TECHNISCHE UNIVERSITÄT MÜNCHEN und
UNIVERSIDAD DE MÁLAGA

Fakultät für Informatik
Lehrstuhl für Computer Vision & Pattern Recognition

Motion Estimation, 3D Reconstruction and Navigation with Range Sensors

Mariano Jaimez Tarifa

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Prüfer der Dissertation:
1. Prof. Dr. Daniel Cremers
2. Prof. Dr. Javier González Jiménez
3. Prof. Dr. Victor Adrian Prisacariu
4. Prof. Dr. Cristina Urdiales García
5. Prof. Dr. Darius Burschka
6. Prof. Dr. Javier Civera Sancho

Die Dissertation wurde am 24.10.2017 bei der Universidad de Málaga eingereicht und durch die Fakultät für Informatik am 10.11.2017 angenommen.

Prof. Dr. Javier González Jiménez and Prof. Dr. Daniel Cremers are the supervisors of the doctoral dissertation entitled "Motion Estimation, 3D Reconstruction and Navigation with Range Sensors" written by Mariano Jaimez Tarifa. They hereby declare that this dissertation is suitable for the attainment of the degree "Doctor en Ingeniería Mecatrónica" awarded by the Universidad de Málaga and the degree "Doktor der Naturwissenschaften" awarded by the Technische Universität München.

Málaga, 12th July 2017

Prof. Dr. Javier González Jiménez

Munich, 18th July 2017

Prof. Dr. Daniel Cremers

Contents

Acknowledgments and Retrospection

Many people have guided, encouraged and motivated me during this long academic journey. I take the opportunity here to thank them all, not only those who have helped me during my PhD but also those who played an important role in my education during my childhood and adolescence.

I am enormously thankful to Prof. Javier González Jiménez. He gave me the chance to join the MAPIR group and pursue a PhD degree on robotics five years ago. This turn up to be a great (and rather international) adventure full of professional and personal challenges. He taught me how to write a scientific paper (which involves undergoing the processes of "lija gorda" and "lija fina"), and pushed me to publish my work when I was skeptical about it. I have always had concerns about the current research system based on the "publish or perish principle" (and I still have them). He has helped me to keep my feet on the ground, and thanks to that a significant percentage of the papers supporting this thesis exist. However, his main strength is, in my opinion, his capacity to motivate us. He is always involved in our projects, being supportive during the hard times and excited about the good results. As a PhD student, this is very rewarding.

These years were full of joyful and often hilarious moments thanks to my colleagues and good friends in MAPIR. Raúl Sarmiento (*aka* Joselito) was always an endless source of entertainment and unpredictable thoughts. He has also provided me with technical support during all these years, fact for which he claims I owe him hundreds of euros (although I paid that debt back long ago). Francisco Meléndez was the best comrade and teammate for the many events and parties we organized in MAPIR at Christmas, Halloween, etc. Javier G. Monroy was my most reliable co-worker; he is committed, honest and good-hearted, and I appreciate him for that. Rubén Gómez was my companion of adventures. We went together to three ICRAs (Seattle, Stockholm and Singapore) and we visited each other during our respective stays in Munich and Zurich. I remember all of these trips with joy and I am happy to have shared those many good moments with him. Besides, there were uncountable enriching conversations about politics, science, education or even philosophy, and for those I thank Eduardo Fernández, Jesús Briales, Carlos Sánchez, Manuel López, Francisco A. Moreno, Andrés Góngora, Juan A. Fernández, Cipriano Galindo, Ana Cruz and Vicente Arévalo. Last, I must mention José Luis Blanco, a

former member of MAPIR. His work opened doors for many of us, MRPT was an invaluable programming tool for the work presented here and his detailed answers to my uncountable emails were highly appreciated.

In 2014 I enjoyed a research stay in the computer vision group at the Technical University of Munich led by Prof. Daniel Cremers. After that, Daniel offered me to pursue a dual PhD under his supervision, which was an incredible opportunity I could not refuse and I am very grateful for. As a supervisor, Daniel has taught me about optimization and the state-of-the-art in computer vision. He has been very supportive and flexible during these years with my changing plans and temporal visits to Munich. The computer vision group at TUM is a melting pot with very talented people from different nationalities and varied research interests. It is often visited by renowned researchers in the field, which is a strongly positive aspect and a great chance for us to grow. I acknowledge Daniel for all of this, and also for fostering a good atmosphere at work and for promoting out-of-work leisure activities like hiking, skiing, etc.

My life in Munich would have been sadder and boring without my colleagues and friends from the lab. Mohamed Souiai (*aka* Mo) taught me the basics of variational methods and also introduced me to many people and places in the city. Robert Maier was always willing to offer me the sofa of his flat when I ran into logistic problems (which happened more than once). Christian Kerl kindly lent me the teddy bear of his son to carry out my experiments (I am indebted to Moritz). Vladyslav Usenko was my most faithful companion for karaoke every week. My office mates, Christiane Sommer and Virginia Estellers, offered me endless and interesting discussions about the world, the people, politics and the right way to phrase ideas in English (on which we normally disagreed). Many others have contributed to turn my time in Munich into a pleasant and exciting experience, and I want to thank them for that: Lingni Ma, John Chiotellis, David Schubert, Laura Leal-Taixé, Jürgen Sturm, Jörg Stückler, Martin Oswald, Jakob Engel, Raluca Scona, Yvain Queau, Matthias Vestner, Zorah Lähner, Emanuel Laude, Rui Wang, Thomas Möllenhoff, Thomas Windheuser, Thomas Frerix, Benedikt Löwenhauser, Caner Hazirbas, Philip Häusser, Björn Häfner, Nikolaus Demmel, Tim Meinhardt, Vladimir Golkov, Tao Wu, Rudolph Triebel, Emanuele Rodola, Michael Möller, Csaba Domokos, Frank Schmidt, Sabine Wagner and Quirin Lohr.

In 2015, barely recovered from a back surgery, I spent two months working in Microsoft Research Cambridge under the supervision of Dr. Andrew Fitzgibbon. Honestly I regard Andrew as my third "unofficial" supervisor. Working with him was a thrilling and uplifting experience. I was lucky to spend hours with him developing and discussing different mathematical models, and I was very surprised by the fact he did not only had a vast theoretical background but also knew about implementation details, libraries, hardware, etc. Tom Cashman was another talented researcher I had the chance to work with in Cambridge. I appreciate much that he was always willing to give me a hand when I needed it, both during these months and afterwards. I thank both for those exciting months of work. Even though my health was far from good by that time, I remember it as a very fulfilling experience.

The doctoral degree is the final step of a long academic journey that also encompasses many years at school, high school and the university (bachelor and masters degrees). I believe that, at this point, it is fair to recall and thank those teachers that made an impact on me before I

started my PhD. I spent most of my childhood at the school Cerrado de Calderón, a place I remember with affection. There the first important teacher was Teresa Alba, who realized when I was 7 years old that I was a special pupil and proposed my parents to promote me to a higher course/class (one year). I barely remember her face but she changed my life with that decision, and I am grateful for it. A few years after, when I was 10 years old, I had a special tutor called Juan José Méndez. He always minded me and encouraged me to go further and learn more than the others if I had the potential to do so. Later at high school there were several influential figures. I remember Enrique Salinas, an exceptional sport teacher. José Luis Espejo taught me Spanish and its history; by that time I was unaware of how important one's language skills are but now I know it (in fact I notice that I automatically become a sillier person when I have to communicate or write in English. That is the price non-native English speakers pay to be part of this globalized world). Juan Carlos Rodriguez introduced me to the marvelous world of physics, and did it brilliantly. Nevertheless, the teacher who influenced me above all, and the one I admired the most, was Pedro Hormigo. He taught us math, and managed to do so in a very inspirational way (if I dig in my memories now after 12 years it feels almost like Harry Potter learning magic at Hogwarts). He is a once-in-a-lifetime teacher and for him is my more nostalgic thank you.

I started to get interested in research during the years of my bachelor. That interest was boosted by Prof. Juan A. Cabrera Carrillo. He offered me to participate in his projects at a very early stage of my studies, which led to subsequent years of collaboration, learning and research in the field of mechanical engineering. I was lucky to share these years with my closest friend (and also my strongest competitor by that time) Pablo Giner Abad. By working together we complemented each other but also pushed each other to the limit to keep up in our personal race. Together with Juan Castillo, the four of us formed a team always eager to address new challenges with imaginative solutions. I thank them all for those many enriching and also amusing moments.

It is time to talk about those who have been there since the day I was born: my family. My father showed me the value of hard working and willpower through sports. We enjoyed endless hours of football, tennis, ping pong, skiing and running together, and with them (and with him) I learned that training and enduring normally comes with a reward. As a pragmatic person, he has questioned the purpose of my work during these years, and I have to concede here that I have also sometimes found it pointless. During my childhood, my mother spent hundreds of hours teaching me and asking me the lessons I should memorize while I was running around our flat with my scooter or my football (or both). By that time she was concerned that I could become a lone, asocial and ruthless person, and she did her best to soften me in order to avoid that. Knowing myself, I can say that she was successful, and I thank her for that. My grandparents, the four of them, have lived exemplary lives even though their circumstances were at times dramatic. They have been references to look up to since I was a child. To finish I simply have to say that I have the best possible relatives in many many senses: my sisters Alejandra and Iria, my uncles and aunts, my cousins, and many others whose affection and tenderness make Málaga and Loja my true homes. But I must confess there is room for improvement in one particular aspect: no matter how many times I explain the topic I work on, they all always forget it almost instantly

(and demand me to explain it again and again and again). I assume I am not the only PhD student suffering this though...

And there is one last and very special person who truly deserves my gratitude: my partner Julia Nagel. She has experienced more than any other the consequences of my PhD during these years. Sometimes these consequences were positive, e.g. when my internship and my dual PhD brought me closer to her (she lives in Germany). Other times, during periods of high workload and deadlines, she had to cope with a less social and often distant me. Our particular issue during these years was travelling: she was always keen on travelling often, to far destinations and for long, and I was always busy with some project. This led to frequent conversations about work-life balance, and I admit now she was right when she said I should schedule time for myself and be able to disconnect from work. I have followed that advice lately. Most importantly, she has provided me with the peace and personal support I need to feel balanced and be able to concentrate on my work. She has also blindly believed in me and celebrated each of my achievements as hers. Moreover, I highly value that she has happily accepted my mother tongue as "our language". For these and many other reasons this thesis would not exist without her, and for her are the only Spanish words here: muchas gracias por todo.

Mariano Jaimez Tarifa
Munich
June 2017

I want to thank the Government of Spain (grant program FPI-MICINN 2012, project *Taroth* DPI2011-25483, project *Promove* DPI2014-55826-R) and the European Union (project *GiraffPlus*, ERC grant *Convex Vision*, ERC consolidator grant *3D Reloaded*) for funding these years of research, internships and attendances to international conferences. I also gratefully acknowledge the support of NVIDIA Corportation with the donation of hardware used for this research.

The Government of Spain has invested a significant sum of money in this thesis during a period when the Spanish economy was in crisis. Unfortunately this investment is not going to be profitable for the Spanish State because the lack of a powerful technological industry in our country pushes me to seek for jobs abroad. I sincerely feel this is a pitty and a detrimental situation for Spain but it is not in my hands to change it. The Spanish government should promote a much tighter collaboration between companies and researchers in order to avoid this. How to do that in an efficient way is beyond my knowledge, but it might be worth thinking about all the work developed by hundreds of PhD students which falls into oblivion after they finish. I believe that huge amount of effort should be exploitable.

1 Introduction

Since its origins, robotics has aimed to create autonomous machines to increase the efficiency of industrial processes, reduce manufacturing costs and free people from tedious and dangerous tasks. Given its complexity, different disciplines of engineering, physics and maths have converged to what we nowadays call *mechatronics* to address the conception, design and manufacture of these complex systems called robots. Among the many challenges still unsolved, one of upmost importance is that of endowing robots with intelligence and autonomy. In order to emulate a person, a robot must be able to perceive its surroundings, interpret its circumstances and decide how to act to achieve its goal. Thus, autonomy is directly related to the robot's capacity to perceive the environment and its ability to process that sensorial data. Like humans, robots are provided with sensors (their "senses") which allow them to see, hear, touch or even smell. And also, like for humans, the most powerful of all senses is vision. Nevertheless, robots are not only equipped with passive sensory systems (those which measure the ambient energy) but also with active sensory systems. In vision, these systems work by emitting a pattern of light which is reflected back from the environment and subsequently detected by a specific sensor. By virtue of this mechanism, active sensory systems are able to infer the geometry of the surrounding obstacles, which is extremely useful in many technological fields.

In robotics, active visual sensory systems, also known as "range sensors", are commonly employed for:

- Knowing the spatial distribution of obstacles around a robot.
- Knowing the location and orientation of objects to be manipulated.
- Building 2D or 3D maps of the environment where the robot operates.
- Estimating the 2D or 3D trajectory of a robot.
- Segmenting the observed scene into the different objects it is composed of.

Additionally, range sensors find a myriad of applications beyond robotics, for example in:

- Human-computer interaction and gaming.
- 3D modelling of objects.
- Motion analysis, both for professional sport training or for therapeutic treatment.

- Motion estimation for devices of virtual/augmented reality.
- Driver assistance systems for cars and other vehicles.

All these applications require complex algorithms able to process the geometric data provided by this kind of sensors. Up to date, numerous solutions have been proposed for each particular problem. However, these solutions are often tuned to work for particular scenarios and under controlled conditions, or present good theoretical results that lack practicality. In this thesis we propose novel algorithms to tackle/solve many of the aforementioned examples. In general, our goal has been the development of methods with a solid theoretical basis which, in turn, could be directly applicable to the addressed problem, not only in theory or in simulation but also in real-world scenarios. To this end, one must pay attention to both the formulation and the implementation of a given algorithm in order to select precise formulations that could be efficiently implemented. Following this utilitarian spirit, we have opted for publishing the code of our works so that the scientific community can use and test them.

1.A Motivation

Range sensors have existed since the beginning of the XX century but they have significantly evolved in the last decades. After the invention of the *sonar* during the I World War, the first range sensors based on light emission were the laser scanners or *lidars*, developed in the 1960s. The accuracy of these sensors was very high but so was their price, and for many years they were mostly utilized for military or spatial applications. Over time their costs dropped and simple versions (2D scanners) started to be equipped in mobile robots for obstacle detection, localization and 2D mapping. Nevertheless, the real revolution came with the advent of Microsoft's Kinect camera by the end of 2010. Kinect was the first low-cost camera (150 euros) able to provide not only colour but also depth images, and at a decently high frame rate (30 Hz). Thus, it was the first low-cost sensor that allowed to "sense" the geometry of the environment with accuracy, and its impact on the robotics community and in many other fields was, and still is, enormous. However, the Kinect sensor also posed a challenge since it provides a huge amount of data to process (colour and depth images with VGA resolution at 30 Hz). In consequence, a significant percentage of the published papers in robotics and computer vision in the last years deals with its use and application to different ends.

Additionally, the steady development of computers has enabled the approach of problems that were, until recently, computationally intractable. CPU's now offer multiple processing units (cores) with larger cache sizes. They also include special registers and instruction sets to efficiently perform certain operations on vectorized data (SSE, AVX). GPU's are no longer mere graphic processing units (as their name indicates) but massive units for parallel computation. They have significantly increased their power, memory and bandwidth and, at the same time, they have become energetically more efficient, broadening the prospects of potential applications. Simultaneously, the appearance of new libraries and compilers has eased the implementation of code that can run on multiple cores of a CPU or a GPU. If we focus on C++, modern compilers (MSVS, GCC, Intel...) generate highly optimized code without requiring much programming

effort. Newer versions of CUDA incorporate the unified memory abstraction that allows programmers to access both GPU and CPU memory within a single memory address space. Many other libraries include functionalities to exploit parallel programming, either through vectorization, multi-thread or GPU implementations: STL (C++11), OpenCL, Eigen, OpenGL, etc.

These two key aspects have motivated the work presented in this thesis. In short, the challenge is about how to exploit the newly available range sensors and the increasing computational power of modern PCs to solve fundamental problems that remain unsolved (or only partially solved) in computer vision and robotics.

1.B Goals

Despite the great advances seen in the last decades in computer vision, there are still many open problems. Among those, the ones that are more relevant in robotics often require certain geometric knowledge of the environment, and therefore can benefit from the use of range sensors. In this thesis we mostly (but not uniquely) address the estimation of 2D and 3D motion with different kinds of range sensors. In some cases the interest is in the motion of the sensor itself, while in other cases the goal is to estimate the translations and rotations of the observed objects. Besides motion estimation, we present new algorithms that also exploit geometric data for 3D reconstruction and autonomous navigation. Thus, this thesis does not tackle a single subject but rather covers a variety of topics. The common goal is to come up with novel solutions that are more precise and faster than existing approaches, or that allow a robot or system to operate under more extreme conditions (e.g. in the dark or in very dynamic environments).

Next, we explain in more detail the addressed topics, their potential applications and the inherent difficulties associated to them:

- **Range-based odometry**. Odometry consists in estimating the motion of a sensor or a set of sensors in an incremental way, i.e. by accumulating pose increments. It is a fundamental component of many robotic systems and virtual/augmented reality devices. In robotics, for instance, it permits us to know the pose of a robot precisely and continuously, which is crucial if the robot must perform an autonomous task. Regarding virtual/augmented reality, knowing the pose of the goggles in real time (and therefore the point of view of the user) is fundamental to render virtual images from the right perspective and with low latency.

  Our goal is to create new odometry algorithms based on range sensors. Given their potential applications, it is essential that these algorithms work in real time and are accurate even in dynamic environments where moving objects are often observed.

- **Scene flow estimation**. Scene flow refers to the vector field that represents the independent 3D motion (or velocity) of each point observed by a camera. It has multiple applications: 3D modelling of non-rigid bodies, manipulation of moving objects, human-machine interaction and motion analysis are a few examples.

  The state of the art in this field is less mature if compared to odometry or optical flow estimation. For that reason, our goal is to overcome (or alleviate) the main limitations that currently prevent its use in real-world scenarios. Among those, the most important one is the

high computational load associated to the estimation process, and that is what we put the focus on.

- **3D Modelling** and **tracking** of objects from a set (or sequence) of images. 3D modelling is useful to measure dimensions, insert real objects in virtual worlds, or virtually combine real objects and environments which are not in the same location (e.g. using a virtual 3D model of a piece of furniture to see how it fits in a given room). Tracking, on the other hand, can be employed for human-computer interaction, face reenactment or character animation for movies or videogames.

  In this field, our goal is to exploit background information (i.e. those image regions that do not observe the object in question) to better constrain the reconstruction or tracking problem, thereby improving the basin of convergence and the quality of the results obtained.

- **Reactive Navigation** with range sensors. This strategy allows a robot or vehicle to "react" to changing environments populated with moving obstacles while advancing towards a predefined target. In contrast to deliberative approaches, these methods do not require previous knowledge of the environment (map), but can only drive the robot towards local targets, i.e., not very far from its pose.

  Most existing approaches simplify the navigation problem by considering only 2D representations of the robot shape and the surrounding obstacles. Our goal is to overcome this conservative assumption and consider more realistic robot shapes and obstacle distributions in 3D. This is possible by virtue of the rich geometric data provided by modern range sensors.

1.C Contributions

In this section we enumerate the contributions of this thesis. A brief summary of each is included, together with the related published papers. Both the code and the demonstration videos associated to them can be found in:

`http://mapir.isa.uma.es/mjaimez`

3D visual odometry from depth images

We present a new method to register consecutive depth images in order to estimate the camera motion. We demonstrate that this method is much faster and more precise than other existing techniques that are also based on geometric alignment [2]. Moreover, we show that our approach is as accurate as other state-of-the-art methods which require both geometric and photometric data to estimate the camera motion [3]. Last, it works in real time (30 Hz or more) running on a single CPU core.

- M. Jaimez and J. Gonzalez-Jimenez, “Fast Visual Odometry for 3D Range Sensors”, *IEEE Transactions on Robotics*, vol. 31, no. 4, pp. 809 - 822, 2015.

2D visual odometry from laser scans

We have developed a method to register consecutive laser scans to estimate the planar motion of a robot. This method is based on the range flow equation, applied for the first time to laser scans in [4]. We demonstrate that our approach is more efficient and precise than some of the most prominent works in scan matching [5, 6]. With a runtime of barely 1 millisecond, this methods is suitable for those robotic applications that are computationally demanding and require planar odometry.

- M. Jaimez, J. G. Monroy, J. Gonzalez-Jimenez, "Planar Odometry from a Radial Laser Scanner. A Range Flow-based Approach", *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4479-4485, 2016.

Such formulation was later improved by including multi-scan alignment and a new symmetric range flow constraint. We also proposed a new technique to select keyscans based on modelling the estimate error as a function of the translations and rotations between the aligned scans. This strategy allows us to set a threshold directly on the error domain and use it to obtain the 2D frontier on the translation-rotation plane which would trigger the selection of a new keyscan. Moreover, we present a thorough experimental section with qualitative and quantitative comparisons both in simulation and with real data.

- M. Jaimez, J.G. Monroy, M. Lopez-Antequera, D. Cremers and J. Gonzalez-Jimenez, "Robust Planar Odometry Based on Symmetric Range Flow and Multi-Scan Alignment", *IEEE Transactions on Robotics*. **Under Review**.

Scene flow estimation in real-time with RGB-D cameras

We have developed the first method able to estimate the scene flow in real time with RGB-D cameras. It imposes photometric and geometric consistency between consecutive images and uses the Primal-Dual algorithm [7, 8] as a solver. This choice is optimal since the Primal-Dual algorithm can be easily parallelized and is therefore straightforward to implement on a GPU. As a result, we achieve runtimes two or three orders of magnitude lower than those of state-of-the-art methods, which typically require several seconds (or even a few minutes) to run.

- M. Jaimez, M. Souiai, J. Gonzalez-Jimenez and D. Cremers, "A Primal-Dual Framework for Real-Time Dense RGB-D Scene Flow", *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 98-104, 2015.

Joint segmentation and scene flow estimation with RGB-D cameras

Inspired by [9], we present a new algorithm to segment the scene into the different rigid bodies that compose it and estimate their underlying rigid motions. The main contribution of this work is the use of a smooth (non-binary) segmentation that allows us to interpolate motions along the transitions between rigid parts. We show this strategy provides better results than traditional binary segmentations when the observed scene contains nonrigid parts/objects (e.g. people).

- M. Jaimez, M. Souiai, J. Stückler, J. Gonzalez-Jimenez, D. Cremers, “Motion Cooperation: Smooth Piece-Wise Rigid Scene Flow from RGB-D Images”, *International Conference on 3D Vision (3DV)*, pp. 64-72, 2015.

Visual odometry and scene flow estimation with RGB-D cameras

This is a problem of great interest and complexity because a high number of applications need to know both the trajectory of a camera and the motion of the objects it observes. The difficulty lies on the fact that, when the camera moves, the whole scene is in "apparent motion" and therefore changes in the image caused by the camera motion and those caused by the own motion of objects are hard to distinguish. In this work we describe a specific strategy to segment the image into static and moving parts. After that, the visual odometry is computed from the static parts and the scene flow is estimated for the moving objects. Furthermore, the whole algorithm runs at a frequency of 10 Hz, which makes it applicable to real-world scenarios.

- M. Jaimez, C. Kerl, J. Gonzalez-Jimenez and D. Cremers, “Fast Odometry and Scene Flow from RGB-D Cameras based on Geometric Clustering”, *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3992-3999, 2017.

New background term for object reconstruction and tracking from depth images

Commonly, a 3D model of an object is computed from a set or temporal sequence of images. In those images the object is seen from different perspectives, but the surrounding objects are also observed (and are normally referred to as "background"). In this work we present a framework based on subdivision surfaces for 3D reconstruction and tracking, and describe a novel strategy to penalize projections of the 3D model onto the background regions of the images. This strategy widens the basis of convergence of the algorithm by keeping the model within the visual hull of the object.

- M. Jaimez, T. Cashman, A. Fitzgibbon, J. Gonzalez-Jimenez, D. Cremers, “An Efficient Background Term for 3D Reconstruction and Tracking with Smooth Surface Models”, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7177-7185, 2017.

Reactive navigation based on exact 3D collision determination

We generalize the algorithm presented in [10] to the "3D world", modelling the robot as a set of prisms (instead of just considering its 2D vertical projection) and exploiting the exact spatial distribution of the surrounding obstacles. The resulting method works with any possible combination of range sensors, being them 2D laser scanners, 3D laser scanners or RGB-D cameras, and it is able to generate motion commands at a frequency of 200 Hz. Moreover, it was tested with different robots for almost 20 km of autonomous navigation at different flats and office-like environments.

- M. Jaimez, J. L. Blanco and J. Gonzalez-Jimenez, “Efficient Reactive Navigation with Exact Collision Determination for 3D Robot Shapes”, *International Journal of Advanced Robotic Systems*, vol. 12, no. 63, 2015.

1.D Framework and Timeline

My research work began in the middle of 2012 at the MAPIR (MAchine Perception and Intelligent Robotics) group[1], which is part of the Department of System Engineering and Automation, at the University of Málaga. This group has extensive experience in mobile robotics, computer vision and robotic olfaction. Given my background on mechanics, control and maths, the natural choice for my research topic was mobile robotics. Nonetheless, and probably because of the boom of Kinect after its appearance in 2010, I ended up working at the frontiers between robotics and computer vision or, more specifically, developing vision algorithms that are directly applicable to robotics.

Initially, we dedicated more than half a year to exploit the geometric information provided by depth cameras for autonomous navigation of wheeled robots. After that, we opted for addressing the visual odometry problem, to which we have dedicated a high percentage of the time and effort of this thesis. In this context, I had the chance to do a 4-month internship at the computer vision group[2] of the Technical University of Munich, led by Prof. Daniel Cremers. This gave me an deeper insight into optimization and variational calculus, and let me explore a new topic which would later become the second pillar of my thesis: scene flow estimation. Beyond that, this collaboration was positive for both sides and, from then on, I have also been a member of that group and pursued a dual PhD doctorate.

Subsequently, I enjoyed another 2-month internship at Microsoft Research Cambridge (UK), under the supervision of Dr. Andrew Fitzgibbon. This stay was suitable because Microsoft is the company that commercialized the Kinect sensor [1], on which most of my thesis is based, and they have also authored an endless number of related publications and applications. Additionally, my months there coincided with the launch of the Hololens [11], a device for augmented reality equipped with two depth cameras. Thanks to this good timing I could also be in contact with some of the projects that will allow the use and interaction with such device in the future.

Through all these years I have had the chance and the pleasure to meet some of the most prominent figures in computer vision and robotics. Apart from the aforementioned internships, I have attended international conferences in Seattle (USA), Lyon (France), Stockholm (Sweden), Singapore and Hawaii (USA). Aside from research, I have also participated in teaching, imparting the "Computer Vision" course together with my supervisor Prof. Javier González Jiménez at the University of Málaga. In summary, these have been vibrant years of constant change and learning, enthusiasm and stress, defeats and victories.

1.E Outline

This thesis is divided into the following chapters:

- **Chapter 1: Introduction.** We present the theme of the thesis, describe its context, list the contributions and detail its structure.

[1] http://mapir.isa.uma.es
[2] http://vision.in.tum.de

- **Chapter 2: Range Sensors.** The concept of *range sensing* is introduced. The different types of range sensors are enumerated and their working principles explained. We describe their advantages and disadvantages if compared to other types of sensors commonly employed in robotics and computer vision, and present their potential applications.
- **Chapter 3: Odometry with range sensors.** We introduce the concepts of odometry and visual odometry. We describe the mathematical tools used to represent rigid transformations and to warp images and scans according to them. Afterwards, we present our works on odometry based on range sensors, both with depth cameras and with 2D laser scanners.
- **Chapter 4: Scene flow estimation with RGB-D cameras.** We introduce the concept of scene flow as a 3D extension of the well-known optical flow. We present two different contributions: the scene flow estimation in real time, and the joint segmentation and scene flow estimation based on a smooth division of the scene into its rigidly moving parts. Moreover, we tackle an ambitious problem that is not often addressed in the literature: the joint estimation of the camera motion and the motion of the objects it observes.
- **Chapter 5: Reconstruction and tracking of objects from depth images.** Since the literature on this topic is very extensive, we give a brief overview of the state-of-the-art focusing on the different strategies that use the "background" information to constraint the model to the visual hull of the object. We present a new background term to exploit such information and demonstrate its advantages over existing approaches when used for modelling or tracking of objects with subdivision surfaces.
- **Chapter 6: Reactive navigation of mobile robots equipped with range sensors.** In this chapter we introduce the concept of reactive navigation which, together with path planning, governs the autonomous motion of mobile robots. In contrast to most existing approaches, which normally use a two-dimensional representation of the robot and the obstacles, we propose to overcome that simplification and tackle the collision determination problem in 3D. Hence, the proposed algorithm works precisely for robots with non-uniform shapes and for any arbitrary combination of different range sensors.
- **Chapter 7: Conclusions.** We finish this thesis by summing up the presented works and their impact. Moreover, we look towards the future, analyzing the upcoming technologies and some of the challenges that still remain unsolved.

2 Range sensing

2.A Introduction

The term *range sensing* refers to a set of technologies that allow the measure of distances between a given device and its surroundings. These technologies are usually (but not exclusively) based on active sensory systems, i.e. systems that probe the environment by emitting energy and measuring how or when it is reflected. There exists a number of different working principles on which these systems are based, but most resort to the emission and reception of sound, light or radio waves.

The ones based on sound are often called *sonars* and are mostly used for marine navigation and submarine military applications. They are also equipped in mobile robots and cars to work as proximity sensors for collision prevention (*e.g.* in the system that beeps when a car gets too close to another car while parking). However, their limited accuracy and resolution prevents them from being used for more complex tasks. Alternatively, systems based on radio waves (*radars*) are a suitable solution to scan the environment under bad visibility conditions since radio waves can penetrate dust, rain, fog or snow. Moreover, this technology has the advantage of allowing the detection of multiple objects at the same bearing. However, radars have a low angular resolution (for small-size antennas) if compared to light-based sensors and their measurements are affected by specular reflections and depend on the different materials of the surrounding objects. For these reasons, radars are often used in middle-to-large scale systems but are not very common in robotics or other applications involving small devices.

This thesis focuses on the range sensors more frequently used in robotics, human-machine interaction and visual/augmented reality: laser scanners and depth-sensing cameras. A more detailed description of their working principles and their characteristics is given next.

2.B Laser Scanners

A laser scanner, also known as *lidar*, is a device that senses distance by emitting laser light and measuring the time it takes to return to the sensor. Since light pulses or waves can only be used to measure distance for a specific bearing, lidars normally incorporate oscillating mirrors to be able to scan in multiple directions. Depending on the degrees of freedom of this oscillating mechanism, lidars will provide 2D or 3D scans of their surroundings (see Figure 2.1). In robotics,

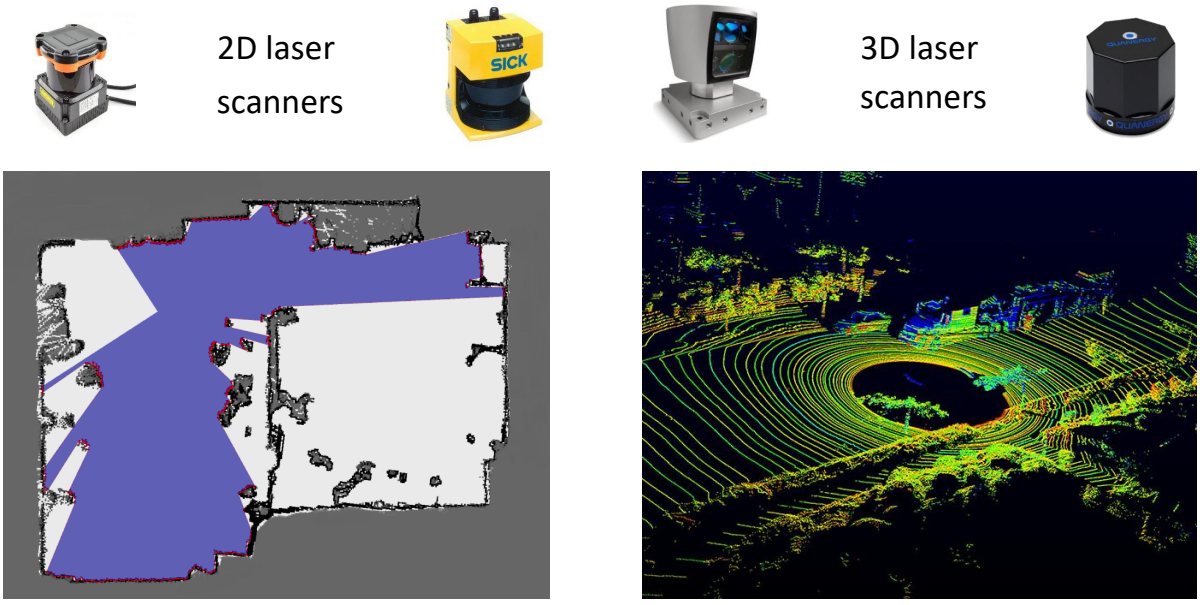


Figure 2.1: **Left**: Hokuyo UTM-30LX and SICK S3000 laser rangefinders, together with an example of a 2D scan (270°) overlapped with a previously-built map of the environment. **Right**: Velodyne HDL-64E and Quanergy laser rangefinders, with and example of a 3D scan (360° × 27°) taken with a Velodyne sensor mounted on a car.

2D lidars are commonly employed to obtain horizontal "slices" or "cuts" of the environment, which is very useful for indoor localization, 2D mapping and obstacle avoidance. On the other hand, 3D lidars are mostly employed for outdoor applications. In these cases, the motion of a robot (or a vehicle) is not always planar and the 3D spatial distribution of both close and distant obstacles becomes more relevant. Aside from robotics and autonomous navigation, the 3D lidar technology is also of interest in geodesy, geology, meteorology and military applications, but this is out of the scope of this thesis.

The main advantages of lidars are:

- Long range. They can provide measurements from few centimeters to more than 100 meters, depending on the model. As a consequence, they are suitable for indoor operation in cluttered environments but also for outdoor operation at high speeds where detecting distant objects/obstacles becomes crucial.
- Wide field of view. Their horizontal aperture typically ranges from 90 degrees to 360 degrees, which is much wider than the standard field of view of digital cameras.
- Good angular resolution, normally below 1 degree.
- High accuracy. They exhibit low measurement errors that are either constant (for short ranges) or linear with the distance.
- Medium-high sampling rate. This is necessary to operate in dynamic environments and react in time to changing conditions.

If compared to other sensors, their main disadvantage is their relatively high price. The cheapest 2D laser scanners on the market cost around 400€ - 500€, but that price rises up to many

thousands of euros for low-to-middle quality 3D laser scanners. Moreover, lidars have a high power consumption, are not robust to shaky motions and their performance deteriorates under the presence of fog, heavy rain or dust. Nowadays, this prevents 3D lidars from being a basic component of mobile robots and vehicles, but advances in solid state-based technologies might help to bring them to the consumer markets.

2.C Depth-sensing Cameras

In contrast to standard digital cameras, which capture the colour (RGB) of the scene, depth-sensing cameras store the distance to the points they observe. With a high (spatial) resolution, they can provide a fine-grained representation of the observed objects. Regarding their accuracy, they tend to be precise for close distances but become inaccurate for mid-long range measurements, with an error that typically grows quadratically with the distance [14]. On the other hand, their field of view is normally equivalent to that of a digital camera equipped with a 25-30 mm lens (e.g. $58° \times 45°$ for Kinect 1). In this sense, depth cameras are inferior to laser scanners because they do not provide a comprehensive view of the scene. Last, they can work at 30 Hz - 60 Hz, which is sufficient for most applications.

Next, we present the two different working principles these cameras rely on, as well as their particular advantages and disadvantages.

2.C.1 Time-of-flight Cameras

This type of cameras emit light pulses and measure their time of flight (ToF) until they are received back at the sensor. Unlike lidars, these cameras do not incorporate rotating parts, flashing the scene only once per image and capturing the reflection by an image sensor / pixel matrix. Examples of these cameras are the Microsoft Kinect 2.0 and the Heptagon Swiss Ranger 4000 (Figure 2.2). Since they normally emit and detect infrared light, they are affected by the sun radiation. However, they can still provide decent measurements in outdoor scenarios if there is no direct sunlight. As can be seen in Figure 2.2, one of their drawbacks is the fact that they are

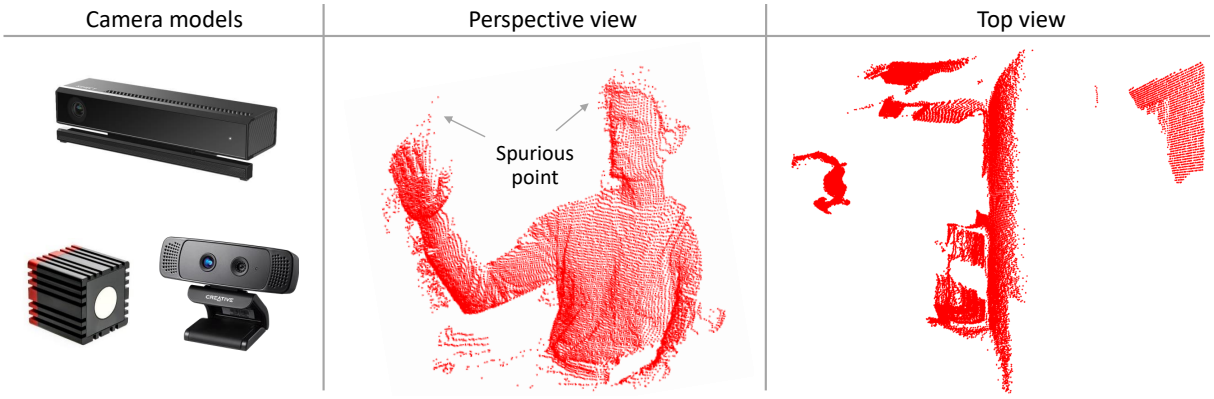


Figure 2.2: **Left**: Kinect 2, Heptagon Swiss Ranger and Creative Gesture cameras. **Middle**: Example of the 3D point cloud computed from a depth image obtained with a Kinect 2. **Right**: Top view of a scene observed with Kinect 2.

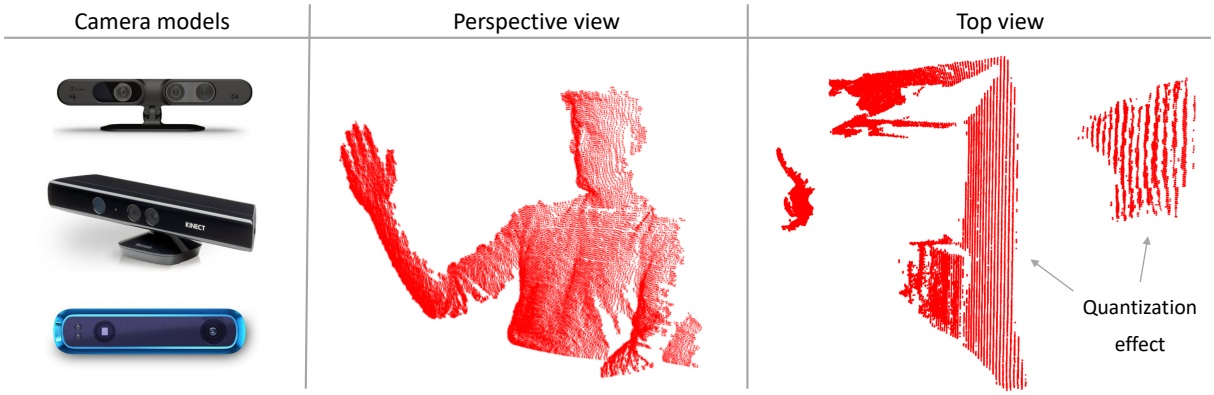


Figure 2.3: **Left**: PrimeSense Carmine, Kinect 1 and Structure sensor. **Middle**: Example of the 3D point cloud computed from a depth image obtained with a PrimeSense Carmine. **Right**: Top view of a scene observed with the same camera.

prone to creating spurious points around the silhouettes of objects or, more precisely, at depth discontinuities. These spurious points can be removed at a post-processing stage, at the expense of consuming computational resources.

2.C.2 Structured light Cameras

Structured light cameras work by projecting a known pattern of light on the scene and inferring the depth field from the way that pattern deforms. The main example of this kind is the Microsoft Kinect 1 (although the technology behind Kinect 1 was developed by PrimeSense, which later also produced their own depth cameras: the PrimeSense Carmine). Structured light cameras typically employ an infrared pattern of light and, hence, become completely blind under the presence of sunlight. The reason is that the sun radiation is much more intense than the pattern emitted by the camera, and therefore the latter is masked and cannot be perceived by the sensor. Commonly, they exhibit another downside: a strong quantization of the measured depth. As this quantization actually affects the inverse depth, it is not noticeable for short distances but becomes extreme for distant regions, as can be seen in Figure 2.3.

2.D RGB-D Cameras

Both structured light and ToF cameras can be found as independent devices on the market, but they are often combined with RGB cameras, leading to the so-called *RGB-D cameras*. This synergy is suitable for robotics and many other fields because it combines both photometric and geometric data to provide a four-dimensional representation of the observed scene. An issue when jointly exploiting RGB and depth images is that of properly registering them. Since RGB and depth images are captured by different lenses, there is no direct correspondence between their pixels. In other words, a point observed by a certain pixel in the RGB image does not coincide with the point observed by that same pixel in the depth image. In fact, there will often be points of the scene which are visible from the RGB lens but not from the infrared lens, and vice versa. Although RGB-D cameras often incorporate a function to automatically register

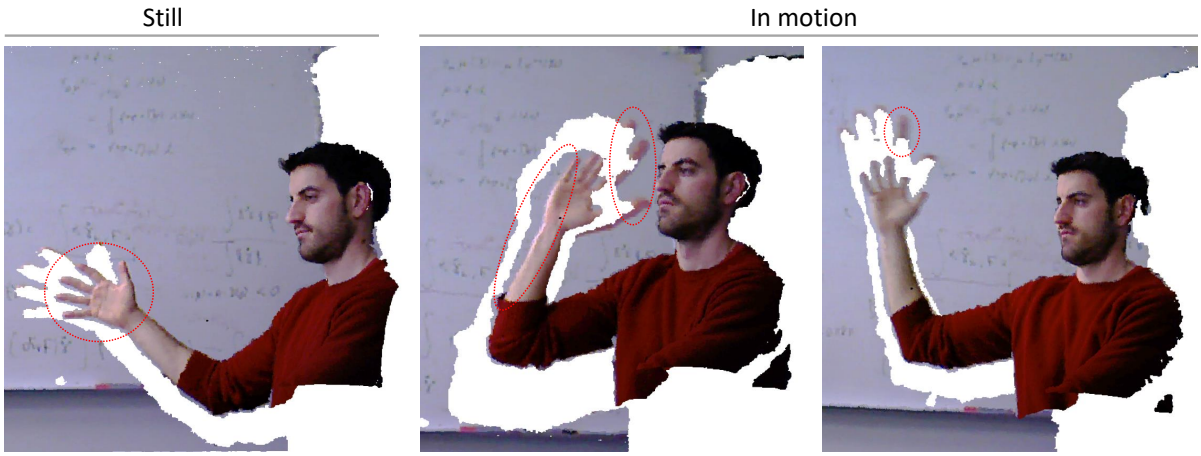


Figure 2.4: Examples of 3D coloured point clouds built from automatically registered RGB-D images taken with a PrimeSense Carmine.

colour and depth images, that registration is far from perfect, as illustrated in Figure 2.4. There exist methods to improve that registration by aligning the edges (regions of high gradients) of the two images or warping them according to the rigid transformation existing between the two lenses (calibration). Both approaches require extra computation and will always imply at least a small increment in the runtime of the proposed algorithm. For this reason, the solution to this problem often consists in developing methods which are robust to the RGB-D misalignment.

3 Range-based Odometry

3.A Introduction

The term *odometry* resulted from the combination of two greek words[1]: *odos*, which means road or path, and *metros*, a measure. Thus, odometry is related to the act of measuring the length of a certain path or trajectory. Originally, the term odometry was specifically used to refer to what we nowadays call *wheel odometry*, and consisted in counting the number of turns of one or several wheels to estimate the planar trajectory of a vehicle. This process is inherently incremental, since the trajectory can only be estimated as a sum of small pose increments (i.e. one cannot estimate the whole trajectory of a vehicle from the final count of wheel turns, this problem is highly underconstraint). In fact, wheel odometry can only be exact if these increments are infinitesimally small and integrated through time, which is impossible in practice. Alternatively, *visual odometry* resorts to align consecutive images to estimate the motion of the sensor. To clarify that we employ range measurements in our work, we will also coin the term *range-based odometry*, although both might be equivalent in many cases (e.g. when working with depth images). Like wheel odometry, visual and range-based odometry work incrementally as they require a certain degree of overlap between the incoming data to be able to align them. In both cases, since odometry continuously builds upon previous estimates, it accumulates and propagates the estimation errors through time. For this reason, having accurate motion estimates becomes crucial.

In general, there exist a number of technologies to track the motion of a given vehicle or device:

- The *Global Positioning System*, or GPS, was developed by the Defense Department of the USA during the Cold War. It consists of a set of satellites that are permanently orbiting the Earth and emitting electromagnetic signals. These signals can be received on earth to pinpoint, through triangulation, the position of the receiver with errors of just a few centimeters. However, GPS's are unable to provide information about orientation and they do not work in indoor scenarios where the satellites' signals cannot be detected.
- Similarly, but at a lower scale, indoor systems use arrays of sensors to track the 3D motion of a special marker. Some are based on infrared cameras and reflectors [15], others on RFID

[1] Etymological reference from https://en.wikipedia.org/wiki/Odometry

tags [16], or even on ultrasounds [17]. The common limitation of all these approaches is the need of a controlled environment, which is not possible in many applications.

- *Inertial Measuring Units*, or IMUs, measure accelerations, rotational velocities and sometimes the magnetic field of the Earth. Through temporal integration, and by detecting the direction of gravity, these devices can be used to estimate translations and orientation. On the one hand, orientation can be obtained quite accurately by leveraging the different measurements that IMUs provide. On the other hand, translations are hard to obtain given the second-order nature of the estimation process and the difficulty associated with canceling the gravitational component of the measurements [18].
- Encoders are utilized to measure wheel turns (wheel odometry) or to know the relative positions of robotic arms, legs, etc. The main and significant drawback associated to wheel odometry is slippage, since in that case the rotation of the wheels does not correspond to the motion of the vehicle/robot. More generally, estimates based on encoders require a precise geometric model of the limbs or parts in motion, which is not always known with such precision.
- Visual and range-based odometries are very flexible solutions since they can be particularized to work with different sensors (RGB or depth cameras, stereo systems, laser scanners) and configurations (2D, 3D or any particular combination of translations and rotations). As a drawback, they are sensitive to the illumination conditions or the solar radiation.

Each of the aforementioned technologies are suitable for some specific tasks, but in recent years visual odometry has demonstrated to be the most precise and versatile alternative. For instance, it is used to provide continuous estimates of the 3D pose of a drone [19], which is mandatory if the drone is to carry out any autonomous task. Similarly, planar odometry based on scan matching is employed to know the position of service robots equipped with laser scanners (e.g. telepresence robots [20]). Planar odometry does not always rely on lidars though, and other platforms like the Roomba vacuum cleaner [21] incorporate solutions based on RGB cameras (probably combined with wheel odometry and proximity sensors). The Microsoft augmented-reality device Hololens [11] includes a set of RGB and depth cameras to perform 3D SLAM, and to do so it must align the incoming photometric and geometric data provided by these cameras in real time. In general, almost any application related to 2D or 3D mapping includes an odometry module to be able to integrate the data from images or scans correctly. As a last illustrative case, visual odometry is even employed for video processing, for example to render smoother versions of time-lapse videos [22].

3.B Rigid Transformations and Lie Algebra

In this section we introduce the mathematical tools utilized throughout this thesis to handle geometric transformations in Euclidean spaces, particularly 2D and 3D rigid transformations.

A rigid transformation is the one that preserves distances between the transformed points, hence moving them as if they were part of a rigid body. Rigid transformations encompass translations and rotations, and are normally expressed as

$$T = \left(\begin{array}{c|c} R & \boldsymbol{t} \\ \hline 0 & 1 \end{array}\right), \tag{3.1}$$

where R is a rotation matrix and $\boldsymbol{t}$ is a translation vector. In the three-dimensional case, $R \in \mathbb{R}^{3\times3}$ and $\boldsymbol{t} \in \mathbb{R}^3$, while in a 2D space $R \in \mathbb{R}^{2\times2}$ and $\boldsymbol{t} \in \mathbb{R}^2$. This matrix can be used to transform points from one reference system to another, or equivalently, to move them according to a certain rigid body motion. If $\boldsymbol{p}$ is a vector with the coordinates of a given point P, its transformed coordinates $\boldsymbol{p}'$ can be computed as

$$\boldsymbol{p}'_h = T\boldsymbol{p}_h, \qquad \boldsymbol{p}'_h = \begin{pmatrix} \boldsymbol{p}' \\ 1 \end{pmatrix}, \qquad \boldsymbol{p}_h = \begin{pmatrix} \boldsymbol{p} \\ 1 \end{pmatrix}, \tag{3.2}$$

where $\boldsymbol{p}_h$ and $\boldsymbol{p}'_h$ are the homogeneous coordinates of the point before and after the transformation, respectively. Using homogeneous coordinates is advantageous because they allow us to concatenate multiple transformations as a simple product of matrices.

The only inconvenience associated to this representation of rigid motions is the fact that it is not minimal because the size of rotation matrices (R) is larger than the actual degrees of freedom (e.g. rotations in 3D can be encoded by a vector of 3 elements but R is 3×3). Fortunately, there exists a theory that connect spatial transformations (Lie groups) and their minimal representation (Lie algebras). A Lie group is a group that is also a differentiable manifold, with the property that the group operations are compatible with the smooth structure. Every Lie group has an associated Lie algebra, which is a vector space generated by differentiating the group transformations along specific directions in the space. This vector space (or tangent space) has the same dimensions as the number of degrees of freedom of the group transformations and is an optimal space to represent differential quantities related to the group [23, 24]. In this thesis we will work with the Lie algebras $\mathfrak{se}(2)$ and $\mathfrak{se}(3)$ associated to 2D and 3D rigid transformations, respectively. The corresponding Lie groups are commonly denoted as SE(2) and SE(3). If $\boldsymbol{\xi} = (\boldsymbol{\nu} \;\; \boldsymbol{\omega})^T$ is a vector of the Lie algebra where $\boldsymbol{\nu}$ encodes the translational component and $\boldsymbol{\omega}$ the rotational one, the rigid transformation associated to it is given by the following exponential map:

$$T = \exp\left(\hat{\boldsymbol{\xi}}\right) = \exp\left(\begin{array}{c|c} \omega_\times & \boldsymbol{\nu} \\ \hline 0 & 0 \end{array}\right), \tag{3.3}$$

where $\omega_\times$ is a skew symmetric matrix built from the rotational components of $\boldsymbol{\xi}$:

$$\omega_\times = \begin{pmatrix} 0 & -\omega \\ \omega & 0 \end{pmatrix}, \quad \text{if } \boldsymbol{\xi} \in \mathfrak{se}(2), \tag{3.4}$$

$$\omega_\times = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}, \quad \text{if } \boldsymbol{\xi} \in \mathfrak{se}(3). \tag{3.5}$$

Similarly, the vector $\boldsymbol{\xi}$ associated to a given transformation T can be computed with the logarithm map:

$$\hat{\boldsymbol{\xi}} = \log(T)\,. \tag{3.6}$$

Last, we want to describe the physical meaning of the vector $\boldsymbol{\xi}$, which is normally missing in the literature. For clarity we will consider only the 3D case, but the explanation is equally valid for 2D. Let's assume $\boldsymbol{\xi}$ represents the velocity of a rigid body, and P is a point that belongs to that rigid body. Under these assumptions, the instant velocity of P can be expressed as

$$\boldsymbol{v}_P = \begin{pmatrix} \nu_x + z\omega_y - y\omega_z \\ \nu_y - z\omega_x + x\omega_z \\ \nu_z + y\omega_x - x\omega_y \end{pmatrix}, \tag{3.7}$$

where (x, y, z) are the coordinates of P. To obtain the position of P at a given time t we integrate its instant velocity:

$$\boldsymbol{p}(t) = \boldsymbol{p} + \int_0^t \boldsymbol{v}_p\, dt\,. \tag{3.8}$$

Note that this integral must be computed numerically because the coordinates are interdependent. It can be proved (although we do not do it in this thesis) that, if we compute the position of P at t=1, that position coincides with $\boldsymbol{p}'$ (3.2):

$$\boldsymbol{p'} = \boldsymbol{p}(1) = \boldsymbol{p} + \int_0^1 \boldsymbol{v}_p\, dt\,. \tag{3.9}$$

Therefore, the vector $\boldsymbol{\xi}$ contains the translational and rotational velocities of a rigid body according to which P moves (temporally normalized). If the transformation does not correspond to any motion but to a change of reference frame, then $\boldsymbol{\xi}$ represents the rigid velocity that would bring one of the reference frames towards the other in one second.

3.C Coarse-to-Fine and Theory of Warping

The problem of dense image alignment has extensively been studied in computer vision. The term *dense* means that all the image pixels must be aligned, not only those observing special features. The two-dimensional motion field that encodes the displacement of each individual pixel on the image plane is called *optical flow*. Virtually every problem related to image alignment involves the estimation of the optical flow implicitly or explicitly. In visual odometry, the relation between the camera motion and the optical flow is normally embedded in the formulation and the projection model (e.g. pinhole). Another case/example of interest is scene flow, which is often decomposed as the estimation of optical flow plus range flow (or disparity). All these problems are non-convex and require linear approximations, commonly based on the optical flow constraint [25], to be solved. However, these linearizations are quite restrictive. Since they rely on the image gradients, each constraint becomes valid only locally in a small region surrounding each pixel, providing very little basin of convergence for many algorithms. Fortunately, this limitation has been overcome in the past by imposing such linearizations in a coarse-to-fine scheme [26, 27].

A coarse-to-fine strategy consists in building a pyramid of images which are subsequently aligned from the coarsest to the finest level. The coarsest levels offer a wider basin of convergence

Figure 3.1: Example of an image pyramid. Resolution goes from VGA (left) to 15×20 (right). The first rows show two consecutive intensity images taken with an RGB-D camera, and the last row shows the residual image (absolute difference between the two). It can be observed that the misaligned areas cover many pixels in the original images, whereas they are only one or two pixels wide in the coarsest image. Hence, any linearization applied on the finest level would not help to align both images while the same process applied on the coarsest level would work (but not with precision).

because each pixel there covers a "larger area" of the scene (Figure 3.1), whereas the finest levels allow for precise alignment once the algorithm gets close to the solution. To avoid alliasing and information loss, image pyramids are built by combining downsampling and smoothing. The former typically divides the image resolution by two at every step, and the latter is computed by convolving the image with a Gaussian kernel. However, when it comes to geometric data, Gaussian smoothing generates spurious points at depth discontinuities and, for that reason, a bilateral filter [28] can be used instead.

Once the solution is obtained for a given level of the coarse-to-fine scheme, that solution is used to warp one of the two images to align it with the other. This step is fundamental because the linearizations at the following levels would not hold otherwise. There are different strategies to perform this warping, depending on the addressed problem. Next, we describe the basic warping strategy based on the optical flow, and another case of interest for this thesis: when both the scene geometry and the rigid transformation between the images are known.

3.C.1 Warping with optical flow

Let $I_1, I_2 : \Omega \to \mathbb{R}$ be two intensity images, where $\Omega \subset \mathbb{R}^2$ denotes the image domain. An image pyramid is computed for each input image; $I^L_{(\cdot)}$ will be used to refer to the image of the L-th level of the pyramid. If $\boldsymbol{u}^L$ is the optical flow that maps I^L_2 towards I^L_1 then:

$$I^L_2(\boldsymbol{x} + \boldsymbol{u}^L) \approx I^L_1(\boldsymbol{x})\,, \tag{3.10}$$

where $\boldsymbol{x}$ represents the coordinates of a given pixel. The warping in this case is performed by creating a new image $I^{L+1}_{2,w}$ for the next level which is as similar/close as possible to I^{L+1}_1, i.e.

$$I^{L+1}_{2,w}(\boldsymbol{x}) = I^{L+1}_2(\boldsymbol{x} + \hat{\boldsymbol{u}}^L)\,, \tag{3.11}$$

where $\hat{\boldsymbol{u}}^L$ is an upsampled version of $\boldsymbol{u}^L$. Subsequently, at level $L+1$, the new warped image $I_{2,w}^{L+1}$ will be aligned with I_1^{L+1} with a finer optical flow $\boldsymbol{u}^{L+1}$. This process is repeated for each level, thereby concatenating the solutions obtained throughout the pyramid and bringing I_2 closer to I_1 at each new step.

3.C.2 Warping with rigid transformations

In this section we describe the warping for visual odometry with RGB-D cameras and therefore assume that the geometry of the scene is known. Let $I_1, I_2 : \Omega \to \mathbb{R}$ and $Z_1, Z_2 : \Omega \to \mathbb{R}$ be two registered pairs of intensity and depth images, respectively. In this case four image pyramids are built, and $I_{(.)}^L$ and $Z_{(.)}^L$ will denote the intensity and depth images of the L-th level of these pyramids. In visual odometry, the optical flow is not computed explicitly but it can be calculated from the estimated transformation and the camera projection model. Let $\pi : \mathbb{R}^3 \to \Omega$ be a function that projects 3D points onto the image plane and $\pi^{-1} : \Omega \times \mathbb{R} \to \mathbb{R}^3$ be the inverse projection function which provides the 3D coordinates of any observed point. If $T_L \in \mathrm{SE}(3)$ is the rigid transformation that aligns the points of the two images at a given level L, then

$$I_2^L \left(\pi(T_L\, \pi^{-1}(\boldsymbol{x}, Z_1^L(\boldsymbol{x})))\right) \approx I_1^L(\boldsymbol{x}) \tag{3.12}$$

It must be remarked that T_L actually results from composing all the transformations obtained from the first until the L-th level of the pyramid. When working with RGB-D data, both intensity and depth images must be warped according to the estimated transformation:

$$I_{2,w}^{L+1} \left(\pi(T_L^{-1}\, \pi^{-1}(\boldsymbol{x}, Z_2^{L+1}(\boldsymbol{x})))\right) = I_2^{L+1}(\boldsymbol{x})\,, \tag{3.13}$$

$$Z_{2,w}^{L+1} \left(\pi(T_L^{-1}\, \pi^{-1}(\boldsymbol{x}, Z_2^{L+1}(\boldsymbol{x})))\right) = \left|T_L^{-1}\, \pi^{-1}(\boldsymbol{x}, Z_2^{L+1}(\boldsymbol{x}))\right|_z\,, \tag{3.14}$$

where $|\bullet|_z$ is the z-coordinate. In this case, the inverse transformation T_L^{-1} is employed to transform the whole coloured point cloud (computed from the second image) and render a new artificial image from the point of view of I_1, Z_1. This is the most accurate procedure to obtain the warped images but it is not the most efficient one because it involves averaging all the projections and/or keeping a z-buffer to take the ones which are visible from the camera. There exists a fast alternative warping, equivalent to the one presented in (3.11), which can be computed as

$$I_{2,w}^{L+1}(\boldsymbol{x}) = I_2^{L+1} \left(\pi(T_L\, \pi^{-1}(\boldsymbol{x}, Z_1^{L+1}(\boldsymbol{x})))\right)\,, \tag{3.15}$$

$$Z_{2,w}^{L+1}(\boldsymbol{x}) = Z_2^{L+1} \left(\pi(T_L\, \pi^{-1}(\boldsymbol{x}, Z_1^{L+1}(\boldsymbol{x})))\right)\,. \tag{3.16}$$

This method computes the warped image pixel-wise and hence it is easy to parallelize, but it might create artefacts because it combines observations from both RGB-D pairs which do not observe exactly the same points of the scene.

3.D Contributions

We have developed a new dense method for range-based odometry. This method expresses the range flow constraint [29, 30] as a function of the motion of the sensor, and provides accurate estimates by minimizing the resulting overconstrained system. The two main advantages of this

method are its low computational runtime and the fact that it does not rely on photometric data. Consequently, it can be particularized to work with any range sensor.

First, we present our initial work on 3D visual odometry with depth cameras (§3.E). The algorithm has a low computational cost if compared to other existing approaches and runs in real time on a single CPU core (QVGA resolution). Aside from the main formulation, the paper describes a new temporal filter for the camera pose based on the uncertainty of the estimates, and a special strategy to compute the image gradients. It does not incorporate robust functions in the minimization problem (weighted squared residuals are minimized) and, consequently, its performance deteriorates in the presence of moving objects.

Second, we adapt that formulation to estimate planar motion with 2D laser scanners (§3.F). In this case we minimize a robust function of the geometric residuals to improve accuracy against moving object. Thus, we include a two-fold strategy to robustify our algorithm: a pre-weighting stage (similar to §3.E) to downweight points for which the range flow constraint (linearization) does not hold, and a robust penalty function to handle the remaining outliers during the optimization. Results show that our method is very precise and much faster (0.9 millisecond) than other scan-matching algorithms.

Third, in §3.G we extend the work presented in §3.F by introducing a novel symmetric range flow constraint and aligning multiple scans at each iteration. The symmetric formulation equidistributes the estimated motion in both scans, which results in a lower linearization error. The multi-scan approach combines consecutive and keyscan-based alignment to reduce drift and increase robustness against moving objects. Moreover, we present a new keyscan-selection criterion that allows us to impose thresholds directly on the error domain (as opposed to traditional strategies which put limits to other related magnitudes like the maximum translation or rotational, average residual, etc.). We include a thorough experimental evaluation demonstrating that our method outperforms state-of-the-art algorithms in scan matching.

3.E Fast Visual Odometry for 3-D Range Sensors

Mariano Jaimez and Javier Gonzalez-Jimenez

Published in IEEE Transactions on Robotics, 2015.

Fast Visual Odometry for 3-D Range Sensors

Mariano Jaimez and Javier Gonzalez-Jimenez

Abstract

This paper presents a new dense method to compute the odometry of a free-flying range sensor in real time. The method applies the range flow constraint equation to sensed points in the temporal flow to derive the linear and angular velocity of the sensor in a rigid environment. Although this approach is applicable to any range sensor, we particularize its formulation to estimate the 3-D motion of a range camera. The proposed algorithm is tested with different image resolutions and compared with two state-of-the-art methods: generalized iterative closest point (GICP) [2] and robust dense visual odometry (RDVO) [3]. Experiments show that our approach clearly overperforms GICP which uses the same geometric input data, whereas it achieves results similar to RDVO, which requires both geometric and photometric data to work. Furthermore, experiments are carried out to demonstrate that our approach is able to estimate fast motions at 60 Hz running on a single CPU core, a performance that has never been reported in the literature. The algorithm is available online under an open source license so that the robotic community can benefit from it.

3.E.1 Introduction

Fast and accurate 6-DoF visual odometry (VO) is gaining importance in current robotics where increasingly demanding applications are pursued. Two clear examples are terrestrial vehicles that must operate on uneven terrains and UAVs, which often need their 3-D pose to be tracked in order to fly autonomously. The alternative to VO in these cases is applying inertial navigation based on IMUs, but they accumulate too much error over time due to their inability to cancel gravity with enough exactitude [3]. On the other hand, traditional solutions like wheel odometry or GPS navigation simply cannot replace VO as they are not able to provide 3-D pose estimates. Another important advantage of VO is that the required sensorial data (provided by cameras or laser scanners) is also exploited by other robotic modules, both for navigation (SLAM, obstacle avoidance, etc.) and for scene understanding.

The emergence of RGB-D cameras has given rise to new and promising prospects in VO but has also posed some challenges. These sensors are able to provide RGB and depth images simultaneously at 30–60 Hz, which is a huge amount of data to process. For this reason, novel VO approaches struggle to maintain a good computational performance while trying to make the most of all these incoming data and frequently tend to focus on either RGB or depth images. In any case, not many VO methods can actually run at 30 Hz, and very few of them reach the maximum frame rate of 60 Hz that some RGB-D cameras offer.

In this paper, we introduce a novel VO method called DIFODO (DIFferential ODOmetry), which takes 3-D range images (or scans) to estimate the linear and angular velocity of the sensor. Its formulation is founded on the spatiotemporal linearization of a range function (the so-called *range flow constraint equation* [29, 31], which is imposed pixel-wise in a coarse-to-fine scheme in order to cope with the estimation of large motions. A distinct feature of DIFODO is that the same input data (range measurements) are exploited both to obtain the camera motion at each level of the coarse-to-fine pyramid and to perform the warping after the level transitions. Thus, although here we particularize its formulation for depth cameras, DIFODO could be

easily adapted to work with any range sensor. Another key characteristic of DIFODO is that, in contrast to other VO methods, it relies on a closed-form solution and runs in real time on a single CPU core. As a consequence, DIFODO is able to estimate fast motions and finer trajectories as it can work at the highest camera frame rate (60 Hz). As a downside, it shares the same weakness than other geometry-based VO approaches, namely: the estimation problem becomes underdetermined when the observations of the scene lack of enough geometric information, which similarly occurs for VO approaches relying on photometric data when observing low textured areas. The idea of our proposal arose from [5] and [6] which, inspired by the concept of optical flow, presented an algorithm to estimate 2-D motion from laser scans.

