local trajectory estimation (via either wheel or simple visual odometry), visual place recognition, and pose graph optimisation (e.g. [90]). However, in either paradigm few systems have gone further than tackling localisation, and few real visual SLAM systems have been proven in autonomous navigation.

Here, we present a method based on a combination of lightweight vision techniques which permits robust, automatic and repeatable mapping and navigation around a large room. In this scenario, there are several demands placed on the robot's sensing systems:

- Local trajectory estimation.
- Place recognition to detect re-visits (loop closures) and permit global map optimisation.
- Detection and avoidance of obstacles in the robot's close proximity.
- Global free-space mapping for path planning.

We describe a novel combination of lightweight methods to provide all of these capabilities using odometry together with multi-camera visual sensing.

As we metioned previously in 2.4, our 'lightweight' concept of a system is one which comprises modules each of which has some of the following characteristics:

- An easy to understand method which is easy to implement and modify.
- Little requirement for calibration.
- A 'black-box' method, which can be used without knowledge of internals.
- Computational efficiency.
- Modularity: can easily be combined with other techniques.

Many techniques in computer vision require significant expertise to use and put together into systems. Many others are brittle in their behaviour, or require precise calibration to use. We do not argue against the development of those algorithms here, and complicated methods are usually inevitable to explore new concepts and to push

the boundaries of performance. But here we would like to show that a simpler, modular, lightweight approach comprising components each of which on its own can be easily understood by a relative non-expert can be surprisingly effective. We believe that this is of importance in particular in the expanding low-cost consumer robotics field where robots for the home or other mass-market domains are becoming a reality.

An important factor in the use of lightweight vision techniques is that a certain technique will often only be useful in a well-defined set of circumstances specified by the application domain. This brings up the incorporation of modularity into our lightweight concept. Consider the simple visual odometry component of the RatSLAM approach which we tested in Chapter 5. This module, which estimates the rotational and linear velocity of a vehicle on the basis of the horizontal shift and rate of change of a 1D image signal, serves well as a module for visual odometry in the road vehicle case, but would not perform so well for robots which have non-planar movement or which operate in more cluttered types of scene. We will look into this further in the experiments later in this thesis.

Another aspect which we consider relevant to the lightweight concept is that we will sometimes make use of modules which although perhaps not 'simple' in their internal workings, are straightforward from the point of view of a use due to their 'black box' well engineered nature, computational efficiency and robustness independent of parameter settings. An excellent example of such a module much used in mobile robotics, and which we will use in this chapter, is the TORO library for efficient graph optimisation [48]. (and its modern, more general and more efficient replacement g^2o [71]). This is an open source library which is relatively easy to use in various different robotics problems. The software is very well engineered and tested, and is computationally efficient even in large problem domains. Further, it can be said that it is fairly robust with regard to calibration, because it has been shown to work well in cases where for instance only rough and constant covariance values are used for the links in the pose graph it optimises rather than specifically derived ones (e.g. [81]).

Our use of a rig of multiple standard cameras is based on the observation that while a wide field of view is often desirable for SLAM and relocalisation, the other requirements placed on a vision system when the goal is full autonomous navigation are not easily satisfied by a single omnidirectional camera. Omnidirectional cameras, since they use a single optical system, normally have relatively low spatial resolution in all directions; and also, they often do not provide a good view of downward angles below the horizontal. They may therefore be unsuited to free space detection and obstacle detection. The lack of resolution in particular directions may also be a future limitation when robots aim to do more than navigate: when they must identify and interact with objects or people.

Pre-calibrated multi-camera rigs in a single unit which offer good resolution across a wide field of view are available, such as Point Grey's Ladybug, but they are expensive, as well as inflexible since they cannot be reconfigured. The advantage of multiple specialised sensors in high performance autonomous systems has been proven in robots such as the DARPA Grand Challenge vehicles or Willow Garage's PR2. In our work, we choose to use just cameras rather than other outward-looking sensor types, but retain the idea of multiple cameras mounted in a weakly coupled way and which provide specific functions as part of a whole navigation solution, without requiring a tedious inter-camera calibration procedure. Note that a multi-camera rig will in general not be fully equivalent to a single omnidirectional device in terms of imaging geometry since it will have multiple optical centres, but this may not a problem for applications such as navigation as long as the locations of those optical centres is known, and may even be an advantage in inferring depth information.

We demonstrate automatic localisation and mapping, and autonomous navigation in a goal-directed scenario where the robot is able to move repeatably between any pair of points indicated in the map. Further, we demonstrate full autonomous exploration; the robot is dropped into the room with no knowledge or a large cluttered room and is able to explore autonomously to build a consistent map of the whole area suitable for autonomous navigation.

6.2 Related Work

The use of a rig of cameras for SLAM has been subject of little research, presumably because of the difficulty of extrinsic calibration.

The most familiar approach to vision-based SLAM and navigation is the geometrical path of aiming to build a global, uniform map representation of the 3D geometry of a scene while simultaneously estimating the robot's motion through this scene. The 3D scene is represented using parametric features. There have been many papers on this approach [30, 18, 34, 70, 87, 131], with increasingly sophisticated approaches and correspondingly accurate and large-scale results. Most of these methods assumed either a monocular rig or binocular stereo rig with a large overlap in the fields of view of the cameras. In the Chapter 4 we presented a method by which a more general multi-camera rig, potentially including cameras without overlap, could be calibrated such that it could be use in such system. As mentioned, there have been some systems that have taken this approach [58], demonstrating a SLAM system with an eight camera rig.

While such geometric work on fused multi-camera SLAM will doubtless continue, in this work rather than seeking to fuse the images from multiple cameras into a single

wide-angle view, we aim to use each of the cameras for one or more specialised tasks without the need for global extrinsic calibration.

Localisation with a predefined hand-drawn map using a rig of cameras is investigated in [77] where images coming from six cameras are converted into a single flattened image, for loop closure they use a combination of colour histograms, image moments and feature points, a final bundle adjustment procedure is used to refine camera positions obtained with a visual odometry method. In [58] a probabilistic SLAM approach is developed using an incremental expectation maximisation algorithm over the structure obtained by an eight camera rig.

6.3 Method

We divide our whole framework into five main sections which are all interconnected:

- Rig camera placement.
- Map representation and loop-closure detection.
- Global map relaxation.
- Free space mapping.
- Autonomous navigation, obstacle avoidance and path planning.

6.3.1 Rig Camera Placement

Our robot's vision capability is to be provided by a rig of up to four standard cameras, each with a lens offering approximately 80° horizontal field of view. There are many possible choices for the configuration of these cameras, since they must support the various tasks required by loop closure detection, free-space mapping and exploration, and we have examined the trade-offs of different set-ups. A goal of our method is not to require either accurate camera placement or extrinsic calibration. Notably, the final configuration chosen is adapted to the characteristics of the office environment used, where movements can be quite restricted and the distance to the objects relatively short (e.g., when traversing narrow corridors between two desks).

In principle, an extremely wide field of view, up to the maximum full cylindrical field of view offered by a single omnidirectional camera, is well suited to relocalisation: not only does this enable the capture of a good variety of appearance data to act as a fingerprint for a location, but it also permits recognition of previously-visited places independent of the orientation of the robot. However, when this wide field of view is provided not by a single omnidirectional camera but by a rig, we found that additional difficulties arose.

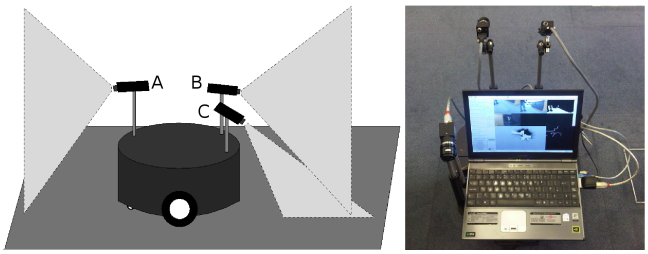


Figure 6.1: Robot camera configuration consisting of a three camera rig where camera C is used for obstacle avoidance and free space mapping and the cameras A and B are used for loop closure detection.

We experimented extensively with histogram-based place recognition (see the next Section) with an ad-hoc four-camera rig designed such that the cameras were mounted horizontally with maximum angular spacing (i.e. at the corners of a square). However, the performance in recognising locations with different robot orientations was disappointing. This was partly due to the fact that the four cameras used left gaps or 'blind spots' in the cylindrical field of view which would not necessarily align when the robot had made a rotation. Another significant point was that the actual distance between the camera centres on the robot was often significant compared to the distance to objects and furniture in the indoor scene, so unmodelled parallax effects came into play.

We found that a good pragmatic solution to camera placement for loop closure detection in an indoor scene is to have one horizontal camera facing forwards, and one backwards (see Figure 6.1). This configuration is able to detect the vast majority of loop closure events the robot will encounter, because in restricted spaces, when a robot revisits a place it is very likely to do it either moving in the same or exactly opposite direction to before. Yes, there will be possible loop closure events sometimes missed when the robot crosses a previous path at right angles. But in fact, such a crossing might well be difficult to reliably detect in any case, as it may correspond to a wide open area where several significantly distant positions at the centre of it have similar appearance, making recognition ambiguous and localisation inaccurate.

Also, as we explain below, and as used in several previous appearance-based loop closure detection algorithms, we make use of a sequence of nearby image matches to validate a loop closure hypothesis and reduce sensitivity to perceptual aliasing, and this would not be possible in a perpendicular crossing.

Together with the front/back camera combination for loop closure detection, we added a third camera pointing down at the ground in front of the robot for free space mapping. As detailed later, we experimented with the downward angle of this camera where there is a trade-off between immediate, local obstacle detection capability and a more forward view suitable for planning. There is an interesting feedback in play

here between the image matching for relocalisation required by SLAM and free space mapping which permits autonomous robot guidance. A free space detection solution which is working well will enable the robot, when exploring or re-visiting, to navigate in a precise way by for instance moving repeatably half-way between gaps or along corridor passages. This makes loop closure detection easier, since the robot is likely to very precisely revisit locations.

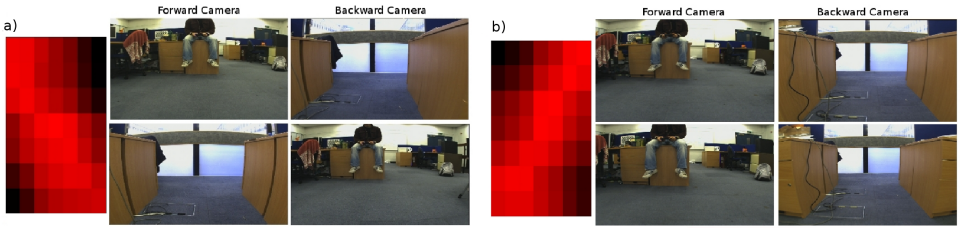


Figure 6.2: Loop Closures using two cameras: forward and backward. a) represents a loop closure found with a difference in robot orientation of 180° with respect to the matching node. b) represents a loop closure found with the same robot orientation as the matching node. The top row in Figures a) and b) indicates the current frame and the bottom row the matching node or keyframe. The single image at the left of Figures a) and b) shows the matrix consistency check, with the bright red indicating strong matching along either the forward or backward diagonal.

6.3.2 Map representation and Loop Closure Detection

Our approach does not assume any prior knowledge about the environment except that is traversable by a wheeled robot and that its visual appearance is descriptive enough.

As the robot autonomously explores or is being driven through an environment, it builds a topological or graph representation of the area. A topological map is a very practical and desirable model for navigation, since it imposes a discrete structure on a continuous space, and because it easily enables the use of different low level (graph relaxation) and high level (path planning) algorithms.

In this undirected graph $G = (V, E)$, a vertex $V = (I_t^N, X_t)$ represents a physical location in the environment which stores all the images I from N cameras of the rig at time t as well as the global position of the robot $X_t = (x_t, y_t, \theta_t)$ (2D position plus heading direction). Each edge E_t in the graph stores the relative 2D transformation between nodes X_t and X_{t+1}. A new vertex is initialised when the distance travelled from the previous position is greater than some threshold β (using the internal odometry of the robot), or when the dissimilarity between consecutive images is above a threshold α.

