CIMAT

Centro de Investigación en Matemáticas A.C.

Environment perception for autonomous vehicles in challenging conditions using stereo vision

---

by

Roberto Guzmán Galán

Thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science with Specialty in Computer Science and Industrial Mathematics

Thesis Advisor

Dr. Jean-Bernard Hayet

December 15, 2014

Abstract

The advances in the field of autonomous cars in recent years have been pretty significant. However much of the research and development work is being performed in developed countries, and thus is not directly applicable to the roads and streets of developing countries, which offer additional challenges that are not often seen in the currently considered scenarios.

For perceiving the environment, today's autonomous cars use a combination of sensors. Among them, stereo cameras offer a good balance between perception capabilities and cost. Several datasets have been created to push forward the development and testing of computer vision algorithms for environment perception and navigation of autonomous vehicles. However, to the best of our knowledge, no dataset exists that offers scenes with the particular characteristics of the streets and roads of developing countries. This makes it difficult to test and adapt the algorithms to perform well in these environments.

In this work, we have created an in-vehicle data acquisition platform that we have used to acquire a dataset featuring stereo images with the particular characteristics of the streets and roads of our country, such as poor road conditions, abundant speed bumpers, insufficient signaling, among others, that add even more complexity to the task of autonomous vehicle navigation. Also, the acquired stereo pairs feature challenges for stereo-matching algorithms typically present in real-world scenarios like perspective distortion, shadowing, occlusions, reflections, transparency, and lens flares.

We also have evaluated several stereo correspondence algorithms and pixel similarity measures under our real-world imagery to determine their suitability to be used in autonomous driving. We offer a qualitative analysis on the obtained disparity maps and perform a runtime comparison on the performance of the selected stereo matchers.

Acknowledgments

First of all, I would like to thank my parents Acela and Rodolfo, for all the effort they have put in raising their sons, offering us the best conditions they could so that we can succeed. They have always believed in me, and have supported me unconditionally. Their love and support have provided me strength in difficult times. Also, I would like to thank my brother Arturo for trusting me, and for making me laugh even in the hardest times.

I would like to specially thank my thesis advisor, Dr. Jean-Bernard Hayet, for his interest in my work and for his patience, understanding and guidance. It has been a real pleasure to work with him. His support and advice, both academically and personally, have been very important.

Also, I would like to thank my examiners, Dr. Héctor Becerra and Dr. Mariano Rivera, for their interest in evaluating my work and for the valuable feedback they provided me. I want to thank also to each one of the professors who gave my classes during my Master's studies for sharing their knowledge with me, and challenging me with difficult tasks that have pushed my limits. Because of that, I am now a better prepared person.

Thanks to CONACyT for the financial assistance which made my graduate studies possible. Also, I would like to thank the whole CIMAT community for all the support that I have received during my Master's studies. Special thanks to the Admissions Committee for giving me the opportunity to continue my studies. Thank you for your confidence in me.

Last but not least I would like to thank my classmates, specially Angel and Ulises (in strict alphabetical order) for their company and help during the course of our studies. We enjoyed a lot of good moments that I will fondly remember. Also, special thanks to my friends Rocky and Norberto from the Department of Industrial Mathematics of CIMAT. Their help and support during my stay in Guanajuato has been invaluable; they certainly have made my life easier.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

According to data from the World Health Organization, the number of deaths caused each year by road traffic accidents around the globe is estimated at almost 1.2 million. Moreover, the number of injured people as a consequence of the crashes could be up to 50 million [100].

The economic losses resulting from these road traffic crashes are enormous, in the order of billions of dollars. Injuries caused by traffic accidents are the eighth cause of death worldwide, and the leading cause of death for young people between 15 and 29 years old. If the current trend continues, by 2030 road traffic deaths will become the fifth leading cause of death globally [101].

In the United States of America, during the year 2012, about 5,615,000 motor vehicle traffic crashes were reported to the police. Approximately 1,634,000 of those accidents caused injuries and 30,800 resulted in fatalities. The economic cost of the traffic crashes is estimated to be around 277 billion dollars [71].

The situation in Mexico is also worrying. Mexico occupies the 13th place in the world in road casualties [101], while being only at the 56th place in number of vehicles per capita [99]. The injuries, disabilities and deaths caused by traffic accidents cost more than 120 thousand million pesos a year. Traffic crashes are the first cause of death in the young population aged 5 – 34, and the second cause of orphanhood. Because of traffic accidents, every year more than 24,000 persons die and approximately 750,000 persons get injured to the point of needing hospitalization. [24].

The accidents may be attributed to different causes: vehicle deficiency or failure, e.g. tire failure, roadway-related factors, e.g. a lane delineation problem, or driver-related factors, like fatigue, speeding, panic, misjudgment, driving under the influence of alcohol, among others. According to John Maddox, former Associate Administrator for Vehicle Safety of the U.S. National Highway Traffic Safety Administration (NHTSA), human error is the critical reason in 93% of the accidents [66]. This percentage is alarming.