In order to validate our method, extensive experimentation has been carried out. First, DIFODO is tested with different resolutions to analyze how its performance changes with the number of points and levels considered. Secondly, it is compared with two prominent methods: Generalized-ICP [2] and the robust dense VO algorithm proposed by Kerl *et al.* [3] (RDVO from here onwards). The former is one of the most widespread VO strategies based on geometry, and hence, it is the best candidate to compare with given that it uses the same input data. The latter is one of the best-performing methods published recently and, conversely to our approach, exploits both geometric and photometric data to estimate the camera motion. Results show that DIFODO is about 30 times faster than generalized iterative closest point (GICP) and 2–3 times more accurate, whereas it achieves a performance similar to RDVO with less input data (only depth). Finally, quantitative and qualitative results are presented to demonstrate that DIFODO is able to estimate fast and real motions, which makes it suitable for real-time applications.

The code has been added to MRPT [32] and is available under an open-source license. An illustrative video of our approach, together with the code, can be found here:

`http://mapir.isa.uma.es/mjaimez`

3.E.2 Related Work

Although the term *visual odometry* was first introduced by Nistér in 2004 [33], the problem of estimating motion from visual inputs has been addressed from different perspectives during the last 30 years. Traditionally, VO systems have been developed for grayscale images coming from one camera or a stereo pair [34]. This general approach usually relies on detecting and matching visual features, having to deal with the problem of data association and outliers [33, 35, 36]. Most solutions resort to RANSAC [37] to solve this limitation and, although the presence of outlier matches is an inherent limitation for these strategies, great results have been achieved (e.g., in planetary exploration [35]). Most recently, the semidense approach of Engel *et al.* [38] obtains very accurate motion estimates by imposing photoconsistency at image regions with non-negligible gradients and estimating depth within a probabilistic framework. However, an important drawback of these methods based only on grayscale images is that their performance deteriorates considerably if the illumination conditions are poor.

Alternatively, range data in the form of 2-D scans have proven to be suitable to estimate planar motion. General point registration methods, most of which are variants of ICP [39], have been extensively utilized to find the homogeneous transformation between consecutive scans

and have been applied not only in VO but also in Localization, Mapping, or SLAM [40, 41]. On the other hand, some methods were specifically conceived to work with 2-D range scans. In [42], a probabilistic framework is proposed to find the rigid-body transformation that maximizes the probability of having observed a scan given the previous one. Rather than trusting a local search to find the global maximum, a multiresolution CPU implementation is proposed to perform a search over the entire space of plausible rigid-body transformations, obtaining good results in simulation. Because of its relation with the proposal here, we have to mention the work of Gonzalez [31], who introduced the concept of *range image flow* and particularized the range flow constraint equation to 2-D scans to estimate the scanner motion in a very straightforward manner. Yet, its applicability was only tested in simulated and simple environments.

Recently, the advent of the new and affordable RGB-D cameras has revolutionized research in robotics. The huge amount of geometric data contained in the depth images, along with the fact that it can be easily combined with traditional RGB ones, have given rise to a fair number of new VO approaches. Some of them are similar to those relying on visual features but they incorporate depth either to directly calculate the geometric transformation between matched features or to improve the outlier rejection stage [43, 44]. Likewise, in [45], visual features are detected and fused to a global model of features against which every new set is registered using ICP. A completely different alternative consists in minimizing an energy function that is usually related to the photometric or geometric error, i.e., the differences between consecutive RGB or depth images when one of them is reprojected or warped against the other. This idea was first presented in [46], although in this case the authors used grayscale images generated by stereo pairs. In [47], this concept is first applied to RGB-D images and both photometric and geometric errors are minimized in a variational framework, yet lacking implementation details and quantitative results. Similarly, in [48], the focus is on the photometric error and the authors employ the depth images only to construct the warping function between consecutive intensity images. This work was extended in [3] by introducing a new probabilistic formulation that produced impressive results. However, despite their high accuracy and robustness, most of these proposals require registered geometric and photometric data to work, which in practice restricts their applicability to systems or robots equipped with RGB-D cameras.

There is another group of methods which, as we do, exclusively make use of the 3D geometric data provided by range sensors. Either ICP or some of its variants have been employed in this regard but it is probably Generalized-ICP [1] which has demonstrated the best performance, being commonly taken as a reference for comparison [48, 49, 50]. On the other hand, other strategies were specifically designed to work with the range images generated by Kinect-like cameras. A remarkable case is KinectFusion [51], which introduces a signed distance function (SDF) to represent the observed scene and applies ICP to align the depth frames against this scene model. In a similar way, the work presented in [52] defines a truncated SDF (TSDF) and registers data directly to the TSDF model, rather than using it to obtain denoised depth images from a virtual sensor (as KinectFusion does). Apart from the great qualitative and quantitative results that these methods have achieved, all of them share two weaknesses. First, they are computationally very expensive, an issue that is typically overcome by developing complex

parallelized CPU or GPU implementations. Second, their performance degrades if the scene does not present sufficient geometric-distinctive features.

Up to date, very few methods truly exploit both depth and RGB images jointly to estimate motion. A recent example is Kintinuous [53], the extension of KinectFusion, which added FOVIS [43] to complement ICP in areas with lack of geometric features. Additionally, in the work of Whelan *et al.* [54] different combinations of [2], [3] and [43] were implemented and tested, reporting good qualitative and quantitative results, and low runtimes by virtue of an exhaustive GPU implementation. The aforementioned work of Tykkala *et al.* [47] could also be considered to belong to this category.

3.E.3 Velocity Constraint derived from the Range Flow Equation

This section describes how the 3-D motion of a range camera can be estimated from two consecutive frames by applying the range flow constraint and the restriction it imposes to the velocities of the observed points. Under the common assumption of a rigid scene, we formulate the point velocities in terms of the camera motion, and hence the latter can be calculated if a sufficient number of points are considered. Let $Z : (\Omega \subset \mathbb{R}^2) \rightarrow \mathbb{R}$ be a depth image provided by a 3-D range camera where Ω is the image domain. According to the work of Spies *et al.* [29], the range flow constraint equation for range cameras reads:

$$\dot{Z} = \frac{\partial Z}{\partial t} + \frac{\partial Z}{\partial u}\dot{u} + \frac{\partial Z}{\partial v}\dot{v} + O(\Delta t, \dot{u}, \dot{v}) \Rightarrow \tag{3.17}$$

$$\dot{Z} \approx Z_t + Z_u\,\dot{u} + Z_v\,\dot{v}\,, \tag{3.18}$$

where $\boldsymbol{w} = (u, v)$ are the pixel coordinates. Equation (3.18) reflects that the total derivative of the depth can be calculated from the optical velocity $\dot{\boldsymbol{w}} = (\dot{u}, \dot{v})$ and the partial derivatives of Z with respect to the time t, u and v (Z_t, Z_u and Z_v, respectively). Since (3.18) is derived from a first-order Taylor series expansion (3.17), it is exact only when the higher order terms $O(\Delta t, \dot{u}, \dot{v})$ are negligible. In practice, this condition is fulfilled if either the displacement between consecutive images is small or the observed points belong to planar patches where the linearization holds.

The three partial derivatives of Z can be directly calculated from the depth images, but $\dot{Z}, \dot{u}$ and $\dot{v}$ are unknowns and should be expressed in terms of the camera velocity. Let $\boldsymbol{\xi} = (v_x, v_y, v_z, \omega_x, \omega_y, \omega_z)^T$ be the camera velocity and $\boldsymbol{P}_c = (x, y, z)$ the spatial coordinates of an arbitrary point P of the scene, both described in the reference frame of the camera. The relationships between $\dot{Z}, \dot{\boldsymbol{w}}$ and $\boldsymbol{P}_c$ can be deduced from the pinhole camera model, assuming that the pixel and spatial coordinates of P are time-varying:

$$v = f_y\frac{y}{z} + v_m \Rightarrow \dot{v} = f_y\left(\frac{\dot{y}z - \dot{z}y}{z^2}\right)\,, \tag{3.19}$$

$$u = f_x\frac{x}{z} + u_m \Rightarrow \dot{u} = f_x\left(\frac{\dot{x}z - \dot{z}x}{z^2}\right)\,, \tag{3.20}$$

where (u_m, v_m) is the image center (principal point) and f_x, f_y are the focal length values, all expressed in pixels. With respect to the camera reference frame, and under the assumption of

a rigid and static scene, all observed points move with a velocity that is equal to the camera velocity but opposite in sign. Thus, the velocity of any 3D point with respect to the camera can be expressed as

$$\dot{\boldsymbol{P}}_c = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix} = \begin{pmatrix} -v_x - z\omega_y + y\omega_z \\ -v_y + z\omega_x - x\omega_z \\ -v_z - y\omega_x + x\omega_y \end{pmatrix} . \tag{3.21}$$

As an intermediate step, we express the range flow in (3.18) in terms of the 3-D velocity of P, according to (3.19) and (3.20):

$$-Z_t = -\dot{z} + Z_u f_x \left(\frac{\dot{x}z - \dot{z}x}{z^2} \right) + Z_v f_y \left(\frac{\dot{y}z - \dot{z}y}{z^2} \right) . \tag{3.22}$$

Notice that Z has been replaced by z because they are equivalent: the capital letter has been used to denote the depth image while the lower case letter directly refers to the spatial coordinate of depth. Finally, rigidity (3.21) is imposed in (3.22):

$$\begin{aligned} -Z_t = &\left(1 + \frac{x f_x}{z^2} Z_u + \frac{y f_y}{z^2} Z_v \right) (v_z + y\omega_x - x\omega_y) \\ &+ \frac{f_x}{z} Z_u (-v_x + y\omega_z - z\omega_y) + \frac{f_y}{z} Z_v (-v_y - x\omega_z + z\omega_x) . \end{aligned} \tag{3.23}$$

Equation (3.23) is a linear restriction that the velocity of a point of the scene (with respect to the camera) has to fulfill and, consequently, imposes a restriction to the camera velocity. Therefore, we can build a solvable algebraic system if at least 6 linearly independent restrictions are available. Notice that not every point will add new information to the system and, as will be described in §3.E.5, the problem could be ill-posed depending on the spatial distribution of the scene points.

The linearization applied to derive the depth flow equation assumes differentiability of the depth images, and either small displacement of the scene or constant depth gradients. As a consequence, in the first place, points on edges must be ruled out since the depth field is not differentiable at the object borders. Secondly, the depth image gradients may not be constant (having higher order derivatives) which implies that, in the most general case, (3.23) only holds for small displacements. Within our formulation, "small displacements" means that the projection of $\dot{\boldsymbol{P}}_c$ onto the image plane is smaller than the neighborhood used to compute the image gradients, typically:

$$|\dot{u}\, \Delta t| < 1, \quad |\dot{v}\, \Delta t| < 1, \tag{3.24}$$

where Δt is the time interval between consecutive frames. Hence, the hypothesis of small displacements involves both the image resolution and the actual 3-D motion of the points.

The solution to the small-displacements restriction was proposed by Brox *et al.* [27] and consists in utilizing a coarse-to-fine scheme. Within this strategy, a Gaussian pyramid is built for the input images and the optical flow is solved from coarser to finer levels, capturing large displacements at the coarsest levels that are subsequently refined throughout the pyramid. At each level, the previously obtained solutions are employed to warp one of the images against the other, leading to image pairs that present less differences than the original pair and for which the

hypothesis of small displacements is fulfilled. This strategy can be applied as well to estimate the camera motion, as explained in [3, 48]. In this case the warping is not performed in the image plane but in the 3D space, and the geometric data captured by range sensors is exploited to warp a given image or measurement according to a spatial transformation. This is the reason why the methods presented in [3, 48] require both geometric and photometric data to work: they impose photo-consistency to estimate motion but they need the geometry of the scene to perform the warping. In our work we adopt the same warping strategy as [3, 48], but since DIFODO relies only on geometric constraints to estimate the camera motion, it can be considered as a purely geometry-based method, and can be generalized to work with any range sensor.

Warping has been extensively applied in computer vision and it is not a contribution of this paper. Therefore, its mathematical formulation is not presented here; more information and details can be found in [3, 48, 27], as well as in §3.C.1.

3.E.4 Solving the Camera Motion

Least-squares solution

As previously mentioned, at least 6 points bearing linearly independent restrictions are required to compute the camera velocity at each level of the pyramid. In practice, however, a much higher number (N) of points are considered to make the solution robust to noise and errors, which leads to an overdetermined linear system that can be solved by weighted least squares:

$$W A \boldsymbol{\xi} = W \boldsymbol{B} \rightarrow \boldsymbol{\xi} = (A^T W A)^{-1} A^T W \boldsymbol{B} = M \boldsymbol{B}\,, \tag{3.25}$$

where $A \in \mathbb{R}^{N\times 6}$ contains the coefficients that multiply ξ in (7), $\boldsymbol{B} \in \mathbb{R}^N$ contains the temporal derivative of depth for each pixel (inverted in sign) and $W \in \mathbb{R}^{N\times N}$ is a diagonal matrix containing the weights associated to the uncertainty of each equation. The $M \in \mathbb{R}^{6\times 6}$ matrix in (9) is symmetric and positive definite or semi-definite, so a Cholesky LDLT factorization (as implemented in Eigen [55]) can be used to compute the solution. Since such factorization applies to a small-size matrix (6×6), it does not condition the computational cost of our method. Although not all the steps of the algorithm have been detailed yet (see §3.E.6), the only operations whose complexity is quadratic with the number of points are the products $A^T W A$ and $A^T W \boldsymbol{B}$ (using dense algebra). Nonetheless, the algebraic system (3.25) can be rewritten in a way that these products do not need to be computed: by multiplying both sides of (3.23) by the square root of the corresponding weight and solving the resulting equation system with the pseudoinverse matrix:

$$A^w \boldsymbol{\xi} = \boldsymbol{B}^w \rightarrow \boldsymbol{\xi} = \left((A^w)^T A^w\right)^{-1} (A^w)^T \boldsymbol{B}^w = M \boldsymbol{B}\,, \tag{3.26}$$

$$A^w = \begin{pmatrix} \sqrt{w_1} a_{11} & \dots & \sqrt{w_1} a_{16} \\ \vdots & \ddots & \vdots \\ \sqrt{w_N} a_{N1} & \dots & \sqrt{w_N} a_{N6} \end{pmatrix},\ \boldsymbol{B}^w = \begin{pmatrix} \sqrt{w_1} b_1 \\ \vdots \\ \sqrt{w_N} b_N \end{pmatrix}. \tag{3.27}$$

Thus, the new formulation allows us to recover the motion parameters with a computational time that grows only linearly with the number of points.

Weighting Function

Weights are necessary to adjust the contribution of every point to the overall motion estimate according to the uncertainty or error associated to its range flow equation. Without loss of generality, this error can be expressed as the addition of two terms: the measurement error, which measures how the sensor noise affects the range equation, and the linearization error from the first order approximation in (3.18). The former term is typically modeled by a zero-mean Gaussian distribution and can be calculated propagating the measurement error to the whole equation. The latter does not follow a Gaussian model and, in general, it is more complex to estimate because it involves studying the second order derivatives of depth to evaluate how significant the neglected terms in (3.17) actually are. In this work we address the analysis of both sources of error and present a weighting function that encompasses information about the camera and the geometry of the scene from which to weight the image pixels accordingly.

In order to estimate the measurement error we need to take into consideration every stochastic variable or parameter that appears in (3.23). Assuming that the parameters of the camera are exactly known, (3.23) can be rearranged and expressed as a function of all noise-prone variables, that is, those dependent on the depth:

$$R(x, y, z, Z_t, Z_u, Z_v) = 0\,, \tag{3.28}$$

being x, y, z linearly related for a given pixel (see (3.19), (3.20)).

To propagate the error of these depth-dependent variables to the range flow equation, their covariance matrix $\Sigma_d \in \mathbb{R}^{6\times 6}$ has to be built as a preliminary step:

$$\Sigma_d = \left(\begin{array}{c|c} \Sigma_{xyz} & \Sigma_{(xyz)(Z_{t,u,v})} \\ \hline \Sigma^T_{(xyz)(Z_{t,u,v})} & \Sigma_{Z_{t,u,v}} \end{array}\right)\,, \tag{3.29}$$

where $\Sigma_{xyz},\ \Sigma_{(xyz)(Z_{t,u,v})},\ \Sigma_{Z_{t,u,v}} \in \mathbb{R}^{3\times 3}$. The mathematical derivation of the submatrices in Σ_d is based on the characteristics of Kinect-like cameras and can be found in the §3.E.9. Knowing Σ_d, the variance of (3.23) associated with the measurement errors can be computed as

$$\sigma_m^2 = \nabla R \cdot \Sigma_d \cdot \nabla R^T\,, \tag{3.30}$$

where ∇R is the gradient of R with respect to x, y, z, Z_t, Z_u, Z_v. Given that R also depends on $\boldsymbol{\xi}$, an approximation of ∇R is computed using the twist $\boldsymbol{\xi}$ estimated at the previous time step. On the other hand, to analyze the error derived from the linearization in (3.17) we must rewrite that equation including the second order terms:

$$\begin{aligned} \dot{Z} =& Z_t + Z_u\dot{u} + Z_v\dot{v} + Z_2(\Delta t, \dot{\boldsymbol{w}}) + O(\Delta t^2, \dot{\boldsymbol{w}})\,, \\ Z_2(\Delta t, \dot{\boldsymbol{w}}) =& \frac{\Delta t}{2}\left(Z_{tt} + Z_{tu}\dot{u} + Z_{tv}\dot{u}^2 + Z_{vv}\dot{v}^2 + Z_{uu}\dot{u}^2 + 2\,Z_{uv}\dot{u}\,\dot{v}\right)\,. \end{aligned} \tag{3.31}$$

It can be observed that, neglecting third or higher order terms, the error is a function of all the second order derivatives and the optical flow. To estimate how (3.31) deviates from linearity we can approximate the second order derivatives from the depth images, but the optical flow is not

known in advance. One possible solution to this problem would consist in computing the optical flow from the previous $\boldsymbol{\xi}$. However, this strategy loses its sense when it has to be applied at finer levels of the Gaussian pyramid where the previous solution gives us no information about the optical flow with respect to warped images. For that reason, we chose a quadratic expression which penalizes the second derivatives homogeneously for all pixels:

$$\delta_l^2 = k_l \left[\Delta t^2 (Z_{tu}^2 + Z_{tv}^2) + Z_{uu}^2 + Z_{vv}^2 + Z_{uv}^2\right] . \tag{3.32}$$

The constant k_l weights the linearization error against the measurement error, and the time increment multiplies the temporal derivatives so that all terms are of the same order of magnitude (depth differences). In practice, it can be noticed that a high penalization of the second order derivatives might discard some points or areas of the scene that, despite its inaccuracy, are useful to constrain the motion estimate. A deeper study of the scene geometry could be performed to detect beforehand how the range equation of every point would constrain the velocity estimate and adapt k_l accordingly. However, this procedure would significantly increase the computational cost of the algorithm and, hence, it is not adopted in this work. Besides, the second temporal derivative of Z is not considered in (3.32) because it cannot be estimated at the different pyramid levels. With the exception of the coarsest level, the others involve warped images which are timeless and for which the concept of second temporal derivative makes no sense.

If N points are considered to build (3.25), for each point $i \in 1, 2 \dots N$, its corresponding weight is inversely related to the uncertainty associated to the range flow equation:

$$W_{ii} = \frac{1}{\sigma_{m,i}^2 + \delta_{l,i}^2} . \tag{3.33}$$

3.E.5 Scene Geometry, Covariance Analysis and Velocity Filtering

Depending on the spatial distribution of the points used to build the algebraic system (3.26), the problem can be well or ill-posed. If the points contain sufficient information about how the scene has changed in the 3 directions of space then the matrix M will be positive definite and (3.26) will stand for a well-posed configuration, otherwise the matrix M will be rank deficient (or close to). Excluding the degenerated singular cases of some surfaces of revolution (i.e. the camera in the center of a sphere), sufficient information is guaranteed if the normals of the observed surfaces can make up a 3D vector basis, that is

$$\text{rank}\left([\boldsymbol{n}_1, \boldsymbol{n}_2, \dots, \boldsymbol{n}_N]\right) = 3 , \tag{3.34}$$

where $\boldsymbol{n}_i \in \mathbb{R}^3$ is the surface normal vector at a given point i. The unfulfillment of this condition leads to the well-known sliding problem of ICP [56], and frequently occurs when the whole point cloud comes from one or two planes (a wall, the floor, etc.).

If M is positive definite, the points of the scene provide enough geometric constraints to estimate the 6 DoF of the motion. On the contrary, if M has not full rank, some linear or angular velocity terms cannot be estimated and the solution that the solver provides for these variables is

arbitrary. This casuistry can be detected analyzing the covariance matrix $\Sigma_\xi \in \mathbb{R}^{6\times6}$ associated to the least squares solution:

$$\Sigma_\xi = \frac{\sum_{i=1}^{N} r_i^2}{N-6} \left((A^w)^T A^w \right)^{-1} , \tag{3.35}$$

where r_i are the geometric residuals. The diagonal elements of the covariance matrix reflect the variance of the components of the estimated $\boldsymbol{\xi}$ in the reference frame of the camera. In general, it is more meaningful to express the matrix Σ_ξ in a diagonal form where the eigenvalues indicate which combinations of motions (eigenvectors) are constrained and which are undetermined. In this way, any uncertainty in the velocity estimate that may appear due to poor geometric information will be revealed: its corresponding eigenvalue will reflect how high or low this uncertainty is while its eigenvector will contain the velocity terms affected by it. This information can be employed to neglect those velocity terms whose variance (eigenvalue) is too high. Thus, if data from other sources like IMUs or wheel odometry were available they could be fused with the VO estimation to generate a more robust solution. When this is not the case, a suitable option consists of applying a smooth filter based on the current and previous estimates, as explained next.

Let $E = \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_6\}$ be an orthogonal basis comprising the eigenvectors ($\boldsymbol{e}_{(.)} \in \mathbb{R}^6$) of Σ_ξ, and $D \in \mathbb{R}^{6\times6}$ the covariance diagonal matrix containing the associated eigenvalues. At a given interval of time $[t, t+1]$, the solution provided by the solver $\boldsymbol{\xi}^{t,s}$ and the previous velocity estimate $\boldsymbol{\xi}^{t-1}$ have to be expressed in the basis E^t, which is always computed from the last covariance matrix $\Sigma_{\boldsymbol{\xi}}^t$. Then, the filtered velocity $\boldsymbol{\xi}_E^t$ in the basis E^t is obtained as a weighted sum of the current estimate and the previous velocity:

$$\left[(1+k_1)I + k_2 D^t\right] \boldsymbol{\xi}_E^t = \boldsymbol{\xi}_E^{t,s} + (k_1 I + k_2 D^t)\, \boldsymbol{\xi}_E^{t-1} . \tag{3.36}$$

Equation (3.36) represents a low-pass filter with dynamic smoothing, where k_1 helps to soften the estimated trajectory and k_2 controls how the eigenvalues in D affect the final estimate. A high value of k_2 implies that those velocity terms with uncertainty will be approximated to their previous value in $\boldsymbol{\xi}_E^{t-1}$, whereas the current estimate (interval $[t, t+1]$) will have a higher importance if k_2 is low.

3.E.6 Framework and Implementation

Overall, the algorithm carries out a sequence of steps that are depicted in Algorithm 1. It receives as inputs the new depth image, the previous pyramid of depth images and the last motion estimate, and yields the average camera linear and angular velocities during the last interval of time, from which the camera pose will be updated. Important aspects and implementation details of this algorithm are explained below.

Algorithm 1 DIFODO

Inputs:
New depth frame $-Z^t$
Previous Gaussian pyramid $-GP(Z^{t-1})$
Previous velocity estimate $-\boldsymbol{\xi}^{t-1}$

Build the new Gaussian pyramid $\rightarrow GP(Z^t)$
for i=1:number of pyramid levels **do**
  if $(i > 1)$ **then**
    Perform warping (see §3.C.2)
  end if
  Discard pixels with null depth
  Compute depth derivatives $\rightarrow Z_t^t, Z_u^t, Z_v^t$
  Compute the weighting function $\rightarrow W^t$
  Solve weighted least squares $\rightarrow \boldsymbol{\xi}_i^{t,s}, \Sigma_{\xi,i}^t$
  Filter level solution $\rightarrow \boldsymbol{\xi}_i^t$
end for
Compute overall solution and update pose $\rightarrow \boldsymbol{\xi}^t$

Gaussian Pyramid

The Gaussian pyramid is built by downsampling the depth images with a standard 5×5 Gaussian kernel. However, a pure Gaussian smoothing would create artifacts in the depth images because it would mix very dissimilar depth values at the object borders. For that reason, a bilateral filter is applied instead to preserve the geometry of the scene. In our work, the Gaussian pyramid starts to be built at a QVGA resolution (240×320). As will be discussed in §3.E.7, in order to estimate fast motions it might be advantageous to run DIFODO with a lower resolution of 120×160 (QQVGA), in which case the image pyramid should be built from this resolution onwards, saving computational time.

Warping

At every new level of the Gaussian pyramid, one of the two depth images must be warped against the other. To perform the warping, all the motion estimates from the previous levels must be integrated to obtain the overall transformation accumulated up to the present level. A special case is, of course, the first level where no warping is needed. In our formulation, the new frame is always warped against the old frame and the motion estimates of every level are expressed in the same reference frame: the last known pose of the camera.

Depth Image Gradients and Weights

The implementation of DIFODO requires a discrete formulation that estimates the average camera velocity between two consecutive depth frames. Unlike in intensity-based image alignment, where intensity gradients remain almost constant for all perspectives from which the scene is observed, the depth gradients change as the camera moves. For this reason, the depth gradients should not be computed from either the initial or the final depth images at a given interval of time.

As an alternative, we demonstrate that a better choice to approximate the depth gradients consists in applying a trapezoidal solution that averages values from both images. If $\Delta \boldsymbol{w} = (\dot{u}\,\Delta t, \dot{v}\,\Delta t)$ is the motion of a given point projected onto the image domain (optical flow), its depth motion is given by:

$$\dot{Z}(\boldsymbol{w})\Delta t = Z^t(\boldsymbol{w} + \Delta \boldsymbol{w}) - Z^{t-1}(\boldsymbol{w})\,. \tag{3.37}$$

Equation (3.37) is the general and non-linear expression of geometric consistency from which (3.18) is derived. Instead of substituting the term $Z^t(\boldsymbol{w} + \Delta \boldsymbol{w})$ by the standard Taylor linearization, we employ a more accurate approximation which weights the initial and final depth gradients at a given time instant:

$$Z^t(\boldsymbol{w} + \Delta \boldsymbol{w}) = Z^t(\boldsymbol{w}) + \nabla \left(\frac{Z^t(\boldsymbol{w}) + Z^t(\boldsymbol{w} + \Delta \boldsymbol{w})}{2} \right) \cdot \Delta \boldsymbol{w}\,. \tag{3.38}$$

At the right-hand side of (3.38), the term $Z^t(\boldsymbol{w} + \Delta \boldsymbol{w})$ is still present and needs to be expressed as a function of the previous depth image and the depth motion, according to (3.37):

$$Z^t(\boldsymbol{w} + \Delta \boldsymbol{w}) = Z^t(\boldsymbol{w}) + \nabla \left(\frac{Z^t(\boldsymbol{w}) + Z^{t-1}(\boldsymbol{w}) + \dot{Z}^t(\boldsymbol{w})\Delta t}{2} \right) \cdot \Delta \boldsymbol{w}\,. \tag{3.39}$$

As commonly done in variational methods, we can impose regularization over the depth motion and assume that the depth motion field is locally constant:

$$\nabla \dot{Z}^t(\boldsymbol{w}) \sim 0\,. \tag{3.40}$$

Finally, this smoothness constraint is imposed in (3.39) which, in turn, is substituted in (3.37) to obtain a more accurate expression of the range flow constraint equation:

$$\dot{Z}^t(\boldsymbol{w}) \simeq \frac{Z^t(\boldsymbol{w}) - Z^{t-1}(\boldsymbol{w})}{\Delta t} + \nabla \left(\frac{Z^t(\boldsymbol{w}) + Z^{t-1}(\boldsymbol{w})}{2} \right) \cdot \dot{\boldsymbol{w}}\,. \tag{3.41}$$

On the other hand, the image gradients must be approximated by some finite difference formula, given that Ω is not a continuous domain but a discrete one. Most of the times this aspect does not receive much attention in the literature and a certain constant kernel is applied over the whole image. However, this is not the best strategy to compute the depth gradients because it leads to very high values at the object borders which does not reflect the real gradients of the surfaces of these objects. As an alternative, we make use of an adaptive formula of finite differences which, for a point P observed at a pixel $\boldsymbol{w}$, computes the depth gradients taking into account those surrounding pixels that observe points close to P. Mathematically, it implies that two nearness functions $r_u, r_v : \Omega \to \mathbb{R}$ must be computed as

$$r_u(u,v) = \frac{1}{\|X(u+1,v) - X(u,v), Z(u+1,v) - Z(u,v)\|}\,, \tag{3.42}$$

$$r_v(u,v) = \frac{1}{\|Y(u,v+1) - Y(u,v), Z(u,v+1) - Z(u,v)\|}\,, \tag{3.43}$$

where $\|\bullet\|$ is the Euclidean norm and $X, Y : \Omega \to \mathbb{R}$ represent the x, y coordinates of the points of the scene. The depth gradients are calculated then from the depth images and the nearness functions:

$$Z_u(u,v) = \frac{r_u(u,v)\, Z_u^+(u,v) + r_u(u-1,v)\, Z_u^-(u,v)}{r_u(u,v) + r_u(u-1,v)}\,, \tag{3.44}$$
$$Z_u^+(u,v) = Z(u+1,v) - Z(u,v), \quad Z_u^-(u,v) = Z(u,v) - Z(u-1,v)\,,$$

$$Z_v(u,v) = \frac{r_v(u,v)\, Z_v^+(u,v) + r_v(u,v-1)\, Z_v^-(u,v)}{r_v(u,v) + r_v(u,v-1)}\,, \tag{3.45}$$
$$Z_v^+(u,v) = Z(u,v+1) - Z(u,v), \quad Z_v^-(u,v) = Z(u,v) - Z(u,v-1)\,.$$

For simplicity's sake, the temporal superscripts have been omitted. These expressions always upweight the closest points in space to approximate the depth gradients, hence avoiding the assignment of huge values at the object borders. If they are evaluated at pixels which do not lie on borders of objects then the right and left derivatives (Z^+ and Z^-) will be similarly weighted, which results in a standard centered approximation of the image gradients.

Regarding the weighting function, the constant k_l that weights the linearization error against the measurement error is set to $5 \cdot 10^{-6}$. The second order derivatives are approximated by constant centered formulas (using the stencil [-1 0 1]) applied over the first order derivatives.

Filter Velocity and Update Pose

The velocity estimate must be filtered at each level of the pyramid because each level can suffer from the lack of geometric distinctive features. However, the last velocity estimate cannot be used directly to filter the solution of every level because all the previous levels have already estimated some motion which should be subtracted from it. The sequential steps that the filter performs at each level of the pyramid are:

1. Compute the overall estimate $\boldsymbol{\xi}^{t,acu}$ accumulated up to the present level i.
2. Subtract $\boldsymbol{\xi}_i^{t,acu}$ from the last velocity estimate $\boldsymbol{\xi}^{t-1}$ to obtain $\boldsymbol{\xi}_i^{t,sub}$.
3. Compute the covariance matrix $\Sigma_{\xi,i}^t$.
4. Calculate the eigenvalues D_i^t and the eigenvectors E_i^t.
5. Express the least squares solution $\boldsymbol{\xi}_i^{t,s}$ and $\boldsymbol{\xi}_i^{t,sub}$ in the base E_i^t.
6. Apply (3.36) with $\boldsymbol{\xi}_{i,E}^{t,s}$ and $\boldsymbol{\xi}_{i,E}^{t,sub}$ as inputs, and obtain $\boldsymbol{\xi}_{i,E}^t$.
7. Transform $\boldsymbol{\xi}_{i,E}^t$ to the reference frame of the camera.

When the process is finished in all the pyramid levels, the final motion estimate $\boldsymbol{\xi}^t$ is computed and integrated over the last time interval to update the camera pose.

Although the same filtering process is followed in every level, there is an important aspect that should be taken into account: the coarsest levels capture the trend of the motion while the

last levels refine it to get an accurate estimate. As a consequence, the coarsest levels are likely to get solutions which are similar to the previous velocity of the camera whereas the estimates of the finer levels are probably quite independent of it. Therefore, our filter should give a weight to the last motion estimate that decreases from coarser to finer levels. To this end, and to smooth the trajectory estimates, we have empirically chosen the following weighting functions:

$$k_1 = 0.5\, e^{-(l-1)}, \quad k_2 = 0.05\, e^{-(l-1)}\,, \tag{3.46}$$

where l is the pyramid level that ranges from 1 (coarsest) to the number of levels considered. These functions have heuristically proved to be a good trade-off between smoothness of the estimated trajectory and capability to accommodate motion changes in a huge variety of scenes (as shown in §3.E.7).

3.E.7 Experiments and Results

This section is divided into four parts corresponding to four distinct series of experiments. Firstly, DIFODO is tested with different resolutions to analyze how its performance changes with the number of pyramid levels considered. Secondly, several experiments are conducted to compare DIFODO, GICP [2] and RDVO [3] focusing on their speed and accuracy. Thirdly, we demonstrate the suitability of our approach to estimate fast motions, comparing results of increasing camera velocities and analyzing its performance using QVGA at 30 Hz against QQVGA at 60 Hz. Fourthly, qualitative results will be shown in the form of 3-D maps built purely from odometry pose estimations.

For the first three groups of experiments we have utilized some data series from the TUM datasets [49], given that they provide the ground truth to calculate the estimate error. The selected datasets reproduce varied camera motions at different speeds and include scenes that contain sufficient geometric and photometric information (the latter is necessary for RDVO). The datasets chosen, according to the authors of [49], belong to 3 different categories:

1. **Handheld SLAM**: It includes general camera motions in office-like or house-like environments. Within this category, we use "Freiburg1/desk" (*f1-desk*), "Freiburg1/desk2" (*f1-desk2*), "Freiburg1/room" (*f1-room*) and "Freiburg2/desk" (*f2-desk*).

2. **3D Object Reconstruction**: It includes trajectories around certain objects. Within this category we use "Freiburg1/teddy" (*f1-teddy*) and "Freiburg1/plant" (*f1-plant*).

Sequence	Duration (s)	Avg. trans. vel (m/s)	Avg. rot. vel (deg/s)	Est. traj. length (m)
f1-desk	23.4	0.413	23.33	9.66
f1-desk2	24.86	0.426	29.31	10.59
f1-teddy	50.82	0.315	21.32	16.01
f1-plant	41.53	0.365	27.89	15.16
f1-room	48.9	0.334	29.88	16.33
f2-desk	99.36	0.193	6.34	19.18
f2-deskwp	142.08	0.121	5.34	17.19

Table 3.1: Dataset characteristics.

3. **Dynamic objects**: In these scenes there are one or more moving objects and, hence, the rigid scene hypothesis is not fulfilled. Within this category we use "Freiburg2/desk with person" (*f2-deskwp*).

Relative and absolute pose errors will be considered, as described in [49], to study the accuracy of our approach. For the relative pose error, both translational and rotational deviations per second will be evaluated with the root mean squared error (RMSE). Besides, we will evaluate the RMSE and the maximum values of the absolute trajectory errors (translational) to analyze the robustness of these approaches to reproduce 3D trajectories faithfully. A brief summary of the datasets considered is presented in Table I. Because of the lack of ground truth at some trajectory stretches, the trajectory length is estimated using the datasets duration and their average translational speed. For all the experiments, the test platform used is a standard desktop PC running Ubuntu 12.04 with an Intel Core i7-3820 CPU at 3.6 GHz.

DIFODO with different resolutions

Several tests have been carried out to analyze how the accuracy and speed of DIFODO are affected by the maximum resolution of the depth images employed in the coarse-to-fine scheme. A total amount of 28 experiments have been performed, 4 for each dataset, with maximum resolutions of 30×40, 60×80, 120×160 and 240×320. In every case, the coarsest level of the pyramid had a resolution of 15×20, implying that the number of levels in the aforementioned experiments were 2, 3, 4 and 5, respectively. No results are presented with a resolution of 15×20 because it is a too low resolution to produce decent estimates. Initially, relative translational and rotational errors are compared (Figure 3.2). As expected, accuracy increases with higher resolutions. It can be noticed that the RMSE of each dataset varies as accuracy does not only depend on the VO method but also on the camera speed and the geometry of the observed scenes. Likewise, in Figure 3.2 the RMSE and maximum absolute translational errors are plotted. It can be observed that, on average, the longest datasets present the highest absolute error since the errors of every iteration are propagated over a longer period of time. It is worth mentioning that the highest

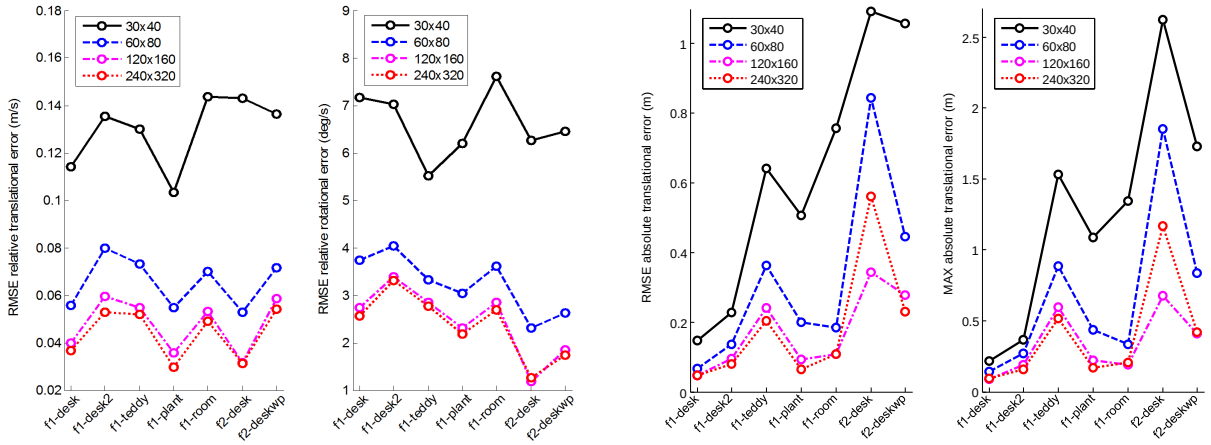


Figure 3.2: **Left**: Comparison of translational and rotational drifts per second with DIFODO at different resolutions. **Right**: Comparison of accumulated translational errors with DIFODO at different resolutions.

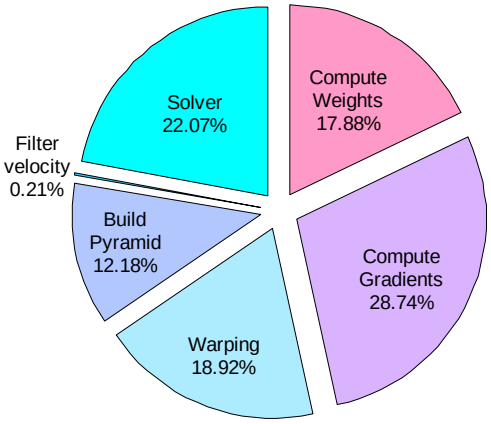


Figure 3.3: Percentage computational load of the main steps that DIFODO performs to estimate motion with QVGA resolution.

resolution (QVGA) does not provide significantly improved estimates respect to those of the previous level (QQVGA). Although it is logical that the solution refinement becomes less and less noticeable with higher resolutions, the main factor that explains this result is the fact that images at QVGA resolution are not filtered because they represent the finest level of the pyramid. As a consequence, the noise of the measurements affects the derivatives calculation more drastically than at any other level. At the finest level, points are closer to each other, and the differences between their depth values are caused not only by the geometry of the scene but also by the noise of the measurements. Hence, the depth gradient approximation becomes imprecise and does not perfectly reflect the real gradients of the observed surfaces. Overall, Figure 3.2 shows that it is beneficial to set QVGA as the maximum resolution although there might be some exceptions like the sequence *f2-desk*. If necessary, this effect could be partially alleviated by applying a previous bilateral filter to the finest level, at the expense of a higher computational cost.

Runtimes are reported in Table 3.2. As DIFODO provides a closed-form solution, its runtime is essentially the same for all the datasets and the exact values might slightly differ depending on how many pixels present null measurements. In Figure 3.3 we show how the runtime of DIFODO at QVGA is distributed among the main steps that comprise it (see Algorithm 1). The velocity filter is significantly faster than any other step because it is the only one which

Sequence	Resolution			
	30×40	60×80	120×160	240×320
f1-desk	3.79	5.12	10.09	28.51
f1-desk2	3.78	5.13	10.03	28.30
f1-teddy	3.94	5.30	10.18	28.99
f1-plant	3.91	5.27	10.17	28.57
f1-room	3.87	5.12	9.92	28.58
f2-desk	3.85	5.23	10.14	29.19
f2-deskwp	3.78	5.09	9.73	28.14
Average	3.85	5.18	10.04	28.61

Table 3.2: Runtime of DIFODO (ms).

does not perform pixel-wise operations. Besides, one aspect should be remarked. In all cases the Gaussian pyramid was built starting from a resolution of 240×320, which on average takes 3.5 milliseconds. However, if DIFODO is configured to work with any inferior resolution, the Gaussian pyramid can be built from 120×160 upwards, which takes less than 1 millisecond. As a consequence, DIFODO runtime can be actually lower than it is shown in Table 3.2 for any resolution inferior to QVGA. This aspect makes QQVGA resolution particularly appropriate for fast robotic applications because it can provide quite accurate results (see Figure 3.2) with a very low runtime.

Comparison between DIFODO, GICP and RDVO

In this subsection we compare our approach with two state-of-the-art methods: GICP and RDVO. For every method we will consider QVGA resolution, although results of DIFODO with QQVGA will be also included to analyze how its performance deteriorates if a faster version of it is needed. To refer to them we will use the names $\text{DIFODO}_{\text{HR}}$ and $\text{DIFODO}_{\text{LR}}$ which stand for high and low resolution, respectively. The results for RDVO have been generated with the code that the authors published online for ROS [57]. Regarding GICP, we make use of the implementation included in PCL [58]. For a fair comparison, a bilateral filter with a 5×5 mask is applied to the depth images before running GICP, which smoothes the raw data and leads to better results. The GICP parameters have been chosen as follows:

- The maximum correspondence distance is set to 0.5 m to cover all the translations and rotations of the datasets where the scenes are sampled at 30 Hz and the maximum observed distance is about 5 meters.
- The maximum number of iterations is set to 10 although we observed in the experiments that the method almost always converges at the 3th – 4th iteration.
- Transformation epsilon is set to 10^{-5} which marks the minimum difference between consecutive transformations to assume that the algorithm has converged. It is actually applied separately to translations and rotations (with this same value).

First, Table 3.3 presents the execution time statistics for each algorithm and dataset. From the results in Table 3.3 we can conclude that our proposal is almost 30 times faster than GICP and

Sequence	Average runtime (ms)			
	$\text{DIFODO}_{\text{LR}}$	$\text{DIFODO}_{\text{HR}}$	RDVO	GICP
f1-desk	10.09	28.51	28.96	838.2
f1-desk2	10.03	28.3	27.00	877.5
f1-teddy	10.18	28.99	27.66	862.4
f1-plant	10.17	28.57	41.82	824.1
f1-room	9.92	28.58	27.39	856.7
f2-desk	10.14	29.19	42.85	769.8
f2-deskwp	9.73	28.14	32.43	677.7
Average	10.04	28.61	32.43	815.2

Table 3.3: Runtime comparative.

Sequence	RMSE Translational deviations (cm/s)				RMSE Rotational deviations (deg/s)			
	$\text{DIFODO}_{\text{LR}}$	$\text{DIFODO}_{\text{HR}}$	RDVO	GICP	$\text{DIFODO}_{\text{LR}}$	$\text{DIFODO}_{\text{HR}}$	RDVO	GICP
f1-desk	3.98	**3.66**	4.08	10.17	2.73	2.56	**2.18**	6.88
f1-desk2	5.96	**5.28**	6.45	11.66	3.39	**3.31**	3.55	6.66
f1-teddy	5.47	**5.18**	9.67	12.73	2.85	2.77	**2.46**	4.71
f1-plant	3.56	**2.98**	3.41	11.17	2.31	2.18	**1.25**	4.78
f1-room	5.31	**4.89**	6.22	9.93	2.85	2.69	**2.62**	4.37
f2-desk	3.17	3.13	**2.39**	7.68	1.18	1.26	**1.02**	2.98
f2-deskwp	5.85	5.42	**3.12**	7.19	1.85	1.74	**0.90**	2.89
Average	4.76	**4.36**	5.05	10.08	2.45	2.36	**1.99**	4.75

Table 3.4: Relative Errors: translational and rotational deviations per second.

Sequence	RMSE (cm)				Maximum (cm)			
	$\text{DIFODO}_{\text{LR}}$	$\text{DIFODO}_{\text{HR}}$	RDVO	GICP	$\text{DIFODO}_{\text{LR}}$	$\text{DIFODO}_{\text{HR}}$	RDVO	GICP
f1-desk	**4.66**	4.76	6.36	18.29	**8.77**	9.4	11.9	36.38
f1-desk2	9.37	**7.93**	8.46	22.75	18.73	**15.54**	42.7	58.08
f1-teddy	24.12	**20.41**	27.1	28.49	59.83	51.47	**48.23**	55.98
f1-plant	9.17	6.45	**4.56**	24.44	22.03	16.67	**9.62**	53.92
f1-room	10.94	**10.88**	33.19	33.86	**19.18**	20.37	56.7	62.38
f2-desk	**34.24**	56.02	34.51	134.3	67.6	116.6	**61.4**	284.8
f2-deskwp	27.74	**22.89**	25.9	67.88	**41.05**	42.23	50.01	151.6
Average	**17.18**	18.48	20.01	47.14	**33.88**	38.90	40.08	100.4

Table 3.5: Absolute translational errors.

as fast as RDVO although our runtime is more deterministic since DIFODO is not iterative. On the other hand, we also compare how precise these methods are in estimating the camera motion. Relative pose errors in the form of translational and rotational deviations per second are analyzed and shown in Table 3.4. It can be noticed that DIFODO, with both QVGA and QQVGA resolutions, is the most accurate method to estimate translations whereas RDVO provides the best results for rotations. GIPC, on the other hand, is always considerably less accurate than the other two approaches. Moreover, absolute trajectory errors are presented in Table 3.5, where it can be seen that DIFODO is the method that estimates the whole trajectories more faithfully. Curiously, $\text{DIFODO}_{\text{LR}}$ provides, on average, the best estimated trajectories although $\text{DIFODO}_{\text{HR}}$ performs better locally. This apparent contradiction is caused by one single sequence: *f2-desk*, where $\text{DIFODO}_{\text{HR}}$ produces a particularly bad overall result.

A special case is the sequence *f2-deskwp* because it is the only one that purposely incorporates moving objects, breaking the rigid scene assumption on which DIFODO relies. RDVO includes in its formulation a weighting function to mitigate (but not eliminate) errors derived from moving objects and, hence, presents a considerable smaller relative RMSE than DIFODO for this dataset. Nevertheless, but for the stretches where the moving person appears, DIFODO performs better than RDVO and, surprisingly, is able to produce the best overall trajectory estimate even for this dataset where RDVO was expected to significantly outperform any other method.

Sequence	Translational RMSE (cm/s)			Rotational RMSE (deg/s)		
	Original	× 2	× 3	Original	× 2	× 3
f1-desk	**3.7**	5.0	8.9	**2.56**	4.28	6.26
f1-desk2	**5.3**	8.9	17.7	**3.31**	3.59	4.00
f1-teddy	**5.2**	7.7	10.9	**2.77**	4.41	7.15
f1-plant	**3.0**	**3.0**	3.5	2.18	**2.13**	2.41
f1-room	**4.9**	**4.9**	9.0	**2.69**	**2.69**	3.77
f2-desk	3.1	2.5	**2.4**	1.26	0.96	**0.88**
f2-deskwp	5.4	5.1	**4.9**	1.74	1.57	**1.47**
Average	**4.4**	5.3	8.2	**2.36**	2.80	3.71

Table 3.6: Relative pose errors with the original and the time-decimated sequences.

Sequence	RMSE (cm)			Maximum (cm)		
	Original	× 2	× 3	Original	× 2	× 3
f1-desk	**4.8**	17.1	23.6	**9.4**	41.8	46.4
f1-desk2	**7.9**	13.1	27.1	**15.5**	30.5	64.7
f1-teddy	**20.4**	25.6	35.0	**51.5**	56.4	61.3
f1-plant	**6.5**	**6.5**	9.0	16.7	**15.1**	21.6
f1-room	**10.9**	18.2	38.8	**20.4**	35.8	76.2
f2-desk	56.0	35.2	**26.2**	116.6	70.4	**46.8**
f2-deskwp	22.9	18.9	**17.8**	42.2	34.6	**32.8**
Average	**18.5**	19.2	25.4	**38.9**	40.6	50.0

Table 3.7: Absolute trajectory errors with the original and the time-decimated sequences.

Estimation of fast motions

In this subsection, we study how the performance of our approach varies when the camera velocity increases. To this end, we simulate faster motions by time-decimating the input data from the TUM dataset. In this set of experiments, which have been carried out with QVGA resolution, the input image sequences are decimated by a factor of 2 and 3, representing velocities that are 2 times and 3 times faster than the original ones. Relative and absolute pose errors are shown in Table 3.6 and Table 3.7 respectively. It can be noticed that the accuracy of our method decreases when a faster motion is simulated on the "f1 dataset", whereas the opposite occurs for the "f2 dataset". To understand these disparate results we have to take into account that the average camera velocities at each dataset are quite different. In particular, the average camera velocities at the "f2 dataset" are about 3 times slower than those at the "f1 dataset", which explains why the decimated sequences of the "f2 dataset" provide good results. On the contrary, the worst estimates are found for the accelerated sequences of *f1-desk* and *f1-desk2*, which present the fastest camera motions. It is worth noting that accuracy improves for the "f2 dataset" when the sequences are decimated or accelerated, effect that can be explained by analyzing the measurement noise of Kinect-like cameras. The depth images provided by this kind of cameras present a flickering or trembling over time that grows quadratically with depth [14]. This implies that no scene looks perfectly static to the camera even if the camera is still. For this reason, every new execution of the algorithm introduces some error in the trajectory estimation which, in the case of small motions, can be partially prevented with a lower sampling rate. This is essentially what causes the improved performance of the decimated "f2-sequences".

In view of these results, we can conclude that DIFODO with QVGA provides very accurate estimates for camera velocities up to 0.7 m/s. If a faster motion needs to be estimated, QQVGA should be chosen to work at 60 Hz which would double the velocity estimation range without compromising the accuracy significantly (see comparison in §3.E.7). For very slow motions, it would be advantageous to reduce the frequency of the estimates, according to the results of the "f2 dataset".

Real-time operation and map building

Last, we have performed real experiments with a handheld camera to demonstrate the accuracy of our approach to estimate geometrically consistent trajectories. To this purpose, DIFODO has been utilized to build 3-D maps of two different scenarios (a living room and our lab). These maps are built as a concatenation of point clouds along the trajectory that DIFODO estimates, without resorting to global consistency or any other mapping strategy. The images are taken with a PrimeSense Carmine 1.08 at 30 Hz with QVGA resolution. A brief summary of the estimated trajectories is presented in Table 3.8. The maximum velocity values have been obtained after applying a median filter to the sequence of velocities to reject possible outliers. Figure 3.4 and Figure 3.5 depict the generated maps from different views. In both cases the real geometry of the mapped environments is preserved quite accurately: flat surfaces remain flat in the map (see the floor in Figure 3.4), walls remain perpendicular to each other, etc. However, the scene colors are not consistent because the shutter speed of the camera was automatic and, therefore, the object colors vary depending on the average brightness of the scene. In any case, we do not aim to address the map building problem but to show that DIFODO is able to provide very accurate estimates not only locally but for full trajectories. Color has simply been added to the maps to enhance their appearance but, as has been repeatedly said throughout the paper, it is not employed to estimate motion.

	Lab	Living room
Duration (s)	33.66	20.31
Length (m)	18.65	8.611
Aver. trans. velocity (m/s)	0.554	0.424
Aver. rot. velocity (deg/s)	16.91	20.55
Max. trans. velocity (m/s)	0.841	0.770
Max. rot. velocity (deg/s)	47.95	49.32

Table 3.8: Real trajectories estimated with DIFODO.

3.E.8 Conclusions and Future Work

In this paper we have introduced a novel visual odometry algorithm based on geometric data, and have detailed its formulation to work with range cameras. Within a coarse-to-fine scheme, the camera motion is estimated by assuming rigid motion of the scene with respect to the camera and finding the rigid body velocity that best describes the velocity of all the points of the scene. A velocity filter is proposed to detect and mitigate wrong estimates under cases of

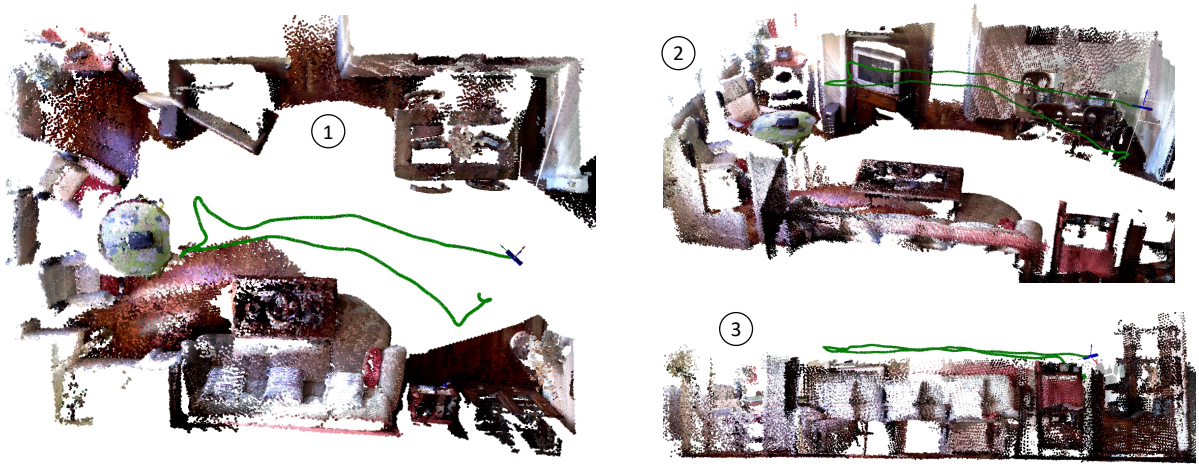


Figure 3.4: Three-dimensional map of a living room generated with an RGB-D camera from DIFODO motion estimates. The map is shown in top (1), perspective (2) and front (3) views. The camera trajectory is depicted with a green line.

geometric uncertainty, and a detailed study on how to accurately approximate the depth gradients is presented.

In terms of speed and precision, our approach has been compared with GICP [2] and RDVO [3]. With respect to GICP, which also estimates motion from geometry, our method is about 30 times faster and more than 2 times more accurate. Regarding RDVO, which needs geometric and photometric data to work, similar results (or even slightly better) are obtained from purely geometric inputs. Maps of 2 different scenarios have been built from real-time odometry estimates to demonstrate qualitatively that DIFODO is able to reproduce full trajectories consistently. Furthermore, DIFODO has proved to provide accurate motion estimates with low image resolutions (120×160) which makes it suitable for real-time robotic applications that might involve fast motions and demand a higher frame rate (60 Hz).

On the other hand, there are some factors that limit the performance of our approach. Firstly, the currently available range cameras are not very precise and generate depth images that considerably deform the real geometry of the scenes they observe. This defect will surely be alleviated at the upcoming new generations of range sensors given the attention that big companies like Microsoft, Apple or Samsung are paying to 3D sensing. Secondly, its accuracy worsens when the rigid scene hypothesis is not fulfilled. We believe that the real solution to this problem will be given by scene flow algorithms that estimate the dense 3-D motion field of the scene points. Particular solutions (like that of [3]) can help to deal with moving objects if a very high percentage of the scene is still rigid, but would fail to estimate motion when the number of moving objects increases or the moving object itself represents a substantial part of the whole scene. Therefore, a more general solution that could be generalized to any arbitrary scene composed of any arbitrary number of moving objects should be found. Last, similarly to GICP or other geometry-based VO methods, DIFODO is unable to estimate some linear or angular velocity components when the scene does not present sufficient geometric-distinctive features. Although the proposed velocity filter helps to mitigate this limitation, a more robust

Figure 3.5: Three-dimensional map of our lab generated with an RGB-D camera from DIFODO motion estimates. An overall top view is shown (left) together with some local views: a detailed top view (1) and two perspective views (2 and 3). The camera trajectory is depicted with a green continuous line.

solution should incorporate RGB or inertial information to effectively tackle the problem of ill-posed configurations.

3.E.9 Appendix: Covariance Computation

This appendix derives the expression of the matrix Σ_d which, according to (3.29), is composed of the following blocks:

$$\Sigma_{xyz} = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{xz} \\ \sigma_{xy} & \sigma_y^2 & \sigma_{yz} \\ \sigma_{xz} & \sigma_{yz} & \sigma_z^2 \end{pmatrix}, \Sigma_{Z_{t,u,v}} = \begin{pmatrix} \sigma_{Z_t}^2 & \sigma_{Z_t Z_u} & \sigma_{Z_t Z_v} \\ \sigma_{Z_t Z_u} & \sigma_{Z_u}^2 & \sigma_{Z_u Z_v} \\ \sigma_{Z_t Z_v} & \sigma_{Z_u Z_v} & \sigma_{Z_v}^2 \end{pmatrix},$$
$$\Sigma_{(xyz)Z_{t,u,v}} = \begin{pmatrix} \sigma_{xZ_t} & \sigma_{xZ_u} & \sigma_{xZ_v} \\ \sigma_{yZ_t} & \sigma_{yZ_u} & \sigma_{yZ_v} \\ \sigma_{zZ_t} & \sigma_{zZ_u} & \sigma_{zZ_v} \end{pmatrix}. \tag{3.47}$$

Assuming that the only source of error is the inaccuracy of the depth measurements, we first characterize this error according to the work of Khoshelham and Elberink [14]:

$$\sigma_{zm} = \frac{\sigma_d}{fb} z^2 \sim 1.4 \cdot 10^{-5} z^2 , \tag{3.48}$$

where σ_{zm} is the standard deviation of the depth measured at a given pixel, f is the focal distance of the IR camera, b is the baseline between the IR camera and projector and σ_d is the disparity measurement error. Since the depth images are filtered with a Gaussian mask of 5×5, the actual standard deviation of z should take it into account:

$$\sigma_z = \frac{\sigma_d}{5fb} z^2 = k_z z^2 \sim 2.8 \cdot 10^{-6} z^2 . \tag{3.49}$$

Knowing the pinhole camera model (see (3.19) and (3.20)), it can be deduced that:

$$\sigma_x = \frac{(u-u_m)}{f_x}\sigma_z = k_z xz\,, \tag{3.50}$$

$$\sigma_y = \frac{(v-v_m)}{f_y}\sigma_z = k_z yz\,. \tag{3.51}$$

On the other hand, the standard deviation of the depth derivatives should be computed as the difference of Gaussians but, for simplicity's sake, it is calculated as if both the subtrahend and minuend of the finite difference expressions were equal to the value of z at the corresponding pixel, i.e.

$$\begin{aligned} z^{t+1}(u,v) \sim z^t(u\pm 1,v) \sim z^t(u,v\pm 1) \sim z^t(u,v) \Rightarrow \\ \sigma_{Z_t} = \frac{k_z z^2}{\sqrt{s}\Delta t},\ \sigma_{Z_u} = \frac{k_z z^2}{2\sqrt{2}},\ \sigma_{Z_v} = \frac{k_z z^2}{2\sqrt{2}}\,. \end{aligned} \tag{3.52}$$

The covariance terms in (3.47) can be obtained analytically, but only those which are not null will be explicitly derived. The null terms correspond to independent variables (Z_t, Z_u and Z_v respect to each other and x, y and z) and, hence, do not need further consideration.

$$\sigma_{xy} = E\left[(x-E[x])(y-E[y])\right] = \frac{(u-u_m)(v-v_m)}{f_x f_y} E\left[(z-E[z])^2\right] = k_z^2 xyz^2\,, \tag{3.53}$$

$$\sigma_{xz} = E\left[(z-E[z])(x-E[x])\right] = \frac{(u-u_m)}{f_x} E\left[(z-E[z])^2\right] = k_z^2 xz^3\,, \tag{3.54}$$

$$\sigma_{yz} = E\left[(z-E[z])(y-E[y])\right] = \frac{(v-v_m)}{f_y} E\left[(z-E[z])^2\right] = k_z^2 yz^3\,, \tag{3.55}$$

$$\begin{aligned} \sigma_{zZ_t} = \sigma_{zZ_u} = \sigma_{zZ_v} = \sigma_{xZ_t} = \sigma_{xZ_u} = \sigma_{xZ_v} = \sigma_{yZ_t} = \sigma_{yZ_u} = \sigma_{yZ_v} = 0, \\ \sigma_{Z_tZ_u} = \sigma_{Z_tZ_v} = \sigma_{Z_uZ_v} = 0\,. \end{aligned} \tag{3.56}$$

This is all the information required to build the matrix Σ_d. A similar study is presented in [45] but based on different assumptions and applied to a different set of variables. It focuses on the 3-D uncertainty of a point (variables x, y and z above) and, although they also model the pixel coordinates u and v as normal distributions, their results are similar to ours if the variance of u and v are set to zero.

3.F Planar Odometry from a Radial Laser Scanner. A Range Flow-based Approach

Mariano Jaimez, Javier G. Monroy and Javier Gonzalez-Jimenez

Published in Proc. International Conference on Robotics and Automation (ICRA), 2016.

Planar Odometry from a Radial Laser Scanner. A Range Flow-based Approach

Mariano Jaimez, Javier G. Monroy and Javier Gonzalez-Jimenez

Abstract

In this paper we present a fast and precise method to estimate the planar motion of a lidar from consecutive range scans. For every scanned point we formulate the range flow constraint equation in terms of the sensor velocity, and minimize a robust function of the resulting geometric constraints to obtain the motion estimate. Unlike traditional approaches, this method does not search for correspondences but performs dense scan alignment based on the scan gradients, in the fashion of dense 3D visual odometry. The minimization problem is solved in a coarse-to-fine scheme to cope with large displacements, and a smooth filter based on the covariance of the estimate is employed to handle uncertainty in unconstraint scenarios (e.g. corridors). Simulated and real experiments have been performed to compare our approach with two prominent scan matchers and with wheel odometry. Quantitative and qualitative results demonstrate the superior performance of our approach which, along with its very low computational cost (0.9 milliseconds on a single CPU core), makes it suitable for those robotic applications that require planar odometry. For this purpose, we also provide the code so that the robotics community can benefit from it.

3.F.1 Introduction

Odometry is an essential component for robot localization. It is commonly solved through three major techniques that are based on inertial devices, wheel encoders or visual odometry (either by feature tracking or by dense image alignment). Inertial measurement units (IMUs) are ideal to estimate spatial orientation but accumulate too much translational error over time due to their inability to cancel the gravitational component of the measurement [18]. Odometry based on enconders has extensively been used to provide fast motion estimates for wheeled or legged robots, though this approach is prone to being inaccurate due to wheel/leg slippage and the impreciseness of the kinematic robot models. Lastly, vision-based methods are arguably the most flexible and powerful solution to the motion estimation problem because they can be adapted to work with different types of robots (wheeled, legged, aerial) and configurations (2D-3D motion).

Our proposal here relies on laser scans and has the advantage over the aforementioned methods to be independent of the vehicle's type of locomotion as well as very fast and precise, as supported by experimental validation. Thus, it turns out to be particularly suitable for those (very common) cases where the robot already uses a laser range scanner for mapping, obstacle avoidance or localization. Our approach, named RF2O (Range Flow-based 2D Odometry), builds upon [30] and represents the apparent motion of any point observed by the sensor as a function of the velocity of the sensor, assuming that the environment is static. Thus, every point defines a geometric residual which can be minimized within a dense formulation to obtain the lidar motion. To overcome the assumption of a motionless environment (i.e. to handle moving objects), we compute the Cauchy M-estimator of the geometric residuals, a more robust estimate than traditional choices like the L_2 or L_1 norms. Furthermore, we solve this estimation problem within a coarse-to-fine scheme, which provides finer results and allows the method to handle larger motions.

We have conducted a set of experiments to compare our method against point-to-line iterative closest point (PL-ICP) [5] and the polar scan matching approach (PSM) [6]. Firstly, their performances are evaluated at different scan rates on simulated scenarios where the ground truth is available. Secondly, qualitative results are shown for a real experiment where 2D maps are built by concatenating the scanned points according to the odometry motion estimates of each method. Thirdly, we devise a real experiment to evaluate how robust the methods are against the presence of noise and moving objects. Overall, results show that RF2O is significantly more precise for both translations and rotations, and presents the lowest runtime (2 times faster than PSM and 20 times faster than PL-ICP). Besides analyzing the results presented herein, we encourage the reader to watch the demonstration video which, together with the available code, can be found at:

`http://mapir.isa.uma.es/work/rf2o`

3.F.2 Related Work

Although low-cost RGB-D cameras have recently favoured the transition to 3D odometry, localization and mapping strategies, it is a matter of fact that a fair number of mobile robots move on planar surfaces and rely on laser scanners to navigate. In this context, very successful results have been achieved in the fields of 2D Localization [59, 60] and SLAM [61], and many algorithms have been proposed to solve the general scan matching problem [62, 5, 39]. In this paper we focus on pure 2D odometry, which can be regarded as a particular case of scan matching where the scans to be aligned are taken consecutively and are normally close to each other.