Reliable loop detection independent of the robot's heading position is certainly an important quality of any SLAM system, it can easily be obtained by using an omnidi-

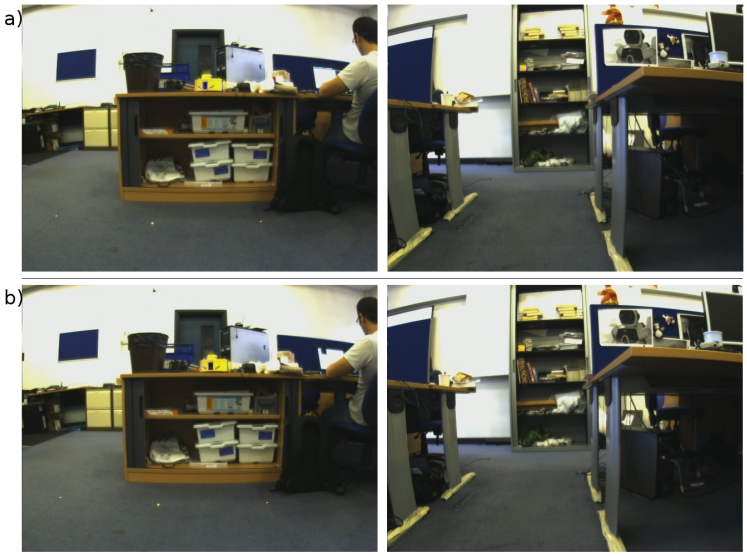


Figure 6.3: Loop closure detection with a forward and backward camera configuration. The top image shows the current image at time t, the bottom shows the best match found over the previous keyframes at time $t-k$. Where k is the number of keyframes in the history of the robot's trajectory.

rectional camera [5] or by building a panoramic image using a calibrated camera rig [77]. Involving a calibration step in our algorithm is not what we really aim due to the fact that it adds a strong initial constrains which makes the system not as "out of the box" as we want. Our solution involves the use of two cameras: one facing forward and one facing backwards, with this configurations we are able to find loop closures when the robot is traversing the same place with a 180° difference in orientation.

In computer vision, there are different approaches that represent images for localisation in different ways. In general we can find those that use features and represent an image in a more local manner and those that represent the image globally using features such as colour.

Due to the robustness provided by local features, cues such as colour has been less studied. It is known that data association is a complex problem in feature based approaches adding complexity to the problem.

When using cues such as colour, data association is usually performed by matching histograms which are a compact manner of representing an image and have proven to serve well in appearance-based topological approaches providing a good level of accuracy and autonomous navigation, [141, 150]. One of the problems of these methods is that because they represent the image globally in a histogram, spatial information is difficult to obtain, in [10], such information is incorporated by obtaining a multiresolution pyramid of multidimensional histograms for each image, leading to a more

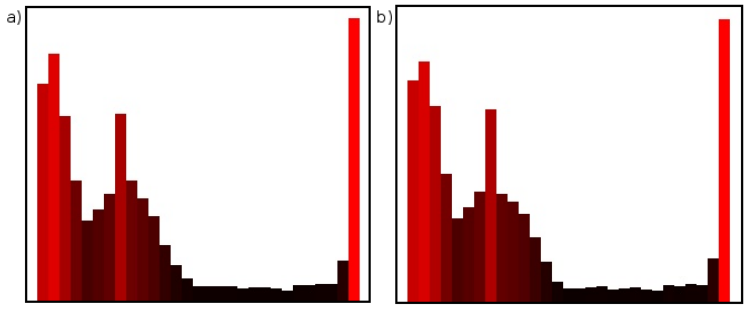


Figure 6.4: Histograms obtained from the images presented in Figure 6.3. Image a) represents the histogram obtained from the two images (backward and forward) at time t. Image b) represents the histogram obtained from the best match.

discriminative characterisation.

Our approach for image comparison relies on a global descriptor implemented in the form of a 1D grey-level histogram. Such minimalistic single-signature place characterisation enables reasonable discrimination, while on the other hand only requiring frugal computational resources. The signature of a location is obtained by sticking the images of both forward and backward cameras next to each other into a single mosaic which serves as the support for the calculation of the descriptor. Different methods for comparison were tested as well as multi-dimension histograms including cues such as gradients, colour or intensity, obtaining the best results using EMD [116] over 1D histograms of grey intensities (see Figure 6.4).

Loop-closure detection is achieved by comparing the most recent location signature against all the previously visited locations. Thanks to the compactness and simplicity of place characterisation, such an exhaustive retrieval procedure can be executed efficiently. Once a candidate for loop closure is found, time consistency is ensured to cope with perceptual aliasing. To this end, we perform a comparison of the appearance of 7 locations $L_{\text{recent}} = \{V_{i-3}, \ldots, V_i, \ldots, V_{i+3}\}$ centred in time around the location of interest V_i, with 7 locations $L_{\text{previous}} = \{V_{j-3}, \ldots, V_j, \ldots, V_{j+3}\}$ similarly sampled around the potentially loop-closing location V_j. This yields a 7×7 matrix $C = \sum_{i,j=1,\ldots,7} C_{ij}$, where each entry corresponds to the distance (in appearance-space) between locations $V_i \in L_{\text{recent}}$ and $V_j \in L_{\text{previous}}$. Note that the procedure has to be postponed until the locations V_{i+1}, V_{i+2} and V_{i+3} are visited and added to the map.

Asserting the time-consistency of a potential loop-closure requires an evaluation of the entries of C which takes care of the heading direction of the robot (see Figure 6.2). However, because of the configuration of the rig retained in our approach, we only need to distinguish between situations where the relative orientation from one passing to another is either 0° or 180°. Therefore, we review the scores of all the elements on

both diagonals of C, and only if they are all below some threshold for one diagonal the loop-closure is accepted. This is enforcing the consistency of the appearance over neighbouring locations in time around both the location of interest and the potential loop-closing location in such a way that the relative orientation of the robot between the 2 passings is properly taken into consideration.

6.3.3 Graph Relaxation and Map Optimisation

Due to the topological nature of our map representation we can optimise the poses by minimising the error between robot positions every time a loop closure is found. Every time a new graph relaxation is applied the map becomes more consistent to the real metric map and therefore becomes usable for navigation and path planning. In our approach we used TORO [48] which provides a highly efficient gradient descent-based error minimisation for constrained graphs.

In order to enable accurate obstacle avoidance and path planning, a global free space map M of the environment is built incrementally as the robot navigates (see Section 6.3.4 for details about free space detection). To this end, we associate to every keyframe of the robot trajectory a relative local free space map M_t, every vertex of the graph G being now represented as $V_t = (I_t^N, X_t, M_t)$, which is a simple 2D occupancy grid with a resolution of $1cm^2$ per cell and whose origin is anchored according to the position and orientation of X_t. To ensure the consistency between topological and global free space maps, M is updated based on the optimised vertex positions whenever the topological graph is relaxed.

It is true that with our featureless technique for loop-closure detection, an accurate position of the robot is not obtained right away. Yet, this is important to impose precision which improves the graph of poses after relaxation. Therefore, to obtain a good relative position estimate at loop-closure between current X_t and past X_p poses, we approximate the displacement $\Delta P = (\Delta x, \Delta y, \Delta\theta)$ between them by aligning the contours of their respective local free space maps M_t and M_p, which can be simply done by solving a cost function minimising the distance between points in those contours:

$$F^{C_1C_2} = \min(\sum_{i,j} \text{dist}(P_i^{C_1}, P_j^{C_2})) \,, \tag{6.1}$$

where the function $\text{dist}(\cdot,\cdot)$ is evaluated obtaining the 2D Euclidian distance between $P_i^{C_1}$ and $P_j^{C_2}$. The cost function of Eq. 6.1 is solved by exhaustive searching. However, a gradient descent method could be used to reach the minimum in less steps.

To obtain the contours we first compute the relative free space maps of the two corresponding robot poses obtained by the loop closure. The relative maps are computed in a window of 4 previous poses, from them the corresponding contours and

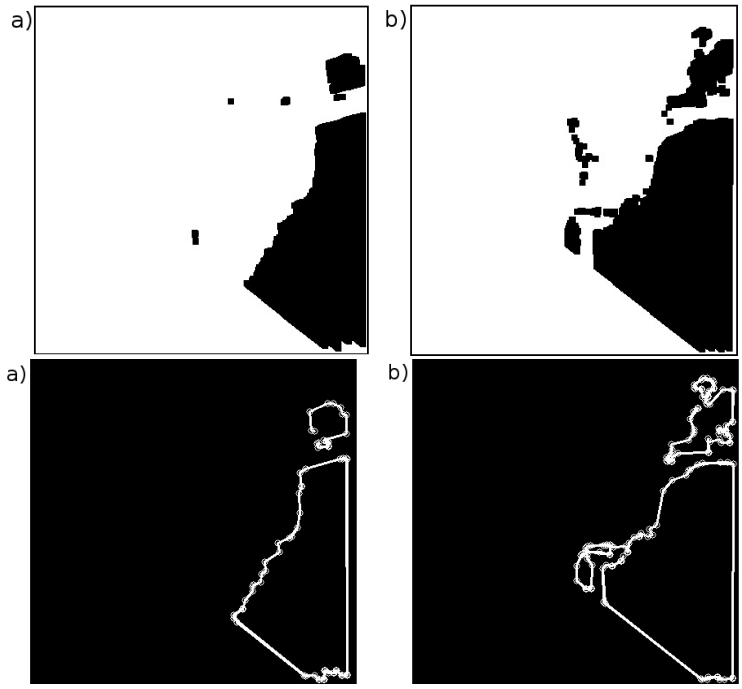


Figure 6.5: These figures show the relative free space maps computed to obtain the relative displacement between forward-forward loop closures. The top images show the free space maps and on the bottom images we can see the contours used for alignment. Image a) represents current robot pose and image b) represents the best keyframe match (the corresponding images can be seen in Figure 6.3).

some points along the contours are extracted. Regions within the free space map with small areas are discarded (see Figure 6.5).

With this method we can obtain an estimate of the displacement $(\Delta x, \Delta y, \Delta\theta)$ for forward-forward loop closures. However, when the robot revisits a location with an opposite direction to the one of the previous passing, it is difficult to match the free space maps, as they do not overlap very much. In such situation though, we can still approximate the relative orientation between the two views by calculating the mean horizontal optical flow between the corresponding images (obtained by the loop closure detection) and this relative orientation is used in the loop closure constraint. We found out that this estimate was better for $\Delta\theta$ in forward-forward and forward-backward loop closures, therefore we decided to use the optical flow estimate in all the cases.

6.3.4 Free Space Mapping

We have developed an accurate algorithm which incorporates geometry and colour information. Our solution assumes that the robot is moving on a plane which is the same

over the whole room, and that the floor is made of large regions with homogeneous colour.

Under the floor planarity assumption, it is possible to map the image pixels of a downward looking camera to the 2D cells of a local free space map of the visible floor in front of the robot. This transformation H_f is a homography which is calibrated once each time by the simple procedure of matching four artificial landmarks on the floor with the corresponding pixel positions in the image.

The inverse mapping H_f^{-1} can be employed to retrieve from the image the RGB value X_i associated to any floor cell i visible in front of the robot, so as to determine if this cell is free of obstacles or not. This is done here by calculating the log-likelihood ratio of occupancy, as follows (statistical dependency on the model is omitted for simplicity):

$$\ln \frac{P(C_i = F|X_i)}{P(C_i = O|X_i)} = \ln \frac{P(X_i|C_i = F)}{P(X_i|C_i = O)} + \ln \frac{P(C_i = F)}{P(C_i = O)} \ , \tag{6.2}$$

where C_i is the class of cell i (F for "floor", O for "obstacle"). We assume constant empirically fixed priors, and a uniform "obstacle" likelihood model, while the "floor" likelihood model is a mixture of L gaussians similar to the one used by Thrun *et al.* [26] for road detection:

$P(X_i|C_i = F) =$

$$\frac{1}{\sum_{j=1}^{L} w_j} \sum_{j=1}^{L} w_j \frac{1}{(2\pi)^{\frac{3}{2}} \sqrt{|\Sigma_j|}} \exp^{-0.5(X_i-\mu_j)'\Sigma_j^{-1}(X_i-\mu_j)} \ , \tag{6.3}$$

where μ_j, Σ_j, w_j respectively are the mean, covariance matrix and mixing coefficient that parametrise the Gaussian i. When the occupancy ratios have been calculated for every cell, the model parameters are updated according to the procedure proposed in [26].