But the cost of car transportation is paid in other ways too. In average, Americans spent 250 hours a year driving to and from work, and in big cities the commuting time can be much worse. That lost time could be better invested, for example doing some work, reading, or simply relaxing. Parking has also become a complicated issue in big cities. According to a report published by the MIT Media Lab, about 40 percent of the gasoline in crowded urban areas is used by drivers looking for a parking spot [60]. This, besides being a waste of time, money and energy, represents a significant source of pollution. In some U.S. cities, parking lots cover more than a third of the land area, making them the most prominent feature of the urban landscape [60]. As it can be seen, current trends cannot be

sustained for much longer, and alternatives to solving these issues are arising. Autonomous cars are one of the most promising ones. They are safer, as a computer system can react much more quickly to an emergency situation than a human does. Also, autonomous cars do not get tired or drunk. They can drive in a more energy-efficient manner. They allow time saving and better use of road infrastructure. When they are ready, they will change the world.

1.1   Objectives

The advances in the field of autonomous cars in recent years have been pretty significant. However much of the research and development (R&D) work is being performed in developed countries, and thus is not directly applicable to the roads and streets of developing countries, which offer additional challenges that are not seen very often in the currently considered scenarios. For perceiving the environment, today's autonomous cars use a combination of sensors. Among them, stereo cameras offer a good combination between perception capabilities and cost. Several datasets have been created to push forward the development and testing of algorithms for environment perception and navigation of autonomous vehicles. However, to the best of our knowledge, no dataset exists that offers scenes with the particular characteristics of the streets and roads of developing countries. This makes difficult to test and adapt the algorithms to perform well in these environments.

The main objective of this work was to create such a dataset, featuring stereo images recorded in Mexican streets, with the particular characteristics of the streets and roads of our country, and with as many as possible additional sensor data to perform road scene understanding. As everyone who has driven in our streets knows, lane marking and road signaling are sometimes either absent or insufficient. Also, the pedestrian flow is less structured, as pedestrians can be commonly found sharing the road with cars and crossing roads at points different than the street corners. Besides, the road quality is often poor, and one can find many speed bumpers and potholes. All these challenges offered by our roads increase even more the complexity of driving in urban scenarios. Also, we wanted to perform a survey on the advances of the field of autonomous cars, specially about environment perception. Finally, we wanted to evaluate the performance of several stereo matchers with the acquired pairs, to find the best option for obtaining quality disparity maps, that could be used in future work with the dataset.

1.2   Related work

In this section, we offer a brief description of the stereo vision datasets that exist in the literature for their use in mobile robotics and autonomous vehicles research.

1.2.1   The Rawseeds project

The Rawseeds project [85, 35] generated a comprehensive high-quality Benchmarking Toolkit for autonomous robotics. The toolkit is mainly focused in the problems of localization, mapping and SLAM in robotics, but certainly can be used for other tasks. The toolkit contains:

- High-quality multi-sensor datasets, with associated ground truth;

- Benchmark Problems based on the datasets;
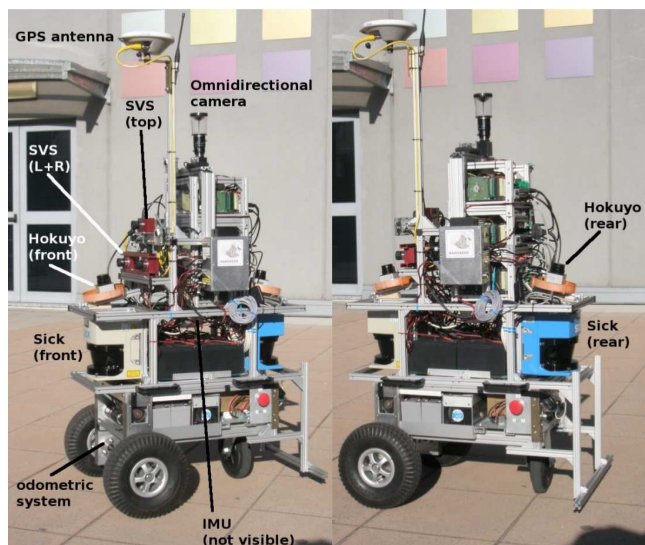
Figure 1.1: The robotic platform used in the Rawseeds project. *Image credit:* The Rawseeds project

- Benchmark Solutions for the problems.

The datasets are made up of raw sensor data and are recorded in indoor and outdoor environments. The Benchmark Problems offer quantitative performance metrics that can be applied to the solutions obtained by the users. Therefore the algorithms can be evaluated quantitatively and be compared in a standard way, allowing to determine the best performing algorithms, and the ones that need further improving. This enables the progress of the robotics field.

The sensors used to acquire the datasets were a "representative sensor suite", defined by the authors as "one including all sensor types usually adopted in mobile robotics" [85]. For each sensor