Traditionally, ICP [39] or some of its variants have been applied to solve the registration problem between consecutive scans. A very successful approach was proposed by Censi [5], where a point-to-line metric is used instead of the point-to-point original metric of ICP. Futhermore, the author presented an implementation which ran one order of magnitude faster than existing ICP variants, and was more precise and efficient than the pioneer point-to-segment work in [4]. More recently, Generalized-ICP [2] showed an improved performance over previous ICP versions, but has been mostly used for the registration of 3D point clouds. For this family of methods, accuracy depends on every particular version and implementation, yet they all share the same weakness: they are computationally expensive.

Alternatively, other methods were specifically designed to solve the 2D scan matching problem. Gonzalez and Gutierrez [30] formulated the *velocity constraint equation*, an adaptation of the optical flow constraint for range scans, and proposed to estimate the lidar motion by imposing this restriction for every point observed in the scans. However, their method was only tested with simple simulated scenarios and provided modest results. Diosi and Kleeman presented the Polar Scan Matching approach [62], where the translation and rotation between two scans are alternately estimated until convergence. In contrast to ICP, this method avoids searching for correspondences by simply matching points with the same bearing, which leads to a higher computational performance. This approach was subsequently extended and further evaluated in [6]. A different method proposed by Olson [42] tries to find the rigid transformation that maximizes the probability of having observed the latest scan given the previous one. Additional

information is used (control inputs or wheel odometry) to ease the method convergence and two different implementations, GPU and multi-resolution CPU, are presented. A thorough evaluation is performed in terms of computational performance but, surprisingly, no results for the method's accuracy are presented.

More recently, other approaches have built upon the aforementioned works. It is the case of [63] and [64], which fuse laser odometry (the Olson's laser odometry [42] and point-to-line ICP [5], respectively) with stereo vision to perform autonomous navigation with UAVs. Furthermore, the work of Pomerleau *et al.* [65] presents a fast implementation and a through evaluation of some ICP variants on real-world 2D and 3D data sets.

3.F.3 Velocity of the Lidar and 2D Range Flow

In this section we describe how the 2D velocity of a laser scanner can be estimated from the apparent motion that it observes, assuming that the environment is static and rigid. Let $R(t, \alpha)$ be a range scan where t is the time and $\alpha \in [0, N) \subset \mathbb{R}$ is the scan coordinate, being N the size of the scan. The position of any point P with respect to the local reference frame attached to the sensor is given by its polar coordinates (r, θ) (see Figure 3.6). Provided that P is visible from the lidar, it will be observed at a scan coordinate α that is directly related to the angular coordinate of P:

$$\alpha = \frac{N-1}{\text{FOV}}\,\theta + \frac{N-1}{2} = k_\alpha \theta + \frac{N-1}{2}\,, \tag{3.57}$$

where FOV is the field of view of the scanner. Similarly to the optical flow constraint equation, a linear constraint can be derived from the general expression of geometric consistency of two scan pairs. Assuming the differentiability of R, the range of any point in the second scan can be expressed as the Taylor expansion

$$R(t + \Delta t, \alpha + \Delta\alpha) = R(t, \alpha) + \frac{\partial R}{\partial t}(t, \alpha)\,\Delta t + \frac{\partial R}{\partial \alpha}(t, \alpha)\,\Delta\alpha + O(\Delta t^2, \Delta\alpha^2)\,, \tag{3.58}$$

where Δt is the time lapse between consecutive scans and $\Delta\alpha$ represents the change in the scan coordinate of the point considered. Neglecting second and higher order terms, and dividing by Δt, we can obtain a simple expression that relates the scan gradients with the change of the range and the scan coordinate of a point during the interval $[t, t + \Delta t]$:

$$\frac{\Delta R}{\Delta t} \simeq R_t + R_\alpha \frac{\Delta\alpha}{\Delta t}\,, \tag{3.59}$$

with

$$\Delta R = R(t + \Delta t, \alpha + \Delta\alpha) - R(t, \alpha),$$
$$R_t = \frac{\partial R}{\partial t}(t, \alpha), \quad R_\alpha = \frac{\partial R}{\partial \alpha}(t, \alpha).$$

If we consider that $\dot{r} = \Delta R/\Delta t$ and $\dot{\alpha} = \Delta\alpha/\Delta t$ are the average velocities of a point in range and scan coordinates during the interval $[t, t + \Delta t]$, we obtain:

$$\dot{r} \simeq R_t + R_\alpha \dot{\alpha} = R_t + R_\alpha k_\alpha\, \dot{\theta}\,. \tag{3.60}$$

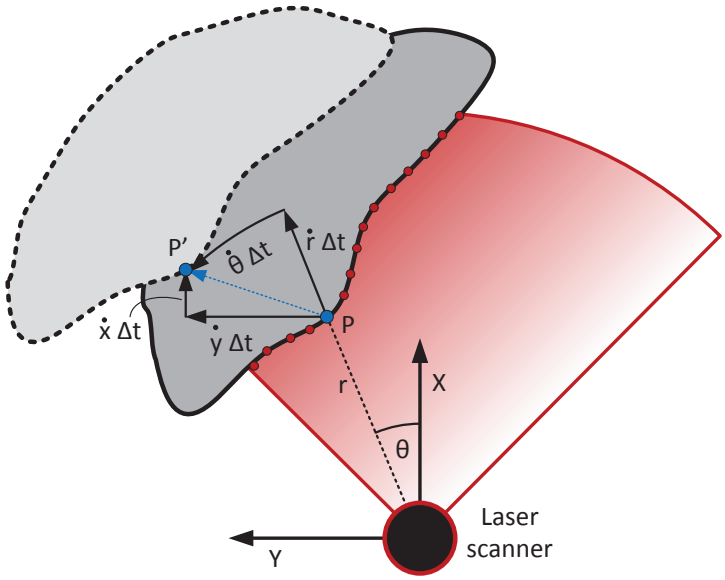


Figure 3.6: Top view representation of a laser scan that intersects with a certain object. An observed point P moves with respect to the scanner to P' after an interval of time Δt.

Equation (3.60) was firstly introduced by Gonzalez and Gutierrez [30] and subsequently generalized and named as the *range flow constraint equation* in [29].

In order to describe the velocities of all points with respect to the same vector basis, we transform the radial and azimuthal velocities $(\dot{r}, \dot{\theta})$ to a cartesian representation $(\dot{x}, \dot{y})$, as shown in Figure 3.6:

$$\dot{r} = \dot{x}\cos\theta + \dot{y}\sin\theta\,, \tag{3.61}$$

$$r\,\dot{\theta} = \dot{y}\cos\theta - \dot{x}\sin\theta\,. \tag{3.62}$$

As a last step, we need to impose that every apparent motion is actually caused by the translation and rotation of the lidar. In other words, we assume that every point moves with respect to the sensor as if it was part of a rigid body whose velocity is the same but opposite in sign to that of the sensor:

$$\begin{pmatrix}\dot{x}\\ \dot{y}\end{pmatrix} = \begin{pmatrix}-v_{x,s} + y\,\omega_s\\ -v_{y,s} - x\,\omega_s\end{pmatrix}\,, \tag{3.63}$$

being $\boldsymbol{\xi}_s = (v_{x,s}\,, v_{y,s}\,, \omega_s)$ a 2D twist (sensor velocity) and (x, y) the cartesian coordinates of P. If the cartesian velocities (3.61) (3.62) are substituted in (3.60) and the rigidity hypothesis (3.63) is imposed, we can transform the range flow constraint equation into a constraint for the lidar velocity:

$$\begin{aligned}\left(\cos\theta + \frac{R_\alpha k_\alpha \sin\theta}{r}\right) v_{x,s} + \left(\sin\theta - \frac{R_\alpha k_\alpha \cos\theta}{r}\right) v_{y,s}\\ + \left(x\sin\theta - y\cos\theta - R_\alpha k_\alpha\right)\omega_s + R_t = 0\,.\end{aligned} \tag{3.64}$$

As a result, every scanned point imposes a restriction to the motion of the sensor and, therefore, three linearly independent restrictions would theoretically suffice to estimate it.

3.F.4 Velocity Estimation

In practice, the motion of the lidar cannot be estimated with only three independent restrictions because, in general, (3.64) is inexact due to the noise of the range measurements, the errors made by the linear approximation (3.59) or the presence of moving object (non-static environment). Therefore, we propose a dense formulation in which all the points of the scan contribute to the motion estimate. For every point, we define the geometric residual $\rho(\boldsymbol{\xi})$ as the evaluation of the range flow constraint (3.64) for a given twist $\boldsymbol{\xi}$:

$$\begin{aligned}\rho(\boldsymbol{\xi}) = R_t + (x\sin\theta - y\cos\theta - R_\alpha k_\alpha)\,\omega \\ + \left(\cos\theta + \frac{R_\alpha k_\alpha \sin\theta}{r}\right) v_x + \left(\sin\theta - \frac{R_\alpha k_\alpha \cos\theta}{r}\right) v_y\,.\end{aligned} \tag{3.65}$$

To obtain an accurate estimate, the sensor motion is computed by minimizing all the geometric residuals within a robust cost function F:

$$\boldsymbol{\xi}_M = \arg\min_{\boldsymbol{\xi}} \sum_{i=1}^{N} F(\rho_i(\boldsymbol{\xi}))\,, \tag{3.66}$$

$$F(\rho) = \frac{k^2}{2}\, ln\left(1 + \left(\frac{\rho}{k}\right)^2\right)\,. \tag{3.67}$$

The function F is the Cauchy M-estimator, and k is an adjustable parameter. Unlike the more common choices of the L_2 or L_1 norms, this function reduces the relevance of those points with very high residuals, and represents an effective and automatic way to deal with outliers. The optimization problem is solved with Iteratively Reweighted Least Squares (IRLS), where the weights associated to the Cauchy M-estimator are:

$$w(\rho) = \frac{1}{1 + \left(\frac{\rho}{k}\right)^2}\,. \tag{3.68}$$

With IRLS, the system is iteratively solved by recomputing the residuals and the weights until convergence.

Pre-weighting strategy

As previously mentioned, there are some factors that can render (3.64) inaccurate, mainly the unfulfillment of the rigidity hypothesis (3.63) and the deviations from the linear approximation made in (3.59). Although the Cauchy M-estimator can alleviate their effect over the overall motion estimates, it does not eliminate it completely. The presence of moving objects is hard to detect before solving the motion and, therefore, we rely on the Cauchy M-estimator to downweight them during the minimization process. On the other hand, deviations from the linear approximation adopted in (3.59) can be detected beforehand, which helps to accelerate convergence in (3.66) and also leads to more accurate results. To this purpose, we propose a pre-weighting strategy to downweight the residuals of those points where the range function is nonlinear or even non-differentiable. We call it *pre-weighting* because it is applied before the minimization problem

(3.66) is solved. In order to quantify the error associated to the linearization of (3.58), we expand the Taylor series to the second order:

$$\dot{r} = R_t + R_\alpha \dot{\alpha} + R_{2o}(\Delta t, \dot{\alpha}) + O(\Delta t^2, \dot{\alpha}) \,, \tag{3.69}$$

$$R_{2o}(\Delta t, \dot{\alpha}) = \frac{\Delta t}{2} \left(R_{tt} + R_{t\alpha}\dot{\alpha} + R_{\alpha\alpha}\dot{\alpha}^2 \right) \,. \tag{3.70}$$

It can be noticed that, neglecting higher order terms, the second order derivatives in $R_{2o}(\Delta t, \dot{\alpha})$ can be used to detect the deviations from linearity. One special case is the second order derivative with respect to time (R_{tt}), which cannot be computed in a coarse-to-fine scheme because the warped images are timeless and, therefore, the concept of second temporal derivative makes no sense (coarse-to-fine is described in §3.F.5). Moreover, it is important to detect those regions of the scans where the range function is not only nonlinear but non-differentiable. These regions are mainly the edges of the different objects observed, and are typically characterized by very high values of the first order derivatives (R_t and/or R_α). To penalize these two effects, nonlinearities and discontinuities, we define the following pre-weighting function for each scanned point:

$$\bar{w} = \frac{1}{\sqrt{\epsilon + {R_\alpha}^2 + \Delta t^2 {R_t}^2 + K_d \left({R_{\alpha\alpha}}^2 + \Delta t^2 {R_{t\alpha}}^2 \right)}} \,. \tag{3.71}$$

The parameter K_d marks the relative importance of first order and second order derivatives, and ϵ is a small constant to avoid the singular case.

Thus, we initially compute a pre-weighted set of residuals

$$\rho_i^w(\boldsymbol{\xi}) = \bar{w}_i \; \rho_i(\boldsymbol{\xi}) \qquad\qquad \forall i \in \{1, 2...N\} \,, \tag{3.72}$$

which are subsequently minimized according to (3.66) (3.67). Although we do not show comparisons in the paper, this strategy provides better results than standard IRLS minimization without pre-weighting and converges faster (approximately by a factor of 2).

3.F.5 Coarse-to-Fine Scheme and Scan Warping

The linearization presented in (3.59) holds either for small displacements between consecutive scans or at areas with constant range gradients (which, in the case of a lidar, would occur for a very unusual geometry: an Archimedean spiral). To overcome this limitation, we estimate motion in a coarse-to-fine scheme, where the coarser levels provide a rough estimate which is improved subsequently in finer levels. The coarse-to-fine scheme was introduced by Battiti *et al.* [26] to solve the optical flow problem for large displacements, and has commonly been adopted ever since [27, 66].

Let $R_0,\ R_1$ be two consecutive laser scans. Initially, two Gaussian pyramids are to be created by successively downsampling (typically by 2) the original scans $R_0,\ R_1$. Normally, a Gaussian mask is applied to downsample RGB or grayscale images but, in the case of range data, a standard Gaussian filter is not the best choice since it creates artefacts on the filtered scans. As an alternative, we employ a bilateral filter [28] that does not mix distant points which are likely to belong to different objects of the scene. Once the pyramids are built, the velocity estimation

problem is iteratively solved from the coarsest to the finest level. At every transition to a finer level, one of the two input scans must be warped against the other according to the overall velocity estimated in previous levels ($\boldsymbol{\xi}_p$). This warping process is always divided into two steps and, in our formulation, is applied over the second scan R_1. Firstly, every point P observed in R_1 is spatially transformed using the rigid body motion associated to the twist $\boldsymbol{\xi}_p$:

$$\begin{pmatrix} x^w \\ y^w \\ 1 \end{pmatrix} = e^{\hat{\boldsymbol{\xi}}_p} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}, \quad \hat{\boldsymbol{\xi}}_p = \Delta t \begin{pmatrix} 0 & -\omega_p & v_{x,p} \\ \omega_p & 0 & v_{y,p} \\ 0 & 0 & 0 \end{pmatrix}. \tag{3.73}$$

Secondly, the transformed points must be reprojected onto R_1 to build the warped scan R_1^w so that:

$$R_1^w(\alpha^w) = \sqrt{(x^w)^2 + (y^w)^2}, \tag{3.74}$$

$$\alpha^w = k_\alpha \arctan\left(\frac{y^w}{x^w}\right) + \frac{N-1}{2}. \tag{3.75}$$

Several points could be warped to the same coordinate α^w, in which case the closest one is preserved (the others would be occluded). If $\boldsymbol{\xi}_p$ is converging to the real velocity, then the warped scan R_1^w will be considerably closer to the first scan R_0 than the original R_1, which allows us to apply the linear approximation in (3.58) with a finer resolution.

3.F.6 Implementation

Our algorithm pays special attention to the computation of the range gradients. Normally, a fixed discrete formula is employed to approximate either scan or image gradients. In the case of range data, this strategy leads to very high values of the gradients at the object borders, which do not represent the real gradients over those objects. As an alternative, we make use of an adaptive formula that regards the geometry of the environment. This formula weights forward and backward derivatives in the scan with the 2D distances between contiguous observations (points):

$$\begin{aligned} R_\alpha(\alpha) &= \frac{d(\alpha+1)\, R_\alpha^-(\alpha) + d(\alpha)\, R_\alpha^+(\alpha)}{d(\alpha+1) + d(\alpha)}, \\ d(\alpha) &= \|((x(\alpha) - x(\alpha-1), y(\alpha) - y(\alpha-1))\|, \\ R_\alpha^- &= R(\alpha) - R(\alpha-1), \quad R_\alpha^+ = R(\alpha+1) - R(\alpha). \end{aligned} \tag{3.76}$$

Thus, the closest neighbour is always contributing more to the gradient computation while distant points barely affect it. In the case that both neighbours are approximately equidistant, the presented formula is equivalent to a centered finite difference approximation. More details about the gradient computation can be found in [66].

Last, it is important to remark that there are some geometric configurations of the environment from which the sensor motion cannot be recovered. The most common case arises when the lidar only observes a wall. Under this circumstance, the motion parallel to the wall is undetermined and therefore the solver would provide an arbitrary solution for it (not only our method but any

approach based purely on geometry). In order to mitigate this problem, we apply a low-pass filter in the eigenspace of the velocity $\boldsymbol{\xi}$ which works as explained next. First, the eigenvalues of the covariance matrix $\Sigma \in \mathbb{R}^{3\times3}$ of the IRLS solution are analyzed to detect which motion (or combinations of motions) are undetermined and which are perfectly constrained. In the space of the eigenvectors, the velocity $\boldsymbol{\xi}_M^t$ provided by (3.66) is weighted with that of the previous time interval $\boldsymbol{\xi}^{t-1}$ to obtain the new filtered velocity $\boldsymbol{\xi}^t$:

$$[(1+k_l)I + k_e E]\,\boldsymbol{\xi}^t = \boldsymbol{\xi}_M^t + (k_l I + k_e E)\,\boldsymbol{\xi}^{t-1}\,, \tag{3.77}$$

where E is a diagonal matrix containing the eigenvalues and k_l, k_e are parameters of the filter. Concretely, k_l imposes a constant weighting between the solution from the solver and the previous estimate while k_e defines how the eigenvalues affect the final estimate. These parameters are set to the following values:

$$k_l = 0.05\, e^{-(l-1)}, \quad k_e = 15 \cdot 10^3\, e^{-(l-1)}\,, \tag{3.78}$$

where l is the pyramid level that ranges from 1 (coarsest) to the number of levels considered. Please refer to [66] for a more detailed explanation on this filter and how it is applied.

3.F.7 Experiments

This section is composed of a set of three different experiments. The two first experiments address the evaluation of the proposed RF2O algorithm and its comparison with other approaches in simulated and real environments, respectively. The third experiment is carried out to analyze the robustness of the motion estimates against noise and the presence of moving objects. For comparison, two state-of-the-art scan matchers are selected: Point-to-Line ICP (PL-ICP) [5], and Polar Scan Matching (PSM) [6]. In both cases, we use the implementations that their authors published online. For quantitative evaluation, the relative pose errors as described in [49] will

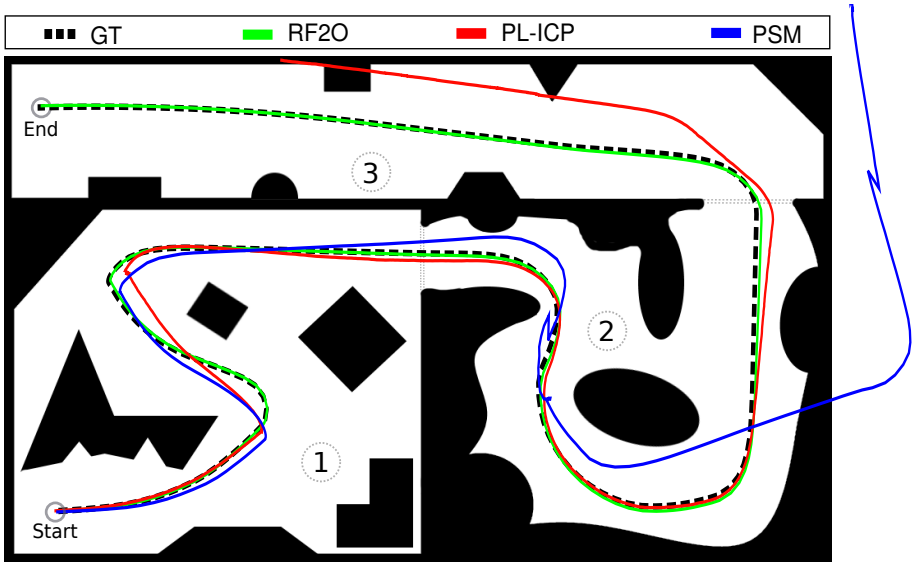


Figure 3.7: Simulated environment and the best path as estimated by each method (RF2O@5Hz, PL-ICP@10Hz, PSM@2Hz). Numbers indicates the different scenarios of the environment.

be considered. Both translational and rotational deviations per second will be evaluated with the root mean squared error (RMSE), which corresponds to a performance measure independent of the experiment duration.

For real experiments, a Hokuyo URG-04LX-UG01 laser scanner mounted on a Giraff mobile robot [20] is used to gather the laser scans at a maximum frequency of 10 Hz. For the case of simulated experiments, the laser characteristics have been imitated (ray number = 682, FOV = 240 °, max distance = 5.5 m). Moreover, a Gaussian noise with σ = 1 cm is added to the simulated scans to make them more realistic.

Comparison in a simulated environment

In this experiment, the compared methods estimate the planar motion of a laser scanner that moves in a simulated environment. We use the precise ground truth available in simulation to perform a quantitative evaluation of the different approaches. The simulated environment is divided into three distinct scenarios (Figure 3.7): a room containing only objects formed by straight line segments (Scen. 1), a room that contains only curved obstacles and curved walls (Scen. 2) and a straight corridor with scattered small objects (Scen. 3). During the experiment, the lidar travels along a predefined path, covering a distance of 43.47 meters at an average speed of 0.398 m/s. Furthermore, four different scan rates (10, 5, 2 and 1 Hz) are tested to analyze the influence of the frequency of execution in the odometry estimates. Table 3.9 depicts the relative pose errors in the form of translational and rotational deviations per second, together with the runtimes of the three compared methods. Figure 3.7 plots the simulated environment with the best estimated trajectory of each method. That is, from all the execution rates, only the one with overall minimum error is plotted for qualitative assessment. As can be noticed, RF2O exceeds the other two approaches for all the scenarios in the experiment, providing much more accurate estimates. PL-ICP presents relatively good estimates for the room scenarios, but it drastically fails at the corridor (specially for translations). On the other hand, PSM presents much higher relative errors in general, and concretely at the second scenario where only curved objects can be found. Furthermore, it presents important problems at narrow places such as doors.

	Scan rate	Translational RMSE (cm/s)			Rotational RMSE (deg/s)			Runtime (ms)		
	(Hz)	RF2O	PSM	PL-ICP	RF2O	PSM	PL-ICP	RF2O	PSM	PL-ICP
Scen. 1	10	**0.425**	14.82	1.860	**0.108**	2.412	0.524	**0.941**	1.837	15.98
	5	**0.308**	7.363	0.759	**0.054**	1.572	0.321	**0.933**	1.979	18.51
	2	**0.248**	3.071	0.584	**0.043**	0.598	0.281	**0.904**	2.205	23.79
	1	**0.273**	12.27	0.396	0.372	2.290	**0.108**	**0.900**	2.675	27.58
Scen. 2	10	**0.398**	19.56	1.904	**0.121**	4.725	0.473	**0.951**	1.994	19.02
	5	**0.346**	18.60	1.084	**0.084**	4.370	0.268	**0.935**	2.642	23.84
	2	**0.785**	18.13	10.14	**0.339**	4.155	3.042	**0.931**	3.351	28.59
	1	**5.250**	42.67	24.07	**3.669**	15.67	7.282	**0.892**	3.656	35.56
Scen. 3	10	**0.461**	4.940	18.44	**0.071**	1.469	0.246	**0.922**	1.826	19.55
	5	**0.382**	5.499	39.74	**0.054**	2.027	0.129	**0.940**	2.296	15.25
	2	**0.249**	7.138	38.48	**0.033**	2.328	0.071	**0.900**	2.911	17.54
	1	**0.439**	33.51	40.19	0.106	3.693	**0.068**	**0.875**	3.677	26.74

Table 3.9: Simulated experiment: translational and rotational deviations per second, and execution times.

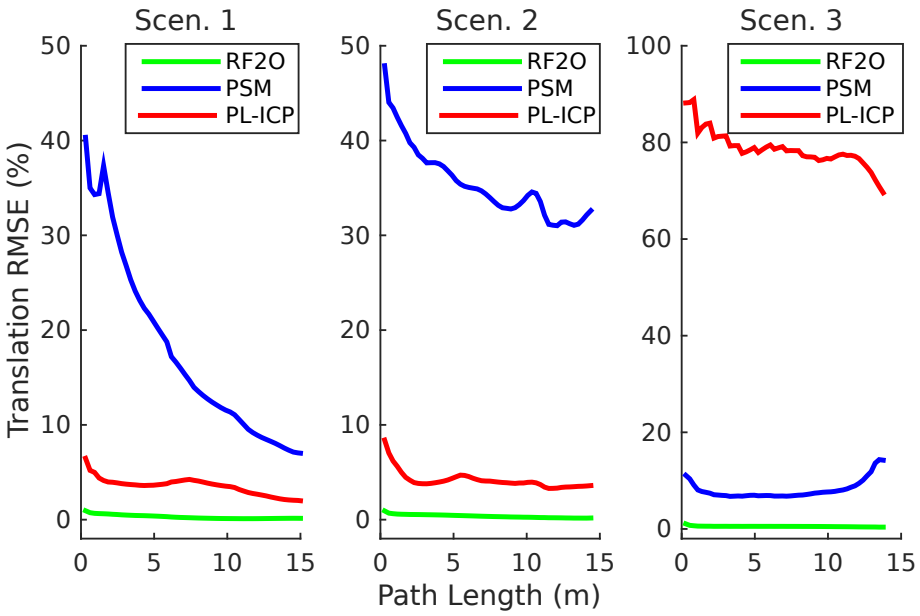


Figure 3.8: Translation errors averaged over all sub-sequences of a given length for the three scenes of the simulated experiment.

It is interesting to notice that the best results are not obtained with the highest frequency. Experiments at 10 Hz provide worse results than those at 5 or 2 Hz, which indicates that data oversampling leads to error accumulation. On the other hand, if the scanning frequency is too little then consecutive scans are too separate and more difficult to align (as occurs at 1 Hz). Thus, the optimal frequency is not always the highest available and depends on the average (or maximum) linear and angular speeds of the lidar.

An alternative and helpful way to compare these methods is to calculate their RMS errors per segment length, as described in [67]. Figure 3.8 depicts these average translational errors as a percentage of the segment length considered (computed independently for the three scenes of the experiment). It can be seen that our approach is in all cases superior to the other two methods, being always under 1% RMSE for all three scenes. PL-ICP is the second best candidate, having around 5% RMSE, except for the long corridor (Scen. 3) where it completely fails.

Finally, from the computational point of view, the last columns on Table 3.9 show the runtimes in milliseconds measured on an AMD Phenom II X6 1035T CPU at 2.6 GHz. Overall, RF2O takes less than 1 ms, followed by PSM with 2.85 ms and PL-ICP with more than 19 ms. Taking this into account, the presented approach not only provides more accurate estimates but it is also much faster than its competitors.

Real experiment

To validate the results obtained in the simulated experiments, we employ a real mobile robot equipped with a Hokuyo laser scanner for navigation in an office-like environment. Making use of a mobile robot allows us to include the odometry estimates from the onboard encoders (a pair of low-cost AMT102-v incremental encoders from CUI Inc.), but prevents us from performing a quantitative comparison given the lack of a precise ground truth. Therefore, in this

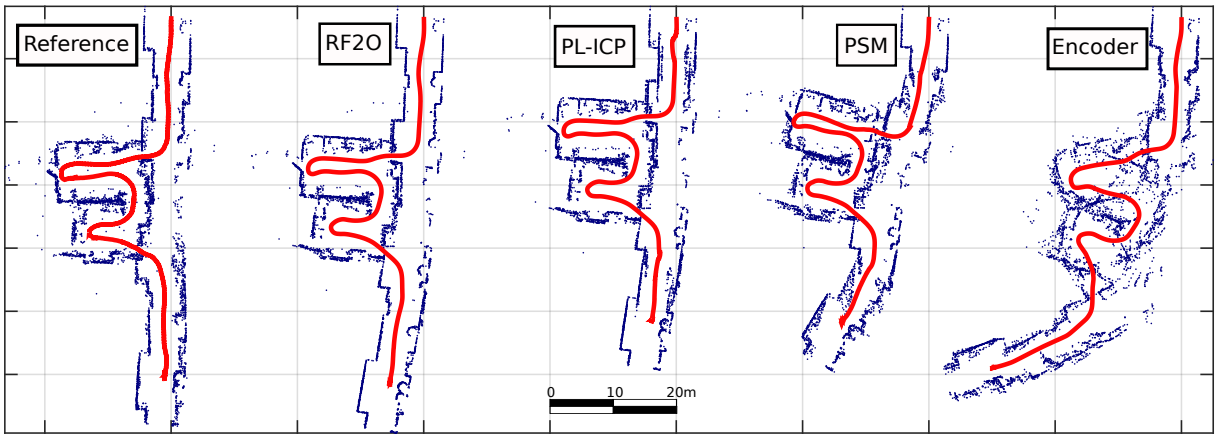


Figure 3.9: Maps built as a concatenation of 2D point clouds along the estimated trajectories for different methods. The reference map is built using the accurate localization of a particle filter-based approach. Trajectories are shown in red and the scanned points in blue.

section the different methods are compared just qualitatively by plotting 2D maps built purely from the odometry pose estimates. In other words, for each method we present maps built as a concatenation of 2D point clouds along their estimated trajectories, without resorting to global consistency or any other mapping strategy.

The path covered by the robot during this experiment is roughly 49 meters long, and is travelled at an average speed of 0.535 m/s (max speed of 0.6 m/s). For all methods, we set the scan rate at 10 Hz. Figure 3.9 shows the maps built from the trajectory estimates of the different approaches. As a reference, we plot the map built from the accurate localization provided by [60], which does not compute odometry but finds the pose of the robot within a previously built map. As can be seen, the map derived from our odometry estimation is noticeably closer to the reference map than any of the others. PL-ICP provides the second best estimation after RF2O, failing mostly in the corridor areas, which results in a shortening of the map and overlapping of scan points in such areas. PSM and the encoder-based maps follow the comparison, being the latter the worst of all of them, with difference.

Robustness against noise and non-static environments

Lastly, we analyze how noise and moving objects affect the motion estimation of the proposed method, i.e., when the assumption of a static environment is violated.

This section is composed of two experiments. The first one aims to evaluate the drift of the compared methods caused by the noise of the measurements. To this end, a real experiment is conducted where a lidar working at 10 Hz is kept still in a static environment for three minutes. Under this setup, since the only error involved is the sensor noise, the outcome represents how noise affects the motion estimates of the different methods. Table 3.10 shows the relative deviations per second of the compared methods. We have also considered a simplified version of our approach (RF2O-NC), where we remove the Cauchy M-estimator and simply minimize the squared residuals (see (3.67)). From these results we can conclude that both RF2O and PL-ICP

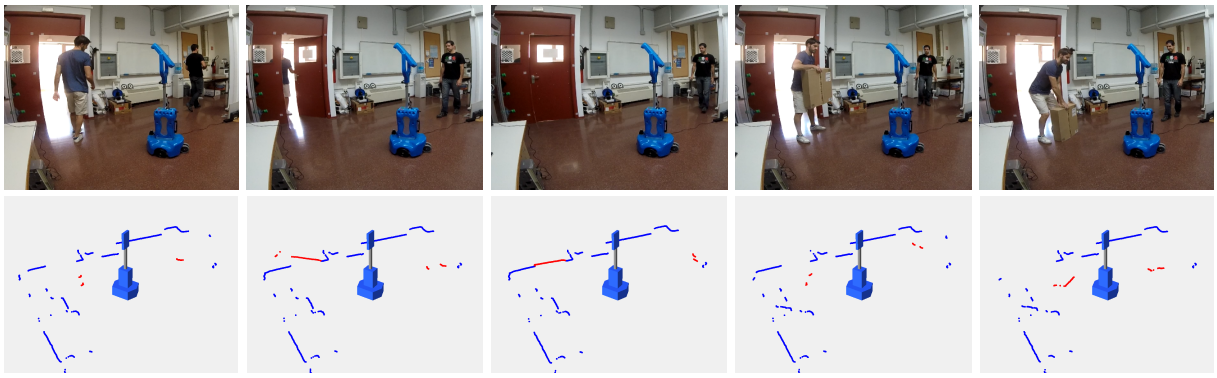


Figure 3.10: **First row**: Sequence of images taken during the second experiment described in §3.F.7. **Second row**: 3D representation of the scans and the robot at those particular instants, where the non-static points are shown in red.

are equally good at translations, being marginally worse than the non-robust RF2O-NC, while, at rotations, PL-ICP is slightly superior than the others.

The second experiment is conducted in the same scenario but, in this case, several moving objects are introduced. During the experiment two persons are walking around the robot, opening and closing a door and displacing a cardboard box (see Figure 3.10). The reader is encouraged to watch the demonstration video where the experiment is shown in detail (`http://mapir.isa.uma.es/work/rf2o`). As can be seen in the second part of Table 3.10, PL-ICP is the most robust method in such situations, followed by the proposed RF2O. It is important to notice that, although PL-ICP is between two and three times better than RF2O, the magnitude of the errors is still pretty small for both methods, unlike the PSM estimates, which show important translational and rotational drifts. Finally, comparing the two versions of our approach, it can be noticed that under the presence of moving objects, the Cauchy M-estimator provides results that are 25% more accurate than those obtained with standard quadratic minimization.

Static Experiment

RMSE	RF2O	RF2O-NC	PSM	PL-ICP
Translation (cm/s)	0.125	**0.113**	0.268	0.125
Rotation (deg/s)	0.075	0.064	0.216	**0.043**

Moving Objects Experiment

RMSE	RF2O	RF2O-NC	PSM	PL-ICP
Translation (cm/s)	0.636	0.879	3.548	**0.412**
Rotation (deg/s)	0.267	0.321	1.091	**0.082**

Table 3.10: Translational and rotational deviations per second: robustness against noise and moving objects.

3.F.8 Conclusions

We have presented a novel approach named RF2O to estimate the planar motion of a lidar by imposing the range flow constraint equation on consecutive scan pairs. Various experiments have been carried out to demonstrate the accuracy of our method, and comparisons with point-to-line ICP, Polar Scan Matching and the standard wheel odometry have been performed in different

scenarios and frame rates. Results show that RF2O provides the most accurate estimates for both translations and rotations, except for non-static environments, where PL-ICP is slightly superior. With a reported runtime of barely 1 millisecond, planar motion can be easily estimated with almost no computational cost, which makes this method attractive for many robotic applications that are computationally demanding and require real-time performance. For future work, we plan to analyze the effect of small deviations from planar motion, which might be useful if this method is applied to estimate the motion of a quadcopter or a vehicle with strong dynamics.

3.G Robust Planar Odometry based on Symmetric Range Flow and Multi-Scan Alignment

Mariano Jaimez, Javier G. Monroy, Manuel Lopez-Antequera,
Daniel Cremers and Javier Gonzalez-Jimenez

Submitted to IEEE Transactions on Robotics, 2017.

Robust Planar Odometry based on Symmetric Range Flow and Multi-Scan Alignment

Mariano Jaimez, Javier G. Monroy, Manuel Lopez-Antequera,
Daniel Cremers and Javier Gonzalez-Jimenez

Abstract

This paper presents a dense method for estimating planar motion with a laser scanner. Starting from a symmetric representation of geometric consistency between scans, we derive a precise range flow constraint and express the motion of the scan observations as a function of the rigid motion of the scanner. In contrast to existing techniques, which align the incoming scan with either the previous one or the last selected keyscan, we propose a combined and efficient formulation to jointly align all these three scans at every iteration. This new formulation preserves the advantages of keyscan-based strategies but is more robust against suboptimal selection of keyscans and the presence of moving objects.

An extensive evaluation of our method is presented with simulated and real data in both static and dynamic environments. Results show that our approach is one order of magnitude faster and significantly more accurate than existing methods in all the conducted experiments. With a runtime of about one millisecond, it is suitable for those robotic applications that require planar odometry with low computational cost. The code is available online as a ROS package.

3.G.1 Introduction

Motion estimation is one of the major challenges in robotics and computer vision. Virtually every robot, be it a drone, a humanoid or a manipulator, needs to accurately keep track of its position to perform an autonomous task. Although different technologies exist for estimating the motion of a robot (e.g. GPS systems, inertial sensors or encoders), visual odometry is arguably the most flexible and powerful solution since it can work with different input data (photometric/geometric) and can be adapted to almost any type of robot.

Among the many robotic platforms used nowadays, a significant percentage of them operate in structured environments and move on a planar surface. Examples of those are:

- Service robots working in hospitals, museums, hotels or airports [68].
- Telepresence robots that operate in domestic environments to monitor and assist old or disabled people [69, 70].
- Autonomous mobile robots employed in warehouses for sorting and delivery of goods [71].
- Modern vacuum cleaners like the iRobot Roomba or the Dyson 360 Eye.

To perceive their surroundings, these robots are often equipped with one or more laser scanners that allow them to survey the environment in a plane parallel to the floor. The data provided by these sensors is suitable for this kind of applications since it can be simultaneously exploited for obstacle avoidance, odometry, localization and 2D mapping.

In this paper we address the problem of estimating planar motion with a radial laser scanner. Our proposal takes inspiration from the latest research on dense and direct 3D visual odometry [48, 47, 3, 66] and expresses the odometry as an energy minimization problem where the scans

are aligned as piecewise continuous functions without searching for explicit correspondences. Despite the extensive body of literature in the field and the remarkable results achieved thus far, we demonstrate that this formulation provides more accurate results than existing techniques. Moreover, these results are achieved with a lower runtime (around 1 millisecond), which renders our method suitable for those robotic systems or applications that are computationally demanding and require real-time operation.

Our approach, which we will refer to as *Symmetric Range Flow-based Odometry* (SRF-Odometry), extends and improves the algorithm presented in [72]. That algorithm is based on the range flow constraint equation and formulates the motion of every observed point as a function of the velocity of the sensor, assuming that the environment is static. In this paper we build upon the same idea, and introduce the following contributions:

- A new symmetric formulation of geometric consistency between scans. To the best of our knowledge, this technique has been applied for the estimation of optical flow but never in the context of visual odometry.
- A new multi-scan formulation which combines the two standard techniques in visual odometry: alignment of consecutive scans/images and alignment against keyscan/keyframe.
- A procedure for modeling the accuracy of our algorithm as a function of the translation and rotation between the registered scans. Based on this model, we propose a new criterion for selecting keyscans by imposing thresholds on the maximum acceptable/desirable translational and rotational errors.
- Faster and more accurate estimates than state-of-the-art techniques, both in static environments or in the presence of moving objects.

We present a thorough evaluation of our method with both synthetic and real data. We analyze how each of its main components contribute to its overall performance and test several versions of it (e.g. two-scan vs multi-scan alignment or robust optimization vs non-robust optimization). Furthermore, we compare our method with four state-of-the-art techniques [5, 6, 73, 74]. Besides analyzing the results presented herein, we encourage the reader to watch the demonstration video and to test the algorithm by themselves. Both the video and the code, which is available as a ROS package, can be found at:

`http://mapir.isa.uma.es/work/SRF-Odometry`

3.G.2 Related Work

Over the last few decades, the scan matching problem has been extensively studied in robotics and computer vision. Although it can be regarded as a general problem, many of the proposed techniques focus on specific applications like localization [59] [60], SLAM [61] or odometry [30]. Since our interest is in the latter, this section will primarily consider those methods which have been particularly designed for (or are commonly applied to) the estimation of planar motion from a sequence of range scans.

In the context of 2D visual odometry, the majority of the existing approaches are based on a dense formulation, i.e., they use all the observations in the scans to align them. Sparse formulations based on interest points like FLIRT [75] or FALCO [76] have been employed for global pose optimization, localization and loop closure, but are rarely used for odometry.

Traditionally, ICP [39] or a number of its variants have been applied to solve the registration problem between consecutive scans. The *Iterative Dual Correspondence* method (IDC) [77] combines two different criteria to find correspondences between the scans: the standard closest-point rule and a new closest-range rule which leads to faster convergence thanks to a better estimation of rotations. Metric-based ICP (MB-ICP) [78] includes a new weighted angular term in its distance metric to improve the search for correspondences under rotation. In [78], MB-ICP obtains very accurate trajectory estimates when the robot wheel odometry provides the algorithm with an initial guess, but no information is provided about how these results would change if no external inputs (wheel odometry) were used. A different approach was proposed by Censi [5], where a point-to-line metric was used instead of the original point-to-point metric of ICP. Furthermore, the author presented an implementation which was an order of magnitude faster than existing ICP variants, while being more precise and efficient than the previous point-to-segment work in [4]. More recently, Generalized-ICP [2] improved the performance of existing ICP versions by including the covariance of both scans in the minimization problem (instead of using only that of the reference scan). However, Generalized-ICP has mainly been used for the registration of 3D point clouds and its performance in aligning 2D range scans does not seem to have been reported yet. In general, for this family of methods, accuracy depends on each particular version and implementation, yet they all share the same weakness: they tend to be computationally expensive.

Alternatively, other methods were specifically designed to solve the 2D scan matching problem:

- Gonzalez & Gutierrez [30] formulated the “velocity constraint equation", an adaptation of the optical flow constraint for range scans, and proposed estimating the lidar motion by imposing this restriction for every observation in the scans. However, their method was only tested on simple simulated scenarios and provided modest results.

- Remarkable results were presented by Biber and Strasser in [73]. Their method, named the *Normal Distributions Transform* (NDT), models the probability of finding a point at a certain position by using a collection of normal distributions to generate a piecewise continuous representation of the 2D plane. This model is created for the reference scan, and is used to evaluate the second scan by projecting it according to the estimated transformation. In this way, the NDT defines and minimizes a cost function which does not include the typical (and slow) search for correspondences. A similar idea based on the Distance Transform was presented by Fitzgibbon [79] to register 2D and 3D point sets and, more recently, a Signed Distance Function-based formulation was proposed by Fossel *et al.* [74] to solve the 2D SLAM problem for laser scanners.

- Censi *et al.* [80] proposed a new method based on the Hough Transform (HT) that permits the combining of the advantages of dense and feature-based scan matching algorithms. Their HT

parameter space is the one associated with lines, and therefore the best results are achieved when the algorithm is tested in polygonal environments. Neither comparisons with other methods nor information about its runtime are provided.

- Diosi & Kleeman presented the Polar Scan Matching approach [62], where the translation and rotation between two scans are alternately estimated until convergence. In contrast to ICP, this method avoids the need to search for correspondences by simply matching points with the same bearing, resulting in better computational performance. This approach was subsequently extended and further evaluated in [6].
- The probabilistic method proposed by Olson [42] attempts to find the rigid transformation that maximizes the probability of obtaining the latest scan given the previous one. Additional information is used (control inputs or wheel odometry) to improve the method convergence and two different implementations, GPU and multi-resolution CPU, are presented. A thorough evaluation of its computational performance is included but, surprisingly, no results for the method's accuracy are presented.

More recently, other approaches have built upon the aforementioned works. This is the case for [63] and [64], which fuse laser odometry (Olson's laser odometry [42] and point-to-line ICP [5], respectively) with stereo vision to perform autonomous navigation with UAVs. Furthermore, the work of Pomerleau *et al.* [65] presents a fast implementation and a thorough evaluation of some ICP variants using real-world 2D and 3D data sets.

3.G.3 Range Flow Constraint for Visual Odometry

In this section we derive a simple and linear constraint for the motion of the sensor by imposing geometric consistency between two consecutive scans. This constraint builds upon two main assumptions: the environment is static and the translation and rotation of the sensor are sufficiently small.

Let r, θ be the polar coordinates of a point with respect to the laser scanner and $R_1, R_2 : \Omega \to \mathbb{R}^+$ be two radial scans taken at consecutive instants of time t_1 and t_2, respectively. For simplicity, we assume that Ω is a continuous domain within the field of view (F_V) of the laser and is given directly in angular coordinates, i.e., $\Omega := [-F_V/2, F_V/2]$. During the time interval $[t_1, t_2]$, the apparent motion $\Delta r, \Delta\theta$ of any point of the environment with respect to the laser scanner must be consistent with the observations of the scans:

$$\Delta r = R_2(\theta + \Delta\theta) - R_1(\theta)\,. \tag{3.79}$$

This constraint is illustrated in Figure 3.11 and is generally valid except in the case of occlusions. Often, this expression is linearized to obtain the so-called "range flow constraint" [29, 72]:

$$\Delta r = R_2(\theta) + \left.\frac{dR_2}{d\theta}\right|_\theta \Delta\theta - R_1(\theta) + O(\Delta\theta^2)\,, \tag{3.80}$$

$$O(\Delta\theta^2) = \frac{1}{2}\frac{d^2R_2}{d\theta^2}\Delta\theta^2 + O(\Delta\theta^3), \tag{3.81}$$

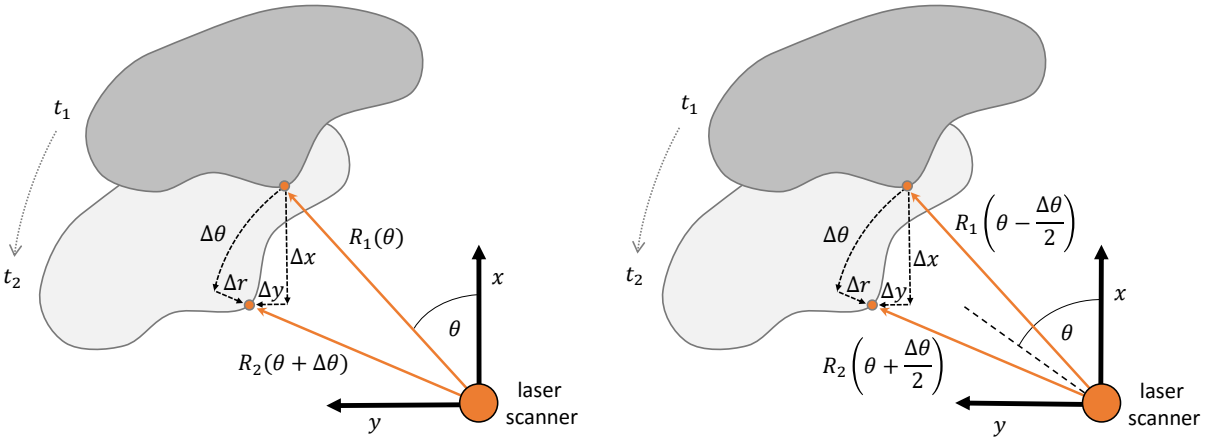


Figure 3.11: **Left**: Standard formulation of geometric consistency between two scans R_1 and R_2 for a given point observed initially at θ. **Right**: Symmetric formulation of geometric consistency applied for the same point considered at the left scheme. The value of θ differs between the schemes because in the symmetric formulation θ represents the average of the initial and the final angles at which the point is observed.

which is a geometric version of the well-known optical flow constraint if the second and higher order terms are neglected. This linearization (3.80) is also a particular 2D case of the general dynamic model presented in [81] for a laser rangefinder.

We propose using a slightly modified version of (3.80), where the motion is equidistributed between the two scans:

$$\Delta r = R_2(\theta + \Delta\theta/2) - R_1(\theta - \Delta\theta/2)\,. \tag{3.82}$$

This alternative representation (Fig. 3.11) has already been used in computer vision to estimate an inherently symmetric optical flow [82]. For us, the major advantage of this formulation is that its linearization is more precise than (3.80):

$$\Delta r = R_2(\theta) - R_1(\theta) + \left(\left.\frac{dR_2}{d\theta}\right|_\theta + \left.\frac{dR_1}{d\theta}\right|_\theta \right) \frac{\Delta\theta}{2} + O\left(\frac{\Delta\theta^2}{4}\right)\,, \tag{3.83}$$

$$O\left(\frac{\Delta\theta^2}{4}\right) = \frac{\Delta\theta^2}{8}\left(\left.\frac{d^2R_2}{d\theta^2}\right|_\theta - \left.\frac{d^2R_1}{d\theta^2}\right|_\theta \right) + O\left(\frac{\Delta\theta^3}{8}\right)\,. \tag{3.84}$$

As can be seen, this symmetric formulation requires more information than the standard range flow constraint (it requires the gradients of both scans) but it has a smaller linearization error for any given $\Delta\theta$.

Next we need to express the motion in Cartesian coordinates. The transformation from polar to Cartesian coordinates (x, y) is exact and linear if applied to instant velocities:

$$\dot r = \dot x \cos\theta + \dot y \sin\theta\,, \tag{3.85}$$

$$r\,\dot\theta = \dot y \cos\theta - \dot x \sin\theta\,. \tag{3.86}$$

To formulate (3.85) and (3.86) in terms of increments, they must be integrated between t_1 and t_2. By assuming that the displacements are small, we can approximate those integrals by the following linear terms:

$$\Delta r = \int_{t_1}^{t_2} \dot{x}\cos\theta + \dot{y}\sin\theta \; dt \approx \Delta x\cos\theta + \Delta y\sin\theta \,, \tag{3.87}$$

$$\Delta\theta = \int_{t_1}^{t_2} \frac{\dot{y}\cos\theta - \dot{x}\sin\theta}{r} \; dt \approx \frac{\Delta y\cos\theta - \Delta x\sin\theta}{\bar{r}} \,, \tag{3.88}$$

with $\bar{r} = (R_1(\theta) + R_2(\theta))/2$ being the best constant approximation of r between t_1 and t_2.

On the other hand, we need to impose the constraint that the relative motion between the environment and the sensor is only caused by the motion of the sensor itself (the environment is static). This motion is encoded by the velocity vector $\boldsymbol{\xi}^s = (\xi_x^s, \xi_y^s, \xi_\omega^s)$, an element of the Lie algebra associated with 2D rigid transformations (i.e. $\boldsymbol{\xi}^s \in \mathfrak{se}(2)$). The motion of every point of the environment can thus be expressed as a function of $\boldsymbol{\xi}^s$ according to the kinematics of a rigid body:

$$\begin{pmatrix}\Delta x\\ \Delta y\end{pmatrix} = \int_{t_1}^{t_2} \begin{pmatrix}\dot{x}\\ \dot{y}\end{pmatrix} dt \approx \begin{pmatrix}-\xi_x^s + \bar{y}\,\xi_\omega^s\\ -\xi_y^s - \bar{x}\,\xi_\omega^s\end{pmatrix} \,, \tag{3.89}$$

where the average coordinates are computed as

$$\begin{pmatrix}\bar{x}\\ \bar{y}\end{pmatrix} = \frac{R_1(\theta) + R_2(\theta)}{2} \begin{pmatrix}\cos\theta\\ \sin\theta\end{pmatrix} \,. \tag{3.90}$$

Finally, plugging (3.87), (3.88) and (3.89) into the range flow (3.83) and discarding the higher order terms, we end up with a linear constraint for the motion of the sensor:

$$\begin{aligned}&\left(\cos\theta + \frac{\bar{R}_\theta\sin\theta}{r}\right)\xi_x^s + \left(\sin\theta - \frac{\bar{R}_\theta\cos\theta}{\bar{r}}\right)\xi_y^s\\ &+ \left(\bar{x}\sin\theta - \bar{y}\cos\theta - \bar{R}_\theta\right)\xi_\omega^s + R_2(\theta) - R_1(\theta) = 0 \,,\end{aligned} \tag{3.91}$$

where

$$\bar{R}_\theta = \frac{1}{2}\left(\left.\frac{dR_2}{d\theta}\right|_\theta + \left.\frac{dR_1}{d\theta}\right|_\theta\right) \tag{3.92}$$

is the average derivative of the two consecutive scans. Therefore, the motion of the sensor $\boldsymbol{\xi}^s$ can be obtained by matching consecutive scans (which should be differentiable or piece-wise differentiable) without searching for and aligning explicit correspondences.

As previously stated, this derivation is valid under the assumption of small motions, i.e. those for which the linearization (3.83) holds. Although there is no sharp transition between "small" and "large" motions, we generally consider that the motion is small if $\Delta\theta$ is always less than or equal to the local neighbourhood used to approximate the range gradients $\bar{R}_\theta$. Commonly, these gradients are approximated with a centred formula using the values of the following and previous observations and, therefore, the angular increment $\Delta\theta$ should be less than or equal to the angle between contiguous observations in the scan.

3.G.4 Optimization Problem

Theoretically, three independent constraints would suffice to obtain the lidar motion but in practice this is unfeasible because (3.91) tends to be inexact due to the noise of the measurements, the errors made by the linear approximation (3.83) or the presence of moving objects (non-static environment). Therefore, we use a dense formulation in which all the scan observations contribute to the motion estimate.

The geometric residual $\rho(\boldsymbol{\xi}, \theta)$ is defined as the evaluation of the range flow constraint (3.91) for a given motion $\boldsymbol{\xi}$ at a given angle θ:

$$\begin{aligned}\rho(\boldsymbol{\xi}, \theta) = R_2(\theta) - R_1(\theta) + \left(\bar{x}\sin\theta - \bar{y}\cos\theta - \bar{R}_\theta\right)\xi_\omega \\ + \left(\cos\theta + \frac{\bar{R}_\theta \sin\theta}{\bar{r}}\right)\xi_x + \left(\sin\theta - \frac{\bar{R}_\theta\cos\theta}{\bar{r}}\right)\xi_y. \end{aligned} \tag{3.93}$$

Since not every arbitrary angle θ can be evaluated, rather only those sampled by the laser scanner, we simplify notation and use $\rho_n(\boldsymbol{\xi})$ to refer to the residual associated with the n-th observation of the scans. To obtain an accurate motion estimate, all the geometric residuals are minimized within a robust cost function:

$$\boldsymbol{\xi}^M = \arg\min_{\boldsymbol{\xi}} \sum_{n=1}^{N} F\left(\rho_n(\boldsymbol{\xi})\right), \tag{3.94}$$

$$F(\rho) = \begin{cases} \frac{\rho^2}{2}\left(1 - \frac{\rho^2}{2c^2}\right) & |\rho| \le c \\ \frac{c^2}{4} & |\rho| > c \end{cases}, \tag{3.95}$$

N being the number of points in the scan. The function $F(\rho)$ is a smooth version of a truncated parabola (Figure 3.12), and c is an adjustable parameter. $F(\rho)$ is continuous and differentiable everywhere, and becomes flat for residuals higher than c, which represents an effective and automatic way to downweight (or even discard) outliers. The parameter c is computed as a ratio of the median absolute deviation (MAD) of the residuals (see §3.G.8).

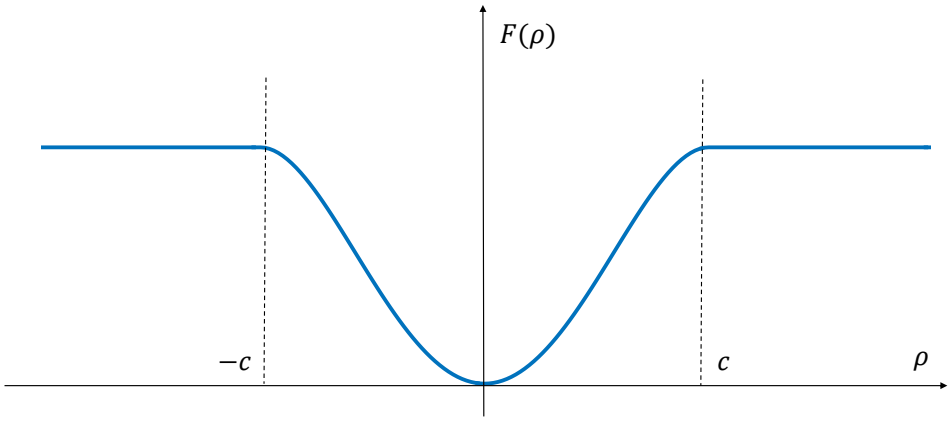


Figure 3.12: Robust penalty function employed to minimize the geometric residuals.

Pre-Weighting Strategy

There are some factors that can render (3.91) inaccurate, mainly the unfulfillment of the rigidity hypothesis (3.89) and deviations from the linear approximation made in (3.83). Although the robust function $F(\rho)$ can alleviate their effect on the overall motion estimate, it does not eliminate it completely. The presence of moving objects is hard to detect before solving the system and we therefore rely on the robust function $F(\rho)$ to downweight them during the minimization process. On the other hand, deviations from the linear approximation adopted in (3.83) can be detected beforehand, which helps to accelerate convergence in (3.94) and also leads to more accurate results. For this purpose, we propose a pre-weighting strategy to downweight the residuals of those observations where the range function (3.82) is highly nonlinear or even non-differentiable. We call it "pre-weighting" because it is applied before the minimization problem (3.94) is solved.

In order to quantify the error associated with the linearization of (3.83), we evaluate the second order terms (3.84) of the Taylor series. Moreover, it is important to identify those regions of the scans where the range function is not only nonlinear but also non-differentiable. These regions are mainly the edges of the various observed objects, and are typically characterized by very high values of the first order derivatives, both the angular $\bar{R}_\theta$ and the temporal $R_t = R_2 - R_1$. To penalize these two effects, nonlinearities and discontinuities, we define the following pre-weighting function:

$$w = \frac{1}{\sigma_s^2 + K_D \left(\bar{R}_\theta^2 + R_t^2\right) + K_{2D}\, \bar{R}_{\theta\theta}^2}\,, \tag{3.96}$$

where $\bar{R}_{\theta\theta}$ is the averaged second-order derivative of R_1 and R_2. The parameters K_D, K_{2D} quantify the relative importance of first and second order derivatives. Furthermore, we add an additional term σ_s^2 to model the noise of the measurements. In this paper we employ a simple constant value for σ_s, but more elaborate and precise noise models could be used instead.

In summary, to estimate the sensor motion we initially compute a pre-weighted set of residuals

$$\rho_n^w(\boldsymbol{\xi}) = w_n\ \rho_n(\boldsymbol{\xi}) \qquad\qquad n \in \{1, 2...N\} \tag{3.97}$$

which are subsequently minimized according to (3.94),(3.95).

3.G.5 Multi-Scan Formulation

Pure odometry always estimates motion between consecutive sets of input data, irrespective of whether these data are wheel rotations, RGB images or range scans. However, this purely incremental strategy has one major drawback in visual odometry: every new increment introduces some error in the pose estimate, even if the real motion is very small or null. This deficiency is commonly solved by periodically selecting a particular scan of the sequence, named as "reference scan" or "keyscan", and aligning every new scan against it. This keyscan acts as a local anchor, helping to reduce the drift of the estimated trajectory. When the incoming scans get too far from the selected keyscan, in the sense that there is not much overlap between the two, a new keyscan must be set and the process continues. A keyscan-based formulation is typically more accurate than purely incremental estimation, but strongly depends on the criterion used to introduce new

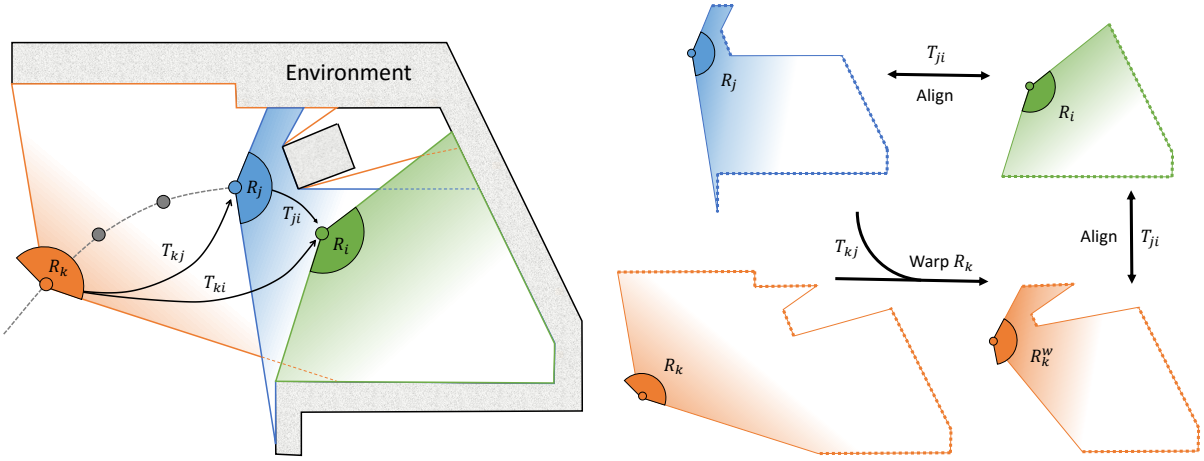


Figure 3.13: **Left**: Schematic representation of our three-scan formulation, with the last scan R_i shown in green, the previous one R_j in blue and the keyscan R_k in orange. **Right**: Diagram of the proposed optimization problem. First, the keyscan R_k needs to be warped according to T_{kj} and, afterwards, both R_k^w and R_j are aligned with R_i. Notice that the warping for R_k^w was performed by keeping the most distant points of R_k after projection, which are likely to represent the structure of the environment and will also provide additional information when compared to R_j.

keyscans. This criterion must prevent the inserting of redundant keyscans and, above all, it must guarantee that the latest scan is always close enough to the keyscan so that they can be aligned.

In this paper we propose a hybrid formulation in which the latest scan is aligned simultaneously against the previous scan and against a keyscan (Figure 3.13). This strategy preserves the advantages of a keyscan-based approach while at the same time reducing the risks originating from inappropriate selection of keyscans. Since three different scans are now fed to the algorithm, the detrimental effects of the sensor noise and the presence of moving objects are also alleviated.

Let $T_{ki}, T_{kj} \in$ SE(2) be the homogeneous transformations between the scans R_i and R_j $(j = i - 1)$ and the last keyscan R_k respectively, and let $T_{ji} \in$ SE(2) be the incremental transformation between R_i and R_j (see Fig. 3.13). Since these transformations form a loop, the following constraint must be fulfilled:

$$T_{kj}\, T_{ji} = T_{ki}\,, \tag{3.98}$$

where T_{kj} is assumed to be known from the previous estimation. Therefore, we can build an optimization problem for jointly aligning the scans R_k, R_j with R_i, subject to (3.98). There are two possible ways of formulating this problem:

- Optimizing for the two sets of unknowns explicitly , i.e., for the vectors $\boldsymbol{\xi}_{ji}$ and $\boldsymbol{\xi}_{ki}$ associated with T_{ji} and T_{ki}.
- Warping R_k towards R_j according to the already known transformation T_{kj} and solving for T_{ji} in both cases.

The second option is more efficient because it involves less unknowns (we only need to estimate $\boldsymbol{\xi}_{ij}$) and implicitly imposes the constraint (3.98) through the warping of R_k towards R_j. Thus, the optimization problem associated with the proposed multi-scan formulation is:

$$\boldsymbol{\xi}^M = \arg\min_{\boldsymbol{\xi}} \sum_{n=1}^{N} F(w_n^{ji}\rho_n^{ji}(\boldsymbol{\xi})) + F(w_n^{ki}\rho_n^{ki}(\boldsymbol{\xi})) \; . \tag{3.99}$$

3.G.6 Solver

Our motion estimation problem is nonlinear and non-convex because both the original constraint of geometric consistency (3.82) and the robust function $F(\rho)$ (3.95) are nonlinear and non-convex. Where the constraint of geometric consistency is concerned, this limitation is solved by deriving the range flow equation (3.83) and defining the geometric residuals as linear functions of the motion of the lidar. However, the resulting linear constraints are only valid for very small displacements and/or rotations and would fail to estimate real motions in practice. This issue has already been addressed in the literature and can be solved by formulating the motion estimation problem within a coarse-to-fine scheme. In a coarse-to-fine scheme, two pyramids of scans (or typically images) are built and aligned, starting from the coarsest level, where the linearization (3.83) holds for larger displacements, and then following this by subsequent refinements in levels with increasing resolutions. Thus, each level incrementally improves the alignment and leaves "less motion" to estimate in the remaining (and finer) levels. More details about coarse-to-fine strategies and warping can be found in [72, 26].

At each level of the coarse-to-fine scheme, the optimization problem (3.99) is solved using Iteratively Reweighted Least Squares (IRLS), where the weights associated with the smooth truncated parabola $F(\rho)$ are:

$$W(\rho) = \begin{cases} 1 - \frac{\rho^2}{c^2} & |\rho| \leq c \\ 0 & |\rho| > c \end{cases} \; , \tag{3.100}$$

Since the robust function $F(\rho)$ is non-convex, we have also contemplated and tested other alternatives for optimizing (3.99), like the lifting strategy proposed in [83], but they do not improve results if compared to IRLS and involve more complicated and slower implementations.

Lastly, for the coarser levels of the coarse-to-fine scheme, it can occur that the motion to be estimated is actually outside the range of motions for which the linearization (3.83) holds. In this case, two different outcomes are possible:

- The real motion is much larger than the valid range for the linearization, and therefore the solver will provide a completely wrong solution.
- The real motion is out of but close to the valid range for the linearization, and therefore the solver will provide a solution which is not precise but comes close to the real motion.

The big failure of the first case cannot be avoided: the scans are simply too far apart and the algorithm will fail. Nevertheless, in the second case the solution can be used to warp one of the two scans towards the other, creating a new configuration in which the remaining motion (to be

estimated) is smaller and, hence, can be obtained more precisely by re-running the algorithm. This process of estimating motion and warping scans can be performed iteratively as long as the last estimated motion is larger than a given threshold ϵ, improving the basin of convergence of our method. The whole estimation process is summarized in Algorithm 2, where the operator $\oplus$ represents

$$\boldsymbol{\xi}_a \oplus \boldsymbol{\xi}_b = \log\left(\exp\left(\hat{\xi}_a\right) \cdot \exp\left(\hat{\xi}_b\right)\right) \tag{3.101}$$

and the $\hat{\xi}$ is the skew-symmetric matrix associated to $\boldsymbol{\xi}$.