Two models overlap when the next criteria is achieved:

$$(\mu_L - \mu_T)^T (\Sigma_L + \Sigma_T)^{-1} (\mu_L - \mu_T) \leq 1, \tag{6.4}$$

then the models are updated by:

$$\begin{aligned} \mu_L &\leftarrow (m_L \mu_L + m_T \mu_T)/(m_L + m_T), \\ \Sigma_L &\leftarrow (m_L \Sigma_L + m_T \Sigma_T)/(m_L + m_T), \\ m_L &\leftarrow m_L + m_T, \end{aligned} \tag{6.5}$$

using only those cells whose ratio is above some threshold. Initially, the gaussians

are learnt in the very first frame using only a small region at the bottom-centre of the image (that is assuming that at startup, the corresponding area on the floor is free of obstacles and is a good representative sample of the overall appearance of the floor).

For better robustness and more efficient planning during navigation, the local occupancy grids corresponding to several consecutive robot poses are fused together (see Figure 6.6): not only does this provide a medium-sized free space map around the robot which is more suited to navigation, but also makes it possible to filter out incorrect obstacle detections due to inaccurate probability scores in the presence of noise in the image or illumination changes in the scene. Similarly, as already mentioned, a global free space map M of the explored environment can be computed at anytime by fusing all the individual local occupancy grids (see Figures 6.9 and 6.10).

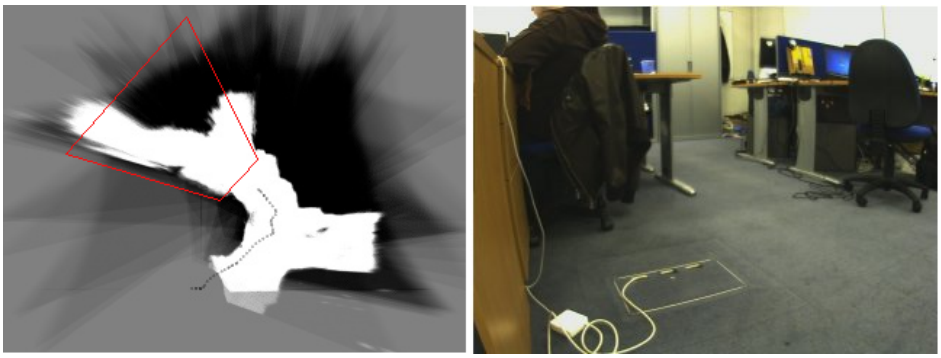


Figure 6.6: The image on the left shows part of a map which has been incrementally built, the region inside the marked area is the current frame being mapped which corresponds to the image on the right.

6.3.5 Autonomous Navigation and Exploration

We use the Dynamic Window Approach (DWA) [40] to plan smooth motions of the robot around nearby obstacles. Having estimated the free space around the robot based on a local window of a fixed number of recent keyframes, the local occupancy map is binarized to obtain boundaries between free-space, obstacles and unexplored areas. Every cell detected as an obstacle boundary in our map is used in DWA to plan the next velocity commands. DWA plans the best route from a current robot position to a target location, avoiding obstacles, and therefore requires the definition of robot goals (see Figure 6.7). We have considered it outside the scope of this paper to investigate ideal exploration strategies, finding that a heuristic technique was sufficient in our application. A goal is randomly selected relative to the robot position and around a square window of $3m \times 3m$. If the robot has spent much time around the same area then a goal is selected further away.

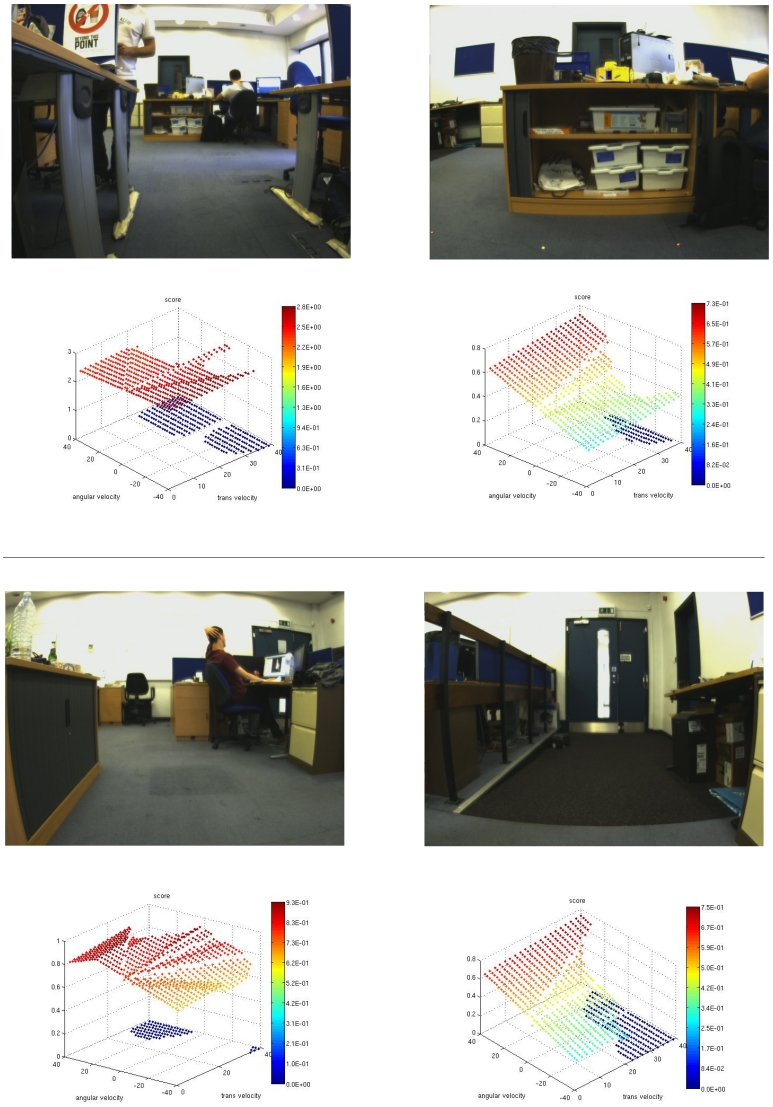


Figure 6.7: The graphs on the bottom of every image, show the output of the DWA algorithm on a particular frame. Axis represent: x - translational velocity, y - angular velocity and z - the final score. The algorithm selects the point with highest score and applies the velocities (x, y) to the robot.

It is also important to obtain a balance between mapping unexplored areas and obtaining accurate maps. Every time the robot has mapped an area completely it revisits previous places within the mapped area to find potential loop closures. By doing this we try to correct drifts in the odometry and also every cell is corrected according to the new robot position.

6.4 Experiments and Results

We have developed our experiments in an office of size 10×10m using a Pioneer robot platform. As can be seen in Figure 6.8, this environment represents a challenging scenario with strong perceptual aliasing and multiple narrow spaces (88cm), making both localisation and autonomous navigation difficult. The quality of our approach is demonstrated by performing manual and autonomous navigation with incremental and real-time free space mapping and loop closure detection.

Figure 6.8: Office environment with strong perceptual aliasing and challenging navigable spaces.

In this system we have decided to use the robot's internal odometry because it provides accurate data in the setting investiagated. Our scenario is indoor and loop closures tend to happen often and drift can be corrected. We could have used Visual Odometry instead but at the cost of using dedicated resources only for this task. The results presented in Section 7.3 evaluate and justify our inclination towards using the robot's odometry.

Our first experiments consider the effect of loop closure detection on map precision: it is well known that odometry-based robot position estimation will continuously drift over time until a loop-closure is detected and the inconsistency is compensated for. To safely navigate, it is therefore extremely important to correct the map accordingly as often as possible. As can be observed in the top left image of Figure 6.9, a constructed map using only odometry information with no loop-closure detection is highly inaccurate, and it would be almost impossible to navigate autonomously or to use the map for further high-level tasks.

When loop-closures are detected with only one camera (top right image), then it can be observed that both the robot trajectory and the map are significantly more accurate. When using two cameras, the number of loop closure detections increases, and so the map becomes even more accurate (see Figure 6.9, bottom).

We have investigated the impact of the downward looking angle of the camera

used for free space detection on the quality of the map (Figure 6.9, bottom images). We have found that if the camera is oriented such that it is only covering a very restricted region (i.e. within 20cm to 110cm) in front of the robot, then the obtained map is very detailed, enabling very accurate obstacle localisation. However, such a downward looking angle penalises motion planning, as it only provides very local information, with obstacles being detected very late, only when the robot is very close to them (see the bottom right part of the figure). When the camera is covering a larger region (i.e. within 60cm to 450cm) in front of the robot, further away obstacles can be detected, enabling more efficient motion planning over a wider time window, leading to smoother robot trajectories (bottom left of the figure).

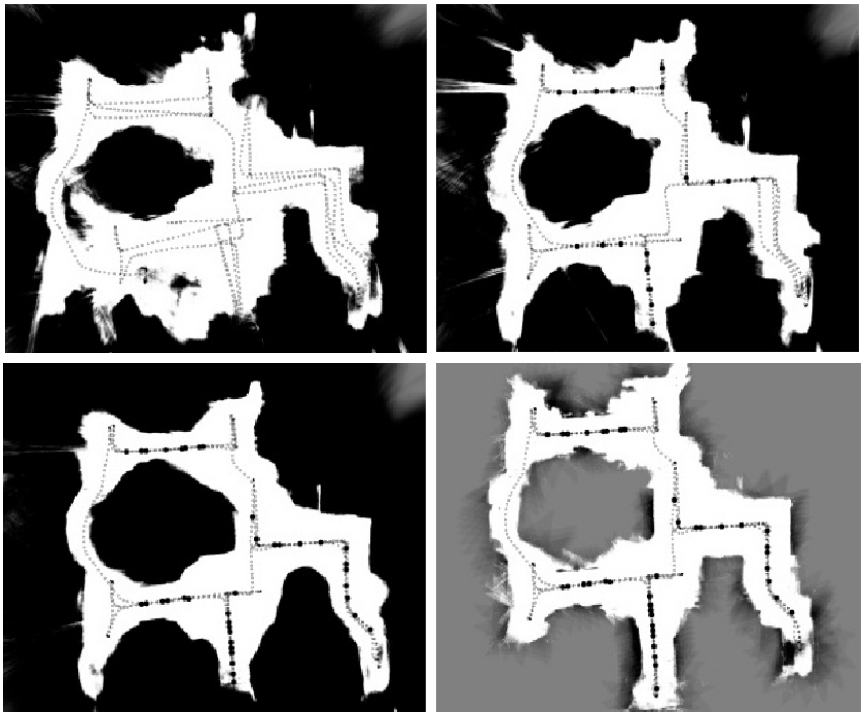


Figure 6.9: Free Space Map built by driving the robot manually (700 keyframes). Top left: map without loop closures, top right: map with loop closures coming from the camera facing forward, bottom left: map with loop closures from both cameras, bottom right: map built using a camera facing to the floor.

To measure the metric quality of our free space map, we have picked 8 random robot positions distributed over the whole room, and compared their coordinates in the map with ground truth obtained by manually measuring the coordinates of these points on the floor. This comparison, presented in Table 6.1, demonstrates the accuracy of our solution, with a mean 2D error of 6.39cm, and a max error of 10.77cm.

With our system the robot was able to successfully achieve several autonomous navigation runs of approximately 20min each.

Ground truth comparison (cm)			
Robot Pose	Ground truth	Robot Pose	Ground truth
(182, 0)	(180, 0)	(96, 445)	(90, 450)
(212, 145)	(210, 150)	(-246, 487)	(-240, 480)
(425, 143)	(420, 140)	(89, -325)	(90, -330)
(512, -271)	(510, -270)	(-280, -326)	(-270, -330)

Table 6.1: Evaluation of Localisation Accuracy. The robot poses were determined by measuring distinctive natural landmarks in the floor (intersection of carpet tiles)

We achieved this by using a relative freespace map computation on every keyframe and using it for local obstacle avoidance, the number of past keyframes we used is a tradeof between having more information but with the cost of increasing computation and the local error due to drifts in previous positions. We found that using only 5 previous keyframes was enough to navigate safely. Another important factor is that the camera oriented to the floor was the most suitable to use since we can adjust the gap between the vertical field of view and the robot which is much less than the camera used for loop closure detection and therefore it can react much better when the robot is close to obstacles.

Figure 6.10 shows examples of global free space maps, providing a comparison of the two downward looking angles already used earlier, again proving the superior accuracy of the proximal sensing configuration. The purpose of this experiment is to demonstrate the reliability of our complete solution comprising simultaneous localisation, mapping, free space detection and navigation: the exploration strategy here is deliberately simplistic, and used only as a proof of applicability of our solution (more advanced policies would certainly result in more efficient exploration).

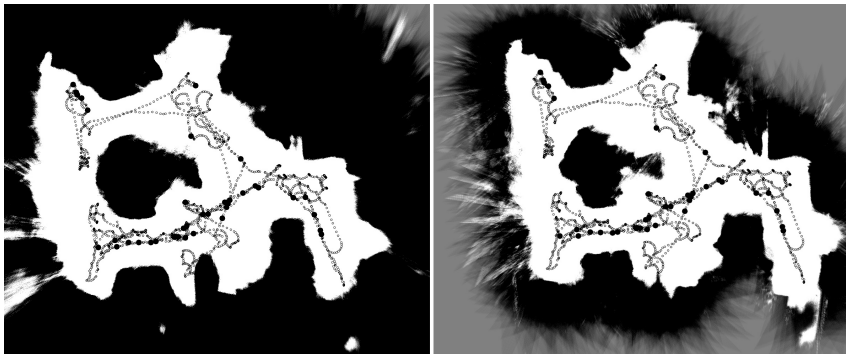


Figure 6.10: Free Space Map built autonomously using 2500 keyframes

6.5 Summary

In this chapter, a lightweight vision-based system for autonomous navigation and exploration has been developed. It has been shown to be effective for surprisingly high performance localisation, and similarly effective for fully autonomous robot navigation and exploration. This system puts together odometry-based trajectory estimation, front/back vision-based loop closure detection, free space identification from a local obstacle detection camera and a forward-looking camera, and local and global occupancy mapping. Our approach relies on an ad-hoc rig of multiple standard cameras, and we have investigated the role each camera should effectively play for the best performance.

7

Camera Placement for Multi-Camera Navigation

In the previous chapter we demonstrated that an effective visual SLAM and navigation system can be put together for a certain domain using a set of lightweight techniques, either off-the-shelf components or simple methods. An important concept in this general framework is the idea of assigning different roles to each camera. For instance, one or two cameras can be assigned to SLAM, one to obstacle avoidance, and another could be specific to object recognition. This is an advantage with respect to omnidirectional cameras. With a camera rig, different lenses could be used and the cameras can be attached in different positions depending on the application to be developed.

7.1 Camera Placement for Loop Closure Detection

In Chapter 6, we developed an integrated approach for autonomous robot navigation and exploration. In that system, three cameras were used: two for appearance based SLAM and the third one for free space mapping, obstacle avoidance and exploration (Section 6.3.1).

The first time we faced the problem of camera positioning was when developing our appearance based SLAM approach. In our early attempts at developing this capability, the first idea was to try to cover the scene as much as possible in such a way that when using four extrinsically uncalibrated cameras we could cover a 360° panorama. However, this is dependable on the number and position of the cameras, and the field of view (FOV) of the lenses. Our cameras' lenses have horizontal FOV of approximately 80°, so when placed on the four corners or the robot with maximal separation there were still gaps in the overall field of view. The thinking was to try to maximise the number of loop closures when the robot arrived to the same position but with different headings. However, we discovered that with our lightweight approach for loop closure detection based on histograms of composite images, it was more difficult to find correct matches mainly due to the gaps between the uncalibrated images. These gaps caused particular problems due to the closeness of the scene in our indoor setting: small rotations and translations of the robot sometimes cause quite different parts of the scene to come into view, particularly in the parts viewed by sideways-facing cameras.

After experimentation, a suitable ad-hoc arrangement of cameras for loop closure detection was discovered. This consisted of one camera facing forward and one backwards with approximately the same downward angle. Because of the narrow spaces of our indoor scenario, places visited by the robot were much more likely to be revisited with the robot travelling in the same or in the opposite direction than when they were first traversed rather than at arbitrary orientations — this and many other indoor scenes are essentially 'corridor-like' in nature. Extended testing reaffirmed that our configuration was correct for this scenario.

7.2 Experiments on Camera Positioning for Loop Closure Detection and Free Space Mapping

Once we found an suitable number of cameras and their corresponding position for our SLAM approach, a second positioning problem arose when computing the free space map. The horizontal angle of both cameras was modified symmetrically from 0° – 30° with 5° increments. In Figure 7.2 we can observe the area covered in one frame with the different angles.

We found a trade off between number of loop closures, false positives and camera coverage. For instance, if the image plane is perpendicular to the ground plane (camera angle 0°) then the image contains more information increasing the distinctiveness of the image and the number of true loop closures. While the camera angle is moved towards observing more of the floor then the images get more similar and it is easier

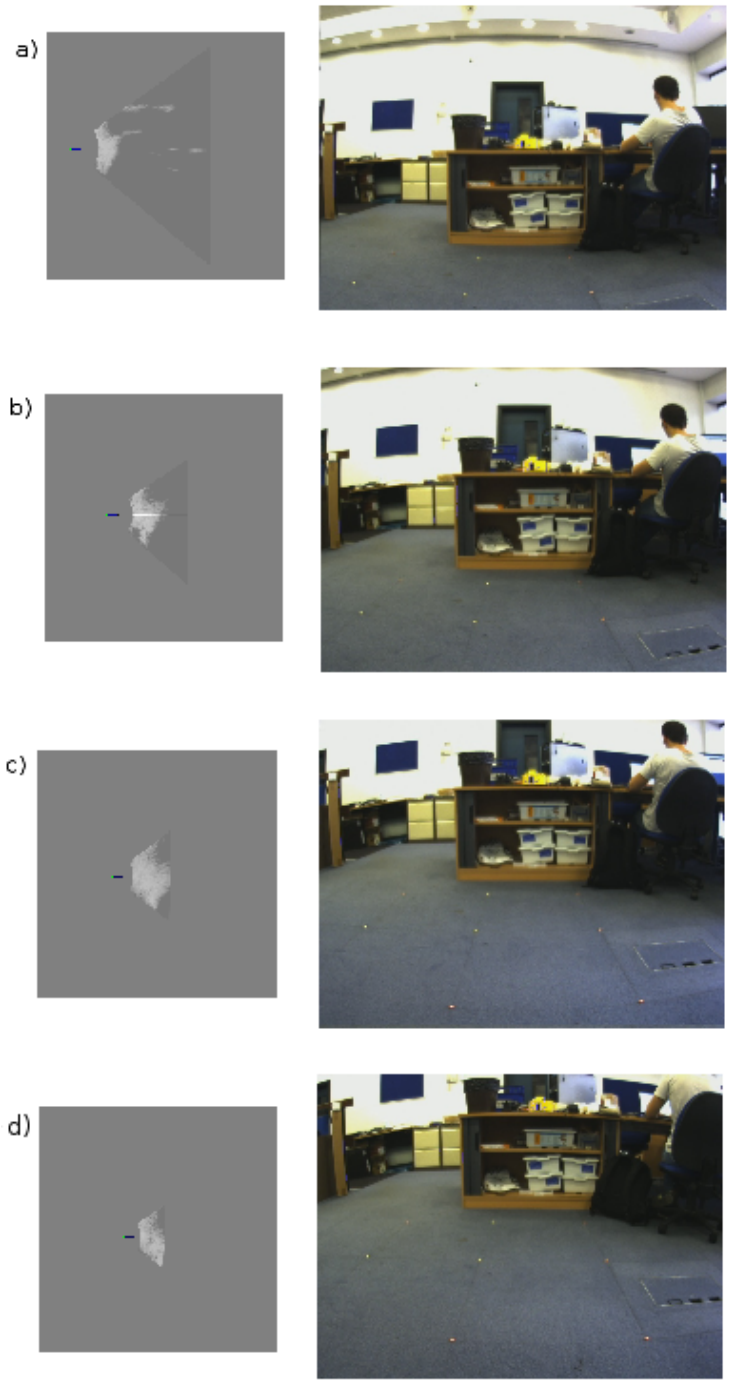


Figure 7.1: This figure shows the area covered per frame when modifying the vertical camera angle: a) 0°, b) 5°, c) 10°, d) 15°. On the right are the corresponding images.

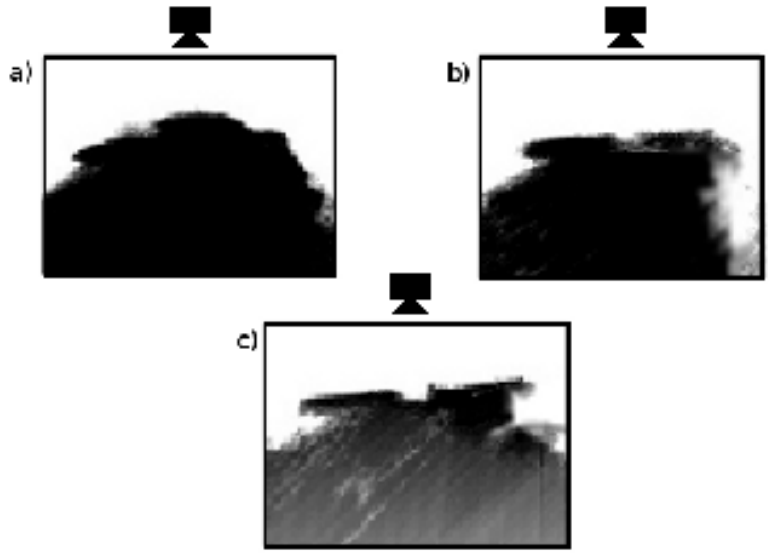


Figure 7.2: This figure shows a zoomed area of the computed free-space map with different downward camera angles: a) 0°, b) 15°, c) 30°. The white area represents free space and the black area occupied space

to obtain perceptual aliasing incrementing the number of false positives.

On the other hand, in its free space mapping role, when we move the camera angle with respect to the floor we observe two characteristics: first, as the camera angle is moved to observe more floor, the area covered is reduced; for instance, when the camera is completely parallel to the floor the area being mapped is more than when the camera is at 5°. This is an advantage for long trajectory planning. Second, when we move the camera angle towards observing more floor we are reducing the distance between the robot and the first cell being mapped — there is a small 'blind spot' in front of the robot. When the camera angle is 0° there is a substantial gap of around 1m between the robot and the first cell mapped. This becomes a problem because the robot cannot navigate too close to obstacles or it might not be able to navigate through narrow spaces. When the camera angle is 30° the gap is reduced and the distance between the robot and the first cell mapped is $\sim 15cm$, allowing the robot to react better to close obstacles and to drive in narrow spaces. Another feature is that as we increase the angle and the camera is observing more floor area, the level of detail is much better. Normally, features that are less than 1 cm in width are not mapped correctly when the camera is almost parallel to the floor. This can be observed in Figure 7.2.

We have performed a series of experiments where the camera angle was moved

Camera Angle	0	5	10	15	20	25	30
Number of Loop Closures	24	25	26	25	29	31	32
Number of False Positives	0	0	1	3	4	7	13

Table 7.1: Evaluation of camera positions as we modify the vertical angle with respect to the floor. The second row shows the total number of loop closures. The third row indicates the number of false positives.

every 5° and we measured the number of true loop closures and false positives. In Table 7.1 it can be observed that when the camera is parallel to the floor there are 0 false positives. When the angle is 5° we also obtain no false positives and 1 more loop closure compared with the 0° position. This might be because when the camera is completely parallel, the image contains more of the ceiling which is of a uniform colour. It can also be observed that the number of false positives increases with the inclination of the camera.

In Figure 7.3 we show the different maps obtained with different camera inclinations corresponding to Table 7.1. As mentioned before, black areas are occupied space, white areas are free space, and grey areas represent unmapped cells. It can be seen that the map coverage decreases with respect to the inclination of the camera but also there is also more detail in the map when the camera is facing more to the floor.