Algorithm 2 Motion estimation in a coarse-to-fine scheme

```
Build Scan Pyramids
Initialize estimated motion: ξ^S = 0
for l = 1 : number of levels do
    Initialize motion in this level: ξ_l^S = 0
    for m = 1 : max iterations do
        Compute ξ_l^M solving (3.99) with IRLS
        Update ξ_l^S = ξ_l^S ⊕ ξ_l^M
        Warp R_i → R_j according to ξ_l^S
        if ‖ξ_l^M‖ < ε then break
        end if
    end for
    Update ξ^S = ξ^S ⊕ ξ_l^S
    Warp R_i → R_j according to ξ^S
end for
```

3.G.7 Keyscan Selection

As previously mentioned, a keyscan-based formulation provides more accurate trajectory estimates than consecutive scan alignment, but requires a suitable keyscan selection criterion. Different strategies have been proposed to introduce new keyscans (or keyframes) when a certain magnitude exceeds a manually set threshold. Typically, this threshold is applied to the estimated angular and linear displacement [84], the residual after image alignment [85] or the entropy of the estimation [86]. The main problem with these approaches is that the selection of thresholds is not directly related to the final performance of the algorithm and, hence, the pose estimation error is not constrained to a well defined range. Consequently, these approaches involve tedious trial-and-error stages to tune their thresholds until the desired performance is achieved.

In this work we propose to model the pose estimation error of our algorithm as a function of the translation and rotation between the registered scans. Using this model, thresholds for the keyscan selection can be set directly over the error domain such that a maximum rotation and/or translation error is not surpassed. These thresholds define a 2D working region over the translation and rotation domains, which will be used during operation to trigger the selection of new keyscans.

Model	Samples	FOV (deg)	Range (m)
Hokuyo UTM-30LX	1080	270	30
SICK LMS-500	361	180	80

Table 3.11: Rangefinders used for modeling the error of the algorithm.

When modeling the error, the specifications of the laser scanner (field of view, maximum range and number of points in the scan) are the main source of performance variability. This means that it is not possible to obtain a single and universal error model, but instead the model must be tuned according to the characteristics of the scanner employed. In this section we obtain the error models for two different laser rangefinders: a Hokuyo UTM-30LX and a SICK LMS-500 (see Table 3.11), used in the simulated and real experiments presented in §3.G.9.

We rely on simulated experiments with precise ground truth to generate sufficient and varied samples to model the estimation error. The data is generated by simulating a laser scanner in a certain environment and applying random displacements and rotations to it in the range of $[0, 1]$ metres and $[0, 1]$ radians. For these simulations we make use of three publicly available scenarios/maps: Belgioioso Castle and Intel Research Lab from the Robotics Data Set Repository (Radish) [87], and the Sarmis domestic environment from Robot@home dataset [88]. Furthermore, Gaussian noise with σ ranging from 5 mm to 25 mm is added to the laser measurements. Overall, 60,000 odometry estimates (i.e 20,000 error samples per scenario) are employed to model the error.

After collecting the samples, we estimate the translational e^T and rotational e^R errors at any location (x^T, x^R) of the translation-rotation plane by calculating a weighted mean of the errors obtained in simulation using an anisotropic Gaussian windowing function with $\sigma_R = 0.1$ rad and $\sigma_T = 0.12$ m:

$$e^T(x^T, x^R) = \frac{\sum_{s=1}^{S} \lambda_s \, e_s^T}{\sum_{s=1}^{S} \lambda_s}\,, \tag{3.102}$$

$$e^R(x^T, x^R) = \frac{\sum_{s=1}^{S} \lambda_s \, e_s^R}{\sum_{s=1}^{S} \lambda_s}\,, \tag{3.103}$$

$$\lambda_s = \exp\left(-\frac{(x^T - x_s^T)^2}{2\sigma_T^2} - \frac{(x^R - x_s^R)^2}{2\sigma_R^2}\right) \tag{3.104}$$

where S is the total number of samples. The resulting surfaces are shown in Figure 3.14. As expected, when the translation and/or rotation between consecutive scans increases, so does the average error in the pose estimation. Comparing the error models of the two laser scanners, we can see that they both have a similar shape, but the error values are higher for the SICK LMS-500. This is to be expected because the SICK rangefinder has a smaller field of view and fewer points per scan than the Hokuyo scanner.

Making use of these models, we set thresholds directly for the translational and rotational errors to restrict them to an acceptable range. Specifically, in this work we set these thresholds to 10 mm and 0.1 degrees, respectively. By intersecting them with the surfaces of the error models, we obtain the final working regions shown in Figure 3.14. It can be seen that the shapes of the regions are similar but their scales differ. The scanner with larger field of view and higher

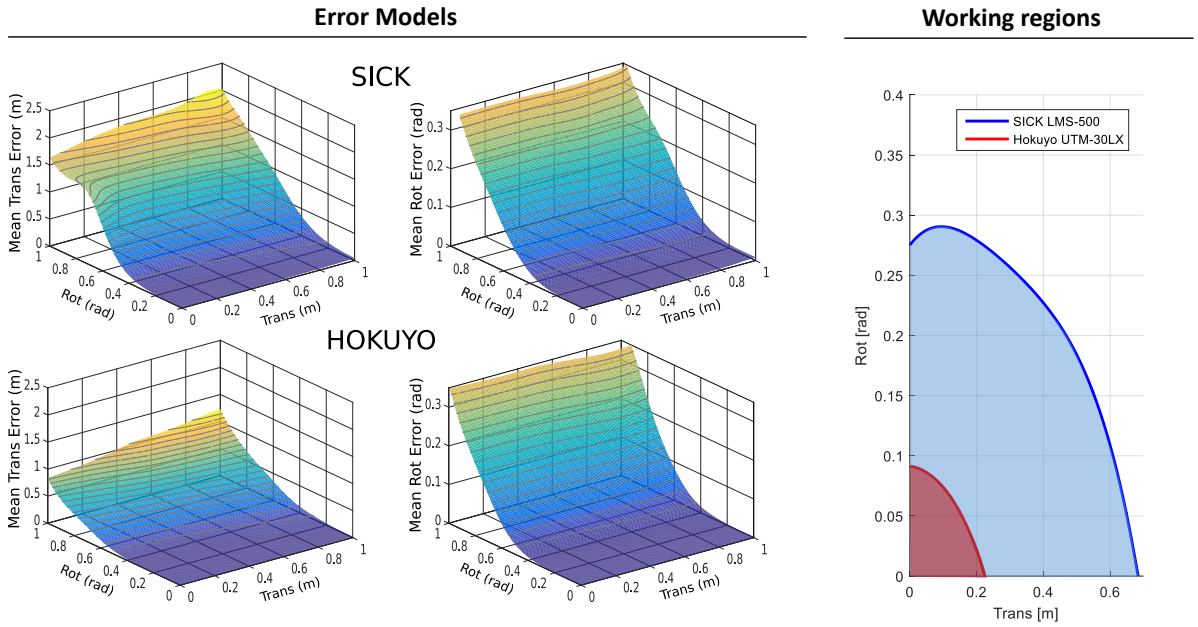


Figure 3.14: **Left**: Translation and rotation error models of the presented odometry algorithm when employing: (top) SICK LMS-500, and (bottom) Hokuyo UTM-30LX. **Right**: Working regions obtained from the error models for two different laser rangefinders, setting a maximum translation error of 10 mm, and a maximum rotation error of 0.1 degrees. If an odometry estimate falls outside the working region, a keyscan update is triggered.

number of points offers a larger working region, indicating that scans carrying more information can work with sparser keyscans while keeping the pose error within the same bounds. Once the working region has been calculated, we fit a fourth-degree polynomial to its boundary, which will be evaluated at each iteration to determine whether the system is inside or outside of the working region. If the estimated pose with respect to the current keyscan falls within the working region, we trust the odometry estimation and keep the current keyscan while otherwise we trigger a keyscan update.

3.G.8 Implementation Details

In this section we describe important details of our algorithm which are not a part of its theoretical core but have an impact on its performance. We also set the values of the parameters introduced throughout the paper and explain how they affect the motion estimates.

Gradient Approximation

Typically, a fixed discrete formula is employed to approximate scan or image gradients. In the case of range data, this strategy leads to very high values of the gradients at the object borders, which do not represent the real gradients of the observed surface(s). As an alternative, we make use of an adaptive formula that weights forward (R_θ^+) and backward (R_θ^-) derivatives in the scan with the 2D distances between contiguous observations (points):

$$R_\theta(n) = \frac{d(n+1)\, R_\theta^-(n) + d(n)\, R_\theta^+(n)}{d(n+1) + d(n)}\,, \tag{3.105}$$

$$d(n) = \|((x(n) - x(n-1), y(n) - y(n-1))\| \ ,$$

where n refers to a specific index in the scan. Thus, the closest neighbour always contributes more to the gradient computation while very distant points barely affect it. If both neighbours are approximately equidistant, the presented formula is equivalent to a centred finite-difference approximation.

Motion Filter

The environments in which the robot operates sometimes includes rooms or areas where the sensor motion cannot be fully recovered, e.g. a long corridor. This is the so-called *aperture problem*: some components of the motion are undetermined and the solver can only provide an arbitrary solution for them. In order to mitigate this problem, we apply a low-pass filter in the eigenspace of the velocity $\boldsymbol{\xi}$ and use the previous estimate to constrain the underdetermined motion. First, we obtain the covariance matrix $\Sigma \in \mathbb{R}^{3\times3}$ associated with the IRLS solution of (3.99). Second, the eigenvalues of Σ are computed and analyzed to detect which components of the motion are undetermined and which are perfectly constrained. In eigenvector space, the velocity $\boldsymbol{\xi}_i^M$ provided by (3.99) is weighted with that of the previous time interval $\boldsymbol{\xi}_{i-1}$ to obtain the new filtered velocity $\boldsymbol{\xi}_i$:

$$[(1 + k_l)I + k_e E]\, \boldsymbol{\xi}_i = \boldsymbol{\xi}_i^M + (k_l I + k_e E)\, \boldsymbol{\xi}_{i-1} \ , \tag{3.106}$$

where E is a diagonal matrix containing the eigenvalues and k_l, k_e are parameters of the filter. Actually, k_l imposes a constant weighting between the solution from the solver and the previous estimate while k_e defines how the eigenvalues affect the final estimate. These parameters are set to the following values:

$$k_l = 0.02\ e^{-(l-1)}, \quad k_e = 5 \times 10^3 e^{-(l-1)} \tag{3.107}$$

where l is the pyramid level that ranges from 1 (coarsest) to the number of levels considered. These values provide good results in all the experiments presented in this paper but they have been obtained heuristically. As a general rule, these values could be decreased if the environment is known to be "geometrically well-constrained", and could be increased in the opposite case. Please refer to [66] for a more detailed explanation on how this filter is applied in a coarse-to-fine scheme.

Parameters for the Robust Optimization

There are several parameters that directly affect the optimization problem (3.99). On the one hand, the pre-weighting function w depends on two parameters (K_D and K_{2D}) and the sensor noise model σ_s. We do not present a formal procedure for tuning K_D and K_{2D} but rather use the values that empirically provided us with the most accurate results. Specifically, we set $K_D = 0.01$ and $K_{2D} = 2\times10^{-4}$. In general, higher values of K_D and K_{2D} lead to higher weights for points close to the sensor (whose coordinates and derivatives tend to be more precise) but will excessively downweight distant points which are sometimes necessary to constrain the estimated

motion. For the sensor noise we set $\sigma_s = 0.02$ m in all cases, which is a representative average value of the noise found in common laser scanners used in robotics. On the other hand, the robust penalty function $F(\rho)$ (3.95) includes the parameter c which marks the limit between inliers and outliers (residuals higher than c lie on the flat area of $F(\rho)$ and therefore do not contribute during the optimization process). We use the median absolute deviation (MAD) of the residuals to tune c and, more specifically, we set $c = 4\,\text{MAD}\,(\rho_n)$. This is a high threshold for outlier rejection, in the sense that it keeps most of the observations as inliers and only those with noticeably high residuals will become outliers (e.g. moving objects).

3.G.9 Experiments

We present a thorough evaluation of our method with simulated and real data in static and dynamic environments. First, we analyze the contribution of each component of our algorithm to its overall performance. We compare different versions of it: with or without pre-weighting (3.97), robust or non-robust minimization, symmetric versus nonsymmetric formulation, and multi-scan versus scan-to-scan alignment. Second, our method is quantitatively and qualitatively compared with some of the most prominent algorithms on scan matching.

For the synthetic experiments we use the environments presented in Figure 3.15 and simulate a Hokuyo UTM-30LX laser scanner (see Table 3.11), including Gaussian noise in the range measurements of $\sigma = 1$ cm. Note that to avoid any bias in these experiments related to the learned error model presented in §3.G.7, we employ different testing scenarios here.

In both the real and simulated experiments, we do not make use of the robot wheel odometry as an initial guess for the motion estimate. Unlike other approaches, which assume that a good initial estimate is always provided, we consider that the scans are the only available inputs for the algorithms.

All the experiments presented here have been run under Ubuntu 16.04 using a single core of an Intel(R) Core(TM) i7-2600K at 3.40GHz.

A) Comparative Analysis of Each Component of the Algorithm

In this section, several versions of our algorithm are evaluated. We simulate a robot equipped with a laser scanner navigating randomly around the free space of the maps shown in Figure 3.15.

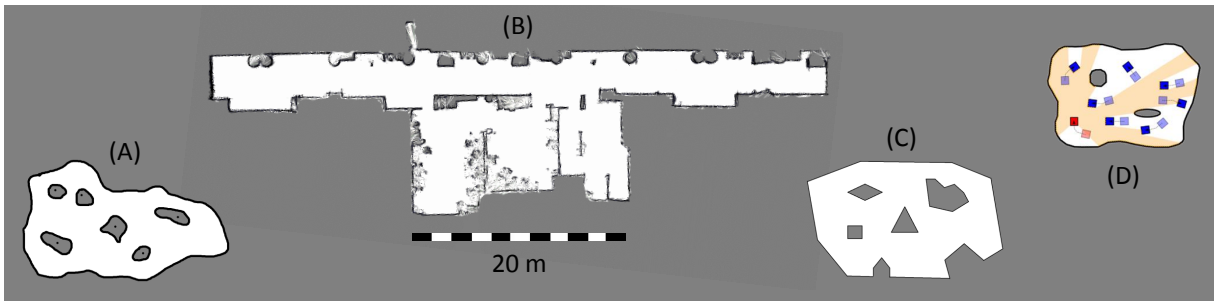


Figure 3.15: Occupancy gridmaps of the environments used in the experiments (simulations). (A) Synthetic map based on curved lines, (B) real map of an office environment, (C) synthetic polygonal map and (D) dynamic environment with additional mobile robots to test robustness to moving objects.

	Traj.	Translational RMSE (cm/s)				Rotational RMSE (deg/s)				Overall drift (m)			
Map	length (m)	SQ	SQ+PW	RF	RF+PW	SQ	SQ+PW	RF	RF+PW	SQ	SQ+PW	RF	RF+PW
A	148.3	86.87	0.688	0.396	**0.298**	7.147	0.096	0.096	**0.026**	12.98	0.799	0.277	**0.225**
B	137.5	617.0	1.212	0.596	**0.505**	6.915	0.105	0.069	**0.044**	483.4	0.862	**0.156**	0.164
C	144.9	193.4	0.759	0.436	**0.225**	15.06	0.129	0.104	**0.023**	42.35	0.201	0.264	**0.084**

Table 3.12: Effect of pre-weighting and robust minimization - Accuracy measured as relative and overall drift.

We use simulations instead of real data because they provide a perfect ground truth, which is necessary for quantitative comparisons. The maximum translational and rotational velocities of the robot are set to 0.5 m/s and 45 degrees/s respectively.

Pre-Weighting and Robust Minimization: We assess the usefulness of the pre-weighting strategy (3.97) and the robust minimization of the residuals (3.94) within the overall motion estimation process. To this end, we compare four basic versions of our method which minimize:

- Squared residuals $|\rho(\boldsymbol{\xi})|^2$ without pre-weighting (SQ).
- Squared residuals $|\rho^w(\boldsymbol{\xi})|^2$ with pre-weighting (SQ+PW).
- Robust residuals $F(\rho(\boldsymbol{\xi}))$ without pre-weighting (RF).
- Robust residuals $F(\rho^w(\boldsymbol{\xi}))$ with pre-weighting (RF+PW).

In these experiments, the motion is estimated by aligning consecutive scans (we do not evaluate the multi-scan formulation yet). We simulate a robot navigating for 10 minutes in the three environments A-C shown in Figure 3.15. The scanning frequency is set to 5 Hz. The estimation errors are measured as the root mean square (RMSE) translational and rotational deviations per second, as described in [49]. Results are presented in Table 3.12, where it can be seen that all versions except SQ provide fairly good estimates with overall translational drifts always far below 1% of the distance travelled by the robot. Robust estimation without pre-weights is more accurate than non-robust estimation with pre-weights, but it is the combination of both strategies which leads to the best results. Moreover, the associated runtimes (Table 3.13) show that the pre-weighting actually accelerates convergence of the solver when combined with the robust function $F(\rho)$.

SQ	SQ+PW	RF	RF+PW
0.629	0.645	1.254	1.139

Table 3.13: Runtime (ms) of the Simplified Versions of Our Algorithm.

Symmetric vs Non-symmetric Formulation: In this section we compare two different versions of our method: one based on symmetric range flow (3.83) (SRF) as described in §3.G.3, and another derived from the standard non-symmetric range flow constraint (3.80) (NSRF). For this

Map	Traj. length (m)	Translational RMSE (cm/s)		Rotational RMSE (deg/s)		Overall drift (m)	
		SRF	NSRF	SRF	NSRF	SRF	NSRF
A (5Hz)	158.0	**0.314**	0.323	**0.025**	**0.025**	0.11	**0.01**
B (5Hz)	122.8	**0.517**	0.537	**0.043**	0.048	0.33	**0.29**
C (5Hz)	163.9	**0.227**	0.231	**0.025**	**0.025**	**0.21**	0.23
A (2Hz)	374.6	**0.233**	0.325	**0.016**	0.274	**0.15**	2.27
B (2Hz)	376.9	**0.704**	1.433	**0.736**	1.293	**2.55**	3.19
C (2Hz)	393.1	**0.191**	0.372	**0.017**	0.219	**0.27**	2.08

Table 3.14: Symmetric vs nonsymmetric formulation - Accuracy measured as relative and overall drift.

and the rest of the experiments presented below we minimize the geometric residuals using pre-weighting and the robust penalty function $F(\rho)$. The methodology is similar to that described in Section 3.G.9 but in this case two different scanning frequencies are employed: 2Hz and 5Hz.

Results are shown in Table 3.14. It can be seen that differences between the two methods are negligible when the scanning frequency is 5Hz, but they become significant for 2Hz. These results are consistent with the theory presented in §3.G.3. When the scanning frequency is 5Hz consecutive scans are close to each other and hence both linear approximations (3.80) and (3.83) are valid and provide accurate results. However, the alignment of consecutive scans taken at 2Hz involves estimating larger translations and rotations for which the non-symmetric linearization (3.80) is not always valid.

Scan-to-Scan vs Multi-Scan Alignment: In this section we compare three different strategies to estimate motion: consecutive-scan alignment (CA), keyscan-based alignment (KA) and the multi-scan approach (MA) described in §3.G.5. Experiments include normal operation at 5Hz in static and dynamic environments, the estimation of large displacements when the laser frequency is set to 2Hz and estimation with noisy measurements. For the experiment with moving objects, we introduce additional robots in the simulation that are permanently wandering and contradict the assumption of a "static environment". In this case we employ a very simple synthetic map (Figure 3.15-D) which, due to its limited number of obstacles and small dimensions, ensures that the additional robots are visible from the laser scanner. As a consequence of the multiple random navigations, which often lead to some robots blocking the way of the others, the distance travelled during these tests is considerably shorter than in the remainder of the experiments.

Results are presented in Table 3.15. It can be observed that KA and MA provide equally good results in static environments when the scanning frequency is 5Hz, while CA always being less accurate both locally and globally. However, the multi-scan formulation is the most precise alternative in the other (more challenging) experiments. It is slightly more precise than KA for large displacements because it is less dependent on the right selection of keyscans. It is also more precise when tested with very noisy scans (σ = 0.1m) since it has more information than CA or KA alone with which to constrain the motion estimate. Lastly, it significantly outperforms CA and KA in the presence of moving objects: the use of multiple scans implicitly facilitates the discernment and downweighting of the moving parts of the scene. In this case, MA is locally as

Experiment	Map	Trajectory length (m)	Translational RMSE (cm/s)			Rotational RMSE (deg/s)			Overall drift (m)		
			CA	KA	MA	CA	KA	MA	CA	KA	MA
	A	162.6	0.327	**0.208**	0.222	0.025	0.018	**0.017**	0.152	0.045	**0.034**
Standard	B	168.7	0.596	**0.421**	0.452	0.043	0.032	**0.030**	0.595	**0.315**	0.433
(5Hz)	C	165.4	0.252	**0.182**	0.186	0.024	**0.018**	**0.018**	0.092	**0.033**	**0.033**
Large	A	175.8	0.216	**0.181**	0.186	0.018	0.016	**0.015**	0.320	0.140	**0.101**
displacements	B	139.7	0.905	0.938	**0.734**	0.614	0.483	**0.360**	**1.417**	1.959	2.646
(2Hz)	C	152.8	0.193	0.178	**0.173**	0.017	0.017	**0.015**	0.017	0.020	**0.015**
noise	A	172.9	1.894	1.557	**1.397**	0.205	0.151	**0.141**	0.372	0.766	**0.256**
σ = 0.1m	B	163.8	1.922	1.613	**1.555**	0.095	0.096	**0.078**	0.502	0.416	**0.302**
(5Hz)	C	148.9	1.803	1.542	**1.418**	0.203	0.149	**0.139**	1.386	0.590	**0.496**
moving	D (3 obj.)	17.65	0.251	0.363	**0.186**	0.046	0.063	**0.037**	0.123	**0.097**	0.104
objects	D (5 obj.)	16.34	0.364	0.943	**0.197**	0.051	0.428	**0.031**	0.524	0.080	**0.076**
(5Hz)	D (7 obj.)	13.94	0.273	0.616	**0.261**	**0.061**	0.184	**0.061**	0.276	0.083	**0.062**

Table 3.15: Different scan alignment strategies - Accuracy measured as relative and overall drift.

smooth as CA (KA provides trembling estimates which lead to high relative errors) and overall precise as KA.

B) Comparisons with Other Methods

In this section we compare our approach (SFR) with the Polar Scan Matcher (PSM) [6], the Canonical Scan Matcher (CSM) [5], the Normal Distributions Transform (NDT) [73] and the Signed Distance Function-based SLAM (SDF) [74]. We consider here the complete version of SFR, i.e. that based on symmetric range flow and multi-scan alignment. For PSM, CSM and SDF we use the original code published by the authors, while for NDT we employ the implementation available in the Point Cloud Library [58][2].

Simulation experiments: We provide a general quantitative evaluation of the different methods in simulation, where ground truth is available. Firstly, we evaluate their performances in static environments (Figure 3.15 A-C) with two scanning frequencies (5Hz and 2Hz). Secondly, we evaluate the accuracy of all methods in the presence of moving objects (map D in Figure 3.15), i.e. when assumption (3.89) is violated. Thirdly we include an additional test in an outdoor environment. The latter case is unfavourable for our approach, which performs dense alignment without explicit correspondences and therefore requires piece-wise differentiable scans to work. The map of the Freiburg Building dataset [87] is used for this test, which is approximately 250×250 meters wide and contains large empty areas and many scattered points.

Results are shown in Table 3.16. Our method clearly outperforms PSM, NDT and SDF for both small (5Hz) and large (2Hz) motions, CSM being the only close competitor with good relative errors on average and a moderate final drift. In general, PSM and SDF perform poorly in all sequences, requiring higher scan frequencies than the ones tested (i.e less displacement between scans) to obtain good estimates. This is specially noticeable for SDF, which often fails to register new scans to its internal map/representation of the environment (even at 5Hz), with consequent errors in the odometry estimation and map updating. NDT is the third best candidate after SRF and CSM. It estimates translations fairly well but copes poorly with large rotations

[2]The original code should provide better results with a lower runtime, as supported by [73], but unfortunately it is not publicly available.

	Translational RMSE (cm/s)					Rotational RMSE (deg/s)					Overall drift (m)				
Map	SRF	CSM	PSM	NDT	SDF	SRF	CSM	PSM	NDT	SDF	SRF	CSM	PSM	NDT	SDF
A (5Hz)	**0.208**	0.826	11.991	2.868	7.802	**0.017**	0.156	1.632	1.095	1.122	**0.035**	0.426	3.985	2.868	8.762
B (5Hz)	**0.374**	0.958	2.039	3.572	23.17	**0.028**	0.073	0.732	1.298	3.644	**0.131**	0.430	4.064	0.800	27.73
C (5Hz)	**0.166**	0.468	11.03	2.793	10.75	**0.016**	0.074	0.920	0.756	2.278	**0.021**	0.202	5.907	1.951	2.739
A (2Hz)	**0.192**	2.311	12.72	10.91	12.91	**0.012**	0.320	3.852	6.454	3.661	**0.046**	2.466	16.76	4.521	14.27
B (2Hz)	**0.409**	4.804	12.70	14.76	22.41	**0.480**	1.556	5.282	7.709	7.449	**1.078**	1.308	5.013	3.224	14.84
C (2Hz)	**0.169**	2.258	13.97	11.23	20.73	**0.016**	0.123	3.802	4.493	6.704	**0.127**	1.079	6.368	7.100	7.749
D (3 obj.)	**0.223**	0.701	9.844	1.226	7.714	**0.038**	0.207	2.974	0.724	2.328	**0.098**	0.572	3.281	0.520	9.052
D (5 obj.)	**0.360**	0.914	3.474	1.457	6.012	**0.050**	0.151	0.665	1.447	2.853	**0.495**	1.142	2.469	1.240	2.569
D (7 obj.)	**0.264**	0.549	3.480	1.138	4.730	**0.059**	0.122	1.436	1.134	2.471	**0.337**	0.691	5.785	2.462	10.06
Outdoor	**0.426**	0.996	4.629	14.99	20.52	**0.028**	0.354	0.675	6.085	9.275	**0.689**	2.014	8.775	52.14	66.39

Table 3.16: General Comparison of Our Approach with Other Methods - Translational and Rotational Deviations per Second, and Overall Drift.

and scans with many sparse points (see outdoor results). It exhibits reasonably good results in the presence of moving objects, with error values similar to those for CSM. As can be seen from Table 3.16, our approach still outperforms the other methods in the outdoor test with an RMSE improvement of more than 50% relative to its closest competitor (CSM). Although the scans were much sparser in this test than in the other experiments, the structure of some buildings was often visible from the lidar (otherwise our method would not have been able to align the scans and CSM would have provided the best results).

The average runtimes shown in Table 3.17 illustrate that our method is not only more accurate but also much faster than the rest of the compared methods, which demonstrates its superiority to current state-of-the-art scan matchers.

SRF	CSM	PSM	NDT	SDF
1.617	10.187	3.947	124.1	12.649

Table 3.17: Runtime (ms) of the Different Methods.

Comparisons with Real Data: In this section we evaluate the different methods with real datasets, and present qualitative and quantitative results. Specifically, we utilize two of the datasets listed in [89]: the *Freiburg indoor building 079* and the *MIT CSAIL building* (introduced in [90]). These datasets are conceived to test SLAM algorithms and cover long distances (423 and 380 meters), very often re-visiting the same place one or more times to test loop-closure strategies. Moreover, consecutive scans are not very close to each other, which makes them particularly challenging for methods based on pure incremental odometry. Given this complexity and the low performance demonstrated by SDF in §3.G.9 (Table 3.16), we exclude it from this comparison.

One important drawback of these datasets is that no ground truth for the robot pose is provided. Instead, they include a precise trajectory estimated with SLAM algorithms, which can be used for quantitative evaluation in some applications. For the odometry estimation problem at hand, only the Freiburg dataset provides a suitable estimated trajectory that can be used as ground truth. In the case of the MIT CSAIL dataset, the laser/pose data is too decimated and therefore insufficient to evaluate odometry. As a consequence, we present quantitative and qualitative results for the Freiburg dataset, but only qualitative results for the MIT CSAIL dataset.

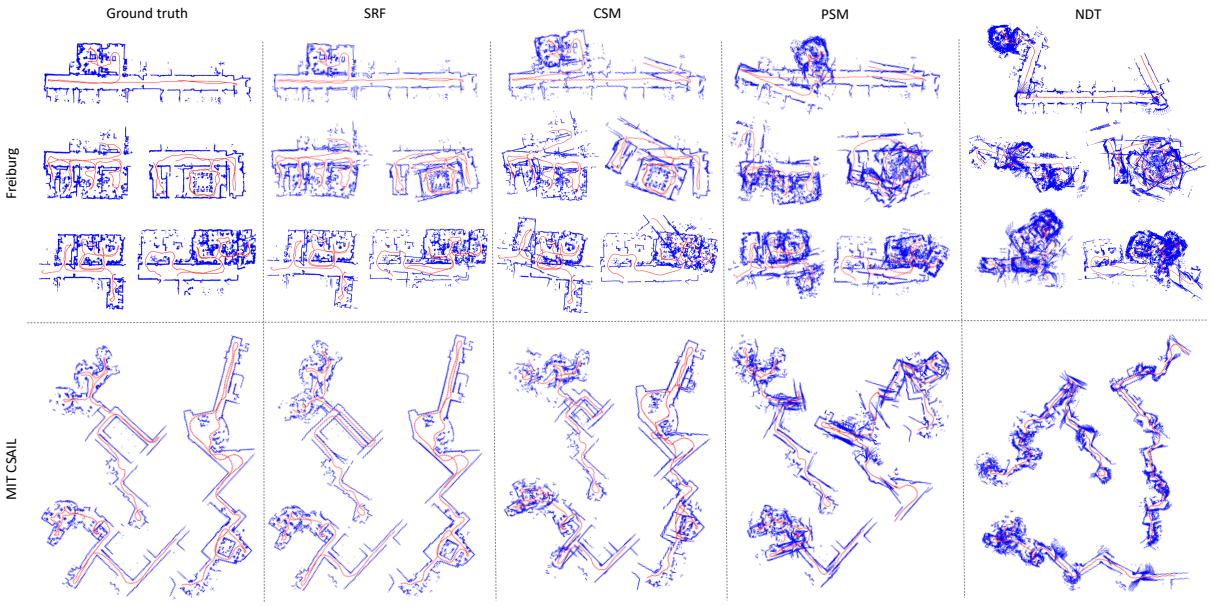


Figure 3.16: Sets of Sub-maps built from the Freiburg and MIT CSAIL datasets. The scan measurements are shown as blue point clouds and the trajectories are plotted in red.

Qualitative results are obtained by fusing the scan observations in a 2D map according to the estimated trajectories provided by each method. Since the datasets are long and the same places are often re-visited, we divide the sequences into a few sub-sequences and build the corresponding sub-maps instead of just one large map per sequence. The resulting estimated maps, together with the maps built from the ground truth, are depicted in Figure 3.16. It can be seen that our approach achieves the best results in both datasets and for every sub-sequence. For the MIT CSAIL dataset, it estimates an almost perfect trajectory for 2 of the 3 sub-maps (bottom-left and right) using only odometry in a sequence which would normally require global pose optimization and loop closure detection.

Quantitative results are obtained by measuring the RMS translational errors per segment length, as described in [67]. We compute these errors for different segment lengths ranging from 1 to 100 meters, and show them in Figure 3.17 as a percentage of the segment length considered. Errors for the shortest segment lengths are not very accurate because the ground truth, although globally consistent, is not exact for local pose increments. Our algorithm has the lowest drift, with RMS errors around 2%. The error associated with CSM ranges from 5% to 7.5%, while the errors associated with PSM and NDT are always above 10% and 20% respectively. These results are less precise than the ones obtained in simulations because the laser scanner employed to record the datasets has a narrower field of view and a smaller size and also because the observed environments contain a high ratio of scattered observations which complicate the scan alignment.

3.G.10 Conclusion

This paper extends and improves the work presented in [72]. We have derived the range flow constraint from a symmetric expression of geometric consistency and have incorporated a multi-

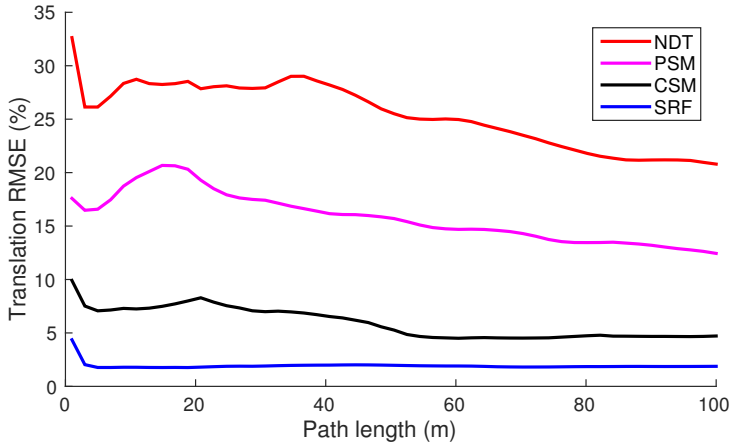


Figure 3.17: Translational RMS errors per segment length computed for the Freiburg dataset, using path lengths from 1 to 100 metres. The errors are shown as a percentage of the segment length under consideration, which illustrates the drift to be expected if trajectories of the given length are to be estimated.

scan formulation that combines the advantages of consecutive scan alignment and keyscan-based approaches. We have also described a procedure to model the average error of our algorithm for different laser scanners and have used this model to define a keyscan-update criterion as a direct function of the maximum desirable translation and rotation errors. We have presented a large set of experiments used to evaluate our method with simulated and real data and to compare it with several state-of-the-art algorithms in scan matching. Quantitative and qualitative results demonstrate that our method is significantly more accurate and faster both in static and dynamic environments.

However, results from the datasets show that the performance of our algorithm deteriorates when the laser primarily sees scattered points in the environment. This is expected because our method relies on range gradients and requires at least a piecewise-differentiable range function to align the scans. For future work, we plan to combine our dense formulation with other sparse techniques, adding an extra term to our cost function to also align interest points.

4 Scene Flow Estimation

4.A Introduction

The term *scene flow* refers to the dense 3D motion field of a scene between two instants of time, and can be regarded as a 3D extension of the well-known optical flow. In contrast to optical flow, the scene flow estimation is a problem that has not often been addressed in the literature since it demands a geometric knowledge of the environment. For this reason, the earliest works on scene flow estimation utilized stereo systems. These methods had to estimate disparity before or simultaneously to the scene flow, which noticeably increases the complexity of the problem. In this context, most scene flow algorithms were extensions of optical flow approaches, where the third component of the motion (shift in depth) was simply obtained as a by-product of the optical flow and the disparity field. However, the arrival of the RGB-D cameras changed this trend. This type of cameras makes it possible to separate scene flow from the disparity estimation problem, which has promoted research in this field.

Scene flow finds a myriad of potential applications. It can be very useful in dynamic environments as a tool to predict the future locations of moving objects. It can also be employed to enhance human-computer interaction by adding information about the velocity of the points of the scene. It is already an important component of many systems developed for tracking and non-rigid 3D reconstruction [91, 92, 93]. Alternatively, it can be exploited to analyze human motion, for both therapeutic and sport purposes. Lastly, it can also be a powerful tool for video processing and compression, but the fact that most video sequences are recorded with single RGB cameras prevents such application at present.

Nevertheless, scene flow expressed as a dense motion field may be hard to exploit because it provides much more data than most applications are able to process (e.g. if computed for QVGA images, it provides 76800 motion vectors for every aligned pair of images). Consequently, it is necessary to develop algorithms that are able to process and simplify such stream of data in a way that existing technologies can directly benefit from it. Moreover, most scene flow algorithms share two limitations. First, they only work with stereo systems or RGB-D cameras and, hence, scene flow cannot be exploited for standard RGB sequences (or at least, to our knowledge, there does not exist any scene flow algorithm for RGB images). Second, it is computationally very expensive, with runtimes that usually range between several seconds and a few minutes per

frame. For these reasons scene flow is still rarely used in robotics or visual&augmented reality, but recent advances suggest that this trend might change.

4.B Representations for Scene Flow

There are different ways to encode scene flow, each of which has distinct advantages and disadvantages. Next, we summarize the most common representations and describe their particular characteristics. [1]

1. Traditionally, scene flow has been expressed as the combination of optical flow $u : \Omega \to \mathbb{R}^2$ and depth displacement $w : \Omega \to \mathbb{R}$. This is ideal for those methods that build upon existing optical flow algorithms, and also for stereo-based scene flow where the problem is inherently decoupled into estimating dense correspondences (optical flow) and depth. Furthermore, it leads to a simple and concise data term $E_D(u, w)$ because photometric and geometric consistency can be formulated directly as a function of u and w:

$$E_D(u,w) = \int_\Omega \|I_2(\boldsymbol{x} + u(\boldsymbol{x})) - I_1(\boldsymbol{x})\| + \mu \, \|Z_2(\boldsymbol{x} + u(\boldsymbol{x})) - Z_1(\boldsymbol{x}) - w(\boldsymbol{x})\| \, \boldsymbol{dx} \,, \quad (4.1)$$

   where μ is a weighting parameter. In general Z can be a depth image (from RGB-D cameras) or a depth field calculated from disparity (from stereo pairs). It must be emphasized that the data term $E_D(u, w)$ shown in (4.1) is a common choice but not the only one: other alternatives or variants of (4.1) have been proposed in the literature.

   On the other hand, this representation provides information about correspondences (endpoints) but not about the trajectory that each 3D point describes between the instants at which the two aligned frames were taken.

2. Alternatively, the motion of the observed points can be represented directly as a 3D displacement field $m : \Omega \to \mathbb{R}^3$. In this case the data term becomes more complex because it involves the projection of the 3D motion to the image plane Ω:

$$\begin{aligned} E_D(m) = \int_\Omega & \left\| I_2\left(\pi\left(\pi^{-1}(\boldsymbol{x}, Z_1(\boldsymbol{x})) + m(\boldsymbol{x})\right)\right) - I_1(\boldsymbol{x}) \right\| \\ & + \mu \, \left\| Z_2\left(\pi\left(\pi^{-1}(\boldsymbol{x}, Z_1(\boldsymbol{x})) + m(\boldsymbol{x})\right)\right) - Z_1(\boldsymbol{x}) - m_z(\boldsymbol{x}) \right\| \boldsymbol{dx} \,. \end{aligned} \quad (4.2)$$

   As an advantage, regularization of the motion field can be directly applied on m. In the case of scene flow, a regularization term E_R is typically imposed to constrain the estimation problem and smooth the motion field. To that end, such a term would often try to minimize the gradients of the 3D motion vectors, thereby enforcing smoothness of the solution:

$$E_R(m) = \int_\Omega \|J_m(\boldsymbol{x})\| \, \boldsymbol{dx} \quad (4.3)$$

   where $J_m(\boldsymbol{x})$ refers to the Jacobian of the motion field with respect to $\boldsymbol{x}$. This is not equivalent to minimizing the gradients of the optical and range flows because a constant optical flow

[1] The notation employed in this section was introduced in §3.B and §3.C, please revisit them if you feel unfamiliar with it.

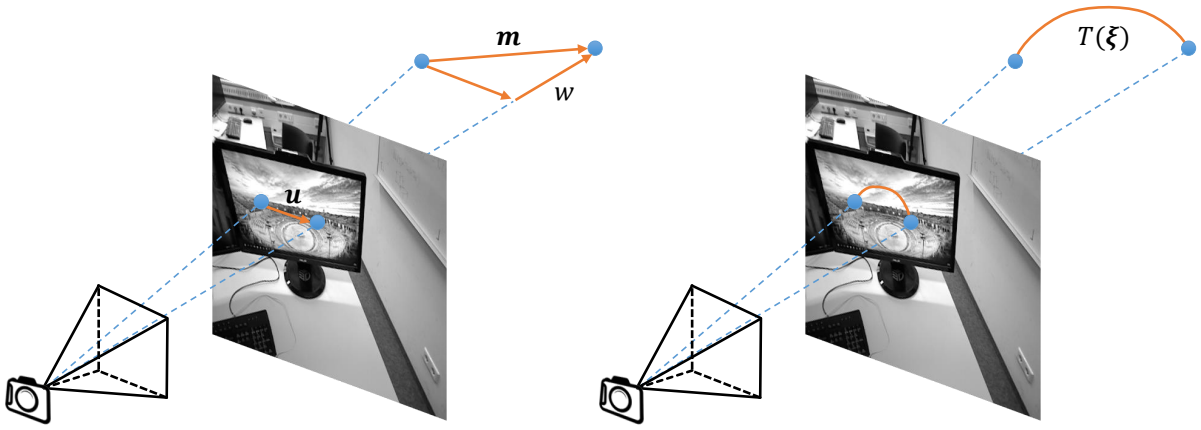


Figure 4.1: Different representations of scene flow[2]. **Left**: Both optical flow + depth shift $(\boldsymbol{u} + w)$ and 3D displacement vector $(\boldsymbol{m})$ provide information only about the final location of each point, but not about its trajectory. **Right**: Encoding scene flow as a rigid body motion makes it possible to recover the trajectory described by every point during the time elapsed between the aligned frames.

(global solution for the regularization term) does not correspond to a constant 3D motion field, and vice versa. In general:

$$E_R(u, w) = \int_\Omega \|J_u(\boldsymbol{x})\| + \mu \|\nabla w(\boldsymbol{x})\| \, \boldsymbol{dx} \neq E_R(m) \tag{4.4}$$

In order to minimize the gradients of the actual motion field using the optical flow-based representation one must include the projection camera model in the regularization term, which complicates the formulation.

Like the optical flow-based formulation, this representation provides information about the final positions of the 3D points of the scene but not about their trajectories.

3. A more elaborate choice consists in overparameterizing the scene flow as a dense field of rigid body motions $\boldsymbol{\xi}$. This parameterization endows each pixel with 6 DoF (in contrast to the 3 DoF of the previous alternatives) and, consequently, requires more constraints to be solved. However, this more complex formulation has some important advantages. First, regularization is more meaningful because rigidity is a more realistic assumption than uniform motion or uniform optical flow:

$$E_R(\xi) = \int_\Omega \|J_\xi(\boldsymbol{x})\| \, \boldsymbol{dx} \;\neq\; E_R(m) \;\neq\; E_R(u, w)\,. \tag{4.5}$$

Second, estimating the underlying rigid motions of the objects of the scene is a powerful tool that can be exploited for image segmentation. Third, regarding each point as part of a rigid body permits us to compute its 3D trajectory between the two aligned frames (see Figure 4.1).

[2]The camera icon was made by Freepik from www.flaticon.com.

4.C Contributions

We contribute several methods to estimate scene flow with RGB-D cameras. Next, we provide a summary of the papers where these methods are described.

In §4.D we present a GPU-based real-time implementation that minimizes an energy function with a primal-dual solver [8]. The data term imposes photometric and geometric consistency between consecutive RGB-D images, and the regularization term enforces smoothness of the motion field. Unlike many existing approaches, which impose regularization on the image plane, we present an alternative strategy to regularize the motion field on the 3D surface of the observed scene, which naturally handles discontinuities of the motion field at the object borders.

A different approach is proposed in §4.E. This paper takes inspiration from [9] and addresses the joint problem of segmenting the scene into the different rigid bodies that compose it and estimating their underlying rigid motions. As a main contribution, we propose a smooth labelling strategy to model the non-rigid motions typically present along the transitions between rigid parts (e.g. in the neck of a person). We incorporate an occlusion mask and include an outlier label to handle those pixels with high residuals for all the motion candidates (those associated to the segments). Results demonstrate that a smooth segmentation based on motion interpolation is more precise than the standard binary alternative, and often leads to a lower number of segments.

Finally, in §4.F we tackle the complex problem of estimating both the camera motion and the scene flow. Here the main difficulty lies in distinguishing static parts of the scene from those which are moving. This step would be straightforward if the camera was still but it becomes challenging when it also moves since all pixels are in apparent motion. We propose to divide the scene into geometric clusters which are, in turn, labelled as static or moving. This strategy also allows us to speed up the scene flow estimation process by two orders of magnitude (if compared to approaches that compute it pixel-wise). Results show that the resulting algorithm provides accurate visual odometry and scene flow with a runtime of 80 milliseconds, which is unprecedented in the literature.

4.D A Primal-Dual Framework for Real-Time Dense RGB-D Scene Flow

Mariano Jaimez, Mohamed Souiai, Javier Gonzalez-Jimenez and Daniel Cremers

Published in Proc. International Conference on Robotics and Automation (ICRA), 2015.

A Primal-Dual Framework for Real-Time Dense RGB-D Scene Flow

Mariano Jaimez, Mohamed Souiai, Javier Gonzalez-Jimenez and Daniel Cremers

Abstract

This paper presents the first method to compute dense scene flow in real time for RGB-D cameras. It is based on a variational formulation where brightness constancy and geometric consistency are imposed. Depth data provided by RGB-D cameras allows us to impose regularization of the motion field on the 3D surface (or set of surfaces) of the observed scene instead of on the image plane, leading to more geometrically consistent results. The minimization problem is efficiently solved by a primal-dual algorithm which is implemented on a GPU, achieving a previously unseen temporal performance. Several tests have been conducted to compare our approach with a state-of-the-art work (RGB-D flow) where quantitative and qualitative results are evaluated. Moreover, an additional set of experiments have been carried out to show the applicability of our work to estimate motion in real time. Results demonstrate the accuracy of our approach, which outperforms the RGB-D flow, and which is able to estimate heterogeneous and non-rigid motions at a high frame rate.

4.D.1 Introduction

Estimating the motion of different objects in a scene is a topic of great relevance in robotics. From a general point of view, and without focusing on particular objects, scene flow is defined as the dense or semi-dense non-rigid motion field of a scene observed at different instants of time. In contrast to optical flow, which provides the projection of the scene motion onto the image plane, scene flow estimates the actual 3D motion field and hence requires more prior information than optical flow (2D). As a consequence, stereo or multi-view camera systems that allow for the estimation of the scene structure have been commonly employed to compute scene flow. However, the new affordable RGB-D cameras, which directly provide registered RGB and depth images at a fairly high frame rate (30 Hz), are an advantageous setting for the implementation of fast scene flow algorithms.

The potential applications of scene flow in the field of robotics are numerous: autonomous navigation and manipulation in dynamic environments, pose estimation or SLAM refinement, human-robot interaction or segmentation from motion are a few examples. Nonetheless, its applicability is highly dependent on its temporal performance because the aforementioned tasks normally need to be executed at a high frame rate. Most existing approaches do not fulfill this requirement and present execution times ranging from several seconds to few hours to compute the scene flow per frame, which in practice limits their usefulness.

In this paper we present the first dense real-time scene flow algorithm for RGB-D cameras. Under a variational framework, a highly parallelizable primal-dual algorithm is proposed to solve in real time the underlying optimization problem. The benefits of this algorithm compared to direct solvers (e.g. SOR) or black box solvers are two-fold: the utilised total variation (TV) regularizer can be minimized with exactitude due to the introduction of dual variables, not needing to resort to differentiable approximations (Charbonnier penalty [94]), and a very fast version of it can be implemented on modern GPUs by reason of its first-order nature. In our variational formulation, both the optical flow and the range flow constraint equations are

applied in a coarse-to-fine scheme to handle large displacements between consecutive frames. Regularization is imposed on the 3D surface in order to smooth the flow field for close points in the real space instead of for contiguous pixels in the image plane. Several experiments have been carried out to evaluate the performance of our approach and compare it with the recent work of Herbst *et al.* [95] (RGB-D flow). Overall, our approach achieves higher levels of accuracy in this comparison while performs three orders of magnitude faster. Quantitative and qualitative results show that our primal-dual scene flow is able to estimate heterogeneous and non-rigid motions precisely on a variety of scenes.

Related Work

Scene flow has traditionally been computed with data coming from stereo or multiple-view camera systems. The term *scene flow* was firstly coined by Vedula *et al.* [96] who proposed to compute the Lucas-Kanade optical flow [97] first and apply the range flow constraint equation at a later stage, obtaining a local solution which does not exploit the geometric data to estimate the optical flow. A global variational approach was presented in [98] where both the optical flow and depth flow are estimated simultaneously using quadratic regularization. To allow for discontinuities in the motion field, Total Variation (TV) was adopted in [99] and [100], where they each present a unified variational formulation to compute the disparity and the motion field jointly. However, other authors claimed that decoupling this two sub-problems is indeed advantageous and developed algorithms [101] that make the most of this decoupling to efficiently solve the global problem, combining FPGA and GPU for disparity and scene flow estimation, respectively, and leading to real-time performance. Moreover, some recent works with stereo cameras [102, 13] propose to exploit the fact that most realistic scenarios are composed of multiple rigid bodies, and impose local rigidity in their formulation which provides more accurate results than standard TV regularization.

The advent of RGB-D cameras few years ago represented a revolution in the field of robotics and computer vision, and much of recent research on scene flow focuses on the use of this kind of cameras. One of the first scene flow algorithms for RGB-D cameras is presented in [103], where a variational approach with quadratic data and regularization terms is proposed. Within a similar variational framework, the work of Letouzey *et al.* [104] makes use of a weighted quadratic regularizer and considers a set of sparse feature correspondences to handle large displacements. The regularization weights are functions of the depth gradients, and are intended to increase regularization between pixels with similar depth values (and vice versa). More recently, the RGB-D flow presented in [95] achieves qualitatively good results by minimizing a functional composed of the L_1 norm of the optical and range flow constraint equations and a weighted TV regularization, where the weighting function encompasses information about the depth, colour and surface normals. The work presented in [105] achieves accurate results too by defining a local/global formulation with adaptive TV regularization, which is a weighted TV regularizer whose weights are similar to those presented in [104], and also resorts to interest points (SURF) to deal with large motions.

Apart from these variational approaches, there exist some Monte Carlo-based methods that provide semi-dense or dense motion fields. Cech *et al.* [106] presented a seed-growing algorithm to compute scene flow in a stereo setup, which represents a good trade-off between local and global methods with respect to accuracy and runtime. Hadfield *et al.* [107] proposed a particle filter to estimate the motion field from depth and intensity images coming from a RGB-D camera, and extended this work in [108] to operate with any combination of photometric and range sensors. Also for RGB-D images, SphereFlow [109] exploits the availability of depth data by seeking correspondences not in the image plane but in spheres of the 3D space, which is demonstrated to be advantageous to handle occlusion and large displacements.

Contribution

The main contribution of this paper is a robust scene flow algorithm for RGB-D cameras that runs in real time. To this end, a primal-dual algorithm is applied for the first time to solve the variational formulation of the scene flow problem. The particular choice of this algorithm is crucial since it is an iterative solver which performs pixel-wise updates and can be efficiently implemented on modern GPUs. Furthermore, a more natural regularization is theoretically justified and imposed, substituting the standard TV by an adaptive TV which represents the line integral of the flow field over the observed surface. Lastly, we take advantage of the geometric data provided by the camera to formulate a more accurate approximation of the image gradients, as well as to filter intermediate solutions in the coarse-to-fine scheme robustly.

The code is available online for public use. We also encourage the reader to watch the demonstration video at:

`http://mapir.isa.uma.es/mjaimez`

4.D.2 Variational Formulation for Scene Flow

We consider the problem of estimating the dense 3D motion field of a scene between two instants of time t and $t+1$ using colour and depth images provided by an RGB-D camera. This motion field $m : (\Omega \in \mathbb{R}^2) \rightarrow \mathbb{R}^3$ is defined over the image domain Ω, is described with respect to the camera reference frame and is expressed in meters. For simplicity's sake, an alternative representation of m is commonly adopted, where m is expressed in terms of the optical flow u, v and the range flow w. For any pixel (x, y) with a nonzero depth value, the bijective relationship $\Gamma : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ between $\boldsymbol{m}(x, y)$ and $\boldsymbol{s} = (u, v, w)^T$ is given by:

$$\boldsymbol{m}(x, y) = \Gamma\left(\boldsymbol{s}(x, y)\right) = \begin{pmatrix} \frac{Z}{f_x} & 0 & \frac{X}{Z} \\ 0 & \frac{Z}{f_y} & \frac{Y}{Z} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix} . \tag{4.6}$$

Equation (4.6) can be obtained directly from the well-known pinhole model, where f_x, f_y are the focal length values and X, Y, Z the spatial coordinates of the observed point. Thus, estimating the optical and range flows is equivalent to estimating the 3D motion field, but the former leads to a simplified implementation since optical and range flow constraint equations apply directly on the image plane.

In order to estimate the motion field we formulate a minimization problem over s (non-bold refers to the whole function in Ω) where photometric and geometric consistency are imposed as well as a regularity of the solution:

$$\min_{s} \{E_D(s) + E_R(s)\} \ . \tag{4.7}$$

In this functional, the data term E_D encodes a two-fold constraint per pixel (intensity and depth matching between pairs of frames), which is insufficient to compute a unique solution. Consequently, a regularization term E_R is crucial because it does not only smooth the flow field but also further constrains the solution space.

Data term

Let I_0, I_1 be the intensity images and Z_0, Z_1 the depth images taken at instants t and $t+1$ respectively. We choose a data term which encourages brightness constancy and geometric consistency of the solution. The former is commonly adopted by existing optical flow and scene flow approaches and encodes that a point should exhibit the same brightness in both intensity images:

$$\rho_I(\boldsymbol{s}, x, y) = I_0(x, y) - I_1(x + u, y + v) = 0 \, . \tag{4.8}$$

Constancy of the intensity gradients is not considered here because our approach runs at a high frame rate, which implies that the brightness constancy assumption is quite accurate (if the source of light remains unaltered). On the other hand, depth does not remain constant over time but its change must be equal to the difference between the first depth image and the second image warped with the optical flow:

$$\rho_Z(\boldsymbol{s}, x, y) = w - Z_1(x + u, y + v) + Z_0(x, y) = 0 \, . \tag{4.9}$$

In contrast to many other approaches, which make use of the L_2 norm or a differentiable approximation (Charbonnier penalty) of the L_1 norm, we minimize the exact L_1 norm of (4.8), (4.9):

$$E_D(s) = \int_\Omega |\rho_I(\boldsymbol{s}, x, y)| + \mu(x, y)|\rho_Z(\boldsymbol{s}, x, y)| dxdy \, , \tag{4.10}$$

where $\mu(x, y)$ is a positive function that weights geometric consistency against brightness constancy. The L_1 norm has shown to be more robust against outliers than the L_2 norm [110] and the choice of the primal-dual algorithm to solve the minimization problem allows us to minimize it exactly without resorting to any approximation (more details will be given in §4.D.3).

The data term E_D is nonlinear, which implies that the energy functional can have multiple local minima. In order to get as close as possible to the global minimum, we use a coarse-to-fine scheme [27]: an image pyramid is built and the solution is computed and upsampled from coarser to finer levels, employing a linearized version of (4.8) and (4.9) at each level:

$$\rho_I(\boldsymbol{s}) \approx I_0(x, y) - I_1(x + u^*, y + v^*) - \nabla I_1(x + u^*, y + v^*) \cdot (u, v)^T \, , \tag{4.11}$$

$$\rho_Z(\boldsymbol{s}) \approx w - Z_1(x+u^*, y+v^*) - \nabla Z_1(x+u^*, y+v^*) \cdot (u,v)^T + Z_0(x,y)\,, \tag{4.12}$$

with u^*, v^* being the solution computed at the previous level. Equations (4.11), (4.12) are the well-known optical flow and range flow constraints whose global minimum can be obtained at each pyramid level.

Regularization term

The regularization term E_R is introduced to overcome the aperture problem associated to the optical and range flows [29], as well as to provide a smooth flow field. In this paper we present a regularizer of the flow field which is based on the total variation but takes into consideration the geometry of the scene, in contrast to standard TV which operates on the image domain Ω and disregards the real-world distances between points. For the sake of clarity, a simplified 2D case (shown in Figure 4.2) will be used to describe the proposed regularization and derive its mathematical formulation. Let $\mathbf{C} : (l \in \mathbb{R}) \rightarrow \mathbb{R}^2$ represent the observed surface of the scene (which becomes a curve in 2D) and $f : \mathbf{C} \rightarrow \mathbb{R}$ be any component of the motion field of the curve $\mathbf{C}$ with respect to the camera. The total variation of f in 2D is defined as:

$$TV(f) = \int_\Omega \left| \frac{\partial f}{\partial x} \right| dx\,. \tag{4.13}$$

However, this regularization does not take into account that contiguous pixels may correspond to distant points in space with different values of f. Thus, if geometric data is available, a more natural regularization would smooth f among points which are close in $\mathbf{C}$ instead of in the image segment:

$$TV_g(f) = \int_\Omega \left| \frac{\partial f}{\partial l} \right| dx\,. \tag{4.14}$$

In both cases we integrate over the image domain Ω because the curve $\mathbf{C}$ is not known, only its projection onto Ω (i.e. depth values and colour information). The derivatives of f with respect to l and x can be decomposed into the two independent directions of space X, Z:

$$\frac{\partial f}{\partial l} = \frac{\partial f}{\partial X}\frac{\partial X}{\partial l} + \frac{\partial f}{\partial Z}\frac{\partial Z}{\partial l}\,, \tag{4.15}$$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial X}\frac{\partial X}{\partial x} + \frac{\partial f}{\partial Z}\frac{\partial Z}{\partial x}\,. \tag{4.16}$$

In (4.15), the vector $(\partial X/\partial l, \partial Z/\partial l)$ is the tangent unitary vector to the curve $\mathbf{C}$ and can be computed as a function of the gradients of the spatial coordinates X, Z over the image domain:

$$\nabla \mathbf{C}_l = \left(\frac{\partial X}{\partial l}, \frac{\partial Z}{\partial l} \right) = \frac{1}{\sqrt{\frac{\partial X}{\partial x}^2 + \frac{\partial Z}{\partial x}^2}} \left(\frac{\partial X}{\partial x}, \frac{\partial Z}{\partial x} \right)\,. \tag{4.17}$$

Substituting (4.17) in (4.15), (4.16) we obtain a weighted TV that takes the geometry of the scene into account:

$$TV_g(f) = \int_\Omega \left| r_x \frac{\partial f}{\partial x} \right| dx = \int_\Omega \frac{1}{\sqrt{\frac{\partial X}{\partial x}^2 + \frac{\partial Z}{\partial x}^2}} \left| \frac{\partial f}{\partial x} \right| dx\,. \tag{4.18}$$

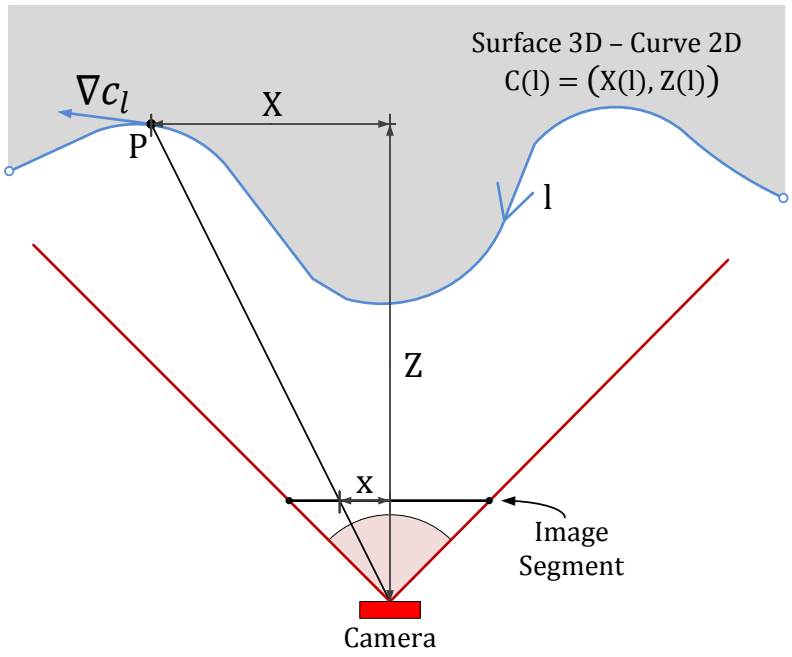


Figure 4.2: Two-dimensional representation (top view) of a camera observing a scene where the image plane Ω becomes an image segment and the frontal 3D surface becomes a 2D curve $\mathbf{C}(l)$.

In contrast to the standard TV (4.13), the regularizer in (4.18) enforces smoothness between close points (high r_x) whereas it barely regularizes f for distant points (low r_x). Such behavior is very convenient since close points are likely to belong to the same object in the scene (hence moving similarly) while distant points might be part of different objects with different motion fields. Equation (4.18) can be easily generalized to the 3D world, leading to:

$$TV_g(f) = \int_\Omega \left| \left(r_x \frac{\partial f}{\partial x}, r_y \frac{\partial f}{\partial y} \right) \right| dxdy \,, \tag{4.19}$$

$$r_x = \frac{1}{\sqrt{\frac{\partial X}{\partial x}^2 + \frac{\partial Z}{\partial x}^2}}, \; r_y = \frac{1}{\sqrt{\frac{\partial Y}{\partial y}^2 + \frac{\partial Z}{\partial y}^2}} \,. \tag{4.20}$$

Finally, the regularization term in our variational formulation imposes the above TV penalization to the three components of the scene flow:

$$\begin{aligned} E_R(s) = \lambda_I \int_\Omega \left| \left(r_x \frac{\partial u}{\partial x}, r_y \frac{\partial u}{\partial y} \right) \right| + \left| \left(r_x \frac{\partial v}{\partial x}, r_y \frac{\partial v}{\partial y} \right) \right| dxdy \\ + \lambda_D \int_\Omega \left| \left(r_x \frac{\partial w}{\partial x}, r_y \frac{\partial w}{\partial y} \right) \right| dxdy \,, \end{aligned} \tag{4.21}$$

where λ_I, λ_D weight these regularization terms within the overall optimization (4.7).

4.D.3 Primal-Dual Algorithm

As pointed out in §4.D.2, our energy formulation is based on a linearization of the data term (4.11), (4.12). Additionally, since we utilize the convex TV regularizer in (4.21), this renders the overall formulation (4.7) convex, which makes our algorithm amenable to convex solvers. As a

non-smooth energy is to be minimized, we have chosen a first order solver [7, 8] for non-smooth problems which tackles the primal-dual formulation of (4.7):

$$\begin{aligned}\min_{s} \sup_{\substack{\{p_u,p_v,p_w\}\in\mathcal{K}\\ \eta\in\mathcal{Q}}} & \lambda_I \int_\Omega \left\langle \boldsymbol{p}_u(x,y), \left(r_x\frac{\partial u}{\partial x}, r_y\frac{\partial u}{\partial y}\right)^T \right\rangle dxdy \\ & +\lambda_I \int_\Omega \left\langle \boldsymbol{p}_v(x,y), \left(r_x\frac{\partial v}{\partial x}, r_y\frac{\partial v}{\partial y}\right)^T \right\rangle dxdy \\ & +\lambda_D \int_\Omega \left\langle \boldsymbol{p}_w(x,y), \left(r_x\frac{\partial w}{\partial x}, r_y\frac{\partial w}{\partial y}\right)^T \right\rangle dxdy \\ & + \int_\Omega |\rho_I(\boldsymbol{s},x,y)| \; + \; \eta(x,y)\; \mu(x,y)\rho_Z(\boldsymbol{s},x,y)\; dxdy\end{aligned} \tag{4.22}$$

where the dual variables p_u, p_v and p_w associated to the regularizer and the dual variable η associated to the data term are constrained by the following sets:

$$\mathcal{K} := \left\{ p : \Omega \to \mathbb{R}^2 \,\middle|\, \|\boldsymbol{p}(x,y)\|_2 \leq 1 \right\} \tag{4.23}$$

$$\mathcal{Q} := \left\{ \nu : \Omega \to \mathbb{R} \,\middle|\, |\eta(x,y)| \leq 1 \right\} \tag{4.24}$$

This iterative solver is suitable for the development of a real-time implementation of scene flow because the primal and dual pixel-wise updates can efficiently be computed in parallel on a GPU. Note that (4.22) is convex with respect to its primal variables and concave with respect to its dual variables which makes it possible to compute a global optimum with the primal-dual algorithm. Regarding the data term (4.10), we decided to just dualize the range flow $|\rho_Z(\boldsymbol{s},x,y)|$ and minimize the photometric term $|\rho_I(\boldsymbol{s},x,y)|$ using the so-called *proximal* or *shrinkage* operator (see [7, 8] for further information and details about the algorithm). In the primal-dual framework, this alternative represents a good balance between fast converge and formulation complexity. On the one hand, the addition of an extra dual variable associated with the term $|\rho_I(\boldsymbol{s},x,y)|$ would slow down convergence and hence the temporal performance of our method. On the other hand, formulating the primal-dual problem without dual variables associated with $|\rho_Z(\boldsymbol{s},x,y)|$ and $|\rho_I(\boldsymbol{s},x,y)|$ would give rise to a more involved and computationally demanding *shrinkage* operator.

4.D.4 Implementation Details

In this section we describe some important aspects associated with the implementation of the primal-dual scene flow. We will mainly focus on the computation of the image gradients, the filter applied to the solutions of each pyramid level and the weighting parameters and functions.