We can conclude that it is useful to assign different roles to the cameras. For our purposes we found it to be a suitable configuration to assign only one camera for free space mapping and obstacle avoidance. This camera makes the robot more reactive to obstacles when performing local free space mapping; it can also navigate in narrow spaces and it can be aware of objects closer to the robot. Two other cameras are used for appearance-based slam and for loop closure detection. Also, in future developments these cameras could also be used for higher-level tasks such as object recognition or human interaction.

7.3 Experimental Investigation of Camera Positioning for Visual Odometry

In this thesis so far, we have not worked with visual odometry, although it is clearly a standard capability which should be considered for a mobile robot navigating based on vision. In the previous chapter we took the choice to base local motion estimation on wheel odometry, due to the ease and good performance of our robot's odometry in the setting considered. However, there will clearly be many cases where wheel odometry is not a good solution.

This section is therefore dedicated to experiments on visual odometry approaches

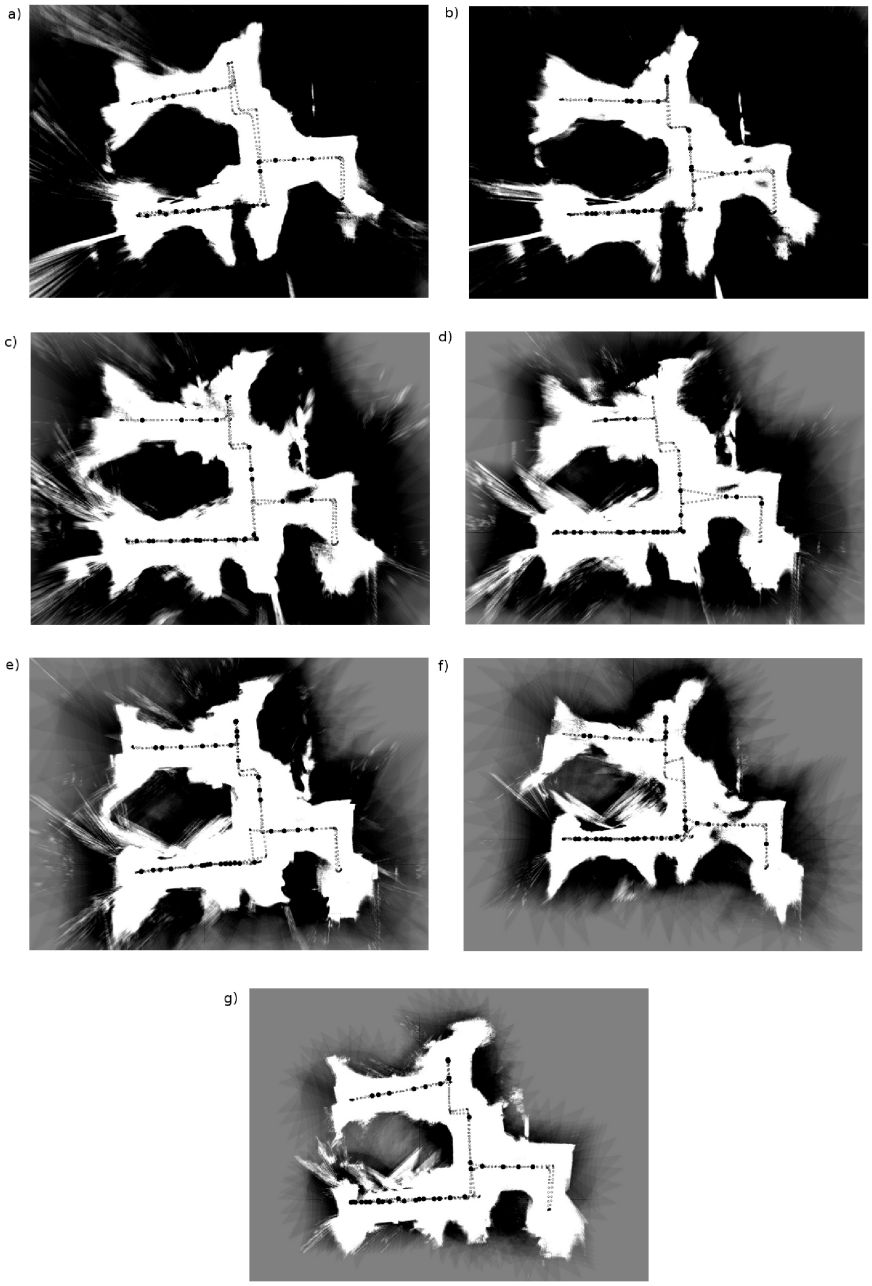


Figure 7.3: This figure shows the different total free-space maps computed with a varying vertical angle of the cameras (forward and backward) with respect to the floor of: a) 5°, b) 10°, c) 15°, d) 20°, e) 25°, f) 30°.

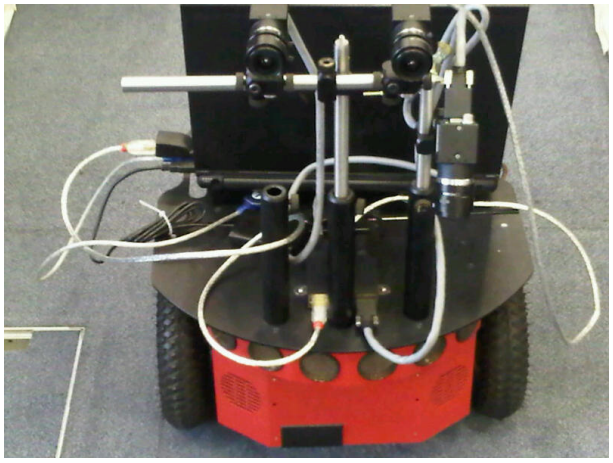


Figure 7.4: Camera rig configuration for our visual odometry experiments: stereo rig setup at the front of the robot and a single camera observing the floor.

with our robotic platform. Given a configurable rig of cameras, we have considered two cases: stereo and monocular odometry (see Figure 7.4). With these two cases we had compare the outputs of the corresponding algorithms against ground truth and the robot's internal odometry.

The stereo set up consisted of two cameras facing forward and calibrated using Matlab's calibration toolbox. Once calibrated, different sequences were taken and the images obtained for each sequence were rectified using OpenCV's library. The stereo VO library used for testing our sequences was an out of the box solution developed by Kitt *et al.* [62] and released under the name *'libvisio'*. It is a cross-platform C++ library for computing the 6 DoF motion of a stereo rig. The method is based on trifocal geometry between image triplets and a Kalman filter combined with a RANSAC outlier rejection scheme is used to estimate the frame to frame motion.

On the monocular visual odometry side we have tried our lightweight LK-RatSLAM implementation (Section 5). Given that the algorithm does not require camera calibration or any particular position we have tried two single camera locations: facing straight forward; and also looking down to the floor. We have found that the camera that presents better results in our indoor scenario is the one looking towards the floor.

In Figures 7.5 and 7.6 we can observe two trajectories followed by the robot, consisting of 690 and 890 frames respectively. In both sequences, the camera looking towards the floor (red) follows a better estimated trajectory than the camera facing forward (green). This is mainly because in the office environment where the sequences were grabbed, the areas tend to be narrow in some parts and more spacious in others. The change in appearance this causes gives unreliable results in the case of the

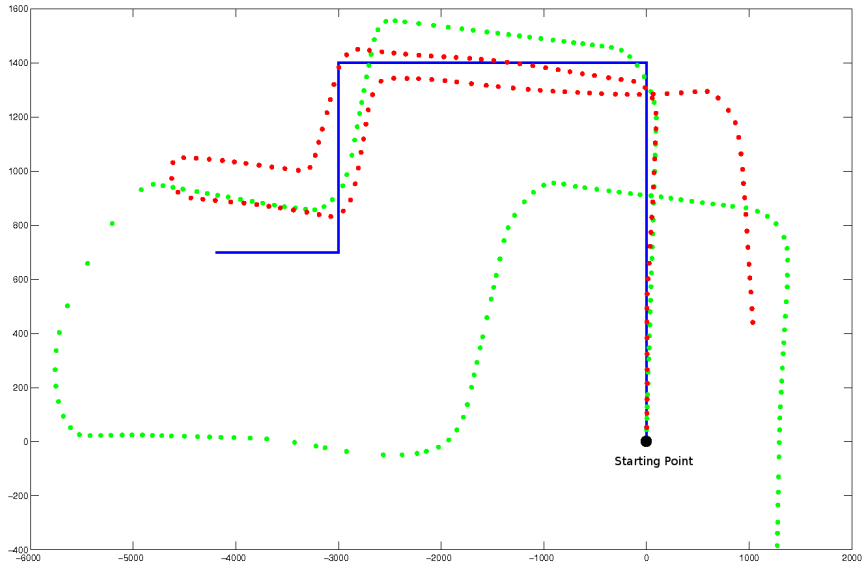


Figure 7.5: Visual odomety sequence number one: short sequence consisting of 690 frames. Monocular LK-RatSLAM visual odometry comparison: camera facing to the floor (red), camera facing forward (blue), ground truth (blue).

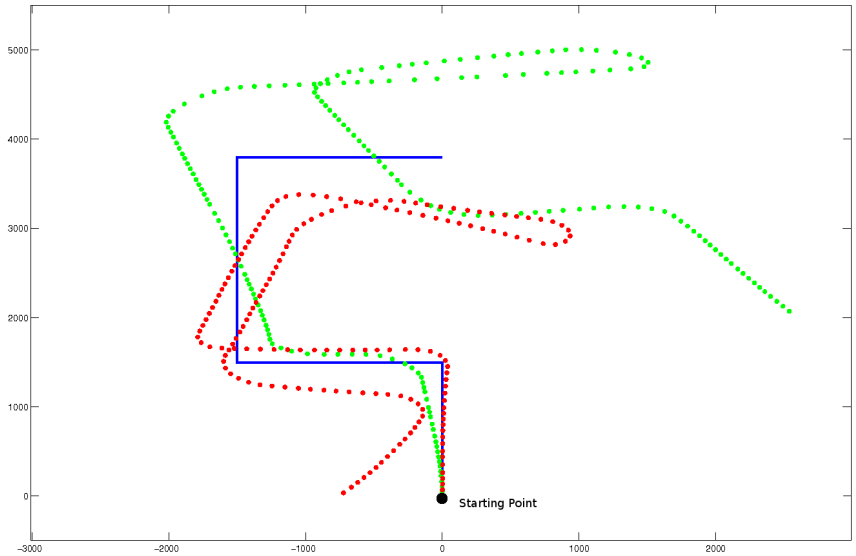


Figure 7.6: Visual odometry sequence number two: short sequence consisting of 890 frames. Monocular LK-RatSLAM visual odometry comparison: camera facing to the floor (red), camera facing forward (green), ground truth (blue).

forward-looking camera.

Even though there is not too much texture in the floor, the whole trajectory is more uniform. One of the problems with this lightweight approach is that since the displacement estimate is based on the amount of change in appearance from the previous to the current image, then when the robot is near to objects or obstacles the change in appearance between them is greater than when the robot is far from obstacles. This produces non-uniform displacements along the whole sequence on the camera facing forward and more uniform and accurate displacements with the camera facing to the floor.

Now that we have selected the camera for monocular odometry using LK-RatSLAM, we compare in Figures 7.7, 7.8 and 7.9, the trajectories computed by the stereo rig (Red), monocular (black), robot's internal odometry (green) and ground truth (blue).

The stereo rig and the camera used for monocular odometry are situated in the front part of the robot and the robot's odometry is given relative to the centre of the robot. Therefore, to obtain an estimate of the error in the trajectories we would need to obtain first the relative position of the stereo rig and the monocular camera with respect to the centre of the robot. This effect can be clearly observed in the Figures by looking at the stereo trajectory which is shifted almost uniformly along the whole trajectory. This can also be observed when the robot is turning on the spot, the stereo trajectory translates and rotates instead only rotate.

Figures 7.7 and 7.8 present short sequences of less than 1000 frames. Figure 7.9 presents a longer sequence of 1790 frames. From these images we can observe that the robot's internal odometry can be used successfully in this kind of environment and corrected every time a loop closure is found. Stereo visual odometry can also be used accurately but at the cost of having two specialised cameras for this purpose. On the other hand our lightweight VO approach does not work very well in this kind of environment.