Regarding the image gradients, most implementations of optical or scene flow choose a particular finite difference approximation (forward, backward or centered formulas) and apply it to the whole image. Nevertheless, when this strategy is applied to compute the depth image gradients it gives rise to very high values at the edges/borders of objects, which are not representative

of the real gradients of the surface (or set of surfaces) observed by the camera. Consequently, the estimated range flow w is prone to taking excessively high values at object borders owing to the inaccurate approximation of gradients adopted. As a solution, we introduce an adaptive approximation of image gradients which is consistent with the geometry of the observed scene. It consists in a weighted forward-backward formula, where the weighting functions capture the geometry derivatives in both directions (forward and backward):

$$\frac{\partial I}{\partial x} \approx \frac{r_x^+ \frac{\partial^+ I}{\partial x} + r_x^- \frac{\partial^- I}{\partial x}}{r_x^+ + r_x^-}, \quad \frac{\partial Z}{\partial x} \approx \frac{r_x^+ \frac{\partial^+ Z}{\partial x} + r_x^- \frac{\partial^- Z}{\partial x}}{r_x^+ + r_x^-} \tag{4.25}$$

and similarly for $\partial I/\partial y$ and $\partial Z/\partial y$. The operators ∂^+ and ∂^- represent right and left derivatives and are approximated using the stencils [0 -1 1] and [-1 1 0] respectively, and the terms $r_x^+, r_x^-, r_y^+, r_y^-$ are right and left approximations of the ones presented in (4.20). These expressions downweight derivatives which contain borders or discontinuities (r_x and r_y very low) adopting that approximation which is more likely to capture the real surface gradient properly. If they are evaluated at pixels which do not lie on object borders then both terms will have practically the same weight and (4.25) will be equivalent to standard centered approximations of the image gradients.

Another important issue is the selection of an appropriate filtering strategy. It is commonly known that the solution to the variational problem might contain spurious wrong estimates that must be removed. These wrong motion vectors are particularly detrimental at the first levels of the coarse-to-fine scheme since they are propagated throughout the pyramid altering significantly the final flow estimate. Traditionally, a median filter is applied to the flow estimate at each level to reject outliers. However, this filter presents an important drawback: it combines the motions of different objects when it is applied at their borders. As an alternative, we opt for using a 3×3 weighted median filter, similar to the one described in [111]. In this filter, pixels are weighted in local histograms which are accumulated to obtain the median value. The weighting function h_{median} is defined as:

$$h_{median} = \frac{1}{1 + k_d(\Delta Z)^2 + k_{dt}\left(\frac{\partial Z}{\partial t}\right)^2} \tag{4.26}$$

ΔZ measures the depth difference between pixels, $\partial Z/\partial t$ represents the temporal derivative of depth and k_d, k_{dt} are parameters. This weighting function allows us to apply a median filter that does not mix the flow fields of different objects, and at the same time penalizes pixels with high temporal derivatives, which are likely to contain outliers.

Last, further details are given about the parameters and weighting functions employed in our variational formulation. The function $\mu(x, u)$ presented in (4.10) which weights the geometric data term is defined as:

$$\mu(x, y) = \frac{\mu_0}{1 + k_\mu\left(\frac{\partial Z}{\partial x}^2 + \frac{\partial Z}{\partial y}^2 + \frac{\partial Z}{\partial t}^2\right)} \tag{4.27}$$

This definition reinforces geometric consistency in areas with low depth gradients and down-weights it otherwise (high depth gradients are normally caused by jumps between objects, the

presence of null depth measurements, occlusions, etc.). Furthermore, the parameters involved in the calculation of the scene flow are empirically set to the following values:

$$\lambda_I = 0.04,\ \lambda_D = 0.35,\ k_d = 5,\ k_{dt} = 10,\ \mu_0 = 75,\ k_\mu = 1000\,. \tag{4.28}$$

4.D.5 Experiments

We have performed two sets of experiments to evaluate our approach quantitatively and qualitatively. Given the lack of a benchmark to test scene flow with RGB-D cameras, some authors resort to the stereo Middlebury datasets [112, 113] and use the depth groundtruth which, together with the intensity images from different views, emulate a pair of RGB-D images. However, this set of images does not exhibit the same characteristics as real RGB-D images: they are not affected by realistic noise or quantization effects, nor contain large empty areas (null depth) that are quite common in depth images. For these reasons, we have created a tool to test scene flow algorithms by generating artificial RGB-D images from a previously captured RGB-D frame and a predefined 3D motion field. Following this procedure, we will present quantitative and qualitative results from this semi-real data. Moreover, qualitative results for real RGB-D streams processed in real time will be shown. In all cases the resolution of the motion field is 240×320 (QVGA).

Evaluation with semi-real data

In this subsection we compare our approach (named PD-flow) with a recent work [95] (RGB-D flow) in terms of accuracy and temporal performance. To this purpose, we have developed a procedure to evaluate the similarity between the estimated and real motion field that exists between two RGB-D frames. This procedure is decomposed into the following steps:

1. Capture an RGB-D frame with the camera.
2. Create a 3D coloured mesh from the RGB-D image.
3. Generate an artificial motion field consistent with the geometry of the scene.
4. Apply this motion field to the vertices of the mesh.
5. Render new intensity and depth images from this deformed mesh.

As a result, a second RGB-D frame is created from a real RGB-D image and a known motion field, which can be used as a groundtruth. The resulting intensity and depth images mainly differ from real ones in one aspect: they do not contain any new information respect to the first RGB-D frame. Thus, some pixels (mainly at the image borders) might not observe the deformed mesh, and their intensity and depth values are set to zero. On the other side, the intensity values at pixels with null depth measurements are neglected (set to zero) because they are not used to build the 3D mesh (their 3D location is unknown) and they will not appear in the second frame. In any case, this last factor is not very relevant since scene flow cannot be evaluated at pixels with null depth measurements (the Γ transformation in (4.6) becomes singular).

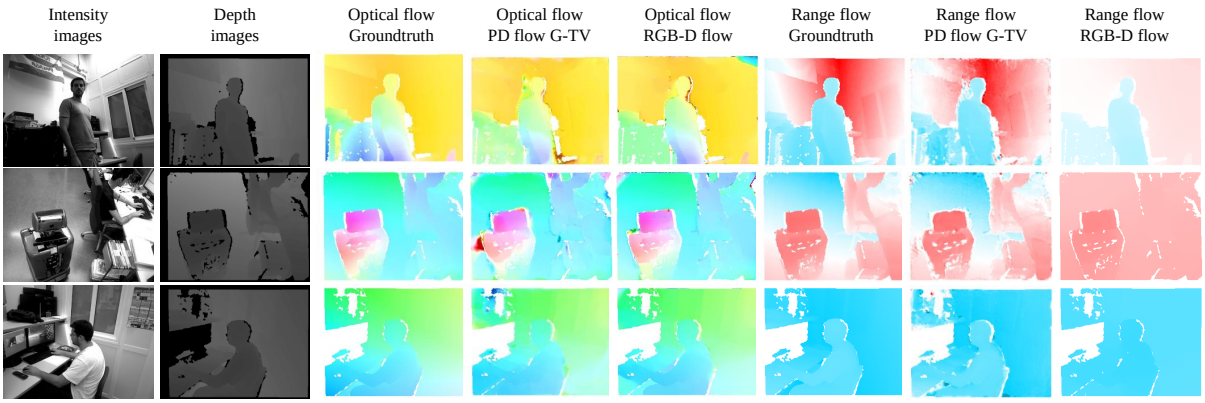


Figure 4.3: Colour representation of the optical and range flows for the "person standing" frames (first row), "robot top" frames (second row) and "person desk" frames (third row) following the Middlebury colour scheme (for the range flow we only consider the colours of the x axis). The flow field is shown in white for the areas with null depth measurements.

RGB-D pair	MAX-V (meters)	NRMS-V PD-flow (TV_g)	NRMS-V PD-flow (TV)	NRMS-V RGB-D flow	AAE PD-flow (TV_g)	AAE PD-flow (TV)	AAE RGB-D flow
person standing	0.073	**0.069**	0.081	0.119	**7.433**	8.788	24.47
person desk	0.064	**0.068**	0.076	0.093	4.852	**4.402**	6.553
robot top	0.110	**0.061**	0.111	0.072	**8.078**	11.72	23.08
robot front	0.155	**0.064**	0.075	0.133	**7.823**	9.747	18.86
room	0.154	0.078	0.077	**0.062**	5.081	7.786	**4.947**
Average	0.111	**0.068**	0.084	0.096	**6.653**	8.489	15.58

Table 4.1: Quantitative evaluation of scene flow with five distinct RGB-D frame pairs.

Five pairs of RGB-D images have been chosen for the evaluation. These images have been generated according to the aforementioned procedure using distinct motion fields with maximum displacements ranging from 7 to 15 centimeters. Besides, these images emcompass information of realistic scenes with close and distant objects moving differently. As in similar works [108], two error measurements are compared: the 3D average angle error (AAE) expressed in degrees and the normalized root mean square error of the velocity magnitude (NRMS-V), where the maximum magnitude of the motion field (MAX-V) is used for normalization. Quantitative results are presented in Table 4.1, where two versions of the PD-flow with standard TV (4.13) and TV_g (4.18), along with the RGB-D flow, are evaluated. On the other hand, qualitative results are displayed in Figure 4.3, which shows that the RGB-D pairs generated for this evaluation contain varied and heterogeneous motion fields. In Figure 4.3 it can be noticed that RGB-D flow is slightly more accurate than our approach computing the optical flow, whereas PD-flow provides considerably better results for the range flow. Regarding the quantitative results, PD-flow with regularization on the 3D surface outperforms the other approaches. Overall, PD-flow is 50% more accurate than RGB-D flow estimating the norm of the motion field and more that 100% more accurate obtaining its direction in space.

As far as temporal performance is concerned, average runtimes are measured for both methods, including different CPU - GPU implementations of our work (Table 4.2). The test platform used is a standard desktop PC running Ubuntu 14.04 with an AMD Phenom II X6 1035T CPU at 2.6 GHz, equipped with an NVIDIA GTX 780 GPU with 3GB of memory. Table 4.2 shows

RGB-D flow	PD flow (CPU)	PD flow (GPU)
119.1s	7.150s	0.042s

Table 4.2: Runtime comparison.

that even the CPU implementation of PD-flow is one order of magnitude faster than RGB-D flow (it should be remarked that RGB-D flow uses GPU). With GPU acceleration, the execution time of our primal-dual scene flow is 2800 times faster than the RGB-D flow, hence reaching a maximum frame rate of 24 Hz. If 30 Hz are needed, the primal-dual solver can be stopped before convergence, providing results which are only 16% less accurate than those presented in Table 4.1.

Evaluation with real data in real time

In this subsection we show qualitative results of the scene flow working at a high frame rate (24 - 30 Hz). We have created two alternative representations of the flow field to illustrate its performance graphically. In the first case, scene flow was computed at 30 Hz (favoring speed) while two people were playing with a basketball. This experiment comprises fast movements, occlusions and heterogeneous and non-rigid motion fields associated to the different objects of the scene. A temporal sequence of pictures is shown in Figure 4.4, where the colour goes from grey/blue for still objects (null norm of the 3D velocity) to intense red for fast motions. It can be observed that the motion field is coherent with the scene: the arms and the ball get different red tones while the rest of the scene remains virtually still. Only the small regions at the background that are occluded by the ball in subsequent frames show a wrong (reddish) flow. It can even be noticed from the sequence that the maximum speed is reached at the moment of launching the ball, and this speed slightly decreases when the ball ascends, which is in accordance with the basic principle of energy conservation.

On the other hand, a distinct representation is displayed in Figure 4.5 to illustrate how the scene flow at 24 Hz (favouring precision) reproduces the real movements of a person. Two 3D point clouds are generated from consecutive depth images, being the first one shown in red and the last one in turquoise, and a vector field represents in blue the magnitude and direction of the estimated motion field. The three images in Figure 4.5 show how our primal-dual scene flow is able to estimate non-homogeneous movements accurately (hand movements in the left images are significantly different than the movement of the body). In contrast to most existing

Figure 4.4: Temporal sequence of two people throwing and receiving a basketball. The magnitude of the motion field is represented by colours ranging from grey/blue (null velocity) to intense red (fast motion).

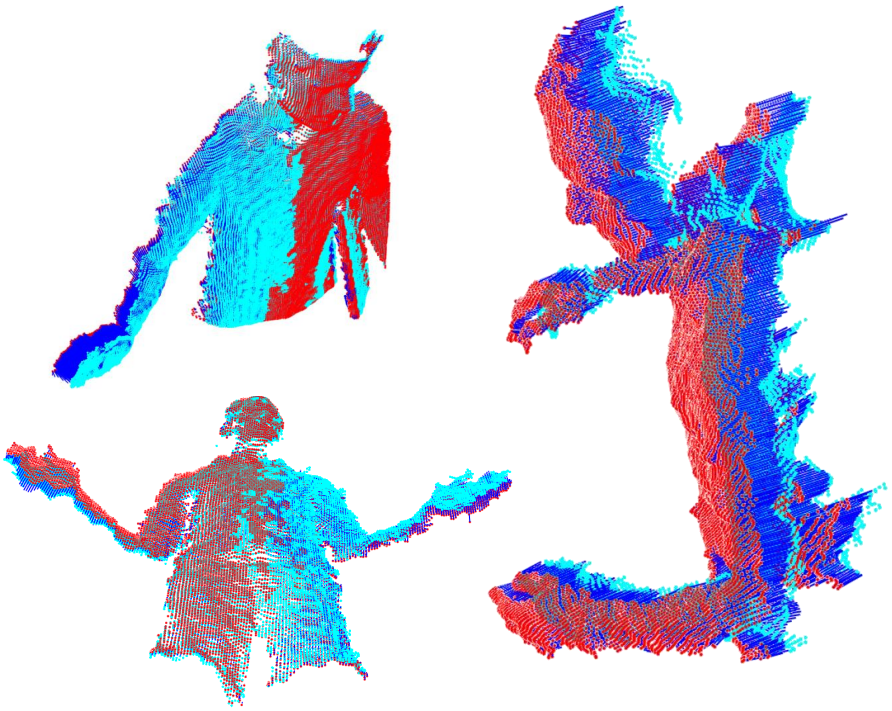


Figure 4.5: 3D representation of the estimated motion field for non-rigid movements of a person. The initial and final point clouds are shown in red and turquoise respectively, and the 3D flow is depicted with blue lines.

approaches, we do not need to handle very large displacements since our approach can be executed at a high frame rate, and hence the changes between consecutive images are drastically smaller for a given motion of the scene. For instance, at a frame rate of 24 Hz and considering a maximum displacement of 15 centimeters between frames (see Table 4.1) movements faster than 3 meters per second can be estimated properly.

4.D.6 Conclusions

A novel scene flow algorithm for RGB-D cameras has been presented. Within a variational framework, geometric data from depth images is exploited to obtain more accurate results: TV regularization is applied over the observed 3D surface and a geometry-dependent formula is proposed to approximate the image gradients. In order to minimize our functional, a highly parallelizable primal-dual solver is proposed and implemented on GPU to achieve real-time performance. The runtime of our algorithm is between two and three orders of magnitude faster than a previous work in scene flow for RGB-D cameras, which makes it suitable for robotic applications that require real-time processing. Quantitative and qualitative results are presented to demonstrate the robustness and accuracy of our approach. The code is available online under an open-source license.

4.E Motion Cooperation: Smooth Piecewise Rigid Scene Flow from RGB-D Images

Mariano Jaimez, Mohamed Souiai, Jörg Stückler,
Javier Gonzalez-Jimenez and Daniel Cremers

Published in Proc. International Conference on 3D Vision (3DV), 2015.

Motion Cooperation: Smooth Piecewise Rigid Scene Flow from RGB-D Images

Mariano Jaimez, Mohamed Souiai, Jörg Stückler,
Javier Gonzalez-Jimenez and Daniel Cremers

Abstract

We propose a novel joint registration and segmentation approach to estimate scene flow from RGB-D images. Instead of assuming the scene to be composed of a number of independent rigidly-moving parts, we use non-binary labels to capture non-rigid deformations at transitions between the rigid parts of the scene. Thus, the velocity of any point can be computed as a linear combination (interpolation) of the estimated rigid motions, which provides better results than traditional sharp piecewise segmentations. Within a variational framework, the smooth segments of the scene and their corresponding rigid velocities are alternately refined until convergence. A K-means-based segmentation is employed as an initialization, and the number of regions is subsequently adapted during the optimization process to capture any arbitrary number of independently moving objects. We evaluate our approach with both synthetic and real RGB-D images that contain varied and large motions. The experiments show that our method estimates the scene flow more accurately than the most recent works in the field, and at the same time provides a meaningful segmentation of the scene based on 3D motion.

4.E.1 Introduction

Scene flow estimation has many applications such as human body pose tracking, articulated object modelling for virtual/augmented reality or traffic scene understanding. In many scenarios, the dynamic scene is composed of rigid parts: human or animal bodies, man-made articulated objects, cars or other vehicles, etc. Many existing methods that work on scene flow do not completely exploit this aspect, and estimate motion fields that are only locally rigid or not rigid at all. Other methods do segment the scene to impose rigidity or strong regularization over the regions (or segments). However, these segmentations are only used as tools to improve the accuracy of the estimates, and do not really correspond to the independent motions of the scene (e.g. [114] partitions the scene into depth layers, [13] divides the scene into piecewise planar regions). Therefore, the segmentation-from-motion problem, which can be particularly useful for scene understanding or human-machine interaction, is not truly addressed by these methods.

On the other hand, assuming purely rigid motions is a strong restriction that is barely fulfilled in organic shapes. When a person moves, there are parts of their body moving rigidly (e.g. upper and lower arms or legs) and others which are transitions between the rigid ones (e.g. the neck). Besides, rigid motions within a fine-grained articulated structure may not be observable with the limited resolution of a camera. For these reasons, a sharp segmentation will never be able to estimate the motion of life beings or some other inanimate objects with exactitude.

In our method, we leverage the natural rigid-part decomposition by allowing for smooth continuous transitions between the parts. We formulate the problem of retrieving a smooth segmentation along with the motion estimates of the rigid parts, where each rigid part is assigned an independent 6-DoF motion. To this end, we solve a non-convex optimization problem by means of coordinate descent consisting of a motion estimation step (in the fashion of visual odometry) and a subsequent variational multilabelling solver. By using a weighted quadratic regularizer

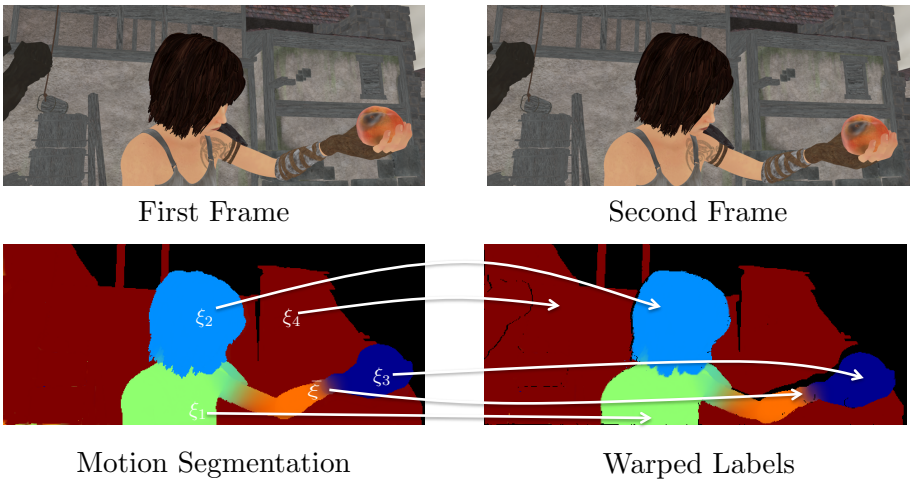


Figure 4.6: The proposed method is based on a motion interpolation model, which allows the emergence of smooth transitions between the segments where the motion is given by a convex combination of adjacent rigid motions (e.g. in $\bar{\boldsymbol{\xi}}$).

over the discontinuity-preserving total variation (TV), we promote smooth transitions between motion models rather than a harsh competition. For this reason, we refer to this approach as *motion cooperation* as opposed to the traditional *motion competition*. We evaluate our motion cooperation scene flow (MC-Flow) algorithm with synthetic and real RGB-D image pairs, and compare it with state-of-the-art approaches. In all cases, our approach achieves a superior performance both qualitatively and quantitatively. Furthermore, this evaluation demonstrates that the combination of a convex relaxation labelling with quadratic regularizer is superior to a sharp traditional segmentation because it naturally relaxes the overly constraining assumption of piecewise rigidity. Additionally, we show that our method retrieves meaningful soft segmentations into rigid parts as depicted in Figure 4.6.

Related Work

Scene flow estimation has been traditionally investigated in the multi-view stereo setting within the computer vision community. Vedula *et al.* [96] proposed one of the first methods based on the optical and range flow constraints. This approach was later extended to regularize the flow field using quadratic [98] and TV regularization [99, 100], the latter optiziming for disparity and flow jointly. In [101], disparity and scene flow estimation were decoupled to achieve real-time performance with a stereo camera system. The approach in [102, 13] oversegments the image into superpixels, assumes the superpixels to cover planar regions, and estimates a rigid-body motion for each superpixel individually. In [13], the planar motion of a superpixel acts as a regularization constraint on the scene flow of the individual pixels. Recently, with RGB-D cameras, the scene flow estimation topic has received further attention due to the availability of depth images at high frame rate. Herbst *et al.* [95] used the L_1 norm on a data term derived from the optical and range flow constraint equations and showed good qualitative results. Jaimez *et al.* [12] devised the first real-time dense scene flow for RGB-D images. A more natural TV regularization for the flow was proposed, where the regularization term minimizes the line integral of the scene

flow gradients over the observed 3D surface. Quiroga *et al.* [115] overparametrize scene flow and estimate a 6-DoF rigid-body motion at each pixel. They regularize the flow in this 6-DoF parametrization such that their model favours locally rigid motions. Hornacek *et al.* [109] also parametrize scene flow as a field of rigid motions, but propose to match corresponding points within a spherical search range instead of using traditional planar patches.

On the other hand, motion segmentation has also been studied in computer vision research. An early variational method for motion segmentation using optical flow constraints was proposed by Cremers and Soatto [9] in their work on *motion competition*. The name stems from the interpretation of the motion segments to compete for the boundaries through the best fit to their individual motion model. Several extensions to this method have been proposed, e.g. using non-parametric motions [116]. Unger *et al.* [117] explicitly model occlusions as an additional label in the multilabel optimization and impose a map uniqueness constraint to avoid ambiguous (non-bijective) data associations. All these methods are 2D and, hence, do not incorporate a 6-DoF motion model. Furthermore, they estimate a discrete segmentation.

3D-motion segmentation has only gained attention recently, mainly due to the current availability of GPUs and RGB-D cameras. Roussos *et al.* [118] propose a variational rigid-body motion segmentation and reconstruction method for monocular video. Zhang *et al.* [119] also pose 3D multi-body structure-from-motion in a variational framework. They require, however, a plane fitting step to make the method robust. Closely related to our method is the approach by Stückler and Behnke [120]. They jointly estimate motion and segmentation of rigid bodies in an expectation-maximization framework in RGB-D video. Each motion segment is assigned one rigid-body motion, but the approach does not interpolate between the motions of the segments. Recently, Sun *et al.* [114] proposed a probabilistic approach which makes use of a depth-based segmentation to estimate motion between RGB-D images. They regularize the estimation process by retrieving a mean rigid-body motion in each layer. This approach also does not explicitly model smooth transitions of motions between layers, but allows for small deviations of the motion field from the layer's mean motion.

Contributions

The MC-Flow algorithm is the first approach to perform joint soft-labelling and scene flow estimation by dissecting the scene into differently moving regions and their underlying motion. Our contributions are the following:

- Our algorithm estimates 3D motion based on a smooth piecewise rigidity assumption and simultaneously finds a soft motion-based segmentation of the scene.
- By choosing a suitable regularizer we are able to interpolate between rigid motions in order to recover non-rigidly moving parts and their underlying motion.
- An arbitrary (and previously unknown) number of rigid parts can be segmented automatically.
- MC-Flow outperforms state-of-the-art RGB-D scene flow algorithms qualitatively and quantitatively.

4.E.2 Problem Formulation

In this work, we assume that the scene can be segmented into n unknown distinct motion labels, each label standing for one rigid motion, as well as non-rigid parts which can be explained by neighbouring rigid motion labels. An illustration of such a smooth segmentation can be seen in Figure 4.6. As inputs, a pair of RGB-D frames (I_1, Z_1) and (I_2, Z_2) is given, where $I_{(.)} : \Omega \to \mathbb{R}$ and $Z_{(.)} : \Omega \to \mathbb{R}$ stand for the intensity and depth images defined on the image domain $\Omega \subset \mathbb{R}^2$. The segments and the rigid motions associated with them are obtained by minimizing a functional which depends on an implicit labelling function $u : \Omega \to [0,1]^n$, the 6-dimensional twist parametrizations $\boldsymbol{\xi}_i \in \mathbb{R}^6$ of the rigid motions and the number n of rigidly moving parts. The label assigment function u encodes the moving scene in the following way:

$$u_i(x) = \begin{cases} 1 & \text{if } x \in \Omega_i, \\ 0 & \text{if } x \notin \Omega_i, \\ (0,1) & x \text{ belongs partially to } \Omega_i \end{cases} . \tag{4.29}$$

Here we denote the i-th segment by $\Omega_i \subset \Omega$, which moves with a velocity $\boldsymbol{\xi}_i$. Note that, in order to permit fuzzy assignments, the label functions u_i can take on values in the interval $[0,1]$, in contrast to classical label assignment problems and their underlying binary representation.

The general problem of jointly solving for motion segmentation and motion estimation can be stated as the following optimization problem:

$$\begin{aligned} E_m(\xi, u, n) = & \int_\Omega G(\xi, I_1, I_2, Z_1, Z_2, u, n)\, dx + R(u, n) \\ & s.t. \sum_{i=1}^{n} u_i(x) = 1,\ u_i(x) \geq 0 \quad \forall x \in \Omega\,. \end{aligned} \tag{4.30}$$

where non-bold ξ refers to the whole set of rigid velocities for $i = 1, 2, \ldots n$. The function G encodes geometric and photometric consistency between the RGB-D images according to a linear combination of rigid-body motions:

$$\begin{aligned} G(\xi, I_1, I_2, Z_1, Z_2, u, n) = & F\left(I_1(x) - I_2\left(W(x, \bar{\boldsymbol{\xi}})\right)\right) \\ & + \alpha\, F\left(\left|T(\bar{\boldsymbol{\xi}})\, \pi^{-1}(x, Z_1(x))\right|_z - Z_2\left(W(x, \bar{\boldsymbol{\xi}})\right)\right) , \end{aligned} \tag{4.31}$$

with

$$\bar{\boldsymbol{\xi}} = \sum_{i=1}^{n} u_i(x)\, \boldsymbol{\xi}_i\,, \qquad W\left(x, \boldsymbol{\xi}\right) = \pi\left(T(\boldsymbol{\xi})\, \pi^{-1}(x, Z_1(x))\right) , \tag{4.32}$$

and $|\bullet|_z$ meaning the z-coordinate. The warping function $W(x, \boldsymbol{\xi})$ involves a projection π which transforms the 3D coordinates of the observed points into pixel coordinates, and $T(\boldsymbol{\xi}) \in \mathrm{SE}(3)$ is the homogeneous transformation associated to the twist $\boldsymbol{\xi}$. The function F in (4.31) is a robust kernel that measures photometric / geometric consistency. In order to obtain a compact labelling, we regularize the labels by imposing a smoothing term $R(u, n)$ in (4.30). Note that problem (4.30) is hard to minimize because the labels are non-linearly involved in the non-convex data term G. To the best of our knowledge, except for performing complete search

on u, which is unfeasible in our application, there is no direct way of tackling problem (4.30). Consequently, we consider a simpler formulation where the labels are pulled out of the data term. This significantly facilitates the optimization process because the label assignment function u is now linearly involved with the data term:

$$E_r(\xi, u, n) = \int_\Omega \sum_{i=1}^{n} u_i \, D(\boldsymbol{\xi}_i, I_1, I_2, Z_1, Z_2) \, dx + R(u, n)$$
$$s.t. \sum_{i=1}^{n} u_i(x) = 1, \; u_i(x) \geq 0 \quad \forall x \in \Omega \, . \tag{4.33}$$

The data fidelity term D_i is now evaluated for every independent rigid motion:

$$D(\boldsymbol{\xi}_i, I_1, I_2, Z_1, Z_2) = F\left(I_1(x) - I_2(W(x, \boldsymbol{\xi}_i))\right)$$
$$+\alpha \, F\left(\left|T(\boldsymbol{\xi}_i)\, \pi^{-1}(x, Z_1(x))\right|_z - Z_2(W(x, \boldsymbol{\xi}_i))\right) \, . \tag{4.34}$$

The optimization problems (4.30) and (4.33) would be equivalent if the labels u were binary. The main difference between the two models is that in (4.30) the motions are interpolated and subsequently used to evaluate the residuals with the exact velocities, whereas in (4.33) the residuals are computed for each independent rigid motion and interpolated afterwards. With binary labels there would not be interpolation between motions or residuals and, hence, both models would turn out to be the same. In this work we aim to solve the motion interpolation model (4.30) but, given its complexity, we resort to the simpler model (4.33) as an approximation of (4.30) to optimize for the labels. For this reason, the regularization term $R(u, n)$ plays a crucial role to estimate accurate interpolated motions at the transitions between rigid bodies/parts.

Overall Optimization

Irrespective of which of the two models we chose, the data terms are nonlinear with respect to the rigid motions. Therefore, the overall optimization problem is not convex and the global minimum cannot be guaranteed to be found.

To tackle this joint problem, we propose a coordinate descent strategy that alternates between estimating the motions for a fixed set of labels and then refining these labels for the recently obtained velocities, as illustrated in Algorithm 3. The motions are computed in the fashion of a visual odometry problem, but considering that the whole scene is not rigid but smooth-piecewise rigid. The labels are solved using the approximate model (4.33) that is convex in u. Note that we are implicitly optimizing for the label count n by adapting the number of labels within the inner iterations, as will be described in §4.E.5. Next, we elaborate on how to solve the main two subproblems in Algorithm 3.

Algorithm 3 Coordinate Descent Optimization

Initialize u^0
for $k = 0, 1, 2, ...$
1: $\xi^{k+1} = \arg\min_\xi E(\xi, u^k)$
2: $u^{k+1} = \arg\min_u E(\xi^{k+1}, u)$
3: Update n
end for

4.E.3 Motion Estimation

Given a precomputed set of labels, at every iteration of Algorithm 3 we need to estimate the rigid-body motions associated to each label (step 1). This problem can be considered as an extension of the well-known visual odometry (VO) problem. In this more general case, the whole scene is not supposed to be moving rigidly; instead, we assume that there are n predominant rigid motions that can be linearly combined to explain the motion of every point of the scene.

Our solution to estimate the motion of the segments builds upon two existing VO methods: DI-FODO [66] and the Robust Dense Visual Odometry [3]. This solution is obtained by minimizing the photometric and geometric residuals, defined as

$$r_I(x, \bar{\boldsymbol{\xi}}) = I_1(x) - I_2(W(x, \bar{\boldsymbol{\xi}}))\,, \tag{4.35}$$

$$r_Z(x, \bar{\boldsymbol{\xi}}) = \left|T(\bar{\boldsymbol{\xi}})\,\pi^{-1}(x, Z_1(x))\right|_z - Z_2(W(x, \bar{\boldsymbol{\xi}}))\,. \tag{4.36}$$

Note that the residuals are defined here according to the motion interpolation model (4.30). To cope with large motions, the process of minimization is applied in a coarse-to-fine scheme where the residuals are linearized at each level of the pyramid. In order to reduce the impact of outliers and to provide an accurate motion estimate, a robust function of the residuals is minimized:

$$\arg\min_{\xi} \left\{ \int_{\Omega} F(r_I) + \alpha\, F(r_Z)\, dx \right\}\,, \tag{4.37}$$

$$F(r) = \frac{c^2}{2}\, ln\left(1 + \left(\frac{r}{c}\right)^2\right)\,. \tag{4.38}$$

The function $F(r)$ is equivalent to the Cauchy M-estimator. Although we do not present comparisons in this regard, it was chosen because it provides considerably better results than other more common choices like the L_2 or L_1 norms. The parameter α balances the two kinds of residuals and c controls the relative weighting between high and low residuals. This minimization problem is solved using Iteratively Reweighted Least Squares (IRLS), where the associated weighting function is

$$w(r) = \frac{1}{1 + \left(\frac{r}{c}\right)^2}\,. \tag{4.39}$$

With this strategy, we are able to solve the motion estimation problem accurately. The minimization of both the photometric and the geometric residuals allows us to estimate the motion of the segments even if they lack either texture or geometric distinctive features. This aspect is crucial because the segments can be considerably small (compared to the whole scene) and might not present sufficient photometric or geometric data to solve the 3D registration problem using only one of these two input data.

4.E.4 Label Optimization

Once the motion ξ^{k+1} at a given iteration $k+1$ is obtained, we optimize the label assignment function as the second step of the overall optimization problem (Algorithm 3). For a fixed set of motions ξ, the functional $E(\xi^{k+1}, u)$ is convex and can be solved using state-of-the-art first-order solvers. In this work, the labelling function is optimized with the primal-dual algorithm

developed by Pock *et al.* [121]. Detailed information about how to apply this algorithm to the addressed problem is given in §4.E.9.

Two different regularizers are considered: total variation and quadratic regularization. Furthermore, the geometrical data that RGB-D cameras provide are exploited to regularize the labels according to the real 3D distances between points. Thus, regularizers are defined as a function of a weighted gradient ∇_r of the labels, whose weights (r_x) are the inverse of the 3D distances between the points:

$$\nabla_r u_i = \left(r_{x_1} \frac{\partial u_i}{\partial x_1}, r_{x_2} \frac{\partial u_i}{\partial x_2} \right) . \tag{4.40}$$

More details on the theory and the implementation of this regularization strategy can be found in [12].

Total Variation Regularization

Total variation was made popular by the seminal work of Rudin *et al.* (ROF) [122] on image denoising. This strategy is often advantageous because it permits the presence of discontinuities in the regularized variable, and can also be used to measure the perimeter of a region if applied on its indicator function. These factors made TV widely used in general reconstruction problems like image denoising [122], image deblurring [123] and image segmentation [124, 125]. In order to incorporate TV regularization into our approach, we simply set:

$$R(u, n) = \lambda \sum_{i=1}^{n} \int_{\Omega} \left\| \nabla_r u_i(x) \right\|_1 dx . \tag{4.41}$$

Quadratic Regularization

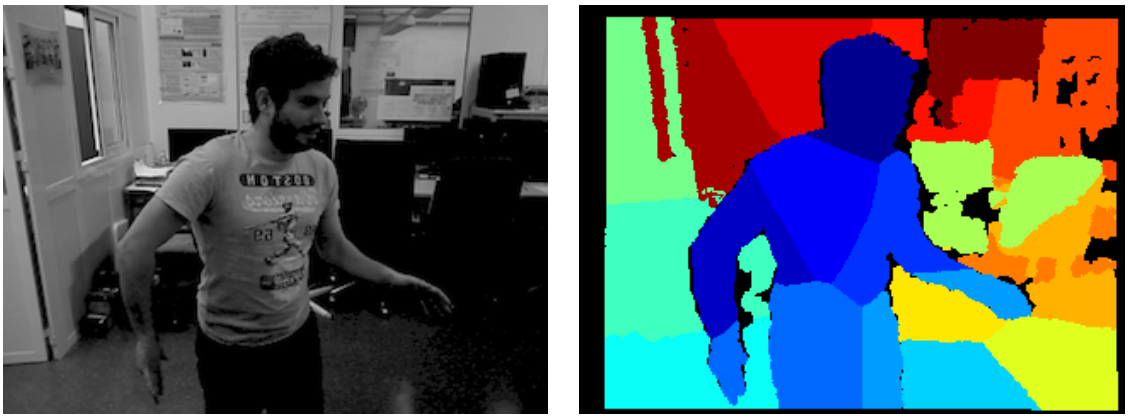
As previously mentioned, TV regularization favours sharp label boundaries. However, in our segmentation we would like to obtain a smooth interface between the regions (or at least between contiguous regions). Hence, a suitable choice to encourage smooth label transitions is the so-called *Tikhonov* or quadratic regularization:

$$R(u, n) = \lambda \sum_{i=1}^{n} \int_{\Omega} \left\| \nabla_r u_i(x) \right\|_2^2 dx . \tag{4.42}$$

Normally, quadratic regularization smoothes all the potential discontinuities in the solution, which does not help to provide a precise segmentation of the scene. However, the geometric weighting presented in (4.40) makes it possible to estimate discontinuities in the labels (for the different depth layers) and soft transitions between rigid parts at the same time.

4.E.5 Initialization and Adaptive Number of Labels

This section describes the adopted strategy to refine the number of labels n so that they represent the actual number of independent rigid motions in the scene. Since we are solving a non-convex problem, it is crucial to start with an initial set of labels u^0 that allows us to converge to the global optimum in Algorithm 3. Instead of including the number of labels in the variational formulation



a) Input image b) K-means initialization u^0

Figure 4.7: We initialize our algorithm by performing K-means on the 3D coordinates of the image pixels.

(which would significantly increase the computational burden), we propose to initialize the labels with a meaningful over-segmentation of the observed scene and iteratively remove those labels that are redundant or not significant for the overall motion estimation. To this end, we create an initial K-means segmentation based on the 3D coordinates of the points of the scene. The initial number of labels is always set to 20 (the number of independent rigid motions in the scene is assumed to be smaller than this quantity). An example of a K-means initialization is shown in Figure 4.7. The refinement of the label count is performed, after a full inner iteration of Algorithm 3, as follows:

- If labels i and j are associated to similar velocities, i.e., if $\|\boldsymbol{\xi}_i - \boldsymbol{\xi}_j\| \leq \delta$ for some small $\delta > 0$, we merge both labels.
- If a label i contains too few pixels, i.e., if $\int_\Omega u_i(x) < \gamma$ for some small $\gamma > 0$, we assign these pixels to the outlier label and remove label i.

4.E.6 Occlusions and Outliers

In our formulation, we include an outlier label (u_n) to capture pixels with null depth measurements and those other pixels that produce very high residuals for all the possible velocity candidates $\boldsymbol{\xi}_i$. To this end, a constant weight K_D is assigned to this label which, according to (4.33) means that $D_n = K_D$ in the whole image plane Ω. As previously mentioned, this outlier label also plays an important role in the process of reducing the number of labels. When a label is removed as a consequence of containing very few pixels, those few pixels need to be assigned to another label. If they were assigned to a wrong label they could affect the subsequent motion estimate and spoil the results. Conversely, if they are assigned to the outlier label, they don't participate in the motion estimation stage and are automatically assigned to the best label afterwards in the label optimization stage.

On the other hand, we detect occlusions to avoid the evaluation of the dataterm (D_i in (4.33)) for those pixels which are not visible in the second RGB-D frame. Occlusions are handled with a binary mask $O(x)$ instead of an extra label, in a way that occluded points can still be segmented

and, therefore, their 3D motion is estimated too. This can be accomplished by virtue of the regularization term, and allows us to provide a complete segmentation of the scene even if some points or areas are occluded after the motion.

In order to detect occlusions, two factors are considered: the amount of pixels of the first frame that are registered to each pixel of the second frame, and the temporal change in the depth images. First, we compute a cumulative function $C(x) : \Omega \rightarrow \mathbb{R}$ that counts how many pixels from the first frame are warped to the pixel x of the second frame (according to the estimated motion). Without occlusions, this function is approximately equal to 1 (or maybe inferior to one for new points appearing in the second frame), meaning that there is a one-to-one (bijective) correspondence between the observed points at both images. On the contrary, if $C(x)$ is noticeably higher than one, there are some pixels in the first frame that are warped to the same pixel x in the second frame, indicating the existence of occlusions. Consequently, we can define a function $O_C(x)$ that finds the pixels candidates for occlusion by applying a warping with the estimated motion and evaluating the cumulative function C:

$$O_C(x) = C(W(x, \bar{\boldsymbol{\xi}})) \,. \tag{4.43}$$

On the other hand, unlike in the optical flow problem, geometric information is available and can be exploited to reason whether a point is occluded or not. The simplest function that can be used to detect occlusions is the temporal change in depth:

$$O_Z(x) = Z_1(x) - Z_2(x) \,. \tag{4.44}$$

Combining these two functions we can detect most of the occluded areas in the scene by imposing a threshold K_o:

$$O(x) = \begin{cases} 1 & \text{if} \quad O_C(x) + K_z\, O_Z(x) > K_o \\ 0 & \text{else} \end{cases} , \tag{4.45}$$

where K_z is a parameter that weights O_Z against O_C. This strategy could be improved by embedding these functions into a variational formulation and imposing regularization over the occlusion mask. However, this has not been implemented in our work because it would significantly increase the runtime of the algorithm.

4.E.7 Experiments

In this section, qualitative and quantitative results are presented to evaluate the accuracy of our approach. These results are divided into two categories: scene segmentation and scene flow estimation. However, the evaluation process is not straightforward given the lack of benchmarks with either scene flow ground truth or segmentation from 3D motion. For this reason, we have selected a set of synthetic and real RGB-D frame pairs that contain varied and challenging motions. First, our approach is tested with some sequences from the Sintel dataset [126]. This dataset contains scenes with heterogeneous and large motions, and provides optical flow ground truth which can be used to measure the scene flow error. Second, the joint segmentation and motion estimation is generated for several RGB-D image pairs that either have been utilized in previous works in the literature (as in [115]) or have been taken with RGB-D cameras in

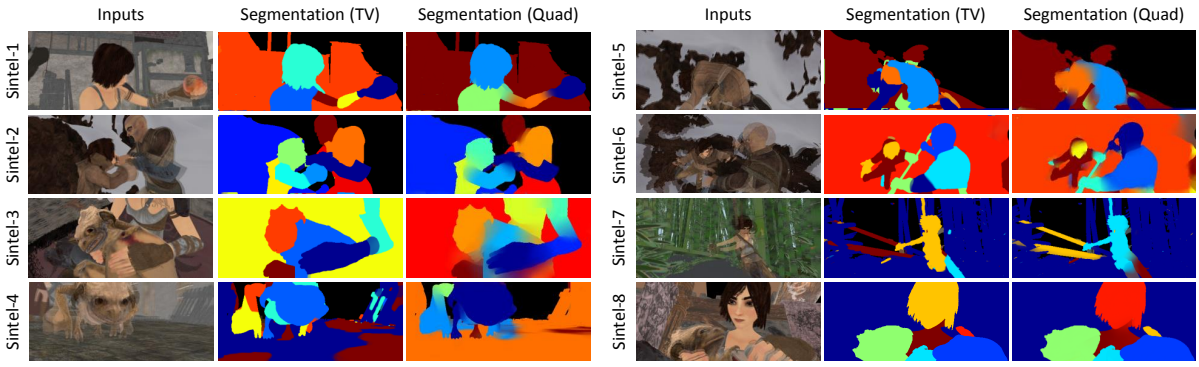


Figure 4.8: Segmentation estimated by our approach for the eight sequences of the Sintel dataset considered. Colours are independent for each result and do not depend on the associated rigid motion. Black represents the outlier label.

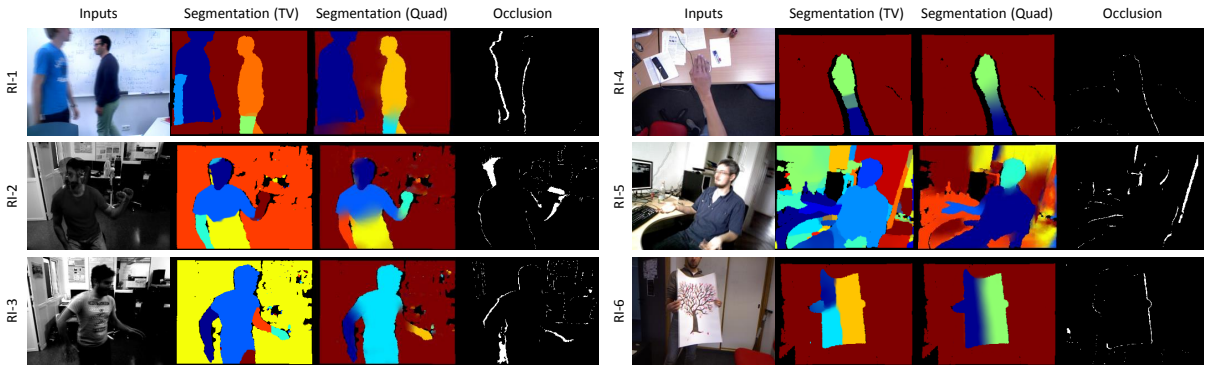


Figure 4.9: Segmentation and the occlusion layer estimated by our approach for 6 image pairs taken with RGB-D cameras. Colours are independent for each result and do not depend on the associated rigid motion. Black represents the outlier label.

our lab. In all cases, two versions of our method are tested, corresponding to the two different regularization strategies for the label optimization problem: total variation (TV) and quadratic regularization (Quad). The resolution adopted for the images is QVGA (240×320) for those taken with an RGB-D camera and 218×512 for the Sintel sequences. The maximum depth is set to 5 meters in all cases. Tests have been performed with a total of fourteen image pairs: eight from the Sintel dataset (named *Sintel-1...8*) and six real image pairs (named *RI-1...6*).

Scene Segmentation

In this subsection we present the motion segmentation that our method provides for all the tested sequences. The occlusion layer is also displayed for some sequences together with the segmentation although the occlusion is not a label itself (but a mask). Figure 4.8 shows the results for the Sintel images. It can be observed that TV produces very sharp labels with very few pixels interpolating between different motions. On the contrary, quadratic regularization gives rise to a smooth segmentation where many pixels adopt an interpolated velocity between two (or maybe more) rigid-body motions. The same behavior can be seen in Figure 4.9 where the results for the real RGB-D images are presented. In general, it can be noticed that the number

of labels to which the method converges is not the same for the two regularization strategies. Normally, TV produces a higher number of labels because it is not able to interpolate motions and tends to keep extra labels to compensate for it. Except for *Sintel-4* (with TV) and *RI-5*, the resulting segmentations represent quite accurately the different objects and rigid parts of the scenes.

Scene Flow Evaluation

For all the sequences, the scene flow is evaluated quantitatively and compared with three state-of-the-art methods: the Primal-Dual flow (PD-Flow) [12], the Semi-Rigid flow (SR-Flow) [115] and the Layered flow (L-Flow) [114]. First, the photometric and geometric residuals are computed by warping the intensity and depth images (respectively) according to the estimated flow. It is important to note that occluded pixels will show very high residuals even if the motion is accurately estimated for them, which considerably disturbs the error metrics (RMSE of the residuals). To overcome this limitation and to provide more precise comparisons, we compute the RMSE of the non-occluded pixels, which is a more reliable metric of the scene flow accuracy. To this end, we assume that the occlusion layer computed by our approach is sufficiently accurate and use it in all cases (neither PD-flow nor SR-flow detect occlusions). This does not represent any bias toward our method because it is a common mask applied to all of them, and if some occluded pixels have not been detected properly then they will affect the error metrics of all the compared methods equally. Table 4.3 shows the results for all the frame pairs. It can be observed that our method provides the most accurate estimates with both TV and quadratic regularization. The differences between TV and Quad are essentially caused by the way they produce transitions between the labels and the number of labels they converge to. As previously analyzed, TV generates a sharp segmentation where the motion is barely interpolated, whereas quadratic regularization provides smooth transitions between the labels that lead to larger areas with interpolated motions. On the other hand, TV tends to converge to a higher number of labels,

	Photometric residual - RMSE					Geometric residual - RMSE				
	PD-Flow	SR-Flow	L-Flow	MC-TV	MC-Quad	PD-Flow	SR-Flow	L-Flow	MC-TV	MC-Quad
Sintel-1	0.060	0.035	0.049	0.022	**0.021**	0.443	0.317	0.420	0.253	**0.186**
Sintel-2	0.057	0.068	0.063	0.026	**0.025**	0.086	0.090	0.108	0.056	**0.053**
Sintel-3	0.048	0.041	0.047	0.032	**0.028**	0.021	0.022	0.035	0.018	**0.017**
Sintel-4	0.091	0.069	0.109	0.063	**0.044**	0.378	0.347	0.607	**0.155**	0.190
Sintel-5	0.074	0.067	0.091	**0.051**	0.055	0.373	0.267	0.498	**0.203**	0.283
Sintel-6	0.120	0.118	0.127	**0.055**	**0.055**	0.224	0.190	0.253	0.114	**0.096**
Sintel-7	0.076	0.071	0.079	**0.035**	0.038	0.407	0.423	0.382	0.233	**0.188**
Sintel-8	0.063	**0.026**	0.045	0.028	0.027	0.083	0.069	0.086	0.038	**0.037**
RI-1	0.038	0.025	0.031	0.024	**0.022**	0.070	0.060	0.046	**0.038**	**0.038**
RI-2	0.032	0.028	0.035	0.021	**0.020**	0.286	0.259	0.294	0.114	**0.102**
RI-3	0.031	0.024	0.027	**0.018**	**0.018**	0.221	0.208	0.217	0.160	**0.145**
RI-4	0.015	0.012	0.011	**0.008**	**0.008**	0.025	**0.024**	0.025	0.025	0.025
RI-5	0.074	0.051	0.056	**0.039**	0.040	0.095	0.087	0.108	**0.079**	0.085
RI-6	0.077	0.050	0.070	0.049	**0.047**	**0.036**	0.038	0.037	0.041	0.040
Average	0.061	0.049	0.060	0.034	**0.032**	0.197	0.172	0.223	0.109	**0.106**

Table 4.3: Photometric and geometric residuals after warping the image pairs with the estimated scene flow.

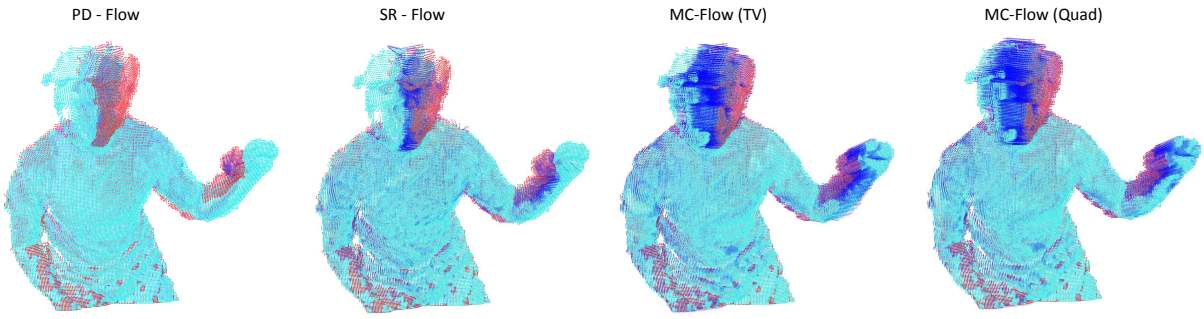


Figure 4.10: Comparison of the 3D motion fields estimated for the *RI-2* images. The initial frame is represented by the red point cloud, the final frame by the turquoise point cloud and the scene flow by the blue lines. The above comparison shows that our approach provides the most accurate estimate of the scene flow.

which helps to compensate for its inability to capture nonrigid motions. Overall, the best results are obtained with quadratic regularization, although the differences are small.

For the sake of clarity, Figure 4.10 is included to illustrate the 3D motion field that the compared methods estimate for the *RI-2* pair. PD-Flow, which was conceived to work in real time, is unable to estimate large motions and can only capture the motion of the body and the upper arms. SR-Flow provides better results but is still unable to reproduce the real motion of the hand and head. Only our approach estimates the whole motion field properly, specially with quadratic regularization of the labels.

Moreover, for the Sintel image pairs, we project the scene flow onto the image plane to obtain the optical flow and compare it with the ground truth provided by the Sintel dataset. In this case we evaluate two error metrics: the average end-point error (EPE) and the average angular error (AAE), as explained in [127]. Again, the results (Table 4.4) are computed for the non-occluded pixels, which is a fairer comparison given that some methods do not manage occlusions and hence provide bad estimates for the occluded areas. It can be seen that our approach with both TV and quadratic regularization clearly outperforms the others, providing a motion estimate that is between 2 and 5 times more accurate than those from the PD-Flow, SR-Flow and the L-Flow.

Regarding the computational performance, our method ranks second with a runtime of 30 seconds. For the experiments, we have utilized a standard desktop PC running Ubuntu 14.04 with an AMD Phenom II X6 1035T CPU at 2.6 GHz, equipped with an NVIDIA GTX 780 GPU with 3GB of memory. The measured runtimes are:

- PD-Flow: 0.042 seconds (GPU).
- SR-Flow: 150 seconds (CPU).
- L-Flow: 8 minutes (CPU).
- MC-Flow: 30 seconds (label optimization on GPU and all the remaining steps on CPU).

4.E.8 Conclusions

In this paper we have addressed the problem of joint segmentation and scene flow estimation from RGB-D images. The overall optimization problem is solved by means of a coordinate

	Optical flow - EPE					Optical flow - AAE				
	PD-Flow	SR-Flow	L-Flow	MC-TV	MC-Quad	PD-Flow	SR-Flow	L-Flow	MC-TV	MC-Quad
Sintel-1	1.940	0.684	1.320	0.221	**0.219**	27.87	7.694	13.26	**2.486**	2.827
Sintel-2	2.299	2.100	2.851	0.367	**0.324**	23.63	16.02	35.50	**4.826**	4.950
Sintel-3	1.223	1.130	0.975	0.383	**0.344**	31.69	20.21	20.80	8.364	**7.721**
Sintel-4	17.04	21.68	15.26	10.23	**3.436**	73.57	90.56	43.09	22.13	**9.694**
Sintel-5	4.381	3.990	3.212	2.316	**1.983**	24.27	26.14	10.43	14.56	**10.16**
Sintel-6	6.045	7.739	7.67	**1.168**	1.498	12.10	18.99	27.52	**3.845**	5.194
Sintel-7	2.875	3.335	3.382	**1.480**	1.591	26.50	21.26	22.48	**7.723**	8.169
Sintel-8	1.674	0.456	1.012	**0.228**	**0.228**	22.45	4.713	8.003	3.762	**3.757**
Average	4.685	5.142	4.461	2.049	**1.203**	30.26	25.70	22.63	8.462	**6.559**

Table 4.4: Average end-point and angular errors of the optical flow computed by projecting the estimated scene flow onto the image plane.

descent method which alternates between motion estimation and label optimization, while at the same time adapts the number of labels to the real number of independent rigid motions of the scene. Two different regularization strategies for the labels are employed, TV and quadratic, leading to sharp and smooth segmentations, respectively. Our method has been tested with both synthetic and real RGB-D image pairs, and the experiments show that joint segmentation and motion estimation provides very accurate results that outperform state-of-the-art scene flow algorithms on RGB-D frames. Comparisons between the two regularization strategies show that quadratic regularization estimates motion more accurately than TV because it generates smooth label transitions between rigid bodies, which models the scene motion more realistically. For future work, we plan to extend this work to RGB-D video streams where temporal regularization can be imposed.

4.E.9 Appendix: Primal-Dual Algorithm for the Label Optimization

As mentioned in §4.E.4, the labelling function u is optimized with the primal-dual algorithm developed by Pock et al [121]. To this end, we consider the following generic problem:

$$\min_{u} \{H(u) + Q(Ku)\} \tag{4.46}$$

where H and Q are given by:

$$\begin{aligned} H(u) &= \int_{\Omega} \sum_{i=1}^{n} u_i \, D_i(\boldsymbol{\xi}_i, I_1, I_2, Z_1, Z_2) \, dx + \delta(u) \,, \\ Q(Ku) &= R(u) \,. \end{aligned}$$

The linear operator K denotes the spatial gradient applied on the labelling. This operator is computed in the same fashion as in [12] where the authors incorporate geometric weights in order to regularize the motion field on the 3D world coordinates instead of on the image plane. The term $\delta(u)$ denotes the indicator function for the feasibility of u according to the constraint $\sum_{i=1}^{n} u_i(x) = 1$:

$$\delta(u) = \begin{cases} 0 & \text{if } \sum_{i=1}^{n} u_i(x) = 1, \\ \infty & \text{else.} \end{cases} \tag{4.47}$$

More specifically, the aforementioned algorithm tackles the problem

$$\min_{u} \max_{p} \quad \{\langle p, Ku \rangle + H(u) + Q^*(p)\} \tag{4.48}$$

and performs the iterative steps illustrated in Algorithm 4 in order to solve (4.48). The dual variable $p : \Omega \to \mathbb{R}^n$ is an auxiliary variable and is maximized during the optimization process. Function Q^* denotes here the Fenchel conjugate function of Q. The operators $(I + \tau \partial Q^*)^{-1}$ and $(I + \tau \partial H)^{-1}$ are the so-called *resolvents* and can be considered as generalized projectors [8]. Regarding the step sizes, we set them according to the diagonal preconditioning strategy explained in [121].

Algorithm 4 Primal-Dual Algorithm for minimizing $E(\xi^{k+1}, u)$

for $k = 0, 1, 2, ...$
1: $p^{k+1} = (I + \tau \partial Q^*)^{-1} \left(p^k + \tau K \bar{u}^k\right)$
2: $u^{k+1} = (I + \tau \partial H)^{-1}(u^k - \sigma K^T p^{k+1})$
3: $\bar{u}^{k+1} = u^{k+1} + \theta \left(u^{k+1} - u^k\right)$
end for

Proximal Operator for Total Variation Regularization

In our algorithm we incorporated TV regularization and we set:

$$R(u) = \lambda \left\| Ku(x) \right\|_1 = \sum_{i=1}^{n} \int_{\Omega} \left\| K_s \, u_i(x) \right\|_2 . \tag{4.49}$$

The operator K_s here represents the gradient operator for each of the label functions u_i. Regarding the computational cost of our method, applying TV has only impact on computing the resolvent $(I + \tau \partial Q^*)^{-1}$ in the primal-dual formulation (4.48) of our problem. This operation can be done in closed-form as follows:

$$\left(I + \tau \partial Q^*\right)^{-1} \left(\boldsymbol{p}(x)\right) = \lambda \frac{\boldsymbol{p}_i(x)}{\max(\left\| \boldsymbol{p}_i(x) \right\|, \lambda)}, \quad \forall i \in \{1, ..., n\} . \tag{4.50}$$

Proximal Operator for Quadratic Regularization

To obtain a smooth transitions between the labels, we have also tested quadratic regularization:

$$R(u) = \lambda \left\| Ku(x) \right\|_1^2 = \sum_{i=1}^{n} \int_{\Omega} \left\| K_s \, u_i(x) \right\|_2^2 . \tag{4.51}$$

The implementation of a quadratic regularizer only affects the evaluation of the proximal operator $(I + \tau \partial Q^*)^{-1}$. Like for TV, it can be done in closed form:

$$\left(I + \tau \partial Q^*\right)^{-1} \left(\boldsymbol{p}(x)\right) = \frac{\boldsymbol{p}_i(x)}{(\lambda + \tau)}, \quad \forall i \in \{1, ..., n\} . \tag{4.52}$$

Implementation of the Differential Operator K

For simplicity, we assume that in the discrete case $u_i \in X$ with $X = \mathbb{R}^{m \times c}$ (where m and c are the number of rows and columns, respectively), and $p_i \in X \times X$. The discrete linear operator K represents a geometrically weighted gradient that is applied pixel-wise over the image domain Ω. Hence, identical K_s operators are applied over each individual labelling functions u_i as follows:

$$K_s u_i = \nabla_r \, u_i = \left(r_{x_1} \frac{\partial u_i}{\partial x_1}, r_{x_2} \frac{\partial u_i}{\partial x_2} \right) \tag{4.53}$$

where (x_1, x_2) represent the pixel coordinates and the weighting functions r_{x_1}, r_{x_2} encode the inverse of the 3D distances between points observed by contiguous pixels (more details about these geometric functions can be found in §4.D.2). On the other hand, K^T represents the weighted divergence operator (inverted in sign). Similarly to K, it is composed of identical K_s^T operators applied pixel-wise to the individual dual variables $\boldsymbol{p}_i$:

$$K_s^T \boldsymbol{p}_i = -\nabla_r^* \cdot \boldsymbol{p}_i = -r_{x_1} \frac{\partial p_i^1}{\partial x_1} - r_{x_2} \frac{\partial p_i^2}{\partial x_2} \tag{4.54}$$

where (p^1, p^2) are the two components of the dual variables.

4.F Fast Odometry and Scene Flow from RGB-D Cameras based on Geometric Clustering

Mariano Jaimez, Christian Kerl, Javier Gonzalez-Jimenez and Daniel Cremers

Published in Proc. International Conference on Robotics and Automation (ICRA), 2017.

Fast Odometry and Scene Flow from RGB-D Cameras based on Geometric Clustering

Mariano Jaimez, Christian Kerl, Javier Gonzalez-Jimenez and Daniel Cremers

Abstract

In this paper we propose an efficient solution to jointly estimate the camera motion and a piecewise-rigid scene flow from an RGB-D sequence. The key idea is to perform a two-fold segmentation of the scene, dividing it into geometric clusters that are, in turn, classified as static or moving elements. Representing the dynamic scene as a set of rigid clusters drastically accelerates the motion estimation, while segmenting it into static and dynamic parts allows us to separate the camera motion (odometry) from the rest of motions observed in the scene. The resulting method robustly and accurately determines the motion of an RGB-D camera in dynamic environments with an average runtime of 80 milliseconds on a multi-core CPU. The code is available for public use/test.

4.F.1 Introduction

The joint estimation of the motion of a camera and the motion of the objects it observes is a problem of great interest with numerous applications in robotics, computer vision and beyond: tracking and mapping in dynamic scenarios, manipulation of fast-moving objects, or autonomous navigation are a few prominent examples. However, it is also a complex and computationally demanding problem that has not been properly solved yet. On the one hand, great progress have been made in visual odometry under the assumption of static or quasi-static environments [3, 38, 66], but the performance of these methods deteriorates when the number of pixels observing non-static parts becomes significant. On the other hand, scene flow (motion of the scene objects) is often estimated as the non-rigid velocity field of the observed points with respect to the camera relative position. This approach alone does not yield the camera motion because all points in the scene are treated equally and, therefore, static and non-static regions are indistinguishable when the camera moves. Moreover, the scene flow estimation tends to be computationally expensive, and most existing approaches require between several seconds and few minutes to align just a pair of images, which prevents them from being used in practice.

In this paper we present a new method to estimate both the motion of an RGB-D camera and the scene flow. Our approach relies on a two-fold segmentation of the scene. First, the scene is divided into geometric clusters by running K-Means on the 3D coordinates of the observed points. These clusters are treated as rigid bodies and are mostly exploited for the scene flow estimation, which greatly reduces its computational cost without sacrificing much accuracy. Second, the scene is also segmented into static and moving parts. The static regions (background) are used to derive the camera motion while the scene flow is estimated for the moving parts. To increase robustness, we propagate the background segmentation through time since static and moving parts of the scene are likely to be consistent along the image sequence.

We perform an extensive evaluation of our approach, comparing it with several state-of-the-art methods in visual odometry and scene flow estimation. Results show that our approach estimates the camera motion more accurately, in particular when the scene is highly dynamic, and ranks second in the scene flow evaluation. Above all, its main advantage is its significantly

lower runtime, of about 80 milliseconds running on multiple CPU cores at QVGA resolution (several orders of magnitude faster than most scene flow algorithms). For this reason, it can be applied online, a feature that existing approaches lack. The code, together with the demonstration video, can be found here:

`http://mapir.isa.uma.es/work/Joint-VO-SF`

4.F.2 Related Work

Thus far, visual odometry and scene flow estimation are two highly related problems that are usually addressed separately because of their intrinsic complexity. Though some joint solutions have been reported, they typically lack precision and efficiency. Next, we review some of the latest proposals for these problems.

Traditionally, visual odometry approaches have exploited sparse feature correspondences to estimate the camera motion [33]. While they are resilient to large numbers of outliers, they usually require optimization over multiple frames to achieve accurate camera localization. In general, these methods cannot provide a dense scene flow, but there exist extensions to estimate the motion of multiple rigid objects [128]. However, they require enough feature points to sufficiently constrain the motion, which is not guaranteed for objects projected as small regions on the image. With the advent of consumer RGB-D cameras, which provide dense depth maps at comparably high resolution, dense direct methods gained popularity. Typical cost functions penalize the intensity error [3, 48], inverse or direct depth error [129, 66], point-to-plane error [51], or an alternative error in the feature space [130]. The main difference to sparse, feature-based methods is that they do not require explicit correspondences, but rather compute and update them implicitly during the optimization. To achieve robustness against unmodelled effects, these approaches combine multiple cost functions and use robust penalties like Huber, Tukey, or Cauchy [54, 3]. This strategy works well if most of the scene is static and only little portions of the input images observe moving parts, but fails when the moving parts become more significant.