We can say that by using the robot's internal odometry we can save the computational time and the resources that a stereo rig or some more elaborate and calibrated monocular VO approach could utilise. However, we can easily substitute the robot's internal odometry for VO if required, or we could formulate a hybrid approach to use VO in places where the robot's terrain might induce more error and switch to the robot's internal odometry when for instance there is not much texture in the scene.

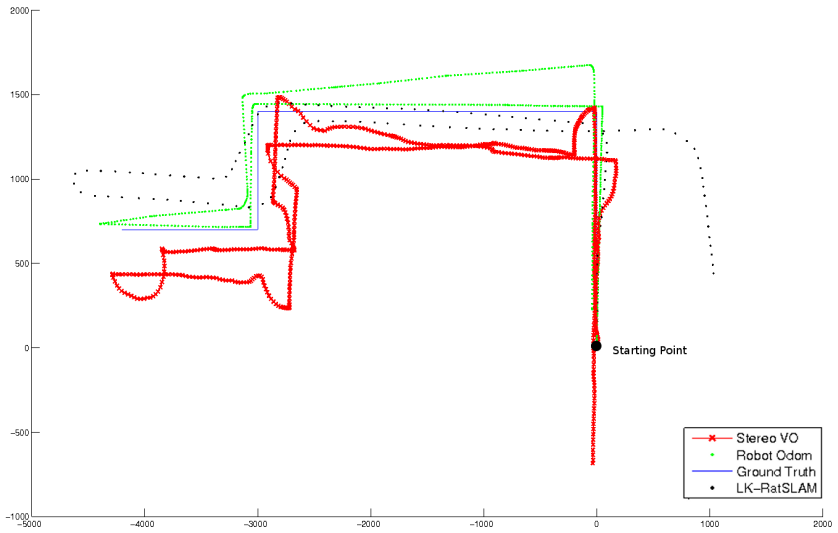


Figure 7.7: Visual odometry sequence number one: short sequence consisting of 690 frames. The figure shows the different paths computed by: stereo visual odometry (red), LK-RatSLAM (black), robot's internal odometry (green) and ground truth (blue).

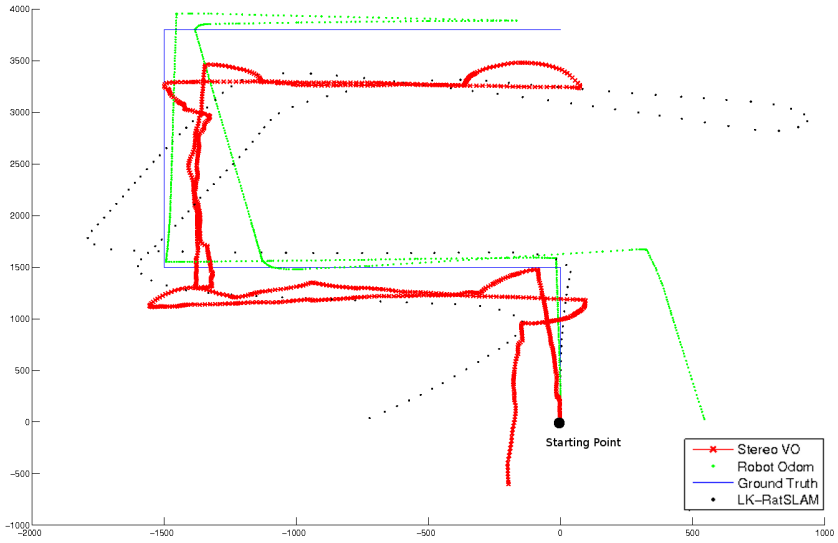


Figure 7.8: Visual odometry sequence number two: short sequence consisting of 890 frames. The graph shows the different paths computed by: stereo visual odometry (red), LK-RatSLAM (black), robot's internal odometry (green) and ground truth (blue).

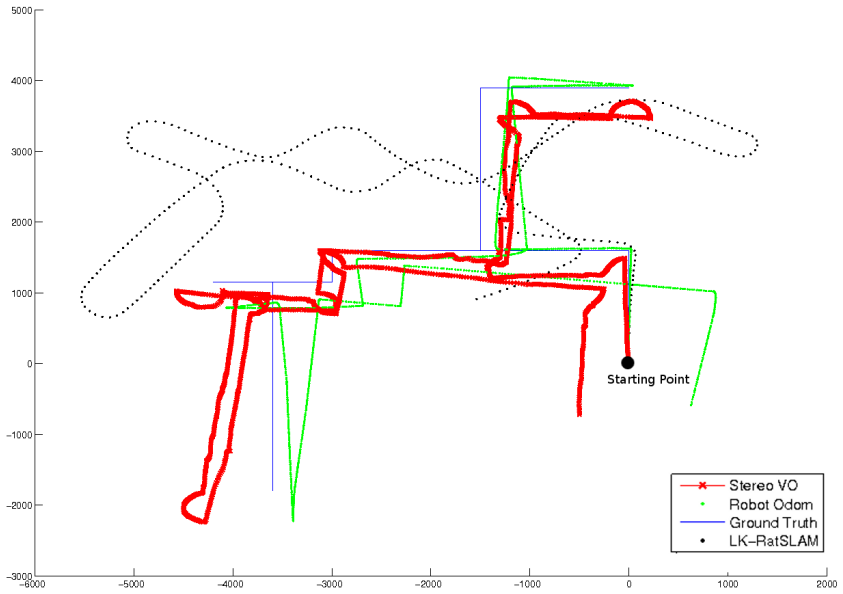


Figure 7.9: Visual odometry sequence number three: longer sequence consisting of 1790 frames. The graph shows the different paths computed by: stereo visual odometry (red), LK-RatSLAM (black), robot's internal odometry (green) and ground truth (blue).

7.4 Multi-Camera Rig Design for Mobile Robots — Discussion

In mobile robotics we can think about task-specific robots and more general robots that can perform different tasks. The aim of this section is to discuss different designs of a multi-camera set-up that could potentially improve the performance of the robot. As we previously mentioned, by having a multi-camera rig we aim to obtain some advantages over omnidirectional cameras, such as: higher resolution, different camera positioning, individual roles depending on the performed task and the capability of using different cameras or lenses.

In the previous sections of this chapter we performed some experiments that showed the advantages and disadvantages by having different camera positions for our navigation system. For our type of scenario it turned out most useful to use one camera forward and one camera backward for the purpose of localisation (given that we could rely on wheel odometry for local motion estimation), and one camera at an angle looking at the floor in front of the robot for free space detection and obstacle avoidance. This result was based on maximising the number of loop closures while also having the ability to build a more detailed free space map and to move safely around narrow spaces. This led us to assign specialised roles to our cameras: forward and backward for localisation and the camera looking to the floor for free space detection and obstacle avoidance.

Multi-camera placement in mobile robotics has not been a subject of much study, possibly because of the inherent difficulty of adding more devices to a system. In our system, we counteract this idea by developing lightweight methods.

A very recently presented piece of work [106] made use of two cameras to compute visual odometry for a skid steered outdoor robot — one forward looking, and one facing to the floor. This type of robot controls the left and right velocities of its wheels independently and the wheels tend to slip when the robot is turning, incrementing the internal odometry error and making it very challenging to compute the heading direction (see Figure 7.4). Both cameras compute VO based on optical flow of tracked features. The information obtained from each individual camera is then fused with a Kalman Filter to obtain a better odometry estimate. The motivation for using both cameras is that the forward facing camera can be easily affected by environmental changes and is less sensitive to small motions. On the other hand, the floor-facing camera is affected by high speed movements. We can observe that the work presented on the previous paper supports the idea of having multiple cameras and advantages can be obtained from each of them. In Chapter 6 we presented a system that uses a multi-camera system with different roles assigned to the cameras. In our case a camera

Figure 7.10: Skid steered robot [106] : two cameras are used on a robot to compute VO: one facing to the floor, the other facing forward.

facing to the floor was found suitable for free space detection and obstacle avoidance, the work of [106] supports the idea that it could also be used to estimate VO as we did in Section 7.3.

Depending on the performed task, one camera configuration could be more useful than others. We confirmed that our forward-backward configuration for navigation was also useful in an urban search and rescue task [59]. In this task a robot is normally teleoperated and the level of autonomy might be minimal. However, this camera configuration improved the level of awareness for the teleoperator.

There are other works that use active vision hardware and calculate view-point direction dynamically. For instance, in [147] a view planning strategy is developed based on information maximisation considering two tasks: localisation and object tracking. Two different stereo camera pairs are used and assigned to each individual task: a wide angle (60° fov) one for localisation and a conventional one (30° fov) for object tracking. The system provides the best position for each camera such as the task can be achieved optimally. However, the system was only tested in simulations on a humanoid robot.

In the state of the art robot PR-2, (see Section 1.3) a large number of visual sensors is mounted on the robot, most of them in the head and others in areas that help in specific tasks such as in the hands to grab objects more accurately. It is interesting to note that a laser scanner is situated at the base of the robot. This suggests that it is still useful to have a low-down sensor for obstacle avoidance tasks, similar to our camera looking to the floor.

We can see that different robots have been subject to various choices with regard to the number, types and positions of the cameras mounted. It might be difficult to agree on an general optimal camera placement and an optimal number of cameras. What we can observe is that it is useful to have different sensors on different parts of the robot.

For instance, an industrial robot that has to navigate in cluttered environments might find it useful to have a stereo camera and a camera pointing to the ceiling on the head. The former could be useful for object inspection and recognition, and the latter perhaps for visual odometry and localisation using texture on the ceiling; another sensor could mounted be at the bottom of the robot for obstacle avoidance. This clearly remains an interesting area to investigate with further research, but we feel that the experiments and discussion in this chapter make clear that the lightweight, multi-camera approach we have taken in the second half of this thesis is highly adaptable and give good possibilities for robust navigation performance in many scenarios with a little re-design and configuration.

In the next and final chapter we will provide our opinions and conclusions to the work done in this PhD thesis.

8

Conclusions

This chapter provides a closing discussion of the work developed in this thesis and what are believed to be the novel contributions. It also suggests ideas for further research in the areas that we have worked on.

8.1 Novel Contributions

The main motivation of this work was to investigate practical solutions for vision-guided robot navigation such as might be suitable for mass-market service robotics in the home and beyond over the coming years. In particular, our research has focused on providing all the sensing capabilities required for navigation by a rig of multiple standard cameras rather than more specialised devices.

We have investigated two main paradigms. The first, more obvious approach is to extend the well-known metric localisation and mapping techniques developed in recent years for single cameras or binocular stereo systems to the case of a reconfigurable multi-camera rig. Building a metric map from multiple cameras, with the possibility of full sharing of that map between the different views, requires very accurate knowledge of the relative locations of those cameras as they are mounted to the robot. Since we

would like to avoid any assumptions or restrictions on the camera positions, this motivated the development of an automatic calibration procedure for the extrinsic locations of the cameras mounted on a robot which could cope even in the case where cameras' fields of view have no simultaneous overlap; and which works without the requirement for calibration patterns or other infrastructure, so that it could be run 'in the field' by a mobile robot whenever its cameras might have been adjusted or even accidentally knocked. Our main assumption is that the cameras can capture synchronised video.

Our calibration procedure, detailed in Chapter 4, relies on the creation of independent feature-based monocular SLAM maps as the robot executes short a pre-planned motion including a full circular loop. These maps will contain shared features, and these are matched using invariant features and then aligned approximately into a single coordinate frame. A final joint bundle adjustment of the whole map with the constraint that the cameras are rigidly mounted with respect to one another achieves very accurate inter-camera reprojection results, proving the global registration achieved. We have demonstrated the technique with both different two-camera configurations and a four-camera, fully non-overlapping rig.

Given this system for multi-camera auto-calibration, we could have chosen to continue down the metric route by developing a system for autonomous navigation that requires calibrated cameras, presumably starting from metric feature-based SLAM. However, we decided at this stage to switch attention to a second major paradigm for achieving autonomous robot capabilities based on the combination of 'lightweight' vision techniques. There had been a number of recent publications showing surprising achievements in SLAM using the right combination of really quite simple methods, and in Chapter 5 we give a thorough review of the particularly impressive RatSLAM approach of Milford and Wyeth [92], discuss our own implementation of the visual odometry part of this system and present an improvement which gives better results in situations different from the road-going vehicle experiments in their work.