Scene flow has traditionally been estimated with stereo systems [101], but this trend also changed in 2010 with the appearance of affordable RGB-D cameras. Several variational approaches have been proposed [104, 95, 105] to compute scene flow from RGB-D image pairs, using different norms (weighted L_2 / L_1) for the data and the regularization terms. Jaimez *et al.* [12] presented a real-time implementation using a Primal-Dual solver, which provides good estimates only for small motions. The semi-rigid scene flow proposed in [115] uses a 6-DoF representation for the flow and also includes the camera motion into its formulation, which makes it the best candidate for comparisons. However, only the accuracy of the scene flow was evaluated in their paper. Sun *et al.* [114] presented a probabilistic approach which relies on a depth-based segmentation of the scene. They regularize the estimation process by retrieving a mean rigid-body motion for each layer, and allow for small deviations of the motion field from the layer's mean motion. A big downside is its high runtime: it requires several minutes to align just a pair of images. The smooth piecewise-rigid flow proposed in [131] achieves very accurate flow estimates by jointly segmenting the scene into its rigidly moving parts and computing their

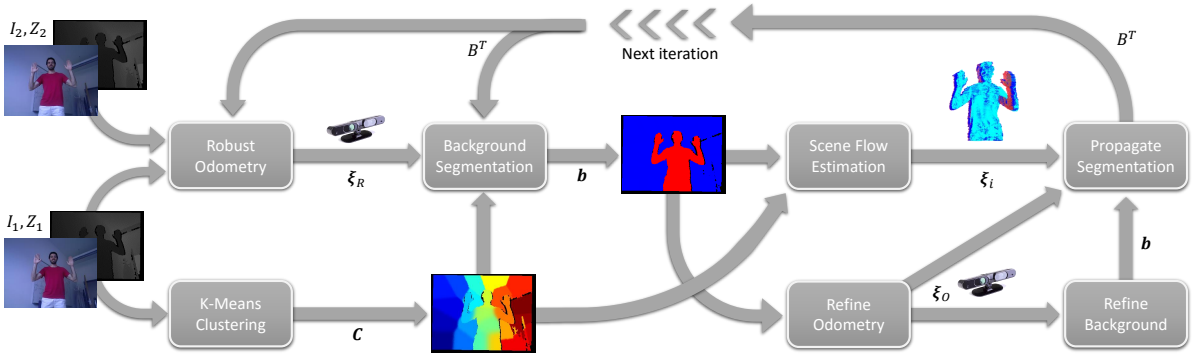


Figure 4.11: Schematic representation of the main components of our algorithm. The estimation process starts (left) by reading a pair of RGB-D images. After the process finishes, the background segmentation $\boldsymbol{b}$ is propagated to the next frame according to $\boldsymbol{\xi}_i$ and $\boldsymbol{\xi}_O$ to be used in the next iteration.

underlying motions. However, optimizing for the segments makes it also computationally very expensive. Other approaches focus on the estimation of large motions. SphereFlow [109] also parametrizes the motion field with 6 DoF, and proposes to match corresponding points within a 3D spherical search range instead of using traditional planar patch comparisons. GraphFlow [132] outperforms SphereFlow by looking for and registering sparse correspondences between points in geometric edges, and densify the flow at a later optimization stage. None of these two methods [109, 132] provide information about their runtime.

Recent works on non-rigid 3D reconstruction also estimate the camera motion and a deformation flow for the particular objects they reconstruct. DynamicFusion [91] estimates the camera motion using KinectFusion [51], which is not prepared to handle moving objects. Afterwards, they estimate a set of sparse volumetric rigid transformations to align the moving object to the model, and interpolate between these transformations to obtain the dense warp-field. VolumeDeform [92] follows a similar sequence, estimating the camera motion by aligning sparse color correspondences and computing the dense deformation field associated to the deformable object at the finest resolution. They both achieve impressive results for moderate camera motions and deformations, but unfortunately their code is not available. Fusion4D [133] addresses the same problem using multiple static cameras, and therefore only a nonrigid motion field is estimated (odometry is not necessary).

4.F.3 Overview of the Method

The proposed method to jointly estimate the camera and the scene motions from RGB-D sequences comprises several sequential blocks, as illustrated in Figure 4.11. As inputs, a pair of I-D frames (I_1, Z_1) and (I_2, Z_2) is given, where $I_{(.)} : \Omega \to \mathbb{R}$ and $Z_{(.)} : \Omega \to \mathbb{R}$ stand for the intensity and depth images defined on the image domain $\Omega \subset \mathbb{R}^2$. First, the frame (I_1, Z_1) is segmented into N geometric clusters $\boldsymbol{C} = \{C_i,\, i = 1, ..., N\}$ by applying K-Means to the 3D coordinates of the scene points. Each cluster is considered to behave as a rigid body, which greatly simplifies the scene flow estimation because the motion is estimated cluster-wise instead of pixel-wise (reducing the number of unknowns by 3-4 orders of magnitude). The velocity associated to the each cluster is represented by $\boldsymbol{\xi}_i \in \mathfrak{se}(3)$. Second, an initial guess for the

odometry $\boldsymbol{\xi}_R \in \mathfrak{se}(3)$ is computed by minimizing the photometric and geometric residuals within a robust penalty function. Thus, we obtain the predominant rigid motion of the scene which, save in cases of very dynamic environments, corresponds to the camera motion itself. Subsequently, this estimate is used to segment the scene into static parts (or background) and moving objects. To this end, the I-D images are warped according to $\boldsymbol{\xi}_R$ and the average residuals per cluster are computed. Only the clusters moving according to $\boldsymbol{\xi}_R$ will have low residuals, and therefore will be segmented as background. Clusters with high residuals after the warping are those whose motion does not coincide with $\boldsymbol{\xi}_R$, and hence are tagged as moving objects. Instead of using a binary segmentation, which would require to set a sharp threshold on the residuals, we use a continuous representation and define $b_i \in [0, 1]$ as the probability of any cluster i to be a moving object. For simplicity, we will use $\boldsymbol{b}$ to refer to the segmentations of all clusters. More details about the background segmentation are given in §4.F.6.

Once the segmentation is known, it can be used to break the motion estimation process into two separate steps. First, all the clusters that have been tagged as background are used to obtain a more precise odometry $\boldsymbol{\xi}_O$, now excluding the non-static parts. Second, a piecewise rigid scene flow is estimated for the rest of the scene, assuming that each cluster behaves as a rigid body. Lastly, the background segmentation is recomputed with the new refined odometry and warped to the next frame, leading to $B^T : \Omega \to [0, 1]$. Since moving objects are likely to be moving for more than one frame, and the same applies to still parts, we make use of B^T in the next iteration to obtain segmentations that are consistent through time.

4.F.4 Geometric Clustering

Our proposal to reduce the complexity of a per-pixel estimation of the motion field is to cluster the scene points in sets that will be treated as rigid bodies. Other existing algorithms have employed this strategy as well to obtain a faster (and often more precise) scene flow either by using superpixels [134] or K-Means [114, 131]. We follow the idea presented in [131] and compute K-Means clusters based on the 3D coordinates of the observed points. This strategy is advantageous because:

- It has a physical ground, in the sense that close points in space are very likely to belong to the same rigid body.
- It is a very convenient representation when working with image pyramids (coarse-to-fine). Once the clusters are computed for a given image, they can be easily computed for the rest of the pyramid by using the spatial coordinates of the k-means that have already been obtained. This is efficient and also provides a consistent clustering throughout all the image levels.
- It is suitable to propagate information from one frame to the next because the cluster centers can be mapped efficiently from (Z_1, I_1) to (Z_2, I_2) with $\boldsymbol{\xi}_O$ and $\boldsymbol{\xi}_i$.

Two additional steps are performed after obtaining the clusters. First, we build a connectivity graph $G = \{G_{ij},\, i = 1, ..., N,\, j = 1, ..., N\}$ which indicates which clusters are contiguous in space ($G_{ij} = 1$) and which ones are separated ($G_{ij} = 0$). This graph is exploited later on for the background segmentation by fostering contiguous clusters to be segmented similarly (spatial

regularization). Moreover, contiguous clusters are smoothed to avoid sharp motion transitions along their boundaries, but this smoothing mostly affects the scene flow estimation and does not play any role in the other modules of our algorithm.

Another important aspect is the number of clusters to consider. Too few leads to very large scene regions which will likely include parts of the scene with different motions. On the other hand, if many clusters are extracted, then they will not contain enough points for their motion to be robustly estimated. We have empirically chosen to use 24 clusters per image, which leads to middle-size clusters that can be homogeneously initialized on the image domain (6×4). More elaborate strategies could be adopted in the future, like using an adaptive number of cluster to fuse redundant regions or to create new ones in areas with high residuals.

The main limitation of our approach is the assumption that each cluster behaves as a rigid body. Since the only criterion to form the clusters is the spatial proximity of points, there can always be clusters which actually contain points from different rigid bodies. For instance, if a person stands close to a wall, there might be some clusters containing points from both the person and the wall. In this case, the estimated motion for those clusters will be the predominant motion between the two (person or wall). This limitation could be alleviated or even solved by increasing the number of cluster and regularizing the motion between them, at the expense of a significantly higher computational cost.

4.F.5 Robust Odometry

The odometry is computed by minimizing the photometric and geometric residuals between consecutive RGB-D pairs. The geometric residuals r_Z and the photometric residuals r_I are defined as

$$r_Z^k(\boldsymbol{\xi}) = Z_2(W(x^k, \boldsymbol{\xi})) - \left|T(\boldsymbol{\xi})\,\pi^{-1}(x^k, Z_1(x^k))\right|_z \ , \tag{4.55}$$

$$r_I^k(\boldsymbol{\xi}) = I_2(W(x^k, \boldsymbol{\xi})) - I_1(x^k) \ , \tag{4.56}$$

where x represents a given pixel of the image (the superscript k is omitted from here on) and $|\bullet|_z$ denotes the z-coordinate of a 3D point. The function $\pi : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ projects 3D points onto the image plane according to the pinhole model, and $T(\boldsymbol{\xi}) \in \mathrm{SE}(3)$ is the homogeneous transformation associated to $\boldsymbol{\xi}$. The warping function is given by:

$$W(x, \boldsymbol{\xi}) = \pi\left(T(\boldsymbol{\xi})\,\pi^{-1}(x, Z_1(x))\right) \ . \tag{4.57}$$

We formulate a dense optimization problem to obtain the camera motion, and compute the Cauchy M-estimator of the residuals:

$$\boldsymbol{\xi}_R = \arg\min_{\boldsymbol{\xi}} \left\{ \sum_{k=1}^{M} \left[F(w_R^k r_Z^k(\boldsymbol{\xi})) + F(\alpha_I w_R^k r_I^k(\boldsymbol{\xi})) \right] \right\} \ , \tag{4.58}$$

$$F(r) = \frac{c^2}{2} \log\left(1 + \left(\frac{r}{c}\right)^2\right) \ , \tag{4.59}$$

where M is the number of pixels in Z_1 with non-null depth. The Cauchy M-estimator represents a good compromise between robustness and basin of convergence, since it is much more robust

than L_2 / L_1 norms but never gets flat like the Tukey's biweight function. The parameter α_I weights the photometric and the geometric terms. The parameter c marks the inflection point of F, and can be tuned to make the estimation more or less robust against high residuals. Furthermore, a pre-weighting (w_R) is applied to help the solver converge to the true camera motion:

$$w_R(x) = Z_1(x)(1 - B^T(x)) \ . \tag{4.60}$$

The pre-weighting has a two-fold function. First, it downweights pixels of the clusters which were previously segmented as moving objects (remember that $B^T(x)$ encodes the probability of pixel x to have been moving in the previous frame). Second, it gives more significance to distant points which are more likely to observe static parts of the scene.

Since (4.58) is highly non-linear, the motion is solved within a coarse-to-fine scheme, linearizing the residuals at every level of the image pyramid, as done in [3, 66]. At each level, the solution of (4.58) is obtained via Iteratively Reweighted Least Squares.

4.F.6 Background Segmentation

In order to estimate both odometry and scene flow, we need to separate the static parts of the scene from the moving ones. This would be straightforward if the camera was still, but when the camera moves every region of the scene is in apparent motion and, hence, static and non-static objects become hard to distinguish. To identify them, we propose to use the robust odometry $\boldsymbol{\xi}_R$ (previously computed) to check which regions/clusters follow this pattern of motion and which do not. This evaluation is not performed pixel-wise but cluster-wise, since we assume that all pixels in a cluster have the same rigid body motion.

Initially, the RGB-D frames are warped according to $\boldsymbol{\xi}_R$. After the warping, clusters belonging to the background will have low photometric and geometric residuals, whereas the residuals associated to moving objects will still be high. In theory, this criterion should suffice to segment the scene into static and non-static parts, but in practice the process is much more complicated because residuals are not always a good metric to evaluate precise image alignment:

- Intensity and depth images are never registered perfectly. This means that some pixels of the background (close to object boundaries) tend to get the color of the foreground and vice versa. Thus, some clusters could be perfectly aligned and still bear high residuals due to this misregistration of color and depth.
- Occluded pixels will always exhibit high residuals even if the images are perfectly aligned.
- Since the depth measurement error grows quadratically with depth, the geometric residuals of distant clusters tend to be much higher than those of clusters close to the camera.

To cope with these issues, the background segmentation is divided into two steps. First, we compute a robust metric of the residuals per cluster ($\boldsymbol{\delta}$). Second, we formulate a minimization problem to obtain the segmentation of the clusters $\boldsymbol{b}$ based on their average residuals, their geometry and their previous segmentations $\boldsymbol{b}^T$ ($\boldsymbol{b}^T$ is computed by averaging $B^T(x)$ per cluster).

The robust average residuals are computed as:

$$\delta_i = \frac{\sum_{k=1}^{S_i - O_i} \alpha_I r_I^k + r_Z^k / \bar{Z}_i}{S_i - O_i} \ , \tag{4.61}$$

where S_i is the size of the i cluster, O_i is the number of occluded pixels in the cluster (which are excluded from the computation of δ_i) and $\bar{Z}_i$ is the average depth of the cluster. The occluded pixels are considered to be those with geometric residuals below a certain threshold, that is, a pixel x is occluded if

$$r_Z(\boldsymbol{\xi}, x) < -\Delta Z_{occ} \ . \tag{4.62}$$

Next, we formulate a minimization problem to obtain the background segmentation. The energy to be minimized is composed of four terms:

$$E(\boldsymbol{b}) = E_D(\boldsymbol{b}, \boldsymbol{\delta}) + E_R(\boldsymbol{b}) + E_T(\boldsymbol{b}, \boldsymbol{b}^T) + E_Z(\boldsymbol{b}, \bar{\boldsymbol{Z}}) \ , \tag{4.63}$$

where $\boldsymbol{b}$ is the only unknown (dependencies with $\boldsymbol{\delta}$, $\boldsymbol{b}^T$ and $\bar{\boldsymbol{Z}}$ are shown for clarity). The dataterm E_D pushes the clusters to be segmented as background when their residuals are low, and vice versa. To this end, we need to define a mapping between $\boldsymbol{\delta}$ and $\boldsymbol{b}$, and specify thresholds for low and high residuals (δ_L and δ_H respectively). For the sake of simplicity, we employ the following piece-wise linear function:

$$g(\delta_i) = \begin{cases} 0 & \delta_i < \delta_L \\ (\delta_i - \delta_L)/(\delta_H - \delta_L) & \delta_L \leq \delta_i \leq \delta_H \\ 1 & \delta_i > \delta_H \end{cases} \ , \tag{4.64}$$

and define the dataterm E_D as

$$E_D(\boldsymbol{b}, \boldsymbol{\delta}) = \sum_{i=1}^{N} w_D(\delta_i) \left(b_i - g(\delta_i)\right)^2 \ , \tag{4.65}$$

with

$$w_D(\delta_i) = \sqrt{\left(\frac{2\delta_i - (\delta_H + \delta_L)}{\delta_H - \delta_L}\right)^2 + 1} \ . \tag{4.66}$$

The function $w_D(\delta_i)$ increases the weight of the dataterm when the residuals are far from the area of uncertainty ($\delta_L < \delta_i < \delta_H$), giving more strength to clusters which are clearly either part of the background or a moving object.

The regularization term E_R tries to force neighbouring clusters to get a similar segmentation, and is defined as

$$E_R(\boldsymbol{b}) = \lambda_R \sum_{i=1}^{N} \sum_{j=i+1}^{N} G_{ij} \left(b_i - b_j\right)^2 \ . \tag{4.67}$$

We have chosen to minimize a quadratic term in (4.67) because it helps to smooth occasional wrong labellings of clusters with misleading residuals. We have also tried to minimize the absolute value of the differences (total variation), which allows for sharp discontinuities between connected clusters, but it did not provide better results.

Temporal regularization (E_T) is also imposed, since both static and dynamic parts of the scene are very likely to remain still and moving (respectively) through time:

$$E_T(\boldsymbol{b}, \boldsymbol{b}^T) = \lambda_T \sum_{i=1}^{N} (b_i - b_i^T)^2 \ . \tag{4.68}$$

Lastly, we include an extra term that introduces a bias towards the background ($b_i \to 0$) for all the clusters which are far from the camera. This models the fact that, in indoor scenarios, moving objects tend to be at the foreground while distant observations are likely to capture the fix elements of the environment (walls, ceiling, floor, furniture, etc.).

$$E_Z(\boldsymbol{b}, \bar{\boldsymbol{Z}}) = \lambda_Z \sum_{i=1}^{N} \max\left(0, e^{\bar{Z}_i} - e^{Z_{Min}}\right) b_i^2 \ . \tag{4.69}$$

Since all the terms in $E(\boldsymbol{b})$ are squares with respect to $\boldsymbol{b}$, the optimization problem (4.63) is convex and its solution can be obtained in closed form. Detailed information about the parameters introduced here is given in §4.F.8.

4.F.7 Scene Flow Estimation and Odometry Refinement

Once the scene is segmented, we divide the motion estimation into two separate processes. All the clusters segmented as background will be considered as a single rigid block and used to re-estimate the odometry. On the other hand, the rigid body motions of the moving clusters $\boldsymbol{\xi}_i$ will be computed independently. Knowing $\boldsymbol{\xi}_i$, the scene flow $\boldsymbol{m}(x)$ associated to each point $\boldsymbol{p}(x)$ of the cluster i is calculated as

$$\boldsymbol{m}(x) = \left(T(\boldsymbol{\xi}_i) - I\right) \boldsymbol{p}(x) \ . \tag{4.70}$$

Since $\boldsymbol{b}$ are continuous in the range $[0, 1]$, we need to create a partition of that interval to separate static and moving objects. Instead of imposing a simple binary threshold at 0.5, we consider the following three regions:

- If $b_i < 1/3$, the cluster i is assumed to be static and is only utilized for the odometry estimation.
- If $b_i > 2/3$, the cluster i is assumed to be moving and is only utilized for scene flow estimation.
- If $1/3 < b_i < 2/3$, the state of the cluster i is uncertain and therefore it is utilized for both the odometry and the scene flow estimation.

In this way, clusters that are not clearly segmented contribute the odometry estimation (because they could be background), but we also compute their own independent motion.

The rigid motions of the moving clusters are obtained by minimizing the following energy:

$$\boldsymbol{\xi}_i = \arg\min_{\boldsymbol{\xi}} \left\{ \sum_{k=1}^{S_i} \left[F(w_Z^k r_Z^k(\boldsymbol{\xi})) + F(\alpha_I w_I^k r_I^k(\boldsymbol{\xi})) \right] \right\} . \tag{4.71}$$

The final odometry $\boldsymbol{\xi}_O$ is computed similarly by minimizing (4.71) for the background pixels. These optimization problems are almost the same as the one described in §4.F.5; the only difference is the pre-weighting strategy. Once the scene is segmented, we use pre-weights that penalize occlusions and discontinuities, favouring smooth regions in the optimization process to find a precise solution:

$$w_Z = \frac{1}{K_Z + (\nabla_x Z_1)^2 + (Z_1 - Z_2)^2} \ , \tag{4.72}$$

$$w_I = \frac{1}{K_I + (\nabla_x I_1)^2 + (I_1 - I_2)^2} \ . \tag{4.73}$$

The weights w_Z and w_I penalize pixels where either the spatial or the temporal gradients are high. Although we do not provide an explicit comparison in the paper, this pre-weighting strategy provides better results than pure robust minimization without pre-weights, and also helps the IRLS solver to converge in fewer iterations.

4.F.8 Implementation Details

In this section we describe important details of our algorithm which are not part of its theoretical core but have an impact on its performance, and set the values of the parameters introduced throughout the paper.

The Cauchy M-estimator includes the parameter c that controls how robustly the residuals are minimized. In all cases, this parameter is set proportional to the average photometric and geometric residuals ($\bar{r}$), which are evaluated after each iteration of the IRLS solver:

$$c = \begin{cases} 0.2\,\bar{r} & \text{Robust odometry (§4.F.5)} \\ \bar{r} & \text{Scene flow (§4.F.7)} \\ 2\,\bar{r} & \text{Odometry refinement (§4.F.7)} \end{cases} \ . \tag{4.74}$$

Another important aspect is the selection of the parameters δ_L and δ_H in the segmentation stage. To obtain them, we compute the median of the robust residuals associated to the clusters ($\hat{\delta}$). Since the clusters of moving objects will typically have residuals considerably higher than $\hat{\delta}$, we set this value as a threshold. We have also observed that average residuals grow with the motion of the camera and, therefore, we also make δ_L and δ_H to be dependent on the norm of the camera motion:

$$\hat{\delta}_t = \min(t_M, \max(t_B, \hat{\delta})) \ , \tag{4.75}$$

$$\delta_L = \hat{\delta}_t + 10 \ \|\boldsymbol{\xi}_R\|_2 \, , \quad \delta_H = 2\hat{\delta}_t + 10 \ \|\boldsymbol{\xi}_R\|_2 \ . \tag{4.76}$$

The median residual $\hat{\delta}$ has to be truncated because clusters with residuals below a certain low threshold (t_B) are always assumed to belong to the background and those above a high threshold ($\sim 2\,t_M$) are assumed to be moving objects.

The rest of the parameters introduced in sections §4.F.5, §4.F.6 and §4.F.7 are set as follows:

- α_I is set to 0.15 so that photometric and geometric residuals contribute equally.

- The occlusion threshold (ΔZ_{occ}) is set to 0.2 meters, and the Z_{Min} of the geometric prior in the background segmentation is set to the mean depth of the scene divided by four.
- The weights of the different terms in the segmentation stage are set to $\lambda_R = 0.5$, $\lambda_T = 1.5$ and $\lambda_Z = 0.15$.
- The constants of the pre-weights in (4.72) and (4.73) are set to $K_I = 0.1$ and $K_Z = 10^{-4}$.

These values are the ones used for all the experiments presented in the paper and, although they have been obtained empirically, they provide good results for the indoor scenarios typically found when using RGB-D cameras. However, these values are not optimal in general and would need to be retuned if this method is applied with different sensors and/or configurations (e.g. stereo system and outdoor).

4.F.9 Experiments

This section is divided into two main parts: evaluation of the odometry and evaluation of the scene flow estimation. In all the experiments, we used registered RGB-D images at QVGA resolution (240×320). The experiments has been run on a laptop with an Intel Core i7-4712 HQ CPU at 2.3 GHz and Ubuntu 14.04. Besides analyzing the results presented herein, we encourage the reader to watch the demonstration video of our method (link above).

Odometry Evaluation

We test the accuracy of our algorithm with several sequences of the TUM dataset [49]. Some of the selected sequences do not contain moving objects (*Freiburg1*), while others (*Freiburg3*) are very challenging and present, at least for some time intervals, scenes mostly composed of moving objects or where the percentage of pixels with null depth goes beyond 50%. For all the tested sequences, we compare our method with DIFODO [66], DVO [3] and the semi-rigid scene flow (SR-Flow) [115] (which is the only method, apart form ours, providing both odometry and scene flow). The accuracy of each method is measured with the root mean square (RMS) translational and rotational drifts per seconds, as proposed in [49]. Besides this quantitative evaluation, and since only pixels with valid depth can be used to estimate the camera motion, we compute two additional statistics per sequence: the average percentage of pixels with valid depth, and the minimum percentage of pixel with valid depth among all the frames (i.e. the RGB-D image with the highest number of null depth measurements).

Sequence	Average % valid depth	Min % valid depth	Translational RMSE (cm/s)				Rotational RMSE (deg/s)			
			DIFODO	DVO	Ours	SR-Flow	DIFODO	DVO	Ours	SR-Flow
Fr1/desk	74.49%	60.07%	**3.66**	4.08	3.79	11.6	2.56	2.18	**1.88**	7.44
Fr1/desk2	74.52%	60.95%	**5.28**	6.45	5.33	12.7	3.31	3.55	**2.51**	9.96
Fr1/teddy	77.53%	64.73%	**5.18**	9.67	6.51	17.7	2.77	2.46	**2.04**	13.47
Fr3/walk static	54.59%	45.20%	45.4	31.2	**11.1**	23.7	10.2	4.56	**1.83**	5.42
Fr3/walk xyz	52.77%	17.67%	69.4	48.1	**30.4**	50.1	12.49	8.45	**5.69**	11.43
Fr3/walk halfsphere	58.59%	30.46%	70.0	41.2	**34.1**	46.5	19.21	7.22	**6.77**	13.8

Table 4.5: Odometry - Sequence statistics and translational/rotational deviations per second.

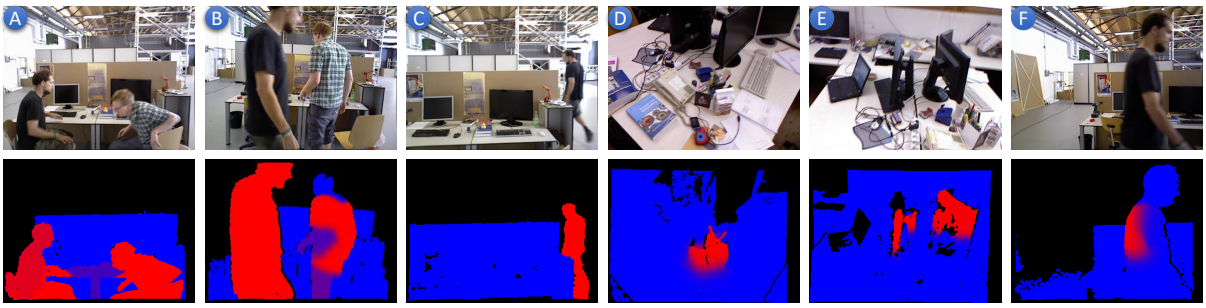


Figure 4.12: Segmentations obtained for some of the RGB-D frames in the tested sequences. Clusters segmented as background are depicted in blue and moving objects in red. It can be noticed that different moving parts can be segmented properly: (A) is segmented perfectly and (B) is not perfect but quite reasonable. (C) shows that, despite the geometric prior (4.69), our method can also detect moving objects which are at the background, far from the camera. (D) and (E) are examples of clusters wrongly segmented as moving parts. This sometimes occurs with clusters that observe non-smooth surfaces with scattered points and depth discontinuities, but they do not have any negative effect on the odometry estimation as long as there are enough clusters segmented as background. Finally, (F) shows one of the mentioned cases where more than half of the image has null depth and, among the pixels with valid depth, only half of them are observing static parts. In this case, our method fails to segment the scene properly and hence also fails to estimate the camera motion.

Results are shown in Table 4.5. It can be noticed that, for the static scenes, DIFODO shows the lowest translational drift and our approach has the lowest rotational error. The worst results are provided by the SR-Flow, being on average between 2 and 4 times less accurate than the other methods. As far as the dynamic sequences are concerned, the best results are always obtained with our approach. However, the drift is quite high in all cases, a fact that can be explained by analyzing the number of valid/used points in each sequence. While static sequences present, on average, 75% of valid depth measurement, this number drops to 55% in the non-static sequences. Moreover, within the non-static sequences, some RGB-D frames contain even less than 20% of valid depth measurements which, together with the fact that some of them are observing moving objects, renders the odometry problem extremely complicated. For the "*Freiburg3/walk static*" sequence, the percentage of used pixels never goes below 45%, and therefore our approach can still provide reasonable results. In the other two cases, the presence of RGB-D pairs with very low percentages of valid depth leads to much higher RMS errors.

We also show some of the segmentations provided by our method (Figure 4.12) and, for the static sequences, we compute the percentage of pixels wrongly segmented as background, and also those considered as uncertain (Table 4.6). Results show that these percentages are quite low for the three sequences considered.

Sequence	Moving objects	Uncertain
Freiburg1/desk	1.95%	3.73%
Freiburg1/desk2	2.53%	4.44%
Freiburg1/teddy	1.2 %	1.58%

Table 4.6: Percentage of pixels wrongly segmented as background or as uncertain regions in the *Freiburg1* sequences.

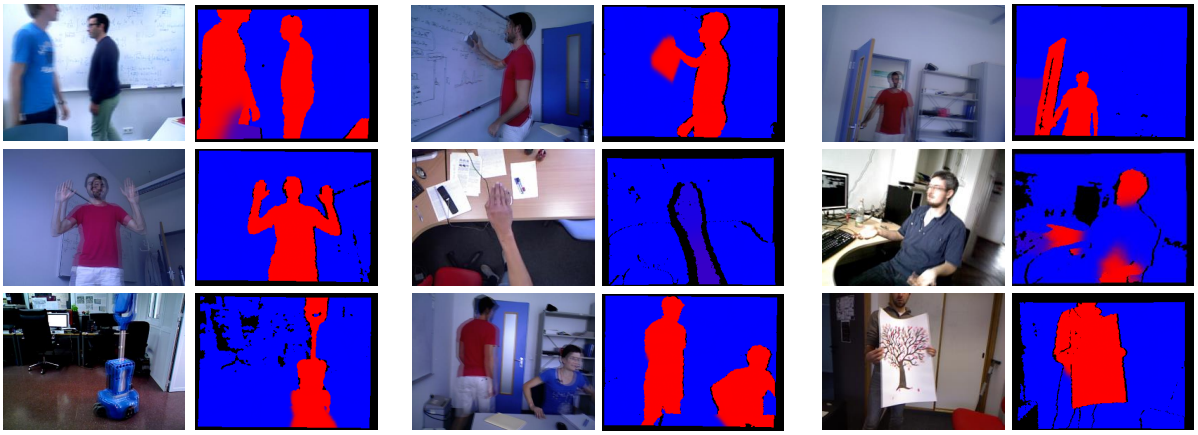


Figure 4.13: Selected images to evaluate scene flow, together with the segmentation provided by our method. Clusters segmented as background are depicted in blue and moving objects in red. It can be seen that the moving objects are segmented precisely in most of the images. The main exception is the "*hand RGB-D pair*", where the hand was not segmented as a moving object but as a region with uncertainty (slightly purple in the picture) due to its very small motion. Other inaccuracies can be observed in small parts of the backgrounds wrongly segmented as moving objects.

Scene Flow Evaluation

In this section we compare our approach with three of the most recent works on scene flow estimation: primal-dual scene flow (PD-Flow) [12], the semi-rigid scene flow (SR-Flow) [115] and the smooth piecewise rigid flow (MC-Flow) presented in [131]. Given the lack of RGB-D datasets with ground truth for the evaluation of scene flow, we have selected a set of RGB-D pairs observing different objects with varied motions. Some of the tested images have been used in previous works [115] [131], while others are new and will be published together with the code. Half of these RGB-D pairs were recorded with a moving camera, while the camera stood still in the other half. The accuracy of the different methods is assessed by warping the RGB-D pairs according to the estimated scene flow and measuring the RMS residuals (geometric and photometric) after alignment. The only inconvenient associated to this procedure is the fact that occluded pixels always generate high residuals even if the images are perfectly aligned, and therefore distort the error metric. To prevent this, for every method we discard pixels with geometric residuals below a certain threshold, i.e., a pixel x is disregarded if $r_Z(x) < -\Delta Z_{occ}$ after warping. Thus, the resulting RMS residuals become a more faithful metric of precise image alignment.

The testing images, together with the segmentations estimated by our approach, are shown in Figure 4.13. It can be observed that the segmentations are precise but not perfect, mostly because pixels are not segmented independently but in clusters, and clusters sometimes contain points of different objects (e.g. in the "*cleaning whiteboard*" image, the hand with the eraser is segmented as a moving object together with a small part of the whiteboard). Quantitative results are presented in Table 4.7. MC-Flow provides the best results for almost every RGB-D pair, followed by our method. This is to be expected because MC-Flow uses as strategy similar to the one described here to estimate motion, but it also optimizes for the clusters in an alternating

RGB-D pair	Camera moving	RMS photometric residuals				RMS geometric residuals			
		PD-Flow	SR-Flow	Ours	MC-Flow	PD-Flow	SR-Flow	Ours	MC-Flow
Two People walking	No	0.0368	0.0309	0.0284	**0.0251**	0.0511	0.0468	0.036	**0.0336**
Person standing	No	0.0532	0.0494	0.0303	**0.0256**	0.1385	0.1282	0.0829	**0.0814**
Robot	No	0.0518	0.0357	0.0251	**0.023**	0.0733	0.0688	0.0643	**0.0601**
Hand	No	0.0135	0.0077	0.0066	**0.0065**	0.0179	**0.0171**	0.0182	0.0182
Tree	No	0.0661	0.0335	0.0303	**0.0258**	**0.0143**	0.0259	0.0242	0.0196
Two People moving	Yes	0.0387	0.0301	0.0245	**0.0241**	0.1355	0.096	0.0928	**0.0654**
Cleaning Whiteboard	Yes	0.034	0.0233	0.0214	**0.0185**	0.0666	0.0561	0.0377	**0.0315**
Opening door	Yes	0.0231	0.0214	0.0204	**0.0153**	0.1681	0.1317	0.1014	**0.0551**
Person sitting	Yes	0.0753	0.0558	0.0452	**0.0428**	**0.0576**	0.0601	**0.0576**	0.0599

Table 4.7: Scene flow - RMS photometric and geometric residuals.

scheme. Consequently, its results are more precise but its computational burden is much higher. The average runtimes of the compared methods are:

- PD-Flow: 40 milliseconds on GPU or 2 seconds on CPU.
- SR-Flow: 105 seconds on a single CPU core.
- MC-Flow: 20 seconds running on CPU and GPU.
- Our approach: 80 milliseconds on multiple CPU cores.

In summary, our approach provides the second best scene flow estimates after MC-Flow (residuals are 12% higher on average), and it is 50% and 21% more accurate than PD-Flow and SR-Flow, respectively. Moreover, it is 250 times faster than MC-Flow and 1300 times faster than SR-Flow (only PD-Flow on GPU is faster than our method).

4.F.10 Conclusions

In this paper we have presented a method to estimate both odometry and scene flow with RGB-D cameras. Results demonstrate that our method performs similarly or better than other state-of-the-art approaches, which normally focus either on odometry or on scene flow, but do not estimate both. The only method which also addresses the joint estimation problem (SR-Flow [115]) is significantly less accurate with the camera motion and 20% less accurate with scene flow, as well as more than 1000 times slower. The main strength of our approach is that it provides accurate results with a very low runtime (80 milliseconds). To the best of our knowledge, this is the first method providing precise odometry and scene flow at a frame rate (>10Hz) that is sufficiently high to work in real-world scenarios.

Nevertheless, some aspects can still be improved. The K-Means clustering sometimes mix different objects in the same cluster, which leads to inaccurate scene flow estimates. As a solution, we will consider ways to exploit color (superpixels) or orientation in the segmentation process to obtain clusters that are more meaningful and still fast to compute. On the other hand, the temporal propagation of the background segmentation could be improved by introducing probabilistic/weighted models that are more consistent through longer periods of time. Lastly, we want to optimize the implementation of our algorithm so that it reaches real-time performance.

5

3D Reconstruction and Tracking with Subdivision Surfaces

5.A Introduction to 3D Reconstruction

In computer vision, *3D reconstruction* consists in estimating the shape of a given object from one or multiple images in which the object is visible. Depending on the sensor and the configuration used for the reconstruction, the resulting model will have the exact dimensions of the real object (depth or RGB-D cameras [135, 136], laser scanners [137], calibrated RGB cameras [138]) or will keep its proportions with a random or previously-fixed size (monocular RGB cameras [139]). It is worth noting that, when the entity to be modelled is not an object but the environment, the process is referred to as *mapping*. These two commonly-separated topics are, in essence, the same. The main difference lies on the fact that the size and topology of the environment to be mapped are unknown (and probably quite large), whereas those of the object to be reconstructed might be delimited beforehand. For this reason, mapping algorithms must be able to accommodate new incoming data while the sensor explores the environment. On the other hand, reconstruction techniques should be able to segment individual objects and might require higher precision to incorporate fine-grained details.

3D models of objects are useful for many applications. In virtual reality, they allow to insert instances of real objects in virtual scenarios. Even more interesting, if models of people are available they can be used to create Skype-like virtual meetings where each person is represented by an avatar that bears their own true appearance. The same idea has been applied in the gaming industry (see Figure 5.1). In augmented reality, 3D modelling can be used to generate a model of a real object and place it at a given location of the environment, as shown in Figure 5.1. As a different example, 3D reconstruction algorithms can be exploited for reverse engineering, to obtain the dimensions of a product or some components of it. Likewise, it can be used to generate miniatures of people as described in [140].

5.B Introduction to Tracking

Tracking is the process of detecting the location and/or pose of an object throughout a sequence of images. Many algorithms aim at finding the object on an image and placing a bounding box around it. However, we focus on a more complex concept of tracking, where a previously tailored

Figure 5.1: Illustrative examples of real applications of 3D modelling[1]. **Left**: Willen Dafoe in the Playstation game "*Beyond: Two Souls*". **Middle**: Hololens users check how a blue couch fits in their living room. **Right**: Miniature figures of former PhD students of the computer vision group at TUM.

model of the object is continuously aligned with the incoming data to know the exact object pose during the whole sequence. That model is typically represented by a 3D mesh composed of triangles or quads, although other alternatives based on smooth spline-like surfaces are recently gaining importance.

Tracking finds numerous applications in computer vision and computer graphics. For instance, body and hand tracking are currently used for human-computer interface and gaming [141, 142]. Face capture and tracking can be employed to animate the facial expressions of an input video, process known as *face reenactment* [143]. From a medical point of view, body tracking can help to diagnose injuries, correct wrong postures and assist during rehabilitation [144]. It could also be exploited by the fashion industry for virtual clothing to promote online sales.

5.C Contributions

Images used for 3D reconstruction or tracking normally observe not only the object to be modelled or tracked but also parts of the environment where this object is present. As a consequence, their pixels must be segmented into two different categories: those from which the object to reconstruct is visible (often called *foreground*) and those which observe other objects of the scene (often referred to as *background*). The foreground pixels contain information that the 3D model must fit, be it colour, position, orientation, etc. The background pixels also impose the restriction that the model should not be visible from them. Our work focuses on this second type of constraints that try to keep the model within the visual hull of the object. To that end, we present a new background term which formulates ray casting as a differentiable energy function. More precisely, this term addresses a min-max problem by first solving ray casting for the background pixels and then deforming the model so that the rays of the background pixels do not intersect it. Aside from that, we describe a complete framework for 3D reconstruction and tracking with subdivision surfaces, and show that the proposed background term can be easily combined with different data terms into an overall optimization problem. Lastly, the experimental section demonstrates that our proposal has several advantages over the distance transform-based term which is commonly employed in the literature.

[1]Pictures from `https://blog.es.playstation.com` (left), `www.matrixinception.com` (middle) and [140] (right).

5.D An Efficient Background Term for 3D Reconstruction and Tracking with Smooth Surface Models

Mariano Jaimez, Thomas J. Cashman, Andrew Fitzgibbon,
Javier Gonzalez-Jimenez and Daniel Cremers

Published in Proc. Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

An Efficient Background Term for 3D Reconstruction and Tracking with Smooth Surface Models

Mariano Jaimez, Thomas J. Cashman, Andrew Fitzgibbon,
Javier Gonzalez-Jimenez and Daniel Cremers

Abstract

We present a novel strategy to shrink and constrain a 3D model, represented as a smooth spline-like surface, within the visual hull of an object observed from one or multiple views. This new "background" or "silhouette" term combines the efficiency of previous approaches based on an image-plane distance transform with the accuracy of formulations based on raycasting or ray potentials. The overall formulation is solved by alternating an inner nonlinear minimization (raycasting) with a joint optimization of the surface geometry, the camera poses and the data correspondences. Experiments on 3D reconstruction and object tracking show that the new formulation corrects several deficiencies of existing approaches, for instance when modelling non-convex shapes. Moreover, our proposal is more robust against defects in the object segmentation and inherently handles the presence of uncertainty in the measurements (e.g. null depth values in images provided by RGB-D cameras).

5.D.1 Introduction

An important problem in computer vision is the recovery of 3D models from one or more images, whether RGB or depth. Examples include human body tracking [145], single-view reconstruction [146], or the acquisition of deformable object class models [147]. A dominant paradigm is to express the problem as energy minimization: find the 3D model parameters (including model shape, camera positions, etc.) which best explain the given data.

Formulations of such problems as energy minimization typically involve two key data terms: a term encouraging *foreground* measurements within the object to be explained by the model, and a *background* or "empty space" term, requiring that the model does not project in front of data samples known to be outside the object. Typically the foreground terms are easily written as variants of a closest-point or closest-intensity objective, which are readily optimized using ICP [39] or lifting algorithms [148, 147]. In contrast, the background terms involve an expensive raycasting or rendering operation, or are approximated by projecting a finite subset of points on the model surface into a distance transform.

This paper's contribution is to illustrate the failings of distance-transform-based background terms, and to introduce a new formulation with the accuracy of raycasting but which admits efficient optimization using smooth-function optimizers such as Levenberg-Marquardt. The key innovation is to write raycasting as an optimization problem in its own right, and to solve the min-of-max optimization that results from combining raycasting with the foreground term.

We will demonstrate the advantages of such a formulation in two very common scenarios in computer vision: 3D reconstruction and non-rigid tracking. In both cases the object to be reconstructed or tracked will be modeled with a subdivision surface, and the input data will consist of one or multiple depth images with unknown camera positions.

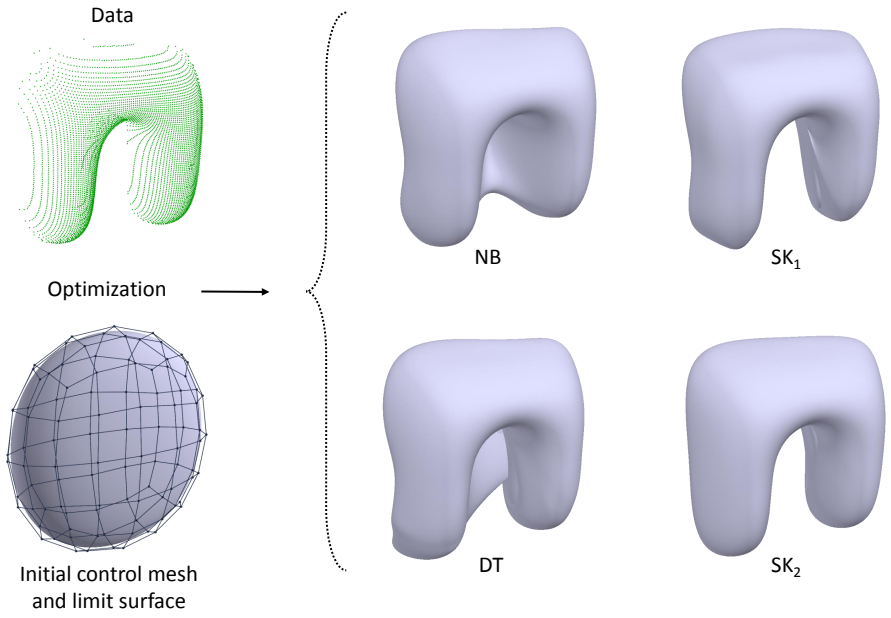


Figure 5.2: Modelling an arch from a single depth image and a challenging initial control mesh topology. Even with depth data, the model spills over into the background region (NB). We show how existing background terms based on the distance transform (DT) fail to capture the concavity, while our new methods based on shrinking kernels (SK_1 and SK_2) succeed.

5.D.2 Related Work

Given the considerable body of related literature, we focus on only a few key examples of the existing approaches.

One class of methods is volumetric or **voxel-based**. For example, the single-view reconstruction work of Töppe *et al.* [149], which imposes the hard constraint that object voxels must project into foreground regions, and expresses the reconstruction problem as energy minimization with one parameter per voxel. Recent work [150, 151] allows significant improvements in optimization, but the large state space limits model resolution, and as noted by Oswald *et al.* [146], the absence of a thresholding theorem means that the relaxation method employed for solution may not yield the optimal Boolean labelling. KinectFusion [51] avoids an optimization over the entire volume by estimating camera position using robust ICP, followed by deterministic carving of a 3D signed distance function, but copes poorly with missing data.

This paper focuses on **mesh-based** methods, such as used by Prasad *et al.* [152] for single-view reconstruction, or Vicente and Agapito [153] in deforming a template 3D mesh to match a given image silhouette. The latter paper used a distance transform penalty for the background term, as did Ganapathi *et al.* [145] in solving the problem of human body tracking. As shown below, the distance transform term has several limitations.

We consider a smooth surface representation based on subdivision surfaces. This spline-like representation has recently been used for 3D morphable model construction [147], hand shape estimation [154] and hand tracking [142]. Subdivision surfaces have also been used to fit 3D point clouds or regular meshes [155, 156, 157, 158, 159], but our guiding example problems

differ in that either the camera positions, or the mesh/object topology, or both, are unknown. Moreover, in real applications a significant percentage of the object to reconstruct or track might be missing due to lack of sufficient views and errors in the measurement process (null pixels in depth images provided by RGB-D cameras). Under these circumstances, the use of an effective and efficient background term becomes crucial.

5.D.3 Definitions and Notation

To illustrate the advantages of our proposal, we address two distinct problems: (A) generating a 3D reconstruction of an object from multiple views and (B) tracking a non-rigid object from an image sequence. In both cases, our input data comprise a set of N depth images $\{Z_i\}_{i=1}^{N}$ of a target object. A depth image is a collection of 3D points $Z_i = [\boldsymbol{p}_{ij}]_{j=1}^{M}$ associated with 2D pixel coordinates $\boldsymbol{x}_{ij}$ through the projection function $\boldsymbol{\pi} : \mathbb{R}^3 \mapsto \mathbb{R}^2$, i.e. $\boldsymbol{x}_{ij} = \boldsymbol{\pi}(\boldsymbol{p}_{ij})$. For each data point $\boldsymbol{p}_{ij}$ we also estimate a unit normal $\boldsymbol{n}_{ij}$ of the target object with $\|\boldsymbol{n}_{ij}\| = 1$. Each image i has an unknown camera pose parametrized using the twist $\boldsymbol{\xi}_i \in \mathbb{R}^6$, inducing a rotation matrix $R(\boldsymbol{\xi}_i) \in \text{SO(3)} \subset \mathbb{R}^{3\times3}$ and a translation vector $\boldsymbol{t}(\boldsymbol{\xi}_i) \in \mathbb{R}^3$. The function $\boldsymbol{\pi}$ is also overloaded to project world-coordinate points $\boldsymbol{s}$ by passing the pose of the camera to which points are projected, so $\boldsymbol{\pi}(\boldsymbol{s}, \boldsymbol{\xi}_i) = \boldsymbol{\pi}\big(R(\boldsymbol{\xi}_i)\,\boldsymbol{s} + \boldsymbol{t}(\boldsymbol{\xi}_i)\big)$.

The pixel indices are segmented into three disjoint regions D_i, B_i and C_i to specify whether each pixel observes the target object, the background, or provides no valid depth respectively. For example, $j \in D_i$ means that 3D point $\boldsymbol{p}_{ij}$ is a foreground measurement, and $j \in C_i$ means that pixel $\boldsymbol{x}_{ij}$ has an invalid 3D point. As we shall never access $\boldsymbol{p}_{ij}$ for invalid pixels, we need not define it in this case.

To model the object we use a Catmull-Clark subdivision surface, the shape of which is defined by a control mesh comprising P control vertices $\{\boldsymbol{X}_p\}_{p=1}^{P} \subset \mathbb{R}^3$ that are referenced as the corners of F quadrilateral faces $\{\boldsymbol{Q}_f\}_{f=1}^{F} \subset \{1..P\}^4$. The subdivision surface is a mapping $\boldsymbol{s} : \Omega \mapsto \mathbb{R}^3$, where the parametric domain Ω of the surface is the union $\square \times \{1..F\}$ of F copies of the unit square $\square := [\,0, 1\,] \times [\,0, 1\,] \subset \mathbb{R}^2$: one copy for each face in the control mesh. The topology of the surface, and hence Ω, is held fixed throughout the optimization, and so the shape of the surface is determined purely by the control vertices $X = \{\boldsymbol{X}_p\}_{p=1}^{P}$. We therefore write the surface as $\boldsymbol{s}(\boldsymbol{u}|X)$ where $\boldsymbol{u}$ is the tuple $\boldsymbol{u} = (u_x, u_y, f) \in \Omega$. The unit surface normal is written similarly as $\boldsymbol{s}^{\perp}(\boldsymbol{u}|X)$.

5.D.4 Optimization Problem

We present a unified framework to address either the 3D reconstruction or the non-rigid tracking of an object. It is based on two main constraints:

- The model must fit the geometric data ($\boldsymbol{p}_{ij}$ and $\boldsymbol{n}_{ij}$) computed from the depth images $\{Z_i\}$ for those pixels $j \in D_i$ where the object is *present*.
- The model should not be observable from pixels $\boldsymbol{x}_{ij}$ which are known to observe the background (i.e. $j \in B_i$), since in these locations we know the target object to be *absent*.

The remaining pixels, referenced by C_i, should not place any restriction on the model since their true depth is unknown, and we therefore have no evidence for either presence or absence of the target object.

Data Term

The first energy term measures the error in position and orientation between pixel $j \in D_i$ and a to-be-estimated corresponding model point $\boldsymbol{u} \in \Omega$:

$$E^p_{ij}(X, \boldsymbol{\xi}_i, \boldsymbol{u}) = \|\boldsymbol{p}_{ij} - R(\boldsymbol{\xi}_i)\, \boldsymbol{s}(\boldsymbol{u}|X) - \boldsymbol{t}(\boldsymbol{\xi}_i)\|^2_T\,, \tag{5.1}$$

$$E^n_{ij}(X, \boldsymbol{\xi}_i, \boldsymbol{u}) = \left\|\boldsymbol{n}_{ij} - R(\boldsymbol{\xi}_i)\, \boldsymbol{s}^{\perp}(\boldsymbol{u}|X)\right\|^2_T\,, \tag{5.2}$$

where $\|\bullet\|_T$ represents a truncated Euclidean norm. We combine (5.1) and (5.2) into a *data term* defined as the weighted combination

$$\hat{E}^d(X, \xi) = \sum_{i=1}^{N} \sum_{j \in D_i} \min_{\boldsymbol{u}} \left(\lambda_p\, E^p_{ij}(X, \boldsymbol{\xi}_i, \boldsymbol{u}) + \lambda_n\, E^n_{ij}(X, \boldsymbol{\xi}_i, \boldsymbol{u})\right). \tag{5.3}$$

This energy allows us to fit the model to the data by *lifting* [148, 147, 154] the latent model-data correspondences $\mathcal{U} = \{\boldsymbol{u}^d_{ij}\}$ for each $i = 1 \ldots N$ and $j \in D_i$ to give the energy

$$E^d(X, \xi, \mathcal{U}) = \sum_{i=1}^{N} \sum_{j \in D_i} \lambda_p\, E^p_{ij}(X, \boldsymbol{\xi}_i, \boldsymbol{u}^d_{ij}) + \lambda_n\, E^n_{ij}(X, \boldsymbol{\xi}_i, \boldsymbol{u}^d_{ij})\,, \tag{5.4}$$

with $\hat{E}^d(X, \xi) \leq E^d(X, \xi, \mathcal{U})$ for all $\mathcal{U}$. We can therefore minimize (5.3) and fit the observed data by finding

$$\arg\min_{X, \xi, \mathcal{U}} \left\{E^d(X, \xi, \mathcal{U})\right\}. \tag{5.5}$$

Background Term

Unfortunately, solving (5.5) often gives poor results, because there is nothing to penalize the model spilling over the observed object silhouette (see Figure 5.2 panel "NB" or the teddy bear's legs in Figure 5.5). It is therefore necessary to define an additional energy term that forces the model to remain within the visual hull of the object, as observed by the depth images.

Our goal is to have a background term that penalizes instances of the model that project into pixels $j \in B_i$ where the object is known to be absent. Essentially this is a sum of terms of the form "if any point anywhere on the model projects to pixel $\boldsymbol{x}_{ij}$, pay a penalty":

$$\sum_{i=1}^{N} \sum_{j \in B_i} \begin{cases} 1 & \text{if } \exists\, \boldsymbol{u} \in \Omega \text{ with } \boldsymbol{\pi}(\boldsymbol{s}(\boldsymbol{u}|X), \boldsymbol{\xi}_i) = \boldsymbol{x}_{ij} \\ 0 & \text{otherwise} \end{cases} \tag{5.6}$$

which can be re-cast as a minimization

$$\sum_{i=1}^{N} \sum_{j \in B_i} \begin{cases} 1 & \text{if } \min\limits_{\boldsymbol{u} \in \Omega} \|\boldsymbol{\pi}(\boldsymbol{s}(\boldsymbol{u}|X), \boldsymbol{\xi}_i) - \boldsymbol{x}_{ij}\| = 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.7}$$

and then written in terms of the finite delta function $\perp(r)$:

$$\sum_{i=1}^{N} \sum_{j \in B_i} \perp\big(\min_{\boldsymbol{u}} \|\boldsymbol{\pi}(\boldsymbol{s}(\boldsymbol{u}|X), \boldsymbol{\xi}_i) - \boldsymbol{x}_{ij}\|\big)\,, \tag{5.8}$$

$$\perp(r) = \begin{cases} 1 & \text{if } r = 0 \\ 0 & \text{otherwise} \end{cases}. \tag{5.9}$$

This transformation expresses raycasting as an optimization problem, but not an easy one: first, the L_0-like function has a zero-sized basin of convergence, and second, we cannot use the lifting trick. We rectify the first deficiency by using a more tractable proxy. A natural proxy to use for bounded terms is the L_1 proxy, but here this introduces complex bound constraints, meaning that the advantages conferred by a convex proxy are lost. We can choose as an alternative the complement of almost any flattening robust kernel, with the following desirable properties. It should be continuous and differentiable to allow the use of smooth-surface optimizers, which have been shown to provide significant improvements in convergence for the data term [142]. It also improves the efficiency of our solver (see §5.D.6) if the proxy is convex almost everywhere and flat (i.e. has a local maximum) at its peak. A suitable choice is the kernel used in the graduated non-convexity algorithm of Blake and Zisserman [160], which we name the *shrinking kernel* (SK) to describe its effect on parts of the model that spill over the background:

$$\Lambda(r) = \begin{cases} \left(1 - \frac{\epsilon}{\tau}\right)\left(1 - \frac{r^2}{\epsilon\tau}\right) & r < \epsilon \\ \left(1 - \frac{r}{\tau}\right)^2 & \epsilon \leq r \leq \tau \\ 0 & r > \tau \end{cases} \tag{5.10}$$

with $\epsilon \ll \tau$ as depicted in Figure 5.3. Other alternatives like a quartic polynomial could be used instead; we chose the shrinking kernel because it is the simplest one that fulfills our conditions.

We thus define our background energy by replacing $\perp$ with Λ in (5.8):

$$\hat{E}^b(X, \xi) = \lambda_b \sum_{i=1}^{N} \sum_{j \in B_i} \Lambda\left(\min_{\boldsymbol{u}} \|\boldsymbol{\pi}(\boldsymbol{s}(\boldsymbol{u}|X), \boldsymbol{\xi}_i) - \boldsymbol{x}_{ij}\|\right). \tag{5.11}$$

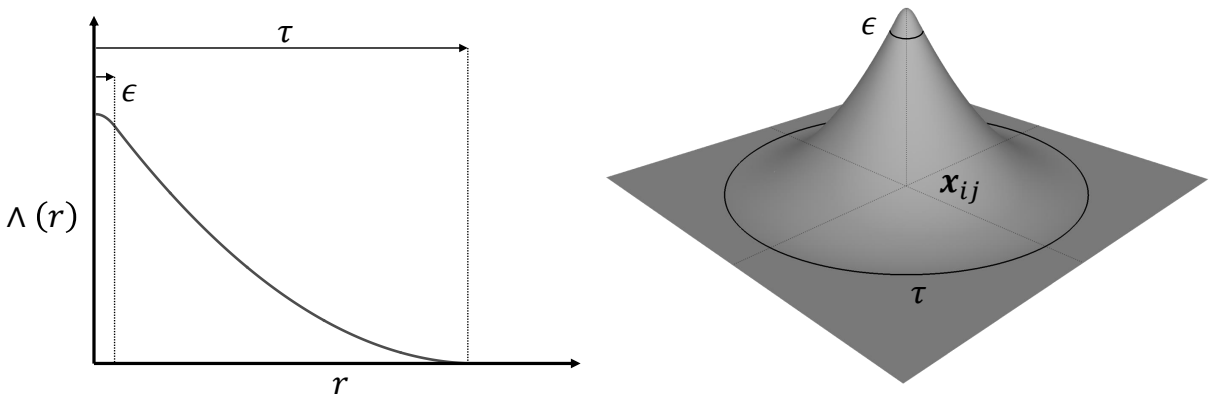


Figure 5.3: **Left**: Plot of the shrinking kernel $\Lambda(r)$ with respect to the reprojection error r. **Right**: 3D representation of the shrinking kernel centered at a particular background pixel $\boldsymbol{x}_{ij}$.

We can also now easily correct the second deficiency in (5.8): access to lifting. By noting that Λ is monotonic, we obtain $\Lambda(\min_x f(x)) = \max_x \Lambda(f(x))$ so

$$\hat{E}^b(X,\xi) = \lambda_b \sum_{i=1}^{N} \sum_{j \in B_i} \max_{\boldsymbol{u}} \Lambda\Big(\|\boldsymbol{\pi}(\boldsymbol{s}(\boldsymbol{u}|X), \boldsymbol{\xi}_i) - \boldsymbol{x}_{ij}\| \Big), \tag{5.12}$$

which can be subject to lifting as above (5.4), by defining latent variables $\mathcal{U}^b = \{\boldsymbol{u}_{ij}^b\}$ for each $i = 1 \dots N$ and $j \in B_i$:

$$E^b(X,\xi,\mathcal{U}^b) = \lambda_b \sum_{i=1}^{N} \sum_{j \in B_i} \Lambda\left(\|\boldsymbol{\pi}(\boldsymbol{s}(\boldsymbol{u}_{ij}^b|X), \boldsymbol{\xi}_i) - \boldsymbol{x}_{ij}\| \right) \tag{5.13}$$

with $\hat{E}^b(X,\xi) \geq E^b(X,\xi,\mathcal{U}^b)$ for all $\mathcal{U}^b$.

Fixed vs adaptive τ

The value of τ in (5.10) significantly changes the effect of the background term (5.13) on the overall optimization. A high value of τ increases the number of pixels pushing the model inwards ($r \leq \tau$) and leads to a smoother energy which is easier to optimize. However, a high τ also implies that the background term competes with the data term at the object boundaries and prevents the model from fitting the data in these areas. Conversely, a low value for τ implies fewer pixels pushing inward and a sharper energy but also less competition between the background and the foreground terms. To overcome these limitations, we employ an adaptive $\tau_i(\boldsymbol{x})$ which depends on the pixel $\boldsymbol{x}$ and the image i. Thus, τ will be low for pixels close to the silhouette and will be higher otherwise. The function which measures the minimum distance from any pixel to the object silhouette is the distance transform $\mathrm{DT}_i(\boldsymbol{x}) : \mathbb{R}^2 \mapsto \mathbb{R}$. Therefore, the adaptive width of the shrinking kernel is given by $\tau_i(\boldsymbol{x}) = \min(\mathrm{DT}_i(\boldsymbol{x}), \tau_{\max})$ for a given image i. A maximum width $\tau_{\max}$ must be set because the shrinking kernel is intended to work close to the model boundaries and would be ineffective and inefficient if τ took arbitrarily high values.

This strategy takes advantage of the distance transform by using information about proximity to the silhouette to avoid pushing the model beyond it, but it flattens appropriately far from the silhouette, unlike the distance transform, whose gradients are often wrong or misleading (see §5.D.7 and Figure 5.4).

5.D.5 Regularization

The data and background terms (5.4) and (5.13) guarantee that the model fits the data and keeps within the convex hull of the object. However, the solution found using these terms alone can include creases and sharp edges that make the 3D model unappealing. The mesh can also degenerate throughout the optimization process, leading to ill-posed configurations that eventually cause the 3D reconstruction or the tracking system to fail. For these reasons, we introduce two regularization terms that encode a smoothness prior on the object we are trying to reconstruct. Moreover, for the tracking problem, we include other two extra terms to keep the subdivision surface as rigid as possible while tracking the target object.

To keep the surface smooth we penalize the gradient of the surface normals, a proxy for surface curvature. We approximate this using a discrete sum by homogeneously sampling the subdivision surface over its parametric domain $\square \times \{1..F\}$ to obtain F sets of K samples per face, denoted $\boldsymbol{\sigma}_{fk}$. Then the surface smoothness regularizer is

$$E^s(X) = \lambda_s \sum_{f=1}^{F} \sum_{k=1}^{K} \frac{\left\|\nabla_{u_x} \boldsymbol{s}^{\perp}(\boldsymbol{\sigma}_{fk}|X)\right\|^2}{\left\|\nabla_{u_x} \boldsymbol{s}(\boldsymbol{\sigma}_{fk}|X)\right\|^2} + \frac{\left\|\nabla_{u_y} \boldsymbol{s}^{\perp}(\boldsymbol{\sigma}_{fk}|X)\right\|^2}{\left\|\nabla_{u_y} \boldsymbol{s}(\boldsymbol{\sigma}_{fk}|X)\right\|^2}, \tag{5.14}$$

where we use forward finite differences to approximate the gradients $\nabla \boldsymbol{s}^{\perp}(u) \in \mathbb{R}^3$ and $\nabla \boldsymbol{s}(u) \in \mathbb{R}^3$.

In addition, we want control vertices to be spread as evenly as possible over the model, so we must avoid the control mesh from stretching and distorting arbitrarily. We enforce this by adding a simplified and discrete version of the membrane energy. If $\boldsymbol{e}_{f,k}$ denotes the kth edge of the quadrilateral face f, this second regularization term is defined as

$$E^h(X) = \lambda_r \sum_{f=1}^{F} \sum_{k=1}^{4} \left\|\boldsymbol{e}_{f,k}(X)\right\|^2. \tag{5.15}$$

E^h penalizes long edges and indirectly favours isometry.

When the goal is to track a non-rigid object, we assume that a mesh with the shape of the object is given. This mesh moves and deforms over time to fit the new incoming data but, at the same, it must keep its original proportions. To enforce such behaviour, we include the as-rigid-as-possible regularizer (ARAP) [161]:

$$E^a(X, \zeta) = \lambda_a \sum_{v=1}^{P} \sum_{k \in \mathcal{n}_v} \left\|R(\boldsymbol{\zeta}_v)(\boldsymbol{X}_v^o - \boldsymbol{X}_k^o) - (\boldsymbol{X}_v - \boldsymbol{X}_k)\right\|^2 \tag{5.16}$$

where X^o gives the initial locations of the control vertices of the mesh employed for tracking, and $\mathcal{n}_v$ contains the neighbours for vertex v. For each control vertex, the relative rotation of the mesh with respect to its original configuration is represented by the matrix $R(\boldsymbol{\zeta}_v) \in \mathrm{SO}(3)$. These rotations are regularized so that the deformation of the mesh is locally smooth:

$$E^r(\zeta) = \lambda_r \sum_{v=1}^{P} \sum_{k \in \mathcal{n}_v} \left\|\boldsymbol{\zeta}_v - \boldsymbol{\zeta}_k\right\|^2 \tag{5.17}$$

The sum of these terms is written

$$E^R(X) = E^s(X) + E^h(X) + \min_{\zeta} \left\{E^a(X, \zeta) + E^r(\zeta)\right\}, \tag{5.18}$$

and the rotations ζ are lifted into the overall problem.

5.D.6 Solver

We combine the data and background terms with regularizers $E^R(X)$ to build the overall optimization problem:

$$\min_{X, \mathcal{U}, \xi} \left\{ E^d(X, \xi, \mathcal{U}) + \max_{\mathcal{U}^b} \left\{E^b(X, \xi, \mathcal{U}^b)\right\} + E^R(X) \right\}. \tag{5.19}$$

This energy is highly non-linear, non-convex and combines minimization and maximization processes that appear challenging to solve jointly. The problem contains some standard components: the combination of minimization and maximization recalls the concave-convex procedure [162] and DC programming [163]. However the natural decompositions of our problem (i.e. the objectives of the min and max as written) are not concave so these techniques do not directly apply.

Our solver is divided into two stages: an inner maximization and an outer minimization. The inner maximization finds the background correspondences $\mathcal{U}^b$ (raycasting) that are needed for every iteration of the outer minimization problem. In turn, the outer minimization will be solved iteratively by generating updates for the control vertices X, the foreground correspondences $\mathcal{U}$ and the camera poses ξ, guaranteeing that each iteration decreases the overall energy.

Inner Raycasting Maximization

The first task is to find the correspondences for the background term, that is

$$\mathring{\mathcal{U}}^b = \arg\max_{\mathcal{U}^b} \left\{ E^b(X, \xi, \mathcal{U}^b) \right\}. \tag{5.20}$$

The correspondences within the vector $\mathcal{U}^b$ are independent from each other, which means that (5.20) can be solved with independent optimizations over Ω for each pixel. These remain nonlinear optimizations, but we can make use of Levenberg-Marquardt. Although E^b is non-quadratic, monotonicity of Λ means that $\arg\max_u \Lambda(f(u)) = \arg\min_u f(u)^2$ for a smooth function $f : \Omega \mapsto \mathbb{R}^2$. Note that this transformation applies only because the optimizations are independent per pixel: it is not the case that $\arg\max_u \Lambda(f_1(u)) + \Lambda(f_2(u)) = \arg\min_u f_1^2(u) + f_2^2(u)$ in the general case when f_1 and f_2 both depend on all of u. To improve efficiency, the background correspondences stop being updated if their projection error is higher than a given threshold $\upsilon > \tau$, thereby discarding all those pixels of the background which are too far from the model to provide any help.

In order to update these correspondences within the optimizer, correspondences may need to transition between different faces of the control mesh. As Catmull-Clark subdivision surfaces are nearly-everwhere C^2 continuous, these transitions do not harm the differentiability of our energy terms, and the correspondence updates can be handled using the strategy described in [147, 154]. Transitions of the foreground correspondences $\mathcal{U}$ in the outer minimization are handled similarly.