Our main motivation was to advance this line of research to determine whether fully autonomous robot navigation could be achieved based on lightweight vision techniques. In this research, we continued to work with a multi-camera rig, but broke away from the idea that accurate inter-camera calibration was necessarily required. Instead, each lightweight component of our approach should work independently with the video from one or perhaps just two inter-calibrated cameras and provide input to an overall representation with a metric-topological character.

Our concept of a lightweight vision technique has characteristics which include at least some of the following aspects:

- An easy to understand method which is easy to implement and modify.
- Little requirement for calibration.

- A ‘black-box’ method, which can be used without knowledge of internals.
- Computational efficiency.
- Modularity: can easily be combined with other techniques.

In Chapter 6 we presented a system for autonomous navigation around an indoor space based on this idea of a combination of lightweight vision techniques which permits robust, automatic and repeatable mapping and navigation. Our system comprises the following components:

- Local trajectory estimation, enabled here via wheel odometry.
- The selection of keyframes to form a graph-based map.
- Place recognition to detect the most frequent loop closure events using a front/back camera pair and histogram matching.
- Global map optimisation using the black-box TORO library.
- Computation of local free space maps for path planning based on a front down-looking camera using colour learning and segmentation and approximate camera calibration.
- Fusion of local free space maps into a standard log-odds format global occupancy map.
- Local path planning using the DWA technique.
- Simple goal definition for autonomous exploration while map-building.

We demonstrate the performance of this technique in several experiments in both manual driving and fully autonomous modes, and stress the robust properties of this system built from multiple simple components.

We round off the novel research in this thesis in Chapter 7 where we analyse in more depth the question of where cameras can most profitably be located on a mobile robot to best make use of lightweight vision techniques to achieve SLAM and navigation. Several trade-offs are considered, regarding the best locations of cameras to achieve visual odometry, loop closure detection and free space identification. Since there is a hardware cost associated with the use of every extra camera, these factors are then considered jointly to decide on some strategies for optimal placement of multiple cameras which most efficiently makes use of their capabilities, sometimes for multiple uses.

8.2 Further Research

There are several possible avenues for future research.

Firstly, with regard to the metric camera calibration work, the automatic extrinsic camera calibration system developed in this thesis provides accurate estimates of the poses of the cameras relative to each other and up to undetermined scale, but usually this will not be enough for practical use of the cameras in a mobile robot system. As a final step, we could estimate the orientation and scale of the whole rig with respect to the robot's own coordinate system. This part of the process, unlike those above, has elements which are specific to the particular robot in question. The Pioneer P3-DX robot is able to drive in straight lines, curves or rotate on the spot via differential drive. Therefore, part of the robot's pre-programmed motion could be set to be a straight line driven forward with distance estimated using odometry, and another part is a pure rotation on the spot.

Also, it would be interesting and beneficial to determine the intrinsic parameters of the individual cameras as part of the full calibration procedure, rather than requiring them to be known *a priori*. It would be straightforward to include and optimise these parameters in the final bundle adjustment step, but the problem is that the construction of individual camera maps using MonoSLAM would be inaccurate without well known intrinsics for each camera. This could be tackled using the approach recently proposed by Civera *et al.* [21] for sequential auto-calibration of a single camera. Another improvement would be to reduce the computation time during the full global optimisation step by exploting the sparseness of the optimisation problem in the final joint bundle adjustment scheme. This problem was recently considered by P. Lebraly [75].

Turning to the lightweight vision approach of the latter part of the thesis, it may be worth considering taking the specialisation route further and creating a system with heterogeneous camera types with even more elaborated configurations and roles. The appearance-based slam module could be tested in outdoor environments to observe how accurate and distinctive it is, and how the graph relaxation works in cases were the distances between loop closures are greater.

To find the relative distance between matches in a loop closure, we developed an approach that works in forward-forward loop closures. However, this approach does not work well in all cases, because it is based on the colour models and on local free space maps, and it might happen that the lighting conditions change and the local free space map is very different from its corresponding match, so the system needs more frames to adapt its colour models and the local free space maps cannot be used to find the relative displacement. Also, further work needs to be done to find how to obtain the relative displacement in forward-backward loop closures.

Finally, in the exploration phase, our method is goal oriented and at the moment the goals are chosen randomly. Exploring the frontiers can be a solution but we might face a problem of speed. In an optimal exploration algorithm, we would like to cover most of the space in a minimum of time. Different exploration algorithms can be applied which would lead to more efficient area coverage.

List of Figures

List of Tables

Bibliography

[1] RoboCup. URL http://www.robocup.org.

[2] Robotvision library, imperial college london. URL: http://www2.imperial.ac.uk/robotvision/, 2011.

[3] Y. I. Abdel-Aziz. Direct linear transformation from comparator coordinates in close-range photogrammetry. In *ASP Symposium on Close-Range Photogrammetry*, 1971.

[4] Y. Alon, A. Ferencz, and A. Shashua. Off-road path following using region classification and geometric projection constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[5] H. Andreasson, T. Duckett, and A.J. Lilienthal. A minimalistic approach to appearance-based visual SLAM. *IEEE Transactions on Robotics (T-RO)*, 24(5):1–11, 2008.

[6] R. Angst and M. Pollefeys. Static Multi-Camera Factorization Using Rigid Motion. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2009.

[7] R.T Azuma. A survey of augmented reality. *PRESENCE: Teleoperators and Virtual Environments*, 6(4):355–385, 1997.

[8] S. Baker, R. Patil, K. M. Cheung, and I. Matthews. Lucas-Kanade 20 years on: Part 5. Technical report, Robotics Institute, Carnegie Mellon University, 2004. Technical Report CMU-RI-TR-04-64.

[9] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2006.

[10] Paul Blaer and Peter K. Allen. A hybrid approach to topological mobile robot localization. Technical report, Computer Science Department, Columbia University, 2005. Technical Report CUCS-020-05.

[11] M. Bosse, P. Newman, J. J. Leonard, M. Soika, W. Feiten, and S. Teller. An atlas framework for scalable mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2003.

[12] J.-Y. Bouget. Pyramidal implementation of the Lucas-Kanade feature tracker: Description of the algorithm. In *OpenCV Documentation*, 1999.

[13] F. Bourgault, A. A. Makarenko, S. B. Williams, B. Grocholsky, and H. F. Durrant-Whyte. Information based adaptive robotic exploration. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2002.

[14] R. A. Brooks. Intelligence without representation. In *Workshop on the Foundations of Artificial Intelligence, Dedham, MA*, 1987.

[15] G. Carrera, A. Angeli, and A. J. Davison. Lightweight SLAM and navigation with a multi-camera rig. In *Proceedings of the European Conference on Mobile Robotics (ECMR)*, 2011.

[16] G. Carrera, A. Angeli, and A. J. Davison. SLAM-based automatic extrinsic calibration of a multi-camera rig. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[17] Anthony R. Cassandra. Partially observable markov decision processes. URL http://www.pomdp.org.

[18] D. Chekhlov, M. Pupilli, W. W. Mayol, and A. Calway. Real-time and robust monocular SLAM using predictive multi-resolution descriptors. In *Proceedings of the International Symposium on Visual Computing (ISVC)*, 2006.

[19] A. Chiuso, P. Favaro, H. Jin, and S. Soatto. "MFm": 3-D motion from 2-D motion causally integrated over time. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2000.

[20] C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968.

[21] J. Civera, D. R. Bueno, A. J. Davison, and J. M. M. Montiel. Camera self-calibration for sequential bayesian structure from motion. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

[22] L. A. Clemente, A. J. Davison, I. Reid, J. Neira, and J. D. Tardós. Mapping large loops with a single hand-held camera. In *Proceedings of Robotics: Science and Systems (RSS)*, 2007.

[23] T.S. Collett, M. Collett, and R. Wehner. The guidance of desert ants by extended landmarks. *Journal of Experimental Biology*, 204(9):1635 – 1639, 2001.

[24] M. Cummins and P. Newman. FAB-MAP: Probabilistic localization and mapping in the space of appearance. *International Journal of Robotics Research (IJRR)*, 27(6):647–665, 2008.

[25] M. Cummins and P. Newman. Highly scalable appearance-only SLAM — FAB-MAP 2.0. In *Proceedings of Robotics: Science and Systems (RSS)*, 2009.

[26] H. Dahlkamp, A. Kaehler, D. Stavens, S. Thrun, and G. Bradski. Self-supervised monocular road detection in desert terrain. In *Proceedings of Robotics: Science and Systems (RSS)*, 2006.

[27] A. J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2003.

[28] A. J. Davison, Y. González Cid, and N. Kita. Real-time 3D SLAM with wide-angle vision. In *Proceedings of the IFAC Symposium on Intelligent Autonomous Vehicles (IAV)*, 2004.

[29] A. J. Davison, N. D. Molton, I. Reid, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 29(6):1052–1067, 2007.

[30] A. J. Davison and D. W. Murray. Mobile robot localisation using active vision. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 1998.

[31] A. J. Davison and D. W. Murray. Simultaneous localization and map-building using active vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 24(7):865–880, 2002.

[32] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1999.

[33] E. Eade and T. Drummond. Scalable monocular SLAM. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[34] E. Eade and T. Drummond. Monocular SLAM as a graph of coalesced observations. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2007.

[35] S. Esquivel, F. Woelk, and R. Koch. Calibration of a multi-camera rig from non-overlapping views. In *Proceedings of the DAGM Symposium on Pattern Recognition*, 2007.

[36] O. D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig? In *Proceedings of the European Conference on Computer Vision (ECCV)*, 1992.

[37] O. D. Faugeras and G. Toscani. The calibration problem for stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1986.

[38] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[39] A. W. Fitzgibbon. Robust registration of 2D and 3D point sets. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2001.

[40] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, 4(1):23–33, 1997.

[41] F. Fraundorfer, C. Engels, and D. Nistér. Topological mapping, localization and navigation using image collections. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2007.

[42] U. Frese, P. Larsson, and T. Duckett. A multilevel relaxation algorithm for simultaneous localisation and mapping. *IEEE Transactions on Robotics (T-RO)*, 21(2):196–207, 2005.

[43] K. Frisch. *The dance language and orientation of bees*. Belknap Press of Harvard University Press Cambridge, 1967.

[44] W.K. Fung, Y.Y. Leung, M.K. Chow, Y.H. Liu, Y. Xu, W. Chan, T.W. Law, S.K. Tso, and C.Y. Wang. Development of a hospital service robot for transporting tasks. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2003.

[45] Willow Garage. OpenCV (open source computer vision library). URL: http://opencv.willowgarage.com/wiki/, 2011.

[46] A. Gee and W. Mayol-Cuevas. Real-time model-based SLAM using line segments. In *International Symposium on Visual Computing*, pages 354–363, 2006.

[47] A.P. Gee, D. Chekhlov, W. Mayol, and A. Calway. Discovering planes and collapsing the state space in visual slam. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2007.

[48] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard. A tree parameterization for efficiently computing maximum likelihood maps using gradient descent. In *Proceedings of Robotics: Science and Systems (RSS)*, 2007.

[49] D. Hahnel, W. Burgard, D. Fox, and S. Thrun. An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2003.

[50] C. G. Harris and J. M. Pike. 3D positional integration from image sequences. In *Proceedings of the Alvey Vision Conference*, pages 233–236, 1987.

[51] C. G. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the Alvey Vision Conference*, pages 147–151, 1988.

[52] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[53] J. Hoffmann, M. Jungel, and M. Lotzsch. A vision based system for goal-directed obstacle avoidance. In *Proceedings of the Robocup Symposium*, 2004.

[54] R. Horaud and G. Csurka. Self-calibration and Euclidean reconstruction using motions of a stereo rig. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 1998.

[55] S. Ikeda, T. Sato, and N. Yokoya. A calibration method for an omnidirectional multi-camera system. In *Proceedings of SPIE Electronic Imaging*, 2003.

[56] V. Ila, J.M. Porta, and J. Andrade-Cetto. Information-based compact pose SLAM. *IEEE Transactions on Robotics (T-RO)*, 26(1):78–93, 2010.

[57] Cambridge in Colour. Understanding camera lenses. http://www.cambridgeincolour.com/tutorials/camera-lenses.htm, 2010.

[58] M. Kaess and F. Dellaert. Probabilistic structure matching for visual SLAM with a multi-camera rig. *Computer Vision and Image Understanding (CVIU)*, 114:286–296, Feb 2010.