Outer Minimization

Now we need to solve

$$\min_{X, \mathcal{U}, \xi} \{ f(X, \mathcal{U}, \xi) + g(X, \xi) \} \tag{5.21}$$

with

$$f(X, \mathcal{U}, \xi) = E^d(X, \xi, \mathcal{U}) + E^R(X), \tag{5.22}$$

$$g(X, \xi) = E^b\left(X, \xi, \mathring{\mathcal{U}}^b(X, \xi)\right). \tag{5.23}$$

The Levenberg-Marquardt algorithm does not appear to be applicable here because not every term is expressed in the form of sum of squares and, moreover, the background term E^b is not convex. However, we will show that thanks to the particular choice (5.10) adopted to approximate the raycasting function (5.6), the Levenberg-Marquardt algorithm can be applied and it leads to an efficient optimization strategy. First of all, the shrinking kernel is flat at its maximum, which makes the Jacobian computation much easier, as the multiplicands of the difficult terms $\frac{\partial \mathring{\mathcal{U}}^b(X,\xi)}{\partial X}$ are zero:

$$\frac{\partial g(X,\xi)}{\partial X} = \frac{\partial E^b(X,\xi,\mathring{\mathcal{U}}^b)}{\partial X} + \cancelto{0}{\frac{\partial E^b(X,\xi,\mathring{\mathcal{U}}^b)}{\partial \mathcal{U}^b}} \cdot \frac{\partial \mathring{\mathcal{U}}^b(X,\xi)}{\partial X}, \tag{5.24}$$

$$\frac{\partial g(X,\xi)}{\partial \xi} = \frac{\partial E^b(X,\xi,\mathring{\mathcal{U}}^b)}{\partial \xi} + \cancelto{0}{\frac{\partial E^b(X,\xi,\mathring{\mathcal{U}}^b)}{\partial \mathcal{U}^b}} \cdot \frac{\partial \mathring{\mathcal{U}}^b(X,\xi)}{\partial \xi}. \tag{5.25}$$

For the sake of clarity, the Jacobians have been written as scalar partial derivatives. Secondly, the shrinking kernel is defined with $\epsilon \ll \tau$ (see Figure 5.3) and, hence, it is convex and can be expressed as a sum of squares almost everywhere (apart from a small area surrounding its peak). Therefore, we use all those pixels with correspondences lying in the convex area and with non-null gradient ($\epsilon \leq r \leq \tau$), and omit those which are just at the maximum or very close to it. In practice, this does not have any detrimental effect over the minimization process because this approximation discards only pixels which have a ray intersecting with the model or very close to it. Rays that intersect with the model ($r = 0$) contribute no gradient as previously shown; only a tiny fraction of the pixels discarded will have a ray which does not intersect and yet still lies in the concave area ($0 < r \leq \epsilon$).

Every iteration of the Levenberg-Marquardt algorithm involves the construction of a sparse and large linear system which is solved by applying a Cholesky LDLT decomposition. Moreover, to overcome/avoid local minima due to wrong correspondence associations, we periodically perform a global search by uniformly sampling the subdivision surface and checking for each pixel $j \in D_i$ (foreground) whether any of these samples reduces its energy E^d_{ij}. A similar search is also performed for the background correspondences.

Coarse-to-Fine

Subdivision surfaces provide a refinement relation $\mathcal{R}$ that densifies the control mesh without modifying the surface. That means that for every parametric coordinate $\boldsymbol{u}$ in the original mesh, there always exist another $\hat{\boldsymbol{u}}$ in the refined mesh that leads to the same spatial coordinate:

$$\boldsymbol{s}(\boldsymbol{u}|X) = \boldsymbol{s}(\hat{\boldsymbol{u}}|\mathcal{R}X) \quad \forall \boldsymbol{u} \in \Omega. \tag{5.26}$$

This refinement can be iterated to define a series of control meshes $X^l = \mathcal{R}^l X$, all of which represent the same limit surface. Here l denotes a given level within the coarse-to-fine scheme. This means we can optimize a coarse model for the control vertices $X = X^0$, then apply $\mathcal{R}$ to obtain a new set of model freedoms X^1 without changing E in (5.19). We then optimize $X = X^1$ using (5.19) to obtain optimal control vertices at level 1, and iterate this procedure until we find a solution at the target control mesh density. Thus, the optimizer is able to fit a

detailed model with many control vertices by using the coarse model to find the energy well for a good local minimum.

5.D.7 Experiments

We conducted a series of experiments to compare our approach with the popular distance transform (DT) method for enforcing silhouette consistency [145, 153]. To perform these comparisons we implemented an alternative background term E^b_{DT} by sampling the subdivision surface uniformly at L fixed locations $\{\boldsymbol{\sigma}_l\}_{l=1}^{L}$ in Ω, and projecting the samples into the distance transforms of each of the N depth images:

$$E^b_{\text{DT}}(X,\xi) = \lambda_{\text{DT}} \sum_{i=1}^{N} \sum_{l=1}^{L} \text{DT}_i^2 \Big(\boldsymbol{\pi}(\boldsymbol{s}(\boldsymbol{\sigma}_l|X), \boldsymbol{\xi}_i) \Big). \tag{5.27}$$

We present five distinct experiments to compare our background term with the standard DT-based term (5.27). These experiments are intended to investigate the behaviour of our proposal in common computer vision scenarios, not to demonstrate state-of-the-art algorithms for 3D reconstruction or tracking. The first three experiments address the 3D reconstruction problem from single or multiple views respectively. In the remaining tests we track a non-rigid object (a person) through a sequence of depth images. Moreover, we analyze the computational cost of the different tested methods.

For a better visualization of the results presented here, we encourage the reader to watch the demonstration video.

Experiment Initialization and Segmentation

For each experiment, each of the N images captured by the depth camera already provides the set of invalid depth pixels C_i. We have implemented three different approaches to segment the remaining pixels into the regions corresponding to object (D_i) and background (B_i):

- Segmentation by plane removal. The object is placed on a flat surface and the camera must mostly observe the object and the plane it is lying on. B_i is defined by proximity to a plane fitted to the flat surface and D_i by the remainder.
- Segmentation by background subtraction. The camera pose is fixed and several images of the object are taken. By capturing a single image of the background without the target object present, we can define D_i using those pixels that change between the former images and the background image.
- Segmentation by depth thresholding. D_i is simply defined to be those pixels within a depth range previously set, and B_i to be the remainder.

Moreover, our algorithm requires an initial guess for the camera poses, which does not need to be very accurate and we assume is provided by the user.

Modeling an Arch

Our first experiment is a synthetic test where the data to fit consists of a single depth image generated as the front view of a smooth arch. To initialize the mesh, we compute the bounding box of the data and apply $\mathcal{R}$ twice to generate a mesh with enough degrees of freedom to deform and adapt to the data. Its corresponding initial surface is roughly an ellipsoid, which is quite far from the arch that we aim to reconstruct (see Figure 5.2). In the experiment, we compare four different strategies for the background term: DT (5.27), SK (5.13) with fixed τ (SK_1), SK (5.13) with adaptive τ (SK_2) and without background term (NB). The weights associated to the different background terms (λ_b and λ_{DT}) are tuned so that all the background energies have the same initial value.

The final solutions are depicted in Figure 5.2. We observe that the distance transform is unable to shrink the model properly while the two versions of our approach do it almost perfectly. The DT's poor behaviour has two causes. First, our formulation of E^b_{DT} implements a discrete sampling over the model instead of integrating each DT_i over all of Ω. However, the second reason for the failure is that the gradients of the DT function (shown in Figure 5.4) are mostly horizontal between the two pillars of the arch. This gives no reason for the model to shrink vertically. Instead, it stretches horizontally in both directions to move the sampled model positions $\{\boldsymbol{\sigma}_l\}$ out of the penalized image region while leaving model surface stretched in-between. On the other hand, the shrinking kernel always pushes the surface in the opposite direction to the surface normals at the model silhouette. The only pixels where the shrinking kernel has non-zero gradient are those whose ray does not intersect the model but comes close to it ($\epsilon \leq d \leq \tau$). Thus, the SK background term projects the 3D model onto the image plane and pushes the surface inward on those parts of the model's silhouette that project out of the real silhouette (see Figure 5.4).

The number of iterations (see demonstration video) required is significantly reduced by SK_2 over SK_1, being in both cases higher than it is with DT. Interestingly, SK_2 also finds a better optimum for the data term that NB is directly optimizing, by helping the model to distribute its freedoms more usefully. Although SK_1 is also able to create the gap between the columns of the arch, its energy after convergence is higher because the data and the background terms compete at the object boundaries.

Incorrect Segmentations

Another problem associated with the DT background term is the fact that an incorrect segmentation, even if there is just a single misclassified pixel, can lead to a very different distance transform which might be detrimental for the 3D reconstruction. A perfect segmentation is hard to obtain in many practical cases, so robustness to segmentation errors is important. In this section we compare basic 3D reconstructions obtained with no background term (NB), with DT and with SK (adaptive τ) when the segmentation of the object is imperfect.

We test on a multi-view 3D reconstruction problem with $N = 4$ depth images of a teddy bear taken from different camera angles. We segment the depth images using background subtraction as explained in the supplementary material. Since the measurement error grows quadratically

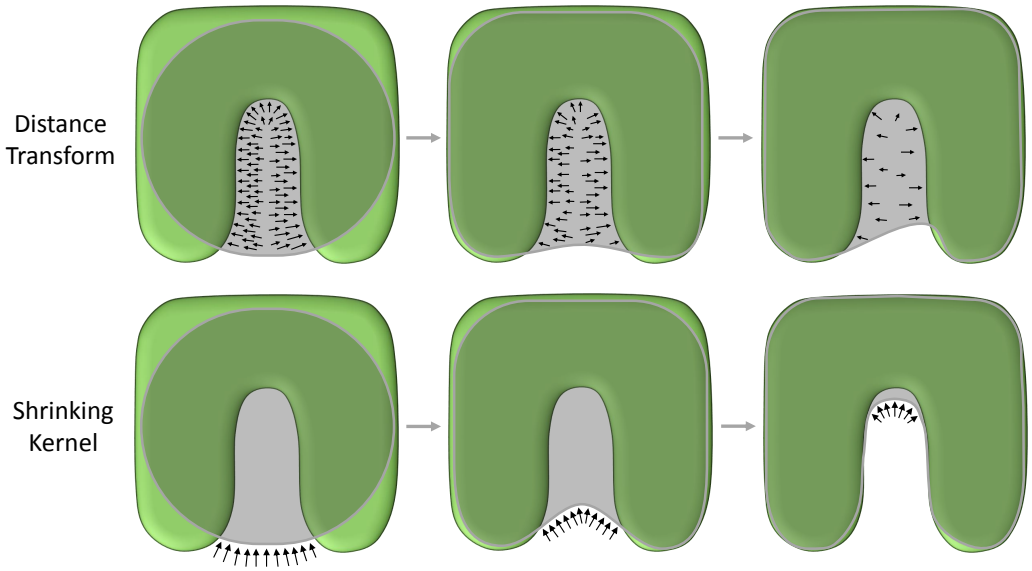
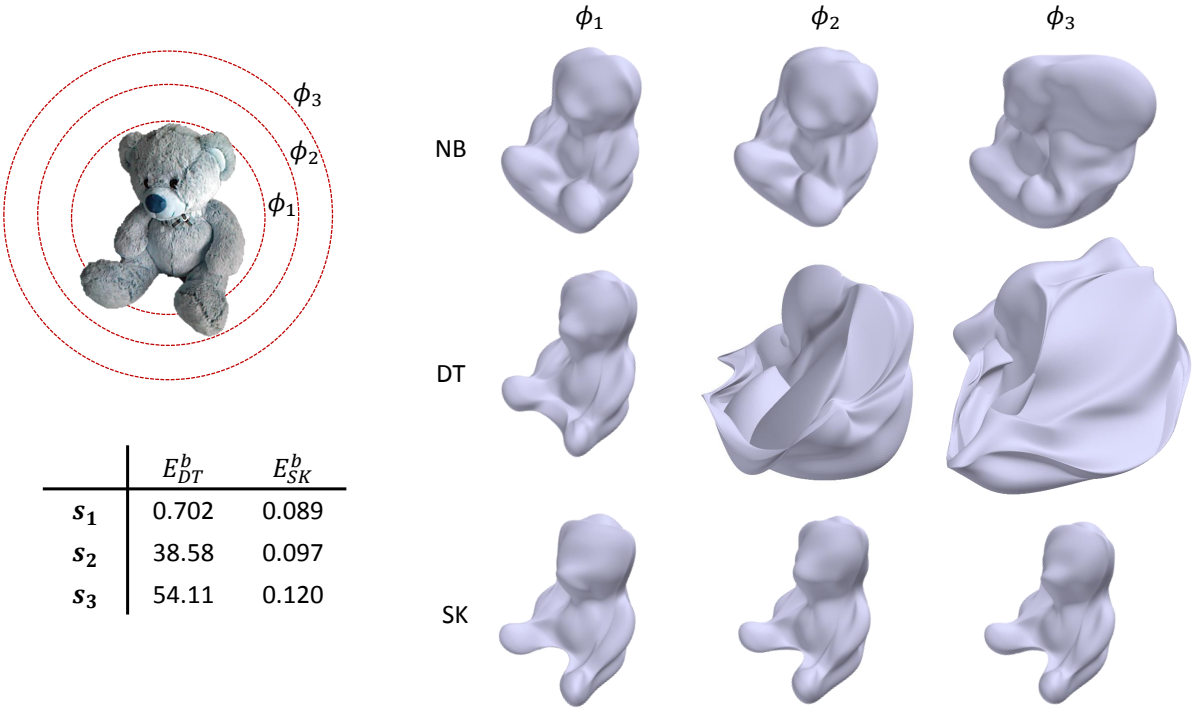


Figure 5.4: Evolution of the surface throughout the optimization process. The images demonstrate how each background term pushes the surfaces: DT creates gradients for every sample of the surface projecting out of the silhouette whereas SK pushes only on its contour.

with depth, we use a comparison threshold between the background and the input images $\{Z_i\}_{i=1}^{N}$ that also grows quadratically with depth, in an attempt to avoid many false positive detections at distant areas. However, the resulting segmentation is still imperfect and every image contains scattered pixels or small distant regions which are mistakenly tagged as object. While it would be possible to post-process the segmentations further for better results, we leave them in this unprocessed state as our intention is to test the robustness of each method to segmentation errors.

We compare the basin of convergence for the reconstruction in each case, by starting with three different initial control meshes. These meshes are cubes placed at the centroid of the data points with edges set to 0.4, 0.5 and 0.6 meters respectively. The control meshes generate roughly spherical surfaces with diameters denoted by ϕ_1, ϕ_2 and ϕ_3 in Figure 5.5. In this experiment we give high weights λ_b and λ_{DT} to the background terms to test whether the algorithm is able to shrink the model to the convex hull. The optimization is run in a coarse-to-fine scheme with 4 levels, each one running a maximum of 25 iterations of LM to solve (5.21).

Quantitative and qualitative results are presented in Figure 5.5, which shows that the data term alone is able to shrink the model partially in the first two cases but completely fails for the largest initialization. It also fails to create the expected gap between the teddy's legs. On the other hand, DT produces a good result for s_1 but fails dramatically for s_2 and s_3 because the DT gradients far from the target object are sometimes directed towards pixels that are incorrectly segmented as data. In fact, they even force the model to protrude and deform in undesirable ways, worsening the solution compared to that without the background term. Finally, SK is able to shrink the model into the convex hull in all cases and successfully separates the teddy bear's legs. The final energies associated to the background terms are also shown in Figure 5.5.



	E^b_{DT}	E^b_{SK}
s_1	0.702	0.089
s_2	38.58	0.097
s_3	54.11	0.120

Figure 5.5: **Top left**: Illustration of the object to model together with the outline of the initial spheres used in the 3 sets of tests. **Bottom left**: Final energy terms after the optimization. **Right**: 3D reconstructions obtained without background term, with DT and with SK for the 3 different initializations ϕ_1, ϕ_2 and ϕ_3. SK is less dependent on initialization, and is better on the gap between the legs.

Robustness to Wrong Initial Camera Poses

For this experiment we have recorded a continuous RGB-D sequence by moving a handheld camera around a teddy bear. In order to get accurate camera poses we have run a voxel-based SLAM method which combines [164] and [165]. The purpose of this experiment is to test the basin of convergence of the different approaches (here SK, DT_{all} and DT_{safe}) when the camera poses are not initialized correctly. To that end, we will consider the pose estimates provided by [164] as ground truth and will generate perturbed initial camera poses by adding Gaussian noise to them (the ground truth poses are actually not error-free but they are precise enough for the evaluation). We decimate the sequence and retain only four depth images to address the 3D reconstruction problem. To be able to measure the deviations with respect to the original camera poses, we fix the first camera and only perturb and optimize for the other three.

The image resolution employed is 60×80, and the optimization runs until convergence within a coarse-to-fine scheme. The initial control mesh is obtained as the bounding box of the data, and is refined once (applying $\mathcal{R}$) before starting the optimization process. For this experiment we employ three coarse-to-fine levels although we run the optimization for the first level twice: first with strong regularization to avoid the mesh to deform too much while the camera poses are far from their true positions, and second with a normal weighting to allow the surface to fit data. Since the teddy bear was lying on a table when the sequence was recorded, the segmentations can be obtained by plane removal.

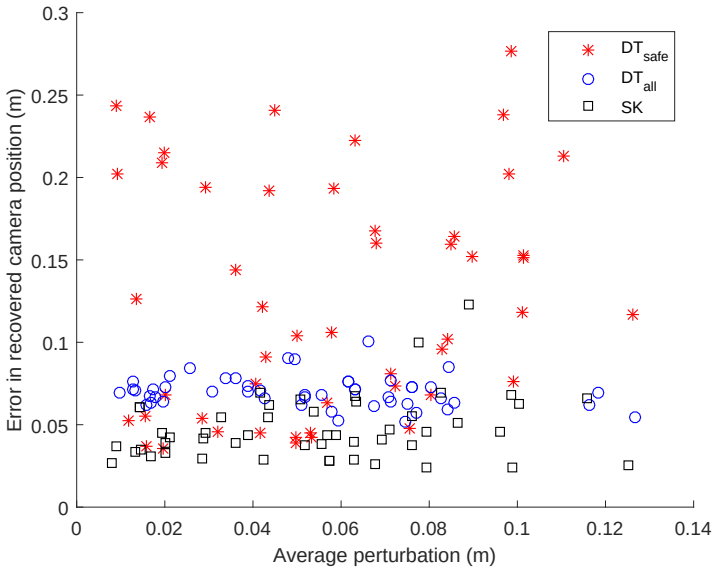


Figure 5.6: Error in the estimation of the camera positions as a function of the initial average perturbations. For simplicity, only the translational component of $\boldsymbol{\xi}_i$ is considered for both the perturbations and the measured errors.

We run a total of 50 tests for each method; results are shown in Figure 5.6. SK provides the lowest error on average since it is able to recover the camera poses and shrink the model without penalizing the data term. On the other hand, $\mathrm{DT}_{\mathrm{all}}$ is always able to bring the cameras to their right configuration but it goes too far by pushing the model inwards when it projects on C_i (null depth), leading to higher average pose errors. Last, $\mathrm{DT}_{\mathrm{safe}}$ shows an erratic behaviour, being sometimes able to bring the cameras to their right poses and sometimes failing dramatically due to the wrong DT gradients.

Tracking under poor illumination conditions

Now we compare the two different background terms focusing on the regions $\{C_i\}$ that can be segmented as neither object nor as background. These regions are common artefacts of the capture mechanisms used by depth cameras, but the same problem could arise with RGB images if there are areas which cannot be segmented properly and remain uncertain.

The goal of the experiment is to track the body of a person who moves in front of an RGB-D camera. To better illustrate the differences between the compared methods, the sequence of images has been recorded outdoors where the depth measurements have a lower quality due to the sun's radiation. The testing sequence consists of 20 images subsampled from a longer sequence of 60 to increase the displacement between consecutive frames. Images are segmented by thresholding depth since the person is always closer to the camera than the background points. The compared background terms are configured so that they have similar weights during the optimization process. For the experiment we assume that an initial mesh (X^o in (5.16)) with the shape of a person is provided. Coarse-to-fine is not used here, i.e., the size and topology of this mesh does not change during the experiment.

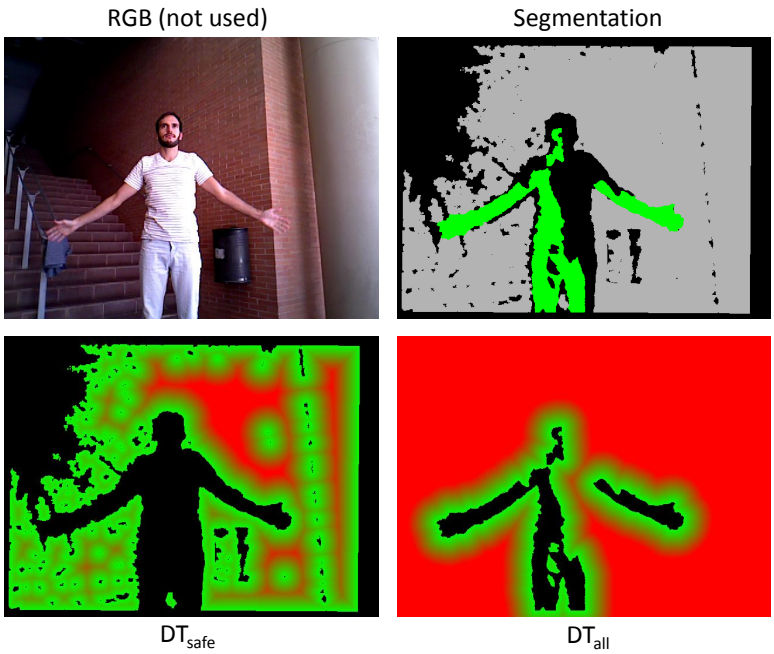


Figure 5.7: Differences between $\mathrm{DT}_{\mathrm{all}}$ and $\mathrm{DT}_{\mathrm{safe}}$ for one of the frames used for tracking. **Top left**: RGB image included for clarity but not used in the algorithm. **Top right**: Segmented image. The foreground pixels are shown in green, background in gray and null depth in black. **Bottom**: $\mathrm{DT}_{\mathrm{all}}$ and $\mathrm{DT}_{\mathrm{safe}}$ starting from zero (green) and truncated at 50 (red) for a better visualization.

Under the presence of invalid or uncertain measurements, we need to decide between two options to compute the distance transform. The first is to compute the distance transform for both background and invalid pixels, setting to zero only those pixels that are segmented as object ($\mathrm{DT}_i(\boldsymbol{x}_{ij}) = 0$ iff $j \in D_i$). We refer to this strategy as $\mathrm{DT}_{\mathrm{all}}$. The main disadvantage of $\mathrm{DT}_{\mathrm{all}}$ is that it shrinks the model beyond the real silhouette of the object, because the object is likely to be visible from some of those pixels tagged as invalid ($j \in C_i$). The second option, denoted here as $\mathrm{DT}_{\mathrm{safe}}$, sets to zero both the pixels observing the object and the invalid depth measurements, in an attempt to create a distance transform that only penalizes pixels that are known to observe the background ($\mathrm{DT}_i(\boldsymbol{x}_{ij}) = 0$ iff $j \notin B_i$). This alternative strategy is less restrictive and gives the model more freedom to adapt to the data properly, but it also has a drawback: the invalid area surrounding the object may have arbitrary size and contour, which can lead to gradients ∇E^b_{DT} with directions that are harmful to the model (see Figure 5.7). In contrast, the shrinking kernel penalizes only background pixels (so it does not overconstrain the final solution) without being affected by the null depth measurements.

Qualitative results are shown in Figure 5.8. It can be observed that, in the absence of a background term, the model fits the data but sometimes protrudes out of the silhouette. Moreover, it allows for wrong correspondences between the model and the data, as occurs sometimes for the head (middle column) which tries to fit some of the arm points. $\mathrm{DT}_{\mathrm{all}}$ provides the worst results because it tries to push the model out of the areas with null depth, and almost half of the pixels observing the person have null depth. $\mathrm{DT}_{\mathrm{safe}}$ performs better but still leads to some artefacts as the gradients of the distance transform are not always directed towards the target.

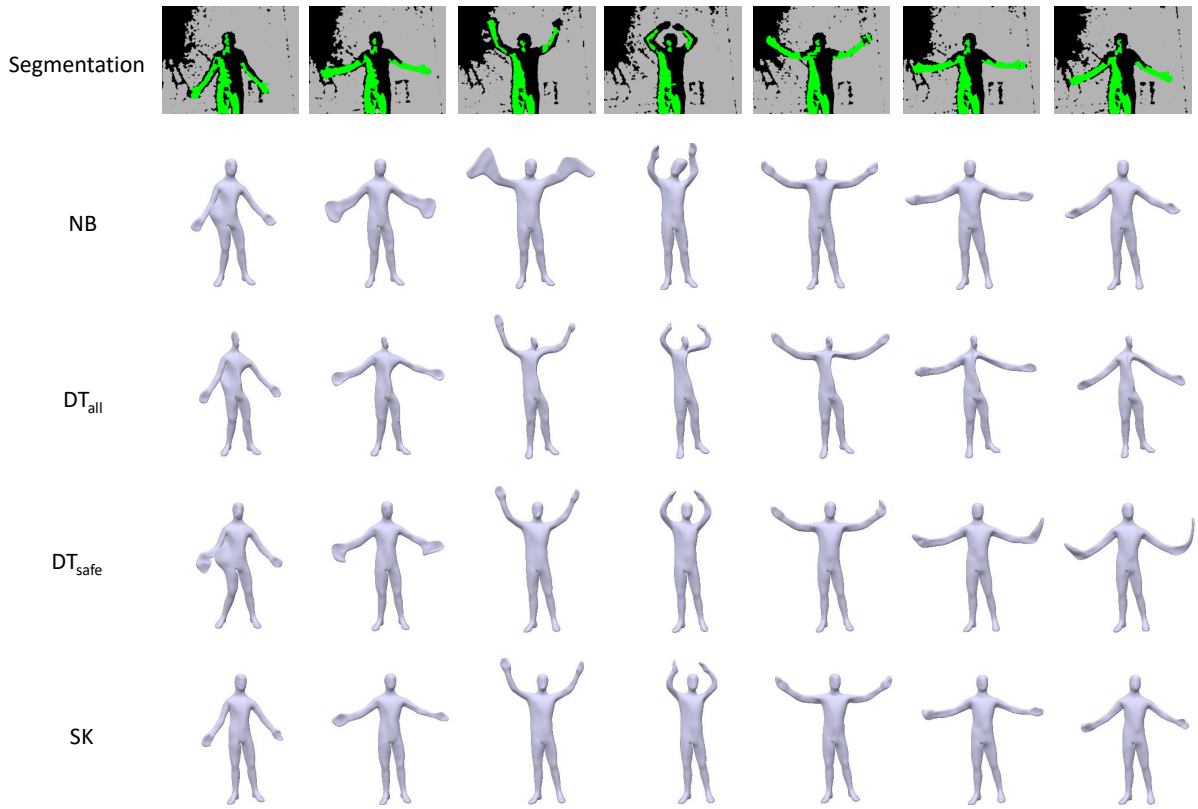


Figure 5.8: **Top row**: Some of the depth images used for tracking. Black represents null depth, pixels in gray observe the background and those in green the object. **Rows 2-5**: Results after convergence for the selected images and the different tested approaches.

Finally, SK achieves the best results, keeping the surface within the silhouette during the whole sequence without leading to artefacts or protusion of the surface.

The same experiment has been carried out indoors under good illumination conditions. In that case, the number of pixels with null depth is much lower and both DT and SK provide equally good results.

Tracking distant data

Lastly, we evaluate the performance of the compared methods when the distance between the data to fit and the mesh increases. To that end we have recorded an RGB-D sequence of a person moving his arms, in this case indoor and with good visibility conditions. Instead of tracking consecutive images as in the previous experiment, the initial mesh is aligned with the data of each image independently to evaluate how the different background terms help the model to converge to the right pose. Apart from that aspect, the procedure is similar to the one described above: an initial mesh is provided and the optimization problem is solved directly for this mesh without resorting to coarse-to-fine.

Six different images with a resolution of 120×160 are considered for the experiment. Results are shown in Figure 5.9. It can be noticed that NB works well for the first three images where data is closer to the initial mesh, but fails to push the arms forward for the last images and wrongly

Figure 5.9: Results after running the optimization problem to align a given initial mesh of a person with geometric data associated to different postures. The first tested images (left columns) observe a person whose posture is close to the initial mesh while the person's posture in the the last ones (right columns) is considerably different from that of the initial mesh.

deforms the main body to fit points which actually correspond to the left arm. $\mathrm{DT}_{\mathrm{all}}$ provides the best results since in this case the segmentations are almost perfect and the number of pixels with null depth around the person is quite low. Hence, the DT gradients help the model to converge to the right solution very quickly. On the other hand, $\mathrm{DT}_{\mathrm{safe}}$ provides results which are even worse than those of NB because some image borders have null depth and therefore attract the arms towards them. In this particular example the testing images could be processed to correct this deficiency but this is not possible in general (see the experiment in §5.D.7) and therefore we keep the original images to illustrate the drawbacks of this approach. Finally, SK provides accurate results but for the fifth image. This is a very challenging experiment for SK because it pushes the model inwards along its silhouette which, in this case, mostly leads to compressing the person's arms. Only the fingertips provide helpful gradients to bring the arms towards the main body. Despite this fact, SK outperforms NB and only fails for the fifth image where a small number of points of the almost-hidden arm causes the model to deform in an undesirable way (and the optimization to fall into a bad local minimum).

Computational Performance

The temporal performance of the different methods is also evaluated. We have chosen to analyse the runtimes of the tracking experiments where coarse-to-fine is not used. In this case a mesh with 1128 vertices is deformed to fit the data contained in depth images with QQVGA resolution (120×160). The time taken by a complete iteration of the Levenberg-Marquardt algorithm is measured for the three cases considered: fitting without background term (NB), with the DT-based background term and with our approach (SK). All the experiments have been run on

a single core of an Intel Xeon E5630 CPU at 2.53 GHz (code compiled for 32 bits and running on Windows 10).

As expected, the fastest method is NB whose iterations take on average 1.46 seconds. When the background terms are included, the average runtime of the LM iterations increases up to 1.62 seconds (DT) and 2.91 seconds (SK). As expected, our proposal is the computationally heaviest alternative because it includes an inner maximization process (raycasting) within the overall optimization.

5.D.8 Conclusions

This paper describes a novel background term that forces a 3D model to shrink within the visual hull of an object observed from one or multiple views. To demonstrate its superior performance over the popular distance transform-based formulation, we introduced a unified framework to address the problems of 3D reconstruction or non-rigid tracking with smooth surface models. Results demonstrate that our proposal enforces silhouette consistency more effectively than the distance transform. Specifically, it works better with real data that often include noise and uncertainty and which cannot always be segmented perfectly. This proposal could therefore be extended to RGB-based reconstruction and tracking systems.

Future work includes finding a better solver for this concave-convex optimization problem, which would optimize all variables jointly, and adapting the topology of the mesh during optimization.

6 Reactive Navigation

6.A Introduction

Autonomous navigation is becoming a key aspect/technology for modern society. The development of machines, vehicles or robots, that can travel autonomously without the need of human intervention is compelling for multiple reasons. From the point of view of manufacturing, endowing robots with such capability would increase efficiency and flexibility, while probably reducing costs at the same time. Regarding the transport system, autonomous vehicles will represent a paradigm shift. They will improve our quality of life, freeing us from driving and parking, and will reduce the number of traffic accidents (which are mostly caused by human mistakes and negligence). Mobile robots currently allow us to explore and monitor remote places that humans cannot reach: other planets, oceanic trenches, caves, warfare scenarios, etc [35, 166]. Furthermore, mastering autonomous navigation is leading to the emergence of new products and services that were not conceivable before. There already exist robots that can clean the floor of our house [21] or mow the lawn autonomously [167]. There are also projects of robots working in museums or airports as assistants to visitors [68]. These are just a few examples (out of many) of how relevant autonomous navigation is becoming nowadays.

Autonomous navigation encompasses two distinct capabilities. First, it involves motion planning, which consists in finding a route from a departing point to a given destination. To do so, this process requires previous knowledge of the environment, some sort of map. Moreover, it is not only important to find a feasible route but to find the best one (either fastest, safest, etc.), since there are normally many possible ways to go from one location to another. Second, the vehicle or robot should be able to react to the obstacles it encounters during its journey (typically people or other vehicles which are in constant motion and therefore do not appear in maps). Thus, reactive navigation receives data from different sensors and modifies the original plan in real time to avoid collisions. Any advanced algorithm of autonomous navigation is composed of these two blocks, following the so-called *hybrid architecture* [168].

Autonomous navigation can be implemented for different types of robots, and would have different requirements to fulfill in each case. Underwater vehicles might take marine currents into account to increase their range and speed, and drones will need to incorporate more complex 3D reactive strategies to guarantee safe operation [169, 170]. Besides, it is also important to con-

sider the mechanical constraints of the robot, and whether these are holonomic or nonholonomic. Holonomic constraints are those that only involve position variables, while nonholonomic constraints involve velocities. For example, planar motion is a simple holonomic constraint while the motion of wheeled vehicles is nonholonomic since the wheels cannot move laterally (assuming there is no slippage).

6.B Contributions

In our work we focus on terrestrial mobile robots, and particularly on those moving on flat surfaces, i.e. with planar motion. In §6.C we present a 3D extension of the reactive navigation algorithm described in [10]. We propose to model the shape of a robot as a set of prisms sorted in height, and group the detected 3D obstacles in the corresponding height bands/levels. Thus, the 3D reactive navigation problem is solved by combining several 2D reactive navigators into a unique space of search where all potential collisions between the robot and the 3D obstacles are taken into account. Both holonomic and nonholonomic constraints are considered by defining path or trajectory families that implicitly fulfill these constraints (as described in [10]). The resulting algorithm is tested with different robotic platforms in various environments. Results demonstrate its effectiveness to drive the robot following a sequence of random destinations in dynamic (and sometimes tight) environments. The autonomous navigation tasks were completed without human intervention in most tests, and the incidents observed where mainly due to unnoticed obstacles or slippage of the wheels, which causes the robot to move in an undesirable way.

6.C Efficient Reactive Navigation with Exact Collision Determination for 3D Robot Shapes

Mariano Jaimez, Jose-Luis Blanco and Javier Gonzalez-Jimenez

Published in the International Journal of Advanced Robotic Systems, 2015.

Efficient Reactive Navigation with Exact Collision Determination for 3D Robot Shapes

Mariano Jaimez, Jose-Luis Blanco and Javier Gonzalez-Jimenez

Abstract

This paper presents a reactive navigator for wheeled robots moving on a flat surface which takes both the actual 3D shape of the robot and the 3D surrounding obstacles into account. The robot volume is modelled by a number of prisms consecutive in height, and the detected obstacles, which can be provided by different kinds of range sensor, are segmented into these height bands. Then, the reactive navigation problem is tackled by a number of concurrent 2D navigators, one for each prism, which are consistently and efficiently combined to yield an overall solution. Our proposal for each 2D navigator is based on the concept of the "Parameterized Trajectory Generator" which models the robot shape as a polygon and embeds its kinematic constraints into different motion models. Extensive testing has been conducted in office-like and domestic environments, covering a total distance of 18.5 km, to demonstrate the reliability and effectiveness of the proposed method. Moreover, additional experiments are performed to highlight the advantages of a 3D-aware reactive navigator. The code is available under an open-source licence.

6.C.1 Introduction

Reactive navigation is a crucial component of almost any mobile robot. It is one of two halves which, together with the path-planner, make up a navigation system according to the commonly used *hybrid architecture* [171]. Within this scheme, a reactive navigator works at the low-level layer to guarantee safe and agile motions based on real-time sensor data.

Traditionally, due to the lack of affordable 3D sensors and the limited computational resources available, reactive navigators have relied on two strong assumptions:

- The world is considered to be two-dimensional. Since robots usually move on a flat surface, this implies that the third dimension (height) is ignored.
- The robot shape is simplified by a polygon or circle projected onto the 2D world.

These two simplifications force the reactive algorithm to adopt the worst-case scenario, that is, to work with the most restrictive section of the robot and the nearest obstacle detected in each direction. This limitation can complicate or even impede many robotic platforms from carrying out their tasks. In the most general case, the 2D reduction overconstrains the robot motion, marking as unfeasible some paths through which the robot could actually navigate. This effect arises when the robot does not have a constant vertical section, and it is particularly detrimental for robotic platforms equipped with a manipulator. In this specific case, any purely 2D approach would not allow the robot to place any object on any horizontal surface (e.g. a table) because the robotic arm and the surface would be superimposed on a 2D projection and would represent a collision for the 2D navigator. With the recent emergence of depth cameras and the current computational resources on board robots, these assumptions are no longer justified.

In this work we address the problem of planar navigation in indoor environments with a reactive navigation system which considers both the 3D shape of the robot and the 3D geometry of the environment (Figure 6.1). The proposed reactive navigator is based on the concept of

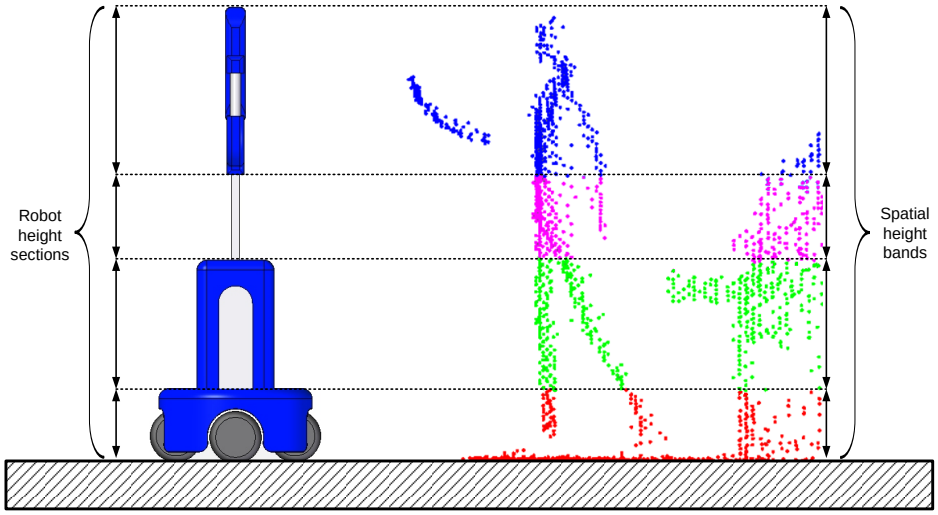


Figure 6.1: 3D obstacles are sorted in height bands according to the 3D shape of the robot.

"Parameterized Trajectory Generator" or PTG [10], a robust and effective 2D reactive navigator that models the robot shape as a polygon and embeds its kinematic constraints into different motion models. The contribution of this paper consists of extending such work to overcome the limitation of modelling the robot in 2D:

- The robot volume is now modelled as a number of prisms consecutive in height. Collisions are evaluated considering those exact prisms, including predicted robot orientations according to a number of path families, unlike many existing approaches which take the conservative circular-robot approximation and only consider circular paths.
- 3D obstacles coming from an arbitrary number of range sensors can feed the reactive navigator.
- The new 3D information is merged consistently and efficiently to yield an overall solution.

This generalization, called 3D-PTG navigator, has been extensively tested in varied and challenging scenarios. Two different robots, Giraff (Figures 6.4 and 6.5) and Rhodon (Figures 6.5 and Figure 6.12), equipped with radial laser scanners and RGB-D cameras, have been employed for the experiments. Overall, more than 20 hours of navigation are analysed, during which the robots covered a distance of 18.5 km in both office-like and house-like scenarios.

This paper is divided into eight sections. Section 6.C.2 describes the state of the art in reactive navigation. A brief summary of the 2D-PTG navigator is presented in §6.C.3. Its generalization to the 3D world is described in §6.C.4, and all the algorithm steps are explained in §6.C.5. Implementation details are given in §6.C.6 and the experiments are presented and analysed in §6.C.7. Results are divided into four subsections: two of them are intended to demonstrate the robustness of our approach, while the other two show the advantages of a 3D navigator as against the classic 2D approach. Finally, conclusions are discussed in §6.C.8.

The code has been added to MRPT [32] and is available under an open-source licence. Two demonstration videos of our approach, together with the code, can be found here:

`http://mapir.isa.uma.es/mjaimez`

6.C.2 Related Work

The first example of reactive navigation was probably that of the tortoises of Walter [172]. From then on, reactive navigation has been well studied, and some authors like Brooks [168] have defined it as the lowest hierarchical layer of a robotic motion control. The first successful methods, such as VFF [173], VFH [174] and VO [175], enabled robots to advance toward a given target while avoiding the obstacles encountered along their path, but they ignored both the robot shape and its kinematic constraints. Afterwards, reactive algorithms started to overcome these simplifications, solving the navigation problem in a velocity space where kinematic and dynamic constraints can be easily considered (typically, speed and acceleration limits). Within this category, Simmons [176] proposed to compute the optimal motion command in a curvature-velocity space where translational and rotational velocities are represented independently. In a similar way, the "Dynamic Window Approach" (DWA) [177], which was arguably the most successful strategy of this kind, minimizes an energy function to obtain the best motion command regarding the reachable obstacles and velocities within a short time interval. DWA is still in use today as a local planner in the popular Robot Operative System (ROS) [57] "navigation stack". These two approaches [176, 177] impose the feasible trajectory to be composed of circular arcs Thus, non-holonomic restrictions are also regarded although, on the other hand, both the robot and the obstacles are still supposed to be circular. This circular shape assumption is also made in the extension of the Velocity Obstacle method [178].

Later improved solutions incorporated the robot shape into the reactive navigator. Minguez and Montano [78] defined the Ego-Kinematic Transformation (EKT): a mathematical procedure to transform the 3D configuration space into a new 2D space which implicitly contains the robot shape and its non-holonomic constraints. In this reduced space, the robot is a free-flying point and any holonomic obstacle-avoidance method can be used to compute the solution. However, this approach still has a shortcoming: only circular paths are considered. This methodology was extended by Blanco *et al.* [10] with the generalization of path models through a novel approach called “Parameterized Trajectory Generator” (PTG). With this tool, several customized path models can be used in the reactive navigator and, at each iteration, the best one is selected according to some specific criteria, such as the collision-free distance for the selected movement, the minimum distance from the path to the target, etc. It must be noted that the robot becoming a free-flying point in this space comes at the cost of having a different set of obstacles in the transformed space, even for real stationary obstacles. However, as will be seen experimentally, this obstacle transformation can be made extremely efficient by means of precomputed look-up tables.

In this context, reactive navigators were effective enough, but they still had to assume that the world was 2D. Nonetheless, the improvement and availability of 3D range sensor during the last few years have made it possible to realistically tackle the problem of navigating in 3D environments. Most of the new approaches are based on processing 3D point clouds and use the resulting information to execute a 2D navigator. For example, the solution proposed by Surmann *et al.* [179] consists in scanning the environment with a tilting laser and extracting semantic information (planes) which is projected onto the floor plane and utilized by a 2D navigator. Holz

et al. [180] also use a tilting laser to generate 3D point clouds, which are processed to obtain the "2D Obstacle Map" and "2D Structure Map". The former contains the minimum distance in each scan direction (i.e., closest obstacles) and is exploited by the reactive navigator, while the 2D Structure Map contains the maximum distance in each scan direction (i.e., furthest obstacles) which is likely to correspond to the environmental bounds and is used for robot localization. In contrast, Marder-Eppstein *et al.* [181] proposed to store the 3D data in a voxel grid: a 3D occupancy grid whose cells are marked as occupied, free or unknown. The robot navigation is controlled by two modules: the "global planner" and the "local planner". The "global planner" creates a high-level plan for the robot to reach the goal location and the "local planner" is a reactive navigator based on the aforementioned DWA [177]. No information is given about how the 3D voxel grid is interpreted by the 2D reactive navigator. Finally, the work recently proposed by Gonzalez-Jimenez *et al.* [182] addresses the problem of adding the 3D information provided by an RGB-D camera to a reactive navigator which was designed to work with radial laser scanners. To this end, they propose to adapt the Kinect depth image into a virtual 2D scan which, in turn, encapsulates the 3D world information.

From a different point of view, 3D navigation is also studied for legged robots and humanoids [183, 184, 185]. In these cases, point clouds are always analyzed to extract semantic information, which is more convenient for the gait of this type of robot. In general, a 3D representation of the world has proved to be advantageous in many other aspects of the robot navigation, e.g. in localization [186].

6.C.3 Reactive Navigation based on PTGs

For the sake of completeness, this section summarizes the PTG-based reactive navigator upon which our proposal is built (more details can be found in [10]). The PTG-based reactive navigator is based on a mathematical transformation that reduces the dimensionality of the Configuration Space (C-Space) [187] from 3D (x, y, ϕ) to 2D, incorporating in the transformation the geometrical and kinematical constraints of the robot. The robot thus becomes a free-flying point over this 2D manifold embedded in C-Space, and the collision avoidance problem is easier and

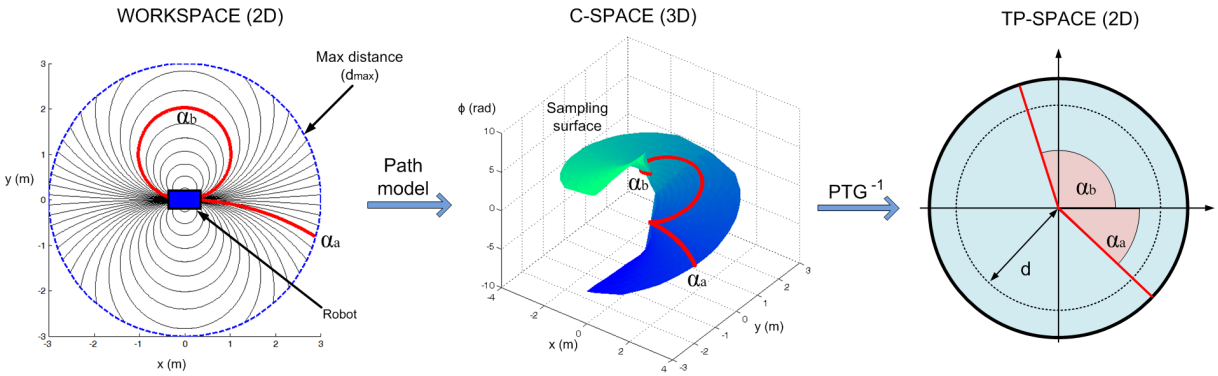


Figure 6.2: Mathematical transformations from the Workspace (2D) to the C-Space (3D) and then to the TP-Space (2D). Individual trajectories (α_a and α_b) form curves in the C-Space and are segments for a particular referenced angle in the TP-Space.

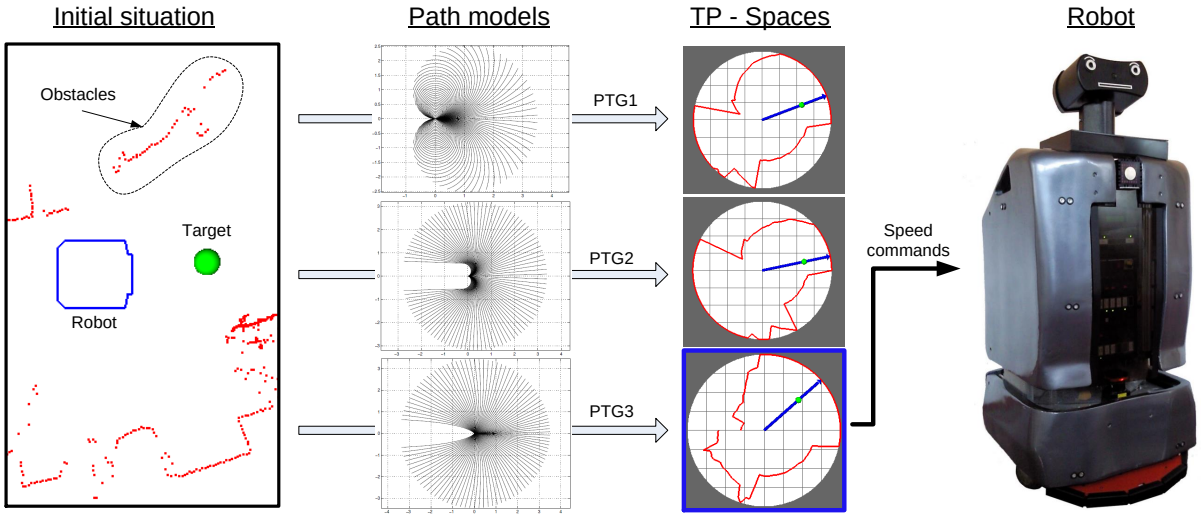


Figure 6.3: In this example three different path models are considered, leading to three different TP-Spaces. In both the Workspace and the TP-Spaces the target is marked as a green circle, while the obstacles are displayed in red. The best candidate for each TP-Space is shown with a blue arrow, and the best of all (PTG_3) is selected to provide the angular and linear robot velocities.

faster to solve. This dimensional reduction is accomplished by restricting the robot motion to one of a set of parametric path models which are compliant with the robot kinematics (e.g. circular paths, as shown in Figure 6.2). The set of all possible robot poses according to any path model constitutes a 2D manifold, referred to as *sampling surface* (S), embedded in the general C-Space. The basic idea behind PTG-based navigation is to map those manifolds by means of 2D *Trajectory Parameter Spaces*, or TP-Spaces. The mathematical transformation between the TP-Space and the C-Space is formulated by a *Parameterized Trajectory Generator* or PTG, a smooth mapping of TP-Space points into C-Space poses according to a certain path model. TP-Spaces are expressed in polar coordinates, where the angle α corresponds to an individual path from the family and the radius d indicates the normalized distance travelled along that path (Figure 6.2). The region of interest in a TP-Space is the circle of unit radius, that is, the subspace $A \times D \subset \mathbb{R}^2$, where $A = \{\alpha \,|\, \alpha \in [-\pi, \pi]\}$ and $D = \{d \,|\, d \in [0, 1]\}$. Therefore, kinematically compliant paths become straight lines in TP-Space and the robot motion can be guided with simple holonomic methods like VFF [173] or ND [188], disregarding the robot shape and non-holonomic constraints.

Since several path models are included in the reactive system, several TP-Spaces are built. The mathematical transformation between the C-Space and the TP-Spaces is done by the inverse PTG function, defined as:

$$\begin{aligned} \text{PTG}^{-1} : S \subset \mathbb{R}^3 &\rightarrow A \times D \subset \mathbb{R}^2 \\ \{(x, y), \phi\} &\rightarrow (\alpha, d)\,. \end{aligned} \tag{6.1}$$

Equation (6.1) transforms both obstacle points and the target from the C-Space to the corresponding TP-Space, using path models such as those shown in Figure 6.3. In the resulting TP-Spaces, the robot becomes a free-flying point because its shape and its kinematic constraints are embedded into the mathematical transformation and, therefore, any holonomic method can

be applied to get the best path $\hat{\alpha}^i$ in the i-th TP-Space. All the $\hat{\alpha}^i$ are subsequently evaluated and compared following some heuristic criteria which take into account the collision-free distance for the selected movement, the minimum distance from the path to the target, whether the robot will be heading to the target or not, etc. The result of this process will be the most suitable movement $\hat{\alpha}$ given the obstacles, the target and the path models being used. Finally, the speed commands associated with $\hat{\alpha}$ are calculated and sent to the robot. This sequence is repeated at a given frequency, typically higher than 20 Hz, such that the robot can move smoothly. In summary, the PTG-based reactive navigator has two inputs –the target relative pose and sensor data– and generates one output: the velocity command for the robot. The sequence of steps to compute the velocity commands is as follows:

1. For each path model, using its corresponding PTG, transform the obstacles and the target to the associated TP-Space.
2. For each path model, apply a holonomic reactive method to get the best path $\hat{\alpha}^i$ in the TP-Space.
3. Select the best path $\hat{\alpha}$ among the candidates $\hat{\alpha}^i$ obtained from the different TP-Spaces.
4. Compute the linear and angular velocities and send them to the robot motor unit.

6.C.4 PTG-based Reactive Navigation in a 3D World

The main limitation of a 2D navigator, such as the one described above or any of those reviewed in §6.C.2, is that both the robot and the world are assumed to be 2D. More specifically, the robot

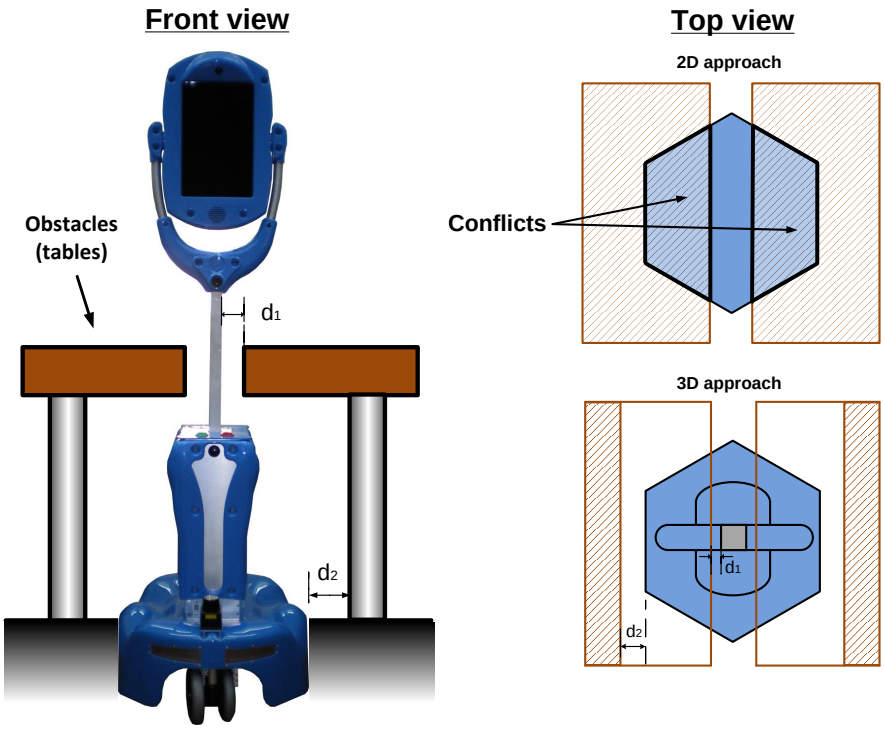


Figure 6.4: Example of how the 2D approach limits the reactive navigator performance.

Figure 6.5: Example of 3D geometric models of the Giraff robot (right) and Rhodon (left).

section is considered to be constant and the detected obstacles are projected onto the floor plane, even if they are provided as 3D data by RGB-D cameras or lidars. This happens to be a valid simplification under the assumption that the robot has roughly the same horizontal profile all the way from bottom to top. However, many wheeled robots have a non-constant section (please visit `http://robots.ros.org` to find many examples like PR2, Gostai Jazz, Amigo, etc.), in which case the 2D solution is suboptimal for two reasons: it takes the biggest section of the robot and also the closest obstacles regardless of their height position in space. Figure 6.4 illustrates this limitation with an obstacle configuration for which a 2D reactive navigator would fail even though the robot has enough space to pass through. This kind of situation is quite common in cluttered environments and demands the addition of the third dimension (height) to the reactive navigator in order to successfully handle it. For that purpose, we model the robot geometry through a set of prisms circumscribing the robot volume, as shown in Figure 6.5. Besides defining the 3D shape of the robot, it is also necessary to include the height coordinates of the obstacles or, more specifically, to sort them into height bands according to the height sections used to model the robot (Figure 6.1). Therefore, we decompose the 3D reactive navigator into N 2D navigators, being N the number of height sections that model the robot geometry. Each 2D navigator comprises an individual robot section and the obstacles in its corresponding height band. In order to obtain an overall solution for the robot, we combine the results for all the 2D navigators, as will be described later.

At this point, it is necessary to give a brief explanation of how obstacles are transformed into TP-Obstacles and what they represent (Figure 6.6). In the Workspace, obstacles are always considered to be points. Focusing on a single obstacle (or point), and given its coordinates, the robot shape and its location, we can calculate all the poses (x, y, α) in the C-Space which imply a

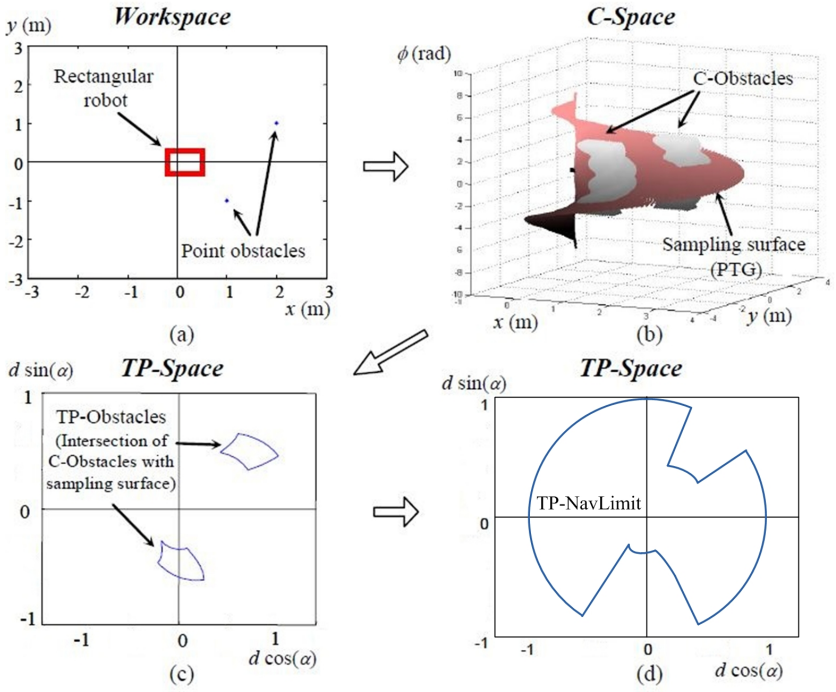


Figure 6.6: A pair of obstacles in the Workspace are transformed to the C-Space and, according to a path family, to the associated TP-Space.

collision between the robot and the obstacle. This set of poses forms a volume called C-Obstacle. In addition, it is useful to recall that in C-Space every path model is a sampling surface. Thus, TP-Obstacles are obtained by transforming the 3D intersection between C-Obstacles and the sampling surface to the TP-Space.

Formally, let $\sigma \in \mathbb{R}^2$ be a real obstacle in the Workspace, C-Obstacle(σ) its representation in C-Space and P a 3D point in C-Space. TP-Obstacles are defined as:

$$\text{TP-Obstacle}(\sigma) = \left\{(\alpha, d) \,\middle|\, (\alpha, d) = PTG^{-1}(P), \forall P \in \text{C-Obstacle}(\sigma) \cap S\right\} . \tag{6.2}$$

Nevertheless, only the closest obstacle for each path α is relevant here as it marks the maximum distance the robot can travel along that path, always from the origin, without collision. The set of closest obstacles in TP-Space is called TP-NavLimit. If there is no obstacle along the path α, its TP-NavLimit is set to 1 (the maximum distance in the normalized TP-Space), that is:

$$\text{TP-NavLimit}(\alpha) = \min\left\{1, d_m\right\} , \tag{6.3}$$

where d_m is the minimum distance of the pairs $(\alpha, d) \in$ TP-Obstacles.

Returning to the 3D reactive navigator, the same process is followed for the N height sections of the robot to obtain a TP-Space with N sets of TP-NavLimits, each indicating the maximum distance that the robot at that height section can travel along a given path model. Hence, the most restrictive TP-NavLimits are used to build the TP-Space for each α (Figure 6.7), that is:

$$\text{TP-NavLimit}(\alpha) = \min_n \left\{\text{TP-NavLimit}_n(\alpha)\right\} , \quad n \in \{1, ..., N\} . \tag{6.4}$$

We want to remark that calculating the minimum of all TP-NavLimits is not equivalent to projecting the most restrictive 3D obstacles onto the floor. Given that the TP-NavLimit of each

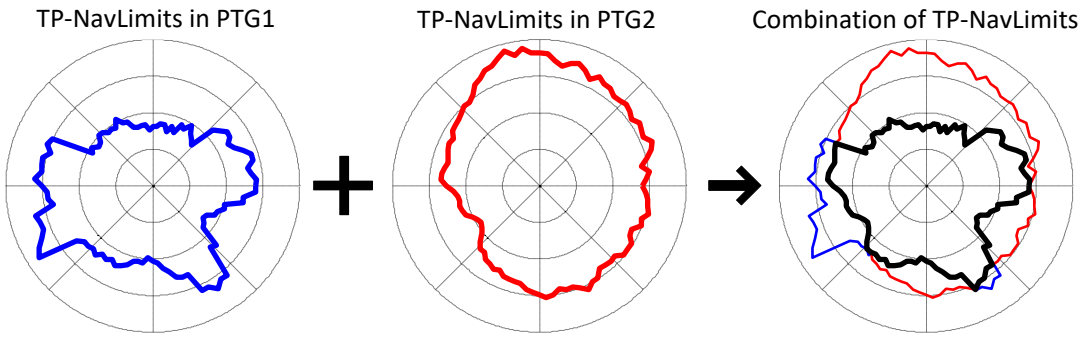


Figure 6.7: Example of combination of TP-NavLimits for a robot with two height sections. PTG1 is related to the first height section (blue) and PTG2 to the second one (red). The resulting TP-NavLimits (black) are calculated as the minimum of the TP-NavLimits associated with each individual PTG.

individual robot section contains information about how far this part of the robot could travel in the 3D world according to some path models, the minimum of them shows how the whole robot could navigate in the same 3D world, since it encompasses the motion restrictions of every part of it. Thus, all these restrictions correspond to poses of the robot that would actually imply collisions with the environment, whereas the typical 2D obstacle projection is prone to creating motion constraints that do not correspond to any potential collision in the real world, hence overconstraining the robot motion.

6.C.5 The 3D Reactive Navigation Framework

In this section we describe the overall operation of the reactive navigator. It consists of a number of steps which are executed periodically at a given frequency (Figure 6.8). The inputs to the reactive system are the obstacles and the relative location of the target. Different kinds of sensors providing 3D obstacle points can be used simultaneously, typically laser scanners and RGB-D cameras. As previously mentioned, these 3D obstacle points are sorted according to the different height sections employed to model the robot volume.

Before applying the PTG transformations, a module called *Short-Term Memory* (STM) stores the position of close obstacles that might eventually become unseen by the robot sensors if they enter into their blind zone. This is particularly relevant for RGB-D cameras which have a narrow field of view and cannot detect obstacles at short distances. The STM module is implemented by N local occupancy grids centred at the robot pose onto which the 3D points within each slice in height are projected. The appropriate grid and cell sizes depend on the sensors used, the accuracy of the localization estimate, how cluttered the environment is, etc. The outputs of the STM block are sets of *virtual obstacles* whose coordinates are generated from those of the occupancy grid cells. These virtual obstacles are merged with the real ones coming from sensors and passed to the TP-Obstacle builder. Detailed information about the occupancy grids, their working principle and how they are implemented can be found in [182].

All the obstacles, real and virtual, are converted into TP-Obstacles for a number of path models and also for each height level. This results in $K \times N$ sets of TP-Obstacles and, subsequently, in $K \times N$ sets of TP-NavLimits. Then, the N sets of TP-NavLimits corresponding to each

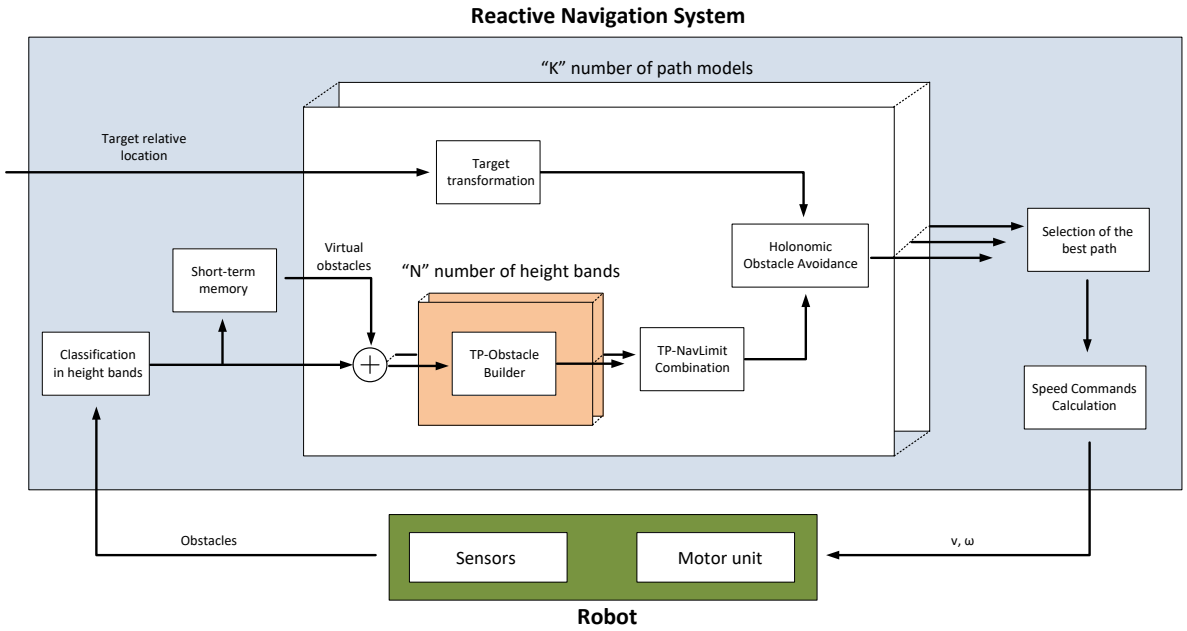


Figure 6.8: Scheme of the 3D reactive navigator with K different path models and N height sections.

path model are combined, as explained in §6.C.4, yielding K TP-Spaces representing the robot navigability for each path model. Concurrently, the relative target location is also transformed to these TP-Spaces, where any holonomic method can be run (e.g. VFF [173] or ND [78]), to get the most suitable path $\hat{\alpha}^i$ for each path model i. The best path candidate $\hat{\alpha}$ among all is then the one that maximizes an objective function that trades off several navigational criteria:

$$\hat{\alpha} = \arg\min_{\alpha} \sum_{i} w_i f_i(\hat{\alpha}^i) \tag{6.5}$$

with w_i being weighting coefficients and f_i factors which measure:

- f_1: The collision-free distance of each candidate (in TP-Space).
- f_2: The angular distance in the TP-Space between the target and the candidate.
- f_3: The minimum distance between the target and the path candidate.
- f_4: How different the new (tentative) and the previous speed commands would be (to soften the robot motion).