[59] B. Keyes, R. Casey, H.A. Yanco, B.A. Maxwell, and Y. Georgiev. Camera placement and multi-camera fusion for remote robot operation. In *Proceedings of the IEEE International Workshop on Safety, Security and Rescue Robotics*, pages 22–24, 2006.

[60] J.-H. Kim and M. J. Chung. Absolute motion and structure from stereo image sequences without stereo correspondence and analysis of degenerate cases. *Pattern Recognition*, 39(9):1649 – 1661, 2006.

[61] Y. Kim and H. Kim. Layered ground floor detection for vision-based mobile robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 13–18. IEEE, 2004.

[62] B. Kitt, A. Geiger, and H. Lategahn. Visual odometry based on stereo image sequences with RANSAC-based outlier rejection scheme. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 2010.

[63] G. Klein and D. W. Murray. Parallel tracking and mapping for small AR workspaces. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2007.

[64] G. Klein and D. W. Murray. Improving the agility of keyframe-based SLAM. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2008.

[65] J. Knight and I. Reid. Binocular self-alignment and calibration from planar scenes. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2000.

[66] O. Koch and S. Teller. Wide-area egomotion estimation from known 3d structure. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, June 2007.

[67] O. Koch and S. Teller. Body-relative navigation guidance using uncalibrated cameras. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2009.

[68] H. Kong, J.Y. Audibert, and J. Ponce. Vanishing point detection for road detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[69] K. Konolige and M. Agrawal. Frame-frame matching for realtime consistent visual mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2007.

[70] K. Konolige and M. Agrawal. FrameSLAM: From bundle adjustment to real-time visual mapping. *IEEE Transactions on Robotics (T-RO)*, 24:1066–1077, 2008.

[71] R. Kuemmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[72] R. K. Kumar, A. Ilie, J. Frahm, and M. Pollefeys. Simple calibration of non-overlapping cameras with a mirror. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[73] T. Läbe and W. Förstner. Automatic relative orientation of images. In *Proceedings of the 5th Turkish-German Joint Geodetic Days*, 2006.

[74] P. Lebraly, O. Ait-Aider, E. Royer, and M Dhome. Calibration of non-overlapping cameras — application to vision-based robotics. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2010.

[75] P. Lebraly, E. Royer, O. Ait-Aider, C. Deymer, and M Dhome. Fast calibration of embedded non-overlapping cameras. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[76] L. Lee, R. Romano, and G. Stein. Monitoring activities from multiple video streams: establishing a common coordinate frame. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 22(8):758–767, 2000.

[77] A. Levin and R. Szeliski. Visual odometry and map correlation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[78] H. Li, R. Hartley, and L. Wang. Auto-calibration of a compound-type omnidirectional camera. In *Proceedings of the Digital Image Computing on Techniques and Applications (DICTA)*, 2005.

[79] Y. Li and S.T. Birchfield. Image-Based Segmentation of Indoor Corridor Floors for a Mobile Robot. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2010.

[80] M. I. A. Lourakis and A. A. Argyros. SBA: A software package for generic sparse bundle adjustment. *ACM Transactions on Mathematical Software*, 36(1):1–30, 2009.

[81] S. J. Lovegrove, A. J. Davison, and J. Ibanez-Guzmán. Accurate visual odometry from a rear parking camera. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 2011.

[82] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110, 2004.

[83] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1981.

[84] Q.-T. Luong and O. Faugeras. Self-calibration of a stereo rig from unknown camera motions and point correspondences. Technical Report INRIA Tech. Report RR-2014, INRIA Sophia-Antipolis, 1993.

[85] A.A. Makarenko, S.B. Williams, F. Bourgault, and H.F. Durrant-Whyte. An experiment in integrated exploration. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2002.

[86] J. Mason, S. Ricco, and R. Parr. Textured occupancy grids for monocular localization without features. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[87] C. Mei, G. Sibley, M. Cummins, P. Newman, and I. Reid. A constant time efficient stereo SLAM system. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2009.

[88] M. Milford and G. Wyeth. Hippocampal models for simultaneous localisation and mapping on an autonomous robot. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2003.

[89] M. Milford and G. Wyeth. Biological navigation systems. In *Robot Navigation from Nature*, volume 41 of *Springer Tracts in Advanced Robotics*, pages 29–39. Springer Berlin / Heidelberg, 2008.

[90] M. Milford and G. Wyeth. Persistent navigation and mapping using a biologically inspired SLAM system. *International Journal of Robotics Research (IJRR)*, 29(9):1131–1153, 2009.

[91] M. Milford, G. Wyeth, and D. Prasser. Simultaneous localisation and mapping from natural landmarks using ratslam. In *Australasian Conference on Robotics and Automation 2004*, 2004.

[92] M. J. Milford and G. Wyeth. Mapping a suburb with a single camera using a biologically inspired SLAM system. *IEEE Transactions on Robotics (T-RO)*, 24(5):1038–1053, 2008.

[93] M. J. Milford and G. Wyeth. Single camera vision-only SLAM on a suburban road network. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2008.

[94] J. M. M. Montiel, J. Civera, and A. J. Davison. Unified inverse depth parametrization for monocular SLAM. In *Proceedings of Robotics: Science and Systems (RSS)*, 2006.

[95] H.P. Moravec and A. Elfes. High resolution maps from angle sonar. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1985.

[96] M. Müller and R. Wehner. Path integration in desert ants, cataglyphis fortis. *Proceedings of the National Academy of Sciences*, 85(14):5287 – 5290, 1988.

[97] R. A. Newcombe and A. J. Davison. Live dense reconstruction with a single moving camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[98] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011.

[99] R. A. Newcombe, S. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011.

[100] D. Nistér. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(6):756–777, 2004.

[101] D. Nistér, O. Naroditsky, and J. Bergen. Visual odometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[102] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[103] University of Bremen. Spatial Cognition Laboratory. Project R3. URL: http://www.sfbtr8.uni-bremen.de/project/r3/research.html.

[104] University of Cambridge. Machine Intelligence Laboratory. libCVD (Computer Vision Library). URL: http://mi.eng.cam.ac.uk/ er258/cvd/, 2010.

[105] T. Oskiper, Z. Zhu, S. Samarasekera, and R. Kumar. Visual odometry system using multiple stereo cameras and inertial measurement unit. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[106] L. Piyathilaka and R. Munasinghe. Multi-camera visual odometry for skid steered field robot. In *International Conference on Information and Automation for Sustainability*, pages 189 – 194, dec. 2010.

[107] M. Pollefeys, R. Koch, and L. Van Gool. Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 1998.

[108] M. Pollefeys, D. Nistér, J. M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S. J. Kim, P. Merrell, C. Salmi, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewénius, R. Yang, G. Welch, and H. Towles. Detailed real-time urban 3D reconstruction from video. *International Journal of Computer Vision (IJCV)*, 78(2-3):143–167, 2008.

[109] E. Prassler and K. Kosuge. Domestic robotics. In *Springer Handbook of Robotics*, pages 1253–1281. 2008.

[110] S. Prince, A.D. Cheok, F. Farbiz, T. Williamson, N. Johnson, M. Billinghurst, and H. Kato. 3D Live: real-time captured content for mixed reality. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2002.

[111] The Player Project. Player and Stage. URL: http://playerstage.sourceforge.net/.

[112] M. Pupilli and A. Calway. Real-time visual SLAM with resilience to erratic motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[113] Haralick R, Lee C, Ottenberg K, and Nölle M. Review and analysis of solutions of the three point perspective pose estimation problem. *International Journal of Computer Vision (IJCV)*, 1994.

[114] C. Rasmussen. Grouping dominant orientations for ill-structured road following. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[115] F.L.W. Ratnieks. How far do honey bees forage? *Beekeepers Quarterly*, 89:26–28, 2007.

[116] Y. Rubner, C. Tomasi, and L.J. Guibas. A metric for distributions with applications to image databases. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 1998.

[117] D. Santosh, S. Achar, and C. V. Jawahar. Autonomous image-based exploration for mobile robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2008.

[118] J. Shi and C. Tomasi. Good features to track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1994.

[119] G. Sibley, C. Mei, I. Reid, and P. Newman. Adaptive relative bundle adjustment. In *Proceedings of Robotics: Science and Systems (RSS)*, 2009.

[120] R. Sim and J. J. Little. Autonomous vision-based exploration and mapping using hybrid maps and rao-blackwellised particle filters. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2006.

[121] R. Simmons. The curvature-velocity method for local obstacle avoidance. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1996.

[122] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2003.

[123] C.C. Slama, C. Theurer, and S.W. Henriksen. *Manual of photogrammetry*. American Society of Photogrammetry, 1980.

[124] R. Smith, M. Self, and P. Cheeseman. A stochastic map for uncertain spatial relationships. In *Workshop on Spatial Reasoning and Multisensor Fusion*, 1987.

[125] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In *Uncertainty in Artificial Intelligence*, pages 435–461. Elsevier, 1988.

[126] J. Solà, A. Monin, M. Devy, and T. Vidal-Calleja. Fusing monocular information in multicamera SLAM. *IEEE Transactions on Robotics (T-RO)*, 24(5):958–968, 2008.

[127] C. Stachniss. *Exploration and Mapping with Mobile Robots*. PhD thesis, University of Freiburg, Department of Computer Science, April 2006.

[128] C. Stachniss and W. Burgard. Exploring unknown environments with mobile robots using coverage maps. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

[129] C. Stachniss, G. Grisetti, and W. Burgard. Information gain-based exploration using Rao-Blackwellized particle filters. In *Proceedings of Robotics: Science and Systems (RSS)*, 2005.

[130] C. Stachniss, D. Hähnel, and W. Burgard. Exploration with active loop-closing for fastslam. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2004.

[131] H. Strasdat, A. J. Davison, J. M. M. Montiel, and K. Konolige. Double window optimisation for constant time visual SLAM. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011.

[132] H. Strasdat, J. M. M. Montiel, and A. J. Davison. Real-time monocular SLAM: Why filter? In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[133] P. F. Sturm and S. J. Maybank. On plane-based camera calibration: A general algorithm, singularities, applications. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1999.

[134] T. Svoboda, D. Martinec, and T. Pajdla. A convenient multi-camera self-calibration for virtual environments. *PRESENCE: Teleoperators and Virtual Environments*, 14(4):407–422, 2005.

[135] Robot Operating System. ROS. URL: http://www.ros.org.

[136] Robot Operating System. ROS 2D Map Display. URL: http://www.ros.org/wiki/rviz/DisplayTypes/Map.

[137] S. Thrun, W. Burgard, and D. Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2000.

[138] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. Cambridge: MIT Press, 2005.

[139] S. Thrun and M. Montemerlo. The graphSLAM algorithm with applications to large-scale mapping of urban structures. *International Journal of Robotics Research (IJRR)*, 25(5-6):403–429, 2006.

[140] R. Y. Tsai. An efficient and accurate camera calibration technique for 3d machine vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1986.

[141] I. Ulrich and I. Nourbakhsh. Appearance-based place recognition for topological localization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2000.

[142] R. Valencia, J. Andrade-Cetto, and J.M. Porta. Path planning in belief space with pose SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[143] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001.

[144] X. Wan, R.F. Wang, and J.A. Crowell. The effect of active selection in human path integration. *Journal of Vision*, 10(11):1 – 11, 2010.

[145] O. Wijk and H. I. Christensen. Localization and navigation of a mobile robot using natural point landmarks extracted from sonar data. *Robotics and Autonomous Systems*, 31(1):31–42, 2000.

[146] Q. Wu, W. Zhang, T. Chen, V. Kumar, et al. Camera-based clear path detection. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 1874–1877. IEEE, 2010.

[147] T. Xu, K. Kuhnlenz, and M. Buss. A view direction planning strategy for a multi-camera vision system. In *International Conference on Information and Automation*, pages 320–325, 2008.

[148] B. Yamauchi. A frontier-based approach for autonomous exploration. In *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*, 1997.

[149] Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 1999.

[150] C. Zhou, Y. Wei, and T. Tan. Mobile robot self-localization based on global visual appearance features. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 1271–1276, 2003.

[151] J. Zhou and B. Li. Robust ground plane detection with normalized homography in monocular sequences from a robot platform. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, 2007.