Finally, the linear and angular velocities are derived from $\hat{\alpha}$ and sent to the robot motion control unit.

6.C.6 Experimental Setup and Implementation Details

The 3D reactive navigator has been intensively tested for months in several scenarios to demonstrate its proper functioning. The Giraff and Rhodon mobile robots (Figure 6.5) have been utilized for the experiments, as their heterogeneous profiles are appropriate to test the 3D reactive navigator's performance. In particular, the Giraff robot has been deployed and utilized for

more than a year in several real apartments in Spain as part of the EU project GiraffPlus [189] using the proposed method to reactively navigate between nodes of a pre-established roadmap.

Giraff is a differential wheeled robot and has been equipped with a Hokuyo URG-04LX-UG01 laser and a PrimeSense Carmine 1.09 RGB-D camera, both facing forwards. Rhodon is a heavier differential wheeled robot equipped with two laser scanners (one Sick LMS200 and one Hokuyo UTM 30-LX, facing forwards and backwards respectively), and one Kinect camera placed at the top of the robot and tilted downwards with an angle of 50 degrees. 3D points provided by all the range sensors are expressed with respect to the robot coordinate system and then merged before feeding the reactive navigator. In this preprocessing stage, the fused point cloud is downsampled, retaining only the most restrictive points at each height band. The frequency at which sensor data are read is adjustable; in our experiments it ranges from 10 Hz for the Hokuyo URG-04LX-UG01 (its maximum) to 30 Hz for the remaining sensors. Due to the lack of 3D sensory information at their back (Rhodon does contain a Hokuyo facing backwards but it is only used for localization), the robots are not allowed to move backwards during the experiments.

Configuration of the 3D Reactive Algorithm

First, we need to define the height sections that model the robot geometry (Figure 6.5). In these experiments we have modelled the Giraff robot with four consecutive prisms and Rhodon with five prisms (see §6.C.4 for further details). Second, path models and their characteristics have to be specified; in our case three different path models are considered (Figure 6.9): circular arcs, trajectories with asymptotical heading and trajectories with a minimal turning radius (see [10] for more details about their mathematical definition).

Furthermore, the holonomic method can be chosen from two options: VFF [173] or ND [78], and their associated parameters can be customized too. We opted for the ND method because, in general, it outperforms the VFF. Finally, the reactive loop frequency is set to 20 Hz, which is fast enough to react to incoming sensor data without overloading the processor.

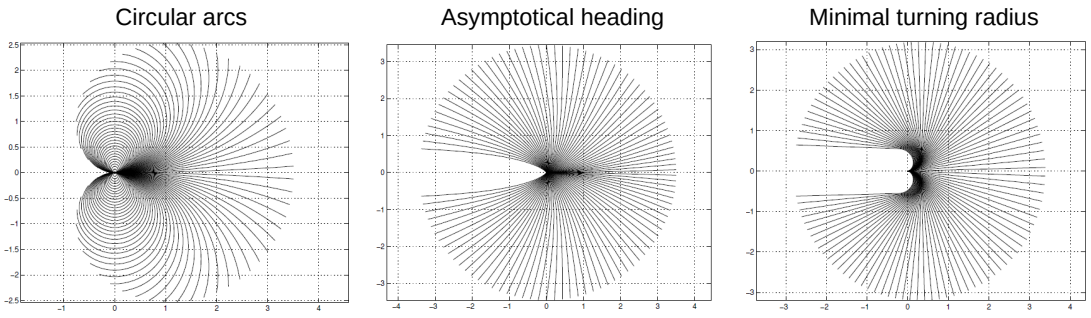


Figure 6.9: Set of path models considered in the experiments by the reactive navigator.

Speed Regulation and Recovery Behaviour

Linear and angular velocity commands sent to the robot come from the PTG associated with the selected path, but they can be modulated or rescaled without violating the kinematic constraints of that path.

As a general rule, the robot should move carefully when it is surrounded by obstacles, but this is not always granted by the PTGs. For this reason, the speed commands are adjusted taking into account the proximity of obstacles, with the frontal obstacles having a greater influence than those at the sides. This speed regulation, which allows us to increase the average robot speed while keeping the navigation safe, can be adjusted depending on the robot dynamics and the desired balance between robot agility and conservativeness.

On the other hand, a basic recovery behaviour is implemented in case the robot gets stuck. Knowing that backward movements are not permitted, the reactive navigator assumes that the robot is trapped when both forward movements and rotations are impeded. In such situations, the robot starts to move backward slowly until it finds a feasible movement from the reactive navigator. If it is unable to find a way out after a few seconds, it stops (to avoid possible backward collisions) and keeps waiting until the environment changes or the user takes control of the situation.

6.C.7 Experiments

A wide variety of experiments have been conducted. The first two sets are intended to study the performance of the 3D navigator under circumstances that are habitual in many robotic applications: navigation in an office-like environment and navigation at home. To validate our proposal, the other two sets of experiments include some specific and demanding situations that cannot be addressed without 3D knowledge of the world. In all the experiments, localization relies on wheel odometry and laser scanners, which feed a particle filter implementation of localization based on a metric map of the environment [60]. Those geometric maps were previously built for each environment by means of a simple ICP-based incremental registration of laser scans.

Computational Burden

In order to check the computational resources that the reactive navigator demands, we have tested how long one complete iteration of the reactive module takes on the Giraff robot, whose processor is an Intel i3 – 2310M 2.10 GHz with 4.0 GB of RAM. Considering four height sections and three different path models, the reactive iteration takes 4.8 milliseconds on a single CPU core, which implies a computational load inferior to 100 milliseconds per second for the 20 Hz implemented frequency. This leaves more than 90% of the CPU to the other robotic modules (localization, sensing, interface, etc.) which have to share the same computational resources.

We can compare this runtime with that of the 2D reactive navigator, i.e., considering only one height section to model the robot. In this case the reactive iteration takes 3.4 milliseconds, which implies that the 3D version is about 40% slower than the 2D for this particular configuration. As can be noticed, the runtime is sublinear with the number of height sections because, within one

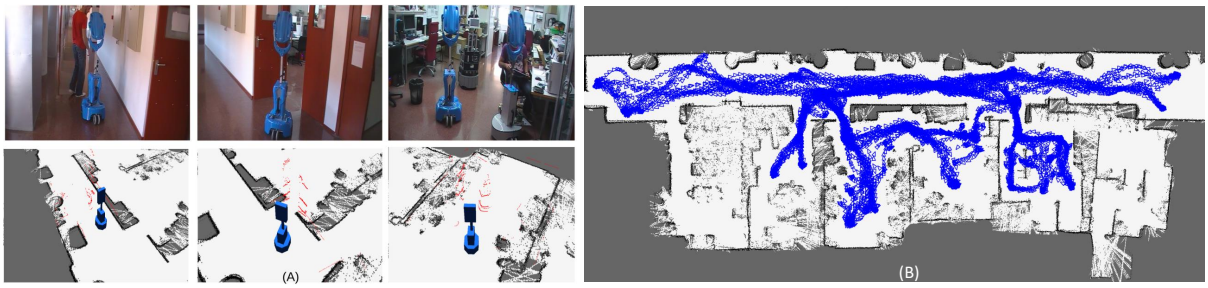


Figure 6.10: A) Some snapshots of the Giraff robot navigating in an office-like environment (first row) together with a virtual representation of the robot and the obstacles detected by the sensors on board (second row). B) The trajectory described by the robot during one of the missions.

complete iteration of the 3D reactive approach, there are only a few steps that are executed for each height section of the robot model (see Figure 6.8).

Navigating in an Office-Like Environment

The Giraff robot has navigated autonomously around our lab floor for more than a year, mainly in an area which includes a long corridor, our two-room lab and the two contiguous labs. This environment presents a wide variety of static and dynamic obstacles that the robot has to detect and dodge (Figure 6.10). Apart from the geometric map for localization, the robot is provided with a topological map from which navigational targets are generated randomly.

Table 6.1 shows the results of some of these navigational missions where navigational data were monitored to evaluate the reactive navigation performance. Overall, the robot has travelled 13.5 km with an average speed of 0.32 m/s and a top speed of 0.7 m/s. The incidents that took place during these sessions were classified into two categories. The first one, called *minor incidents*, refers to smooth contacts or grazes that the robot itself can manage and solve autonomously without human intervention. The second category comprises those cases where the robot gets stuck and cannot resolve the situation by itself, needing human intervention. The incidents recorded are explained according to their nature:

- Minor incidents are due to wheel slippage and the relatively high response time of the whole system. Wheel slippage depends on the robot mechanics and the surface it is moving on, and causes the robot to move in an uncontrolled way. On the other hand, we have checked that the elapsed time from the moment an obstacle is detected until the time the robot starts to react to it is slightly higher than 0.5 seconds. This latency is the sum of a number of small response times, inertias, and communication delays between modules of the robotic architecture.

- Human intervention is mainly needed when the robot gets stuck due to unnoticed obstacles, most of them being small pieces or part of objects lying on the floor.

Figure 6.10 gives an idea of how the robot has been wandering during the experiments. The point map shown was built with a laser scanner and, hence, white areas may contain obstacles invisible to the laser scanner (tables, chairs or other objects) but not to the RGB-D camera. This explains why in the trajectory plot there are apparently free areas that the robot did not visit.

Duration (s)	Distance travelled (m)	Average speed (m/s)	Minor incidents	Human intervention
5330	1771	0.332	0	0
1352	399	0.295	0	0
597	184	0.308	1	0
564	179	0.317	0	0
1500	479	0.319	1	0
1737	535	0.308	0	0
4749	1435	0.302	1	1
3974	1206	0.303	1	1
3764	1207	0.321	1	0
1614	537	0.333	2	0
5450	1811	0.332	4	0
4975	1587	0.319	5	1
1840	601	0.327	3	0
5228	1571	0.300	1	1
Overall Results				
11.85 h	13.5 km	0.316	20	4

Table 6.1: Results of the experiments in an office-like environment.

Navigating in a flat

As mentioned, the Giraff robot has been deployed for more than a year in several flats in Málaga (Spain) as part of the objectives of the EU project GiraffPlus [189]. The flat selected for the experiments has four rooms and presents a narrow navigable space with a reduced margin for manoeuvre (Figure 6.11). As a consequence, the Giraff maximum speed was lowered to 0.4 m/s in this case. The followed procedure is similar to that explained in the previous section: both metric and topological maps were built and provided to the robot which used them to navigate autonomously. Results are listed in Table 6.2. We can observe that the average speed has decreased, which is not only a consequence of the maximum speed reduction, but is also caused by the many situations in which the robot performs a pure rotation to turn round, contributing zero to the average velocity (see Figure 6.11 B). The incidents that took place during these tests are explained following similar criteria to those mentioned in the analysis of the previous set of experiments:

- Minor incidents are mainly due to the high response time of the robot working loop, which becomes more relevant when moving in tight spaces.
- Only one human intervention was necessary because there were not many objects lying on the floor.

Navigation with an outstretched robotic arm

During this experiment, Rhodon is commanded to perform a task that would be unfeasible using a 2D navigator. This task consists in visiting different desks with a robotic arm in a stretched position, emulating the process of collecting and delivering objects autonomously, but omitting

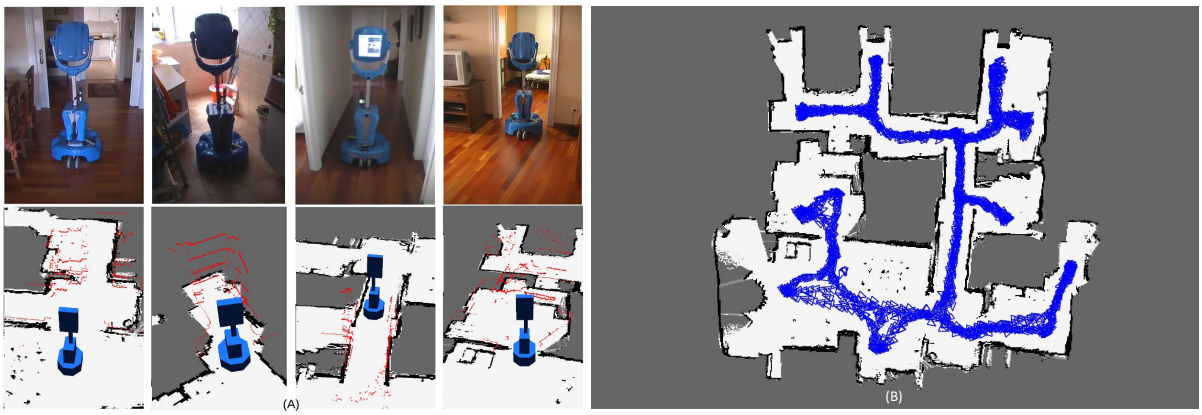


Figure 6.11: A) Some snapshots of the Giraff robot navigating in a flat (first row) together with a virtual representation of the robot and the obstacles detected by the sensors on board (second row). B) The trajectory described by the robot during one of the missions.

Duration (s)	Distance travelled (m)	Average speed (m/s)	Minor incidents	Human intervention
3003	448	0.149	2	0
4331	644	0.149	2	0
3845	571	0.149	1	1
4241	671	0.158	0	0
4263	654	0.153	1	0
4321	716	0.166	1	0
4384	731	0.167	3	0
3207	574	0.179	1	0
Overall Results				
8.78 h	5.01 km	0.159	11	1

Table 6.2: Results of the experiments in an four-room flat.

the manipulation phase as it is outside the scope of this work. This is an illustrative example of a robotic application that necessarily requires 3D knowledge of the environment and the robot. As PTG-based navigation does not support changeable robot shapes, the arm is maintained at the same position during the whole navigation so that the same five prisms model the robot shape properly throughout this test (if the arm moved, its corresponding height section could be modelled according to the range of motion of the manipulator). We specify a blind pixel region for Kinect and neglect all the points observed at that region of the depth images since, given the camera pose on the robot (see Figure 6.5), the robotic arm is necessarily observed by the Kinect and would be considered as an obstacle otherwise. Taking the weight ($\sim$50 kg) and height (1.8 m) of Rhodon into account, its maximum linear and angular speeds have been set to 0.4 m/s and 45 deg/s respectively. During the experiment, the robot has travelled 160 m around our lab visiting a total of seven different desks several times at random (Figure 6.12).

Aside from accomplishing its task, the 3D reactive navigator has shown an improved behaviour with respect to the 2D version since the robot was able, for example, to turn round when surrounded by chairs, tables or boxes while the arm was moving above them. No incidents

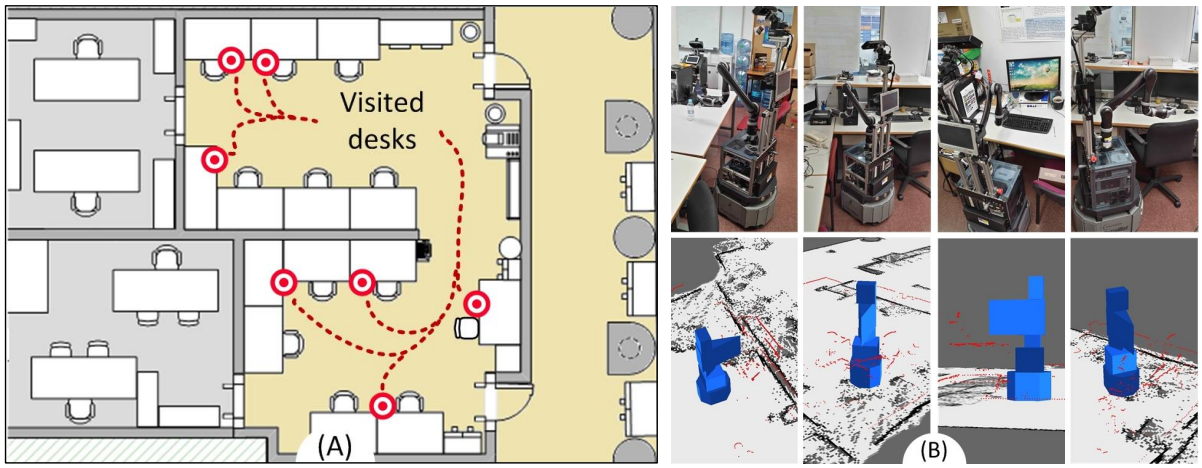


Figure 6.12: A) Schematic map of the lab and the desks that Rhodon visited during the experiment. B) Snapshots of Rhodon reaching its destinations or turning round surrounded by obstacles (first row) together with a virtual representation of the robot and the obstacles detected by the sensors on board (second row).

occurred during this test. In Figure 6.12 we can observe the final pose of the robot when it reached some of its destinations and how the robotic arm lay above the corresponding desks.

Testing the limits of the reactive navigator

A more extreme test to challenge the functioning of the 3D reactive navigator consists in commanding the robot to go through a contour which has the same profile as the robot itself. For this purpose we cut out a piece of fabric and placed it at the door frame (Figure 6.13). The experiment was carried out placing the robot about 5 meters away from the door at different positions and giving it a target outside of the lab. Sometimes we put an additional piece of fabric crossing the contour so as to check that the robot realized it could not go across it. The maximum velocity was set to 0.3 m/s and the short-term memory (STM) was not used (please see the demonstration video at the link attached in the introductory section).

We repeated this run 20 times and the robot always passed through the silhouette if the blocking piece of fabric was not present and always stopped otherwise. Quite frequently (about 50% of the time), however, the robot slightly touched the cloth as the field of view of the RGB-D camera is not wide enough to sense the whole clearance when the robot gets close to the door (Figure 6.13). We also made some trials activating the STM, but we found that using it had a counterproductive effect. As the localization module has a precision of few centimeters, virtual obstacles are inserted in the map carrying such positioning errors, which on occasion prevents the robot from seeing the clearance or cause undesired jittery behaviour in the reactive navigator. In this case, the errors in localization are higher than the spatial margins to pass through the clearance and, hence, the STM becomes useless. Nevertheless, the STM would be a good solution as long as the robot pose was estimated more precisely.

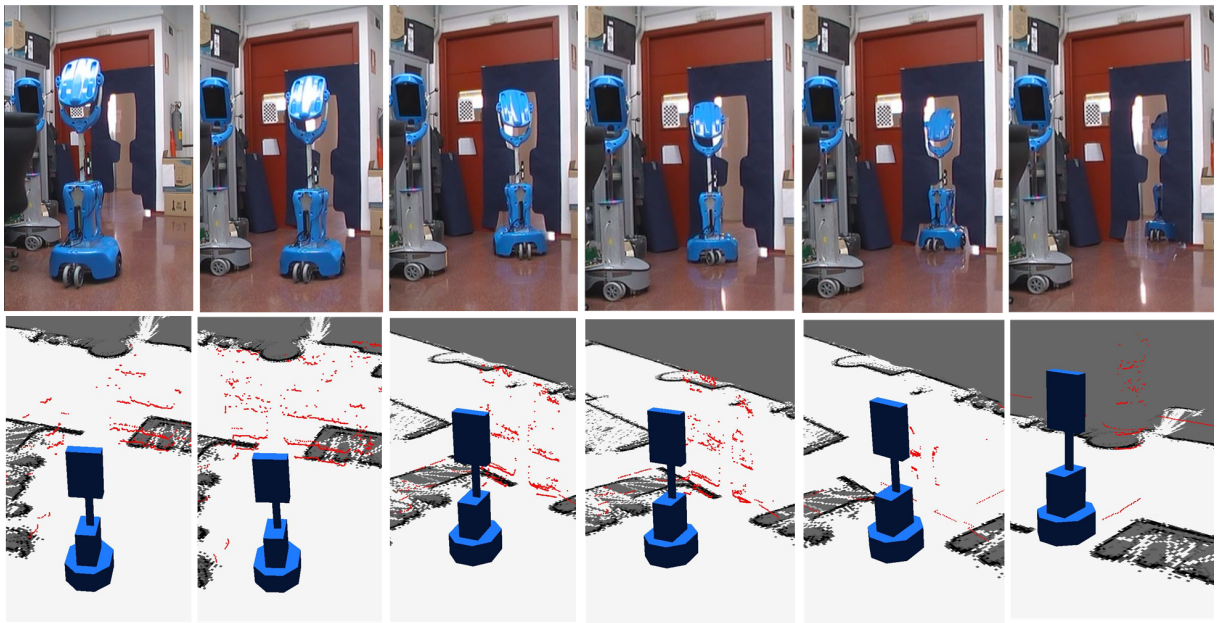


Figure 6.13: Real images (up) and representations of the robot, the detected obstacles and the map (down) showing how the robot goes through the contour.

6.C.8 Conclusions

We have presented a 3D reactive navigator that can be adapted to almost any robot moving on a flat surface. We achieve high levels of versatility and manoeuvrability, as only very weak assumptions have been made to formulate this 3D approach, namely:

- The robot can be properly modelled in three dimensions as a set of prisms.
- Measurements coming from different kinds of sensor can be directly merged and read by the reactive navigator, provided they are expressed as 3D points sets.

The robot is allowed to move according to several kinematically compliant path models. Two robots with heterogeneous height sections were chosen to test the reactive navigator in different environments. A fair number of experiments were conducted and the results support its proper functioning, although it may be conditioned by some factors. First, the robot mechanics has been shown to play an important role and can spoil the reactive navigator performance if the robot is not able to reproduce the motion commands quickly and accurately enough. Second, the coverage of the surroundings by the robot sensorial system is also a key factor that clearly delimits the quality of the reactive navigator. The number, type and placement of sensors needed for the robot to comprehensively sample its surroundings are aspects that require a great deal of attention. In this respect, we believe that active perception could be an effective solution to improve obstacle detection without demanding many sensorial resources. Although it has not been contemplated here, active perception represents a potential improvement and will be studied in future works.

7 Conclusions

Conclusions

In this thesis we have addressed distinct problems that lie at the interface between robotics and computer vision. As a common denominator, the proposed methods have exploited the geometric data provided by range sensors for motion estimation, reconstruction, tracking and navigation.

Range sensors have demonstrated to be a powerful alternative to traditional solutions based on monocular or stereo cameras. Knowing the geometry of the environment is often mandatory for many vision-related tasks and, as a consequence, range sensors present a genuine advantage over passive sensory systems. While the former directly provide such data, the latter entail depth estimation as a preliminar step, consuming valuable computational resources in the process. Moreover, range sensors are able to work under poor illumination conditions or even in complete darkness, where RGB cameras render useless. On the other hand, range sensors can only detect objects below a certain distance threshold and are prone to being affected by the solar radiation, which sometimes prevents their use for outdoor applications.

Next, we present individual summaries of the proposed methods, highlighting their pros and cons as well as possible lines of future work.

1. Regarding visual odometry, we have demonstrated that the camera motion can be estimated fast and precisely from depth images. The main drawback of the proposed method is its lack of robustness to moving objects, but this limitation could be overcome by minimizing the geometric residuals within a robust penalty function (as we have proposed in posterior works) or by extending the existing formulation to perform multi-frame alignment. From the point of view of its application, this method could be utilized to estimate the motion of those virtual&augmented reality devices equipped with depth cameras (e.g. Hololens) or, more generally, as the front-end of SLAM systems.

2. Likewise, inspired by the most recent advances in 3D visual odometry, we have developed a direct method based on dense scan alignment for planar odometry. Results show that our method outperforms the most popular techniques on scan-matching while having a lower runtime. Our approach requires scans to be at least piecewise differentiable, which prevents its use in outdoor environments composed of trees, plants and other scattered objects. Nevertheless, this limitation is not that relevant because motion in outdoor scenarios

is seldom planar and, hence, odometry based on 2D laser scanners is not really an option. Given its characteristics, this method is suitable for service or telepresence robots operating in office buildings, museums, hotels, homes, etc.

3. We have presented the first real-time algorithm for scene flow estimation with RGB-D cameras. We have made the code public so that all those robots or systems equipped with RGB-D cameras (and NVIDIA GPUs) can exploit it for different purposes. The main limitation of this method is that it works only for small displacements between the incoming frames. However, this is not a serious restriction in practice because it runs in real time and the image pairs to be aligned are very close/similar (save in the presence of very fast object). In order to widen the range of potential applications, this algorithm should be combined with additional post-processing strategies to simplify and refine the extensive information contained in the estimated motion field.

4. A completely different approach for scene flow estimation is described in §4.E. This method achieves very accurate results by jointly segmenting the rigid bodies that form the scene and their underlying rigid body motions. We demonstrate that a smooth motion-based segmentation is beneficial if compared to the standard binary approach when the scene contains non-rigid parts like people, animals, toys or other flexible objects. As a drawback, it is computationally very expensive (20 to 30 seconds running on GPU), which prevents its direct application in real-world scenarios. Given that scene flow is computed from rigid transformations, this method can be used to render "virtual images" for temporal interpolation between the aligned frames. Applying this process to an entire RGB-D sequence would result in a "slow motion" version of the original video.

5. We have tackled an uninvestigated problem: the joint estimation of odometry and scene flow. Our solution relies on a two-fold segmentation of the scene, dividing it into rigid geometric clusters that are, in turn, classified as static or moving elements. Identifying the static parts of the scene is paramount to compute a robust odometry, while the geometric clustering is essential to reduce the computational complexity of the problem. By virtue of this piecewise rigid formulation, our method achieves a runtime of about 80 milliseconds running on CPU, which is some orders of magnitude faster than most existing scene flow algorithms. Despite the promising results, there is still room for improvement since the method fails to distinguish static from moving parts when the former represent a small percentage of the scene ($< 50\%$). Possible solutions to this problem would involve multi-frame alignment or a more elaborate strategy for temporal regularization. This method would be useful for virtually any mobile system that requires some degree of autonomy and operates in a dynamic environment (since it works with RGB-D cameras, it would be constrained to indoor use).

6. We have also presented a new background term to enforce silhouette consistency within 3D reconstruction and tracking systems. This term overcomes important limitations of the popular formulation based on the distance transform, but it comes at the expense of a higher computational cost. We embed this term into an overall optimization framework to fit geometric data (obtained from sets of depth images) with subdivision surfaces. Results show the advantages

of this new background term, but the overall framework described for 3D reconstruction and tracking is not mature enough to compete with state-of-the-art algorithms. Concerning the reconstruction stage, one significant difficulty is the lack of an initial topology. In our work, the fitting process always starts with a sphere placed at the centroid of the geometric data, which is subsequently deformed and refined in a coarse-to-fine scheme. The topology of the control mesh is never modified (just refined) during the optimization, fact that renders the reconstruction process extremely complicated. A tentative solution consists in formulating a continuous-discrete optimization strategy which alternates between data fitting and topology rearrangement. This and other alternatives should be explored to improve the quality of the reconstructions.

7. We have generalized an existing reactive navigation algorithm [10] for robots moving on a flat surface. In contrast to most existing approaches, our algorithm takes the 3D shape of the robot into account, as well as the real distribution of obstacles detected by different range sensors. It has been tested with various robotic platforms operating in different environments, allowing robots in MAPIR to cover hundreds of kilometers autonomously. The main limitation of this approach is that it assumes the robot shape is fixed, which is not true for mobile platforms equipped with a manipulator. Therefore, a natural extension of this approach should incorporate the possibility to model changing 3D shapes.

Outlook

We can contend that, nowadays, both autonomous navigation and odometry are solved problems if the environment is static. As Prof. Dieter Fox pointed out in his talk "The 100-100 tracking challenge"[1], the next milestone is achieving the same level of accuracy and robustness in changing environments where robots are surrounded by people or other robots in permanent motion. As a consequence, scene flow, either by itself or combined with segmentation or odometry, will become a more frequent and relevant research topic in robotics and computer vision.

Despite the significant progress made in the last years, including the works presented here, there are still two main problems to solve:

- Scene flow is computationally too expensive. The fact that some algorithms (like the PD-Flow [12] proposed in this thesis) run in real time does not imply that this problem is solved. Those algorithms are fast because they run on powerful GPUs or other dedicated hardware like FPGAs. Most phones, tablets, virtual reality devices or consumer robots cannot afford to be equipped with such hardware, and even if they were, they could not overload it just for scene flow estimation because there are tens of other processes they must run. Therefore, we must find more intelligent strategies to compute scene flow. A positive step forward is the clustering strategy proposed here and in other works like [13]. Estimating motions per cluster greatly reduces the number of unknowns (by two or three orders of magnitude) and often improves accuracy. The next significant improvement might come from studying how to compute the motion of each individual cluster more efficiently.

[1] International Conference on Robotics and Automation (ICRA), Stockholm (Sweden), 2016.

- It is not known how to estimate scene flow with monocular RGB cameras. This is a tremendous limitation since this kind of cameras are cheap, require low power and are equipped in many consumer products. The critical problem is that depth estimation with monocular cameras is performed assuming that disparity between images is explained by the camera motion. When objects move, this hypothesis is violated and there is no obvious way to estimate their 3D position (up to scale) in space. Solutions to this problem must involve the detection of parts of the scene that are not static (high residuals after warping according to the camera motion) and the joint estimation of the position and motion of those elements.

From a technological point of view, in the last decade we have witnessed a spectacular development of range sensors accompanied by a noticeable drop of their prices. In this thesis we have mainly worked with two types of sensors: depth (or RGB-D) cameras and 2D laser scanners. Consciously, and maybe mistakenly, we have disregarded 3D laser scanners. The main reason justifying that decision is that, because of their high prices, our research group does not own such sensor. Admittedly there are datasets with 3D lidar scans and simulators that could be used to generate synthetic data. Yet, during all these years we have focused on working with real sensors, which allowed us to test our algorithms under real conditions, and this was not possible for 3D lidars. Nonetheless, a paradigm shift seems to be near in time because a few manufacturers have announced the future release of solid-state 3D lidars at considerably lower prices than their mirror-based counterparts. This innovation is encouraged by the imminent arrival of autonomous cars and the fact that they rely to a great extent on this kind of sensors to work. As occurred with the advent of Kinect in 2010, robotics might benefit from a technological breakthrough that was not originally conceived for such purpose (although, anyway, an autonomous car is nothing but a mobile robot).

Bibliography

[1] Microsoft, "Kinect sensor." [Online]. Available: https://en.wikipedia.org/wiki/Kinect

[2] A. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," in *Proc. of Robotics: Science and Systems (RSS)*, vol. 2, no. 4, 2009.

[3] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for RGB-D cameras," in *Int. Conf. on Robotics and Automation (ICRA)*, 2013, pp. 3748–3754.

[4] J. Gonzalez, A. Stentz, and A. Ollero, "A mobile robot iconic position estimator using a radial laser scanner," *Journal of Intelligent and Robotic Systems*, vol. 13, no. 2, pp. 161–179, 1995.

[5] A. Censi, "An ICP variant using a point-to-line metric," in *Int. Conf. on Robotics and Automation (ICRA)*, 2008, pp. 19–25.

[6] A. Diosi and L. Kleeman, "Fast laser scan matching using polar coordinates," *The International Journal of Robotics Research*, vol. 26, no. 10, pp. 1125–1153, 2007.

[7] T. Pock, D. Cremers, H. Bischof, and A. Chambolle, "An algorithm for minimizing the Mumford-Shah functional," in *Proc. Int. Conference on Computer Vision (ICCV)*, 2009, pp. 1133–1140.

[8] A. Chambolle and T. Pock, "A first-order primal-dual algorithm for convex problems with applications to imaging," *Journal of Mathematical Imaging and Vision*, vol. 40, no. 1, pp. 120–145, 2011.

[9] D. Cremers and S. Soatto, "Motion competition: A variational approach to piecewise parametric motion segmentation," *International Journal of Computer Vision*, vol. 62, pp. 249–265, 2005.

[10] J. Blanco, J. Gonzalez-Jimenez, and J. Fernandez-Madrigal, "Extending obstacles avoidance methods through multiple parameter-space transformation," *Autonomous Robots*, vol. 24, no. 1, pp. 29–48, 2008.

[11] Microsoft, "Hololens." [Online]. Available: https://www.microsoft.com/microsoft-hololens

[12] M. Jaimez, M. Souiai, J. Gonzalez-Jimenez, and D. Cremers, "A primal-dual framework for real-time dense RGB-D scene flow," in *Int. Conf. on Robotics and Automation (ICRA)*, 2015, pp. 98 – 104.

[13] C. Vogel, K. Schindler, and S. Roth, "Piecewise rigid scene flow," in *Proc. Int. Conference on Computer Vision (ICCV)*, 2013, pp. 1377–1384.

[14] K. Khoshelham and S. O. Elberink, "Accuracy and resolution of kinect depth data for indoor mapping applications," *Sensors*, vol. 12, no. 2, pp. 1437–1454, 2012.

[15] Vicon, "Infrared-based motion capture." [Online]. Available: https://www.vicon.com/products/camera-systems

[16] R. Krigslund, S. Dosen, P. Popovski, J. L. Dideriksen, G. F. Pedersen, and D. Farina, "A novel technology for motion capture using passive UHF RFID tags," *IEEE Transactions on Biomedical Engineering*, vol. 60, no. 5, pp. 1453–1457, 2013.

[17] Nexonar, "Ultrasonic-based motion capture." [Online]. Available: http://www.nexonar.com/en/products/beacons/single-beacon/

[18] O. J. Woodman, "An introduction to inertial navigation," *University of Cambridge, Computer Laboratory, Tech. Rep. UCAMCL-TR-696*, vol. 14, 2007.

[19] J. Engel, J. Sturm, and D. Cremers, "Camera-based navigation of a low-cost quadrocopter," in *Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012, pp. 2815–2821.

[20] J. Gonzalez-Jimenez, C. Galindo, and J. R. Ruiz-Sarmiento, "Technical improvements of the Giraff telepresence robot based on users' evaluation," in *IEEE International Symposium on Robot and Human Interactive Communication*, 2012, pp. 827–832.

[21] iRobot, "Roomba vacuum cleaner." [Online]. Available: http://www.irobot.com/For-the-Home/Vacuuming/Roomba.aspx

[22] N. Joshi, W. Kienzle, M. Toelle, M. Uyttendaele, and M. F. Cohen, "Real-time hyperlapse creation via optimal frame selection," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, p. 63, 2015.

[23] E. Eade, "Lie groups for computer vision," 2014. [Online]. Available: http://ethaneade.com/

[24] P. J. Olver, *Applications of Lie groups to differential equations*. Springer Science & Business Media, 2000, vol. 107.

[25] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artificial intelligence*, vol. 17, no. 1-3, pp. 185–203, 1981.

[26] R. Battiti, E. Amaldi, and C. Koch, "Computing optical flow across multiple scales: An adaptive coarse-to-fine strategy," *International Journal of Computer Vision*, vol. 6, no. 2, pp. 133–145, 1991.

[27] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," in *Proc. European Conference on Computer Vision (ECCV)*, 2004, pp. 25–36.

[28] M. Elad, "On the origin of the bilateral filter and ways to improve it," *IEEE Transactions on Image Processing*, vol. 11, no. 10, pp. 1141–1151, 2002.

[29] H. Spies, B. Jähne, and J. L. Barron, "Range flow estimation," *Computer Vision and Image Understanding*, vol. 85, no. 3, pp. 209–231, 2002.

[30] J. Gonzalez-Jimenez and R. Gutierrez, "Direct motion estimation from a range scan sequence," *Journal of Robotic Systems*, vol. 16, no. 2, pp. 73–80, 1999.

[31] J. Gonzalez, "Recovering motion parameters from a 2D range image sequence," in *Int. Conf. on Pattern Recognition*, 1996, pp. 433–440.

[32] J. Blanco *et al.* (2017) The Mobile Robot Programming Toolkit (MRPT). [Online]. Available: http://www.mrpt.org/

[33] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proc. Int. Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004, pp. 652 – 659.

[34] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *IEEE Robotics and Automation Magazine*, vol. 18, no. 4, pp. 80–92, 2011.

[35] M. Maimone, Y. Cheng, and L. Matthies, "Two years of visual odometry on the Mars exploration rovers," *Journal of Field Robotics*, vol. 24, no. 3, pp. 169–186, 2007.

[36] K. Konolige, M. Agrawal, and J. Sola, "Large-scale visual odometry for rough terrain," in *International Symposium on Robotics Research*, 2011, pp. 201–212.

[37] 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*, vol. 24, no. 6, pp. 381–395, 1981.

[38] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," in *Proc. Int. Conference on Computer Vision (ICCV)*, 2013, pp. 1449–1456.

[39] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, p. 239–256, 1992.

[40] D. M. Cole and P. M. Newman, "Using laser range data for 3D SLAM in outdoor environments," in *Int. Conf. on Robotics and Automation (ICRA)*, 2006, pp. 1556–1563.

[41] J. S. Gutmann and K. Konolige, "Incremental mapping of large cyclic environments," in *Proc. Int. Symposium on Computational Intelligence in Robotics and Automation*, 1999, pp. 318–325.

[42] E. B. Olson, "Real-time correlative scan matching," in *Int. Conf. on Robotics and Automation (ICRA)*, 2009, pp. 4387–4393.

[43] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Visual odometry and mapping for autonomous flight using an RGB-D camera," in *International Symposium on Robotics Research*, 2011, pp. 235–252.

[44] M. Fiala and A. Ufkes, "Visual odometry using 3-dimensional video input," in *Canadian Conf. on Computer and Robot Vision*, 2011, pp. 86–93.

[45] I. Dryanovski, R. G. Valenti, and J. Xiao, "Fast visual odometry and mapping from RGB-D data," in *Int. Conf. on Robotics and Automation (ICRA)*, 2013, pp. 2305–2310.

[46] A. I. Comport, E. Malis, and P. Rives, "Accurate quadrifocal tracking for robust 3D visual odometry," in *Int. Conf. on Robotics and Automation (ICRA)*, 2007, pp. 40–45.

[47] T. Tykkala, C. Audras, and A. I. Comport, "Direct iterative closest point for real-time visual odometry," in *Proc. Int. Conference on Computer Vision Workshops (ICCV Workshops)*, 2011, p. 2050–2056.

[48] F. Steinbrücker, J. Sturm, and D. Cremers, "Real-time visual odometry from dense RGB-D images," in *Proc. Int. Conference on Computer Vision Workshops (ICCV Workshops)*, 2011, p. 719–722.

[49] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012, p. 573–580.

[50] V. Panwar, "Interest point sampling for range data registration in visual odometry," Ph.D. dissertation, Queen's University, Canada, 2011.

[51] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *Int. Symp. on Mixed and Augmented Reality (ISMAR)*, 2011, pp. 127–136.

[52] D. R. Canelhas, T. Stoyanov, and A. J. Lilienthal, "SDF tracker: A parallel algorithm for on-line pose estimation and scene reconstruction from depth images," in *Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013, pp. 3671–3676.

[53] T. Whelan, M. Kaess, M. Fallon, J. Johannsson, H. amd Leonard, and J. McDonald, "Kintinuous: Spatially extended KinectFusion," in *3rd RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, 2012.

[54] T. Whelan, H. Johannsson, M. Kaess, J. J. Leonard, and J. McDonald, "Robust real-time visual odometry for dense RGB-D mapping," in *Int. Conf. on Robotics and Automation (ICRA)*, 2013, pp. 5724–5731.

[55] G. Guennebaud, B. Jacob *et al.* (2017) Eigen: A C++ template library for linear algebra. [Online]. Available: http://eigen.tuxfamily.org/

[56] N. Gelfand, L. Ikemoto, S. Rusinkiewicz, and M. Levoy, "Geometrically stable sampling for the ICP algorithm," in *Int. Conf. on 3-D Digital Imaging and Modeling*, 2003, pp. 260–267.

[57] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, 2009, p. 5.

[58] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *Int. Conf. on Robotics and Automation (ICRA)*, 2011, pp. 1–4.

[59] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust Monte Carlo localization for mobile robots," *Artificial intelligence*, vol. 128, no. 1, pp. 99–141, 2001.

[60] J. L. Blanco, J. Gonzalez-Jimenez, and J. A. Fernandez-Madrigal, "Optimal filtering for non-parametric observation models: applications to localization and SLAM," *The International Journal of Robotics Research*, vol. 29, no. 14, pp. 1726–1742, 2010.

[61] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[62] A. Diosi and L. Kleeman, "Laser scan matching in polar coordinates with application to SLAM," in *Int. Conf. on Intelligent Robots and Systems (IROS)*, 2005, pp. 3317 – 3322.

[63] M. Achtelik, A. Bachrach, R. He, S. Prentice, and N. Roy, "Stereo vision and laser odometry for autonomous helicopters in GPS-denied indoor environments," in *Unmanned Systems Technology XI, SPIE*, 2009, pp. 733 219–733 219.

[64] T. Tomic, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. L. Grixa, F. Ruess, M. Suppa, and D. Burschka, "Toward a fully autonomous UAV: Research platform for indoor and outdoor urban search and rescue," *IEEE Robotics and Automation Magazine*, vol. 19, no. 3, pp. 46–56, 2012.

[65] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing ICP variants on real-world data sets," *Autonomous Robots*, vol. 34, no. 3, pp. 133–148, 2013.

[66] M. Jaimez and J. Gonzalez-Jimenez, "Fast visual odometry for 3-D range sensors," *IEEE Transactions on Robotics*, vol. 31, no. 4, pp. 809–822, 2015.

[67] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *Proc. Int. Conference on Computer Vision (ICCV)*, 2012, pp. 3354–3361.

[68] R. Triebel, K. Arras, R. Alami, L. Beyer, S. Breuers, R. Chatila, M. Chetouani, D. Cremers, V. Evers, M. Fiore *et al.*, "Spencer: A socially aware service robot for passenger guidance and help in busy airports," in *Field and Service Robotics*, 2016, pp. 607–622.

[69] A. Orlandini, A. Kristoffersson, L. Almquist, P. Björkman, A. Cesta, G. Cortellessa, C. Galindo, J. González-Jiménez, K. Gustafsson, A. Kiselev, A. Loufti, F. Melendez-Fernandez *et al.*, "ExCITE Project: A review of forty-two months of robotic telepresence technology evolution," *Presence: Teleoperators and Virtual Environments*, 2017.

[70] S. Coradeschi, A. Cesta, G. Cortellessa, L. Coraci, C. Galindo, J. González-Jiménez, L. Karlsson *et al.*, *GiraffPlus: A System for Monitoring Activities and Physiological Parameters and Promoting Social Interaction for Elderly*, ser. Human-Computer Systems Interaction: Backgrounds and Applications 3. Springer, 2014, pp. 261–271.

[71] E. Ackerman, "Fetch robotics introduces Fetch and Freight: Your warehouse is now automated," IEEE Spectrum online, 2015.

[72] M. Jaimez, J. G. Monroy, and J. Gonzalez-Jimenez, "Planar odometry from a radial laser scanner. A range flow-based approach," in *Int. Conf. on Robotics and Automation (ICRA)*, 2016, pp. 4479–4485.

[73] P. Biber and W. Strasser, "The Normal Distributions Transform: A new approach to laser scan matching," in *Int. Conf. on Intelligent Robots and Systems (IROS)*, 2003, pp. 2743–2748.

[74] J. D. Fossel, K. Tuyls, and J. Sturm, "2D-SDF-SLAM: A signed distance function based SLAM frontend for laser scanners," in *Int. Conf. on Intelligent Robots and Systems (IROS)*, 2015, pp. 1949–1955.

[75] G. D. Tipaldi and K. O. Arras, "FLIRT - Interest regions for 2D range data," in *Int. Conf. on Robotics and Automation (ICRA)*, 2010, pp. 3616–3622.

[76] F. Kallasi, D. L. Rizzini, and S. Caselli, "Fast keypoint features from laser scanner for robot localization and mapping," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 176–183, 2016.

[77] F. Lu and E. Milios, "Robot pose estimation in unknown environments by matching 2D range scans," *Journal of Intelligent and Robotic Systems*, vol. 18, no. 3, pp. 249–275, 1997.

[78] J. Minguez, L. Montano, and J. Santos-Victor, "Abstracting vehicle shape and kinematic constraints from obstacle avoidance methods," *Autonomous Robots*, vol. 20, no. 1, pp. 43–59, 2006.

[79] A. W. Fitzgibbon, "Robust registration of 2D and 3D point sets," *Image and Vision Computing*, vol. 21, no. 13, pp. 1145–1153, 2003.

[80] A. Censi, L. Iocchi, and G. Grisetti, "Scan matching in the Hough domain," in *Int. Conf. on Robotics and Automation (ICRA)*, 2005, pp. 2739–2744.

[81] A. Censi and R. M. Murray, "Bootstrapping bilinear models of simple vehicles," *The International Journal of Robotics Research*, vol. 34, no. 8, pp. 1087–1113, 2015.

[82] L. Alvarez, C. A. Castano, M. Garcia, K. Krissian, L. Mazorra, A. Salgado, and J. Sanchez, "Symmetric optical flow," in *Int. Conference on Computer Aided Systems Theory*, 2007, pp. 676–683.

[83] C. Zach, "Robust bundle adjustment revisited," in *Proc. European Conference on Computer Vision (ECCV)*, 2014, pp. 772–787.

[84] J. Stückler and S. Behnke, "Integrating depth and color cues for dense multi-resolution scene mapping using RGB-D cameras," in *IEEE Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 2012, pp. 162–167.

[85] M. Meilland, A. I. Comport, and P. Rives, "Dense visual mapping of large scale environments for real-time localisation," in *Int. Conf. on Intelligent Robots and Systems (IROS)*, 2011, pp. 4242–4248.

[86] C. Kerl, J. Sturm, and D. Cremers, "Dense visual SLAM for RGB-D cameras," in *Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013, pp. 2100–2106.

[87] A. Howard and N. Roy, "The robotics data set repository (radish)," 2003. [Online]. Available: http://radish.sourceforge.net/

[88] J. R. Ruiz-Sarmiento, C. Galindo, and J. Gonzalez-Jimenez, "Robot@home, a robotic dataset for semantic mapping of home environments," *International Journal of Robotics Research*, vol. 36, no. 2, pp. 131 – 141, 2017.

[89] C. Stachniss, "Robotics datasets." [Online]. Available: http://www2.informatik.uni-freiburg.de/~stachnis/datasets.html

[90] C. Stachniss, G. Grisetti, W. Burgard, and N. Roy, "Analyzing Gaussian proposal distributions for mapping with rao-blackwellized particle filters," in *Int. Conf. on Intelligent Robots and Systems (IROS)*, 2007, pp. 3485–3490.

[91] R. A. Newcombe, D. Fox, and S. M. Seitz, "DynamicFusion: Reconstruction and tracking of non-rigid scenes in real-time," in *Proc. Int. Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 343–352.

[92] M. Innmann, M. Zollhöfer, M. Nießner, C. Theobalt, and M. Stamminger, "VolumeDeform: Real-time volumetric non-rigid reconstruction," *arXiv:1603.08161*, 2016.

[93] M. Slavcheva, M. Baust, D. Cremers, and S. Ilic, "KillingFusion: Non-rigid 3D reconstruction without correspondences," in *Proc. Int. Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[94] A. Bruhn, J. Weickert, and C. Schnörr, "Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods," *International Journal of Computer Vision*, vol. 61, no. 3, pp. 211–231, 2005.

[95] E. Herbst, X. Ren, and D. Fox, "RGB-D flow: Dense 3-D motion estimation using color and depth," in *Int. Conf. on Robotics and Automation (ICRA)*, 2013, pp. 2276–2282.

[96] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade, "Three-Dimensional scene flow," in *Proc. Int. Conference on Computer Vision (ICCV)*, vol. 2, 1999, pp. 722–729.

[97] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. Int. Joint Conference on Artificial Intelligence*, 1981, pp. 674–679.

[98] Y. Zhang and C. Kambhamettu, "On 3D scene flow and structure estimation," in *Proc. Int. Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 2001.

[99] T. Basha, Y. Moses, and N. Kiryati, "Multi-view scene flow estimation: A view centered variational approach," *International Journal of Computer Vision*, vol. 101, no. 1, pp. 6–21, 2013.

[100] F. Huguet and F. Devernay, "A variational method for scene flow estimation from stereo sequences," in *Proc. Int. Conference on Computer Vision (ICCV)*, 2007, pp. 1–7.

[101] A. Wedel, T. Brox, T. Vaudrey, C. Rabe, U. Franke, and D. Cremers, "Stereoscopic scene flow computation for 3D motion understanding," *International Journal of Computer Vision*, vol. 95, no. 1, pp. 29–51, 2011.

[102] C. Vogel, K. Schindler, and S. Roth, "3D scene flow estimation with a rigid motion prior," in *Proc. Int. Conference on Computer Vision (ICCV)*, 2011, pp. 1291–1298.

[103] J. M. Gottfried, J. Fehr, and C. S. Garbe, "Computing range flow from multi-modal Kinect data," in *Advances in Visual Computing*. Springer, 2011, pp. 758–767.

[104] A. Letouzey, B. Petit, and E. Boyer, "Scene flow from depth and color images," in *Proc. British Machine Vision Conference (BMVC)*, 2011, pp. 46.1–46.11.

[105] J. Quiroga, F. Devernay, J. L. Crowley *et al.*, "Local/global scene flow estimation," in *Proc. Int. Conference on Image Processing*, 2013, pp. 3850–3854.

[106] J. Cech, J. Sanchez-Riera, and R. Horaud, "Scene flow estimation by growing correspondence seeds," in *Proc. Int. Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 3129–3136.

[107] S. Hadfield and R. Bowden, "Kinecting the dots: Particle based scene flow from depth sensors," in *Proc. Int. Conference on Computer Vision (ICCV)*, 2011, pp. 2290–2295.

[108] ——, "Scene particles: Unregularized particle based scene flow estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 3, pp. 564–576, 2014.

[109] M. Hornacek, A. Fitzgibbon, and C. Rother, "SphereFlow: 6 DoF scene flow from RGB-D pairs," in *Proc. Int. Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 3526–3533.

[110] A. Wedel, T. Pock, C. Zach, H. Bischof, and D. Cremers, "An improved algorithm for TV-L1 optical flow," in *Statistical and Geometrical Approaches to Visual Motion Analysis*. Springer, 2009, pp. 23–45.

[111] Z. Ma, K. He, Y. Wei, J. Sun, and E. Wu, "Constant time weighted median filtering for stereo matching and beyond," in *Proc. Int. Conference on Computer Vision (ICCV)*, 2013, pp. 49–56.

[112] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International Journal of Computer Vision*, vol. 47, no. 1-3, pp. 7–42, 2002.

[113] H. Hirschmuller and D. Scharstein, "Evaluation of cost functions for stereo matching," in *Proc. Int. Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007, pp. 1–8.

[114] D. Sun, E. B. Sudderth, and H. Pfister, "Layered RGBD scene flow estimation," in *Proc. Int. Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[115] J. Quiroga, T. Brox, F. Devernay, and J. Crowley, "Dense semi-rigid scene flow estimation from RGBD images," in *Proc. European Conference on Computer Vision (ECCV)*, 2014, pp. 567–582.

[116] T. Brox, A. Bruhn, and J. Weickert, "Variational motion segmentation with level sets," in *Proc. European Conference on Computer Vision (ECCV)*, 2006, pp. 471–483.

[117] M. Unger, M. Werlberger, T. Pock, and H. Bischof, "Joint motion estimation and segmentation of complex scenes with label costs and occlusion modeling," in *Proc. Int. Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 1878–1885.

[118] A. Roussos, C. Russell, R. Garg, and L. Agapito, "Dense multibody motion estimation and reconstruction from a handheld camera," in *Int. Symp. on Mixed and Augmented Reality (ISMAR)*, 2012, pp. 31 – 40.

[119] G. Zhang, J. Jia, and H. Bao, "Simultaneous multi-body stereo and segmentation," in *Proc. Int. Conference on Computer Vision (ICCV)*, 2011, pp. 826 – 833.

[120] J. Stückler and S. Behnke, "Efficient dense rigid-body motion segmentation and estimation in RGB-D video," *International Journal of Computer Vision*, vol. 113, no. 3, pp. 233–245, 2015.

[121] T. Pock and A. Chambolle, "Diagonal preconditioning for first order primal-dual algorithms in convex optimization," in *Proc. Int. Conference on Computer Vision (ICCV)*, 2011.

[122] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D*, vol. 60, pp. 259–268, 1992.

[123] T. F. Chan and J. Shen, "Variational image deblurring - a window into mathematical image processing," *Lecture Note Series, Institute for Mathematical Sciences, National University of Singapore*, 2004.

[124] A. Chambolle, D. Cremers, and T. Pock, "A convex approach to minimal partitions," *SIAM Journal on Imaging Sciences*, vol. 5, no. 4, pp. 1113–1158, 2012.

[125] M. Nikolova, S. Esedoglu, and T. F. Chan, "Algorithms for finding global minimizers of image segmentation and denoising models," *SIAM Journal on Applied Mathematics*, vol. 66, no. 5, pp. 1632–1648, 2006.

[126] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *Proc. European Conference on Computer Vision (ECCV)*, ser. Part IV, LNCS 7577, 2012, pp. 611–625.

[127] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," *International Journal of Computer Vision*, vol. 92, no. 1, pp. 1–31, 2011.

[128] D. Katz, M. Kazemi, J. A. Bagnell, and A. Stentz, "Interactive segmentation, tracking, and kinematic modeling of unknown 3D articulated objects," in *Int. Conf. on Robotics and Automation (ICRA)*, 2013, pp. 5003–5010.

[129] D. Gutiérrez-Gómez, W. Mayol-Cuevas, and J. J. Guerrero, "Inverse depth for accurate photometric and geometric error minimisation in RGB-D dense visual odometry," in *Int. Conf. on Robotics and Automation (ICRA)*, 2015, pp. 83–89.

[130] J. Stückler and S. Behnke, "Integrating depth and color cues for dense multi-resolution scene mapping using RGB-D cameras," in *Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 2012, pp. 162–167.

[131] M. Jaimez, M. Souiai, J. Stückler, J. Gonzalez-Jimenez, and D. Cremers, "Motion Cooperation: Smooth piece-wise rigid scene flow from RGB-D images," in *Int. Conf. on 3D Vision (3DV)*, 2015, pp. 64–72.

[132] H. A. Alhaija, A. Sellent, D. Kondermann, and C. Rother, "Graphflow – 6D large displacement scene flow via graph matching," in *German Conference on Pattern Recognition*, 2015, pp. 285–296.

[133] M. Dou, S. Khamis, Y. Degtyarev, P. Davidson *et al.*, "Fusion4D: real-time performance capture of challenging scenes," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, 2016.

[134] C. Vogel, K. Schindler, and S. Roth, "3D scene flow estimation with a piecewise rigid scene model," *International Journal of Computer Vision*, vol. 115, no. 1, pp. 1–28, 2015.

[135] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison *et al.*, "KinectFusion: real-time 3D reconstruction and interaction

using a moving depth camera," in *Proc. of the 24th annual ACM symposium on User Interface Software and Technology*, 2011, pp. 559–568.

[136] S. Choi, Q.-Y. Zhou, S. Miller, and V. Koltun, "A large dataset of object scans," *arXiv preprint arXiv:1602.02481*, 2016.

[137] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans, "Reconstruction and representation of 3D objects with radial basis functions," in *Proc. of the 28th annual conference on Computer Graphics and Interactive Techniques*, 2001, pp. 67–76.

[138] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, "A comparison and evaluation of multi-view stereo reconstruction algorithms," in *Proc. Int. Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2006, pp. 519–528.

[139] P. Ondrúška, P. Kohli, and S. Izadi, "MobileFusion: Real-time volumetric surface reconstruction and dense tracking on mobile phones," *IEEE Transactions on Visualization and Computer Graphics*, vol. 21, no. 11, pp. 1251–1258, 2015.

[140] J. Sturm, E. Bylow, F. Kahl, and D. Cremers, "CopyMe3D: Scanning and printing persons in 3D," in *German Conference on Pattern Recognition*, 2013, pp. 405–414.

[141] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore, "Real-time human pose recognition in parts from single depth images," *Communications of the ACM*, vol. 56, no. 1, pp. 116–124, 2013.

[142] J. Taylor, L. Bordeaux, T. Cashman, B. Corish, C. Keskin, T. Sharp, E. Soto, D. Sweeney, J. Valentin, B. Luff *et al.*, "Efficient and precise interactive hand tracking through joint, continuous optimization of pose and correspondences," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, p. 143, 2016.

[143] J. Thies, M. Zollhofer, M. Stamminger, C. Theobalt, and M. Nießner, "Face2Face: Real-time face capture and reenactment of RGB videos," in *Proc. Int. Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2387–2395.

[144] F. A. Moreno, J. A. Merchán-Baeza, M. González-Sánchez, J. González-Jiménez, and A. I. Cuesta-Vargas, "Experimental validation of depth cameras for the parameterization of functional balance of patients in clinical tests," *Sensors*, vol. 17, no. 2, p. 424, 2017.

[145] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun, "Real-time human pose tracking from range data," in *Proc. European Conference on Computer Vision (ECCV)*, 2012, pp. 738–751.

[146] M. R. Oswald, E. Töppe, and D. Cremers, "Fast and globally optimal single view reconstruction of curved objects," in *Proc. Int. Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 534–541.

[147] T. Cashman and A. Fitzgibbon, "What shape are dolphins? Building 3D morphable models from 2D images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 232–244, 2013.

[148] W. Gander, G. H. Golub, and R. Strebel, "Least-squares fitting of circles and ellipses," *BIT Numerical Mathematics*, vol. 34, no. 4, pp. 558–578, 1994.

[149] E. Töppe, M. R. Oswald, D. Cremers, and C. Rother, "Image-based 3D modeling via Cheeger sets," in *Proc. Asian Conference on Computer Vision (ACCV)*, 2010, pp. 53–64.

[150] N. Savinov, L. Ladicky, C. Hane, and M. Pollefeys, "Discrete optimization of ray potentials for semantic 3D reconstruction," in *Proc. Int. Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 5511–5518.

[151] D. Cremers and K. Kolev, "Multiview stereo and silhouette consistency via convex functionals over convex domains," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 6, pp. 1161–1174, 2011.

[152] M. Prasad, A. Zisserman, and A. W. Fitzgibbon, "Single view reconstruction of curved surfaces," in *Proc. Int. Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 2006, pp. 1345–1354.

[153] S. Vicente and L. Agapito, "Balloon shapes: Reconstructing and deforming objects with volume from images," in *Int. Conf. on 3D Vision (3DV)*, 2013, pp. 223–230.

[154] J. Taylor, R. Stebbing, V. Ramakrishna, C. Keskin, J. Shotton, S. Izadi, A. Hertzmann, and A. Fitzgibbon, "User-specific hand modeling from monocular depth sequences," in *Proc. Int. Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 644–651.

[155] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle, "Piecewise smooth surface reconstruction," in *Proc. Annual Conference on Computer Graphics and Interactive Techniques*, 1994, pp. 295–302.

[156] N. Litke, A. Levin, and P. Schröder, "Fitting subdivision surfaces," in *Proc. Conference on Visualization*, 2001, pp. 319–324.

[157] K.-S. Cheng, W. Wang, H. Qin, K.-Y. Wong, H. Yang, and Y. Liu, "Fitting subdivision surfaces to unorganized point data using SDM," in *Proc. Pacific Conference on Computer Graphics and Applications*, 2004, pp. 16–24.

[158] S. Ilic, "Using subdivision surfaces for 3-D reconstruction from noisy data," in *Workshop on Image Registration in Deformable Environments (DEFORM)*, 2006, pp. 1–10.

[159] S. Ivekovic and E. Trucco, "Fitting subdivision surface models to noisy and incomplete 3-D data," in *Proc. Computer Vision/Computer Graphics Collaboration Techniques: MIRAGE*, 2007, pp. 542–554.

[160] A. Blake and A. Zisserman, *Visual Reconstruction*. Cambridge, USA: MIT Press, 1987.

[161] O. Sorkine and M. Alexa, "As-rigid-as-possible surface modeling," in *Symposium on Geometry processing*, vol. 4, 2007.

[162] A. L. Yuille and A. Rangarajan, "The concave-convex procedure (CCCP)," in *Advances in neural information processing systems (NIPS)*, vol. 2, 2002, pp. 1033–1040.

[163] P. D. Tao and L. T. H. An, "Convex analysis approach to DC programming: Theory, algorithms and applications," *Acta Mathematica Vietnamica*, vol. 22, no. 1, pp. 289–355, 1997.

[164] E. Bylow, J. Sturm, C. Kerl, F. Kahl, and D. Cremers, "Real-time camera tracking and 3D reconstruction using signed distance functions." in *Robotics: Science and Systems*, 2013.

[165] R. Maier, J. Stückler, and D. Cremers, "Super-resolution keyframe fusion for 3D modeling with high-quality textures," in *International Conference on 3D Vision (3DV)*, 2015, pp. 536–544.

[166] C. Kunz, C. Murphy, R. Camilli, H. Singh, J. Bailey *et al.*, "Deep sea underwater robotic exploration in the ice-covered arctic ocean with AUVs," in *Int. Conf. on Intelligent Robots and Systems (IROS)*, 2008, pp. 3654–3660.

[167] Husqvarna, "Automower." [Online]. Available: http://www.husqvarna.com/us/products/robotic-lawn-mowers

[168] R. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal on Robotics and Automation*, vol. 2, no. 1, pp. 14–23, 1986.

[169] D. Scaramuzza, M. C. Achtelik, L. Doitsidis, F. Friedrich, E. Kosmatopoulos *et al.*, "Vision-controlled micro flying robots: from system design to autonomous navigation and mapping in GPS-denied environments," *IEEE Robotics & Automation Magazine*, vol. 21, no. 3, pp. 26–40, 2014.

[170] A. Giusti, J. Guzzi, D. Cireşan, F. L. He, J. P. Rodríguez, F. Fontana *et al.*, "A machine learning approach to visual perception of forest trails for mobile robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 661–667, 2016.

[171] R. C. Arkin, *Behavior-based robotics*. MIT press, 1998.

[172] W. Walter, *The living brain*. WW Norton, 1953.

[173] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, no. 5, pp. 1179–1187, 1989.

[174] ——, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 278–288, 1991.

[175] L. Tychonievich, D. Zaret, J. Mantegna, R. Evans, E. Muehle, and S. Martin, "A maneuvering-board approach to path planning with moving obstacles," in *Proc. of the International Joint Conference on Artificial intelligence*, vol. 2, 1989, pp. 1017–1021.

[176] R. Simmons, "The curvature-velocity method for local obstacle avoidance," in *Int. Conf. on Robotics and Automation (ICRA)*, vol. 4, 1996, pp. 3375–3382.

[177] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics and Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[178] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.

[179] H. Surmann, A. Nüchter, and J. Hertzberg, "An autonomous mobile robot with a 3D laser range finder for 3D exploration and digitalization of indoor environments," *Robotics and Autonomous Systems*, vol. 45, no. 3, pp. 181–198, 2003.

[180] D. Holz, C. Lorken, and H. Surmann, "Continuous 3D sensing for navigation and SLAM in cluttered and dynamic environments," in *Proc. International Conference on Information Fusion*, 2008, pp. 1–7.

[181] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige, "The office marathon: Robust navigation in an indoor office environment," in *Int. Conf. on Robotics and Automation (ICRA)*, 2010, pp. 300–307.

[182] J. Gonzalez-Jimenez, J. Ruiz-Sarmiento, and C. Galindo, "Improving 2D reactive navigators with Kinect," in *Proc. International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, 2013, pp. 393–400.

[183] B. Morisset, R. B. Rusu, A. Sundaresan, K. Hauser, M. Agrawal, J. C. Latombe, and M. Beetz, "Leaving Flatland: Toward real-time 3D navigation," in *Int. Conf. on Robotics and Automation (ICRA)*, 2009, pp. 3786–3793.

[184] K. Nishiwaki, J. Chestnutt, and S. Kagami, "Autonomous navigation of a humanoid robot over unknown rough terrain using a laser range sensor," *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1251–1262, 2012.

[185] J. S. Gutmann, M. Fukuchi, and M. Fujita, "3D perception and environment map generation for humanoid robot navigation," *The International Journal of Robotics Research*, vol. 27, no. 10, pp. 1117–1134, 2008.

[186] R. Kümmerle, R. Triebel, P. Pfaff, and W. Burgard, "Monte Carlo localization in outdoor terrains using multilevel surface maps," *Journal of Field Robotics*, vol. 25, no. 6-7, pp. 346–359, 2008.

[187] T. Lozano-Perez, "A simple motion-planning algorithm for general robot manipulators," *IEEE Journal on Robotics and Automation*, vol. 3, no. 3, pp. 224–238, 1987.

[188] J. Minguez and L. Montano, "Nearness diagram (ND) navigation: collision avoidance in troublesome scenarios," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 1, pp. 45–59, 2004.

[189] S. Coradeschi, A. Cesta, G. Cortellessa, L. Coraci, J. Gonzalez, L. Karlsson, F. Furfari, A. Loutfi, A. Orlandini, F. Palumbo *et al.*, "Giraffplus: Combining social interaction and long term monitoring for promoting independent living," in *Proc. International Conference on Human System Interaction (HSI)*, 2013, pp. 578–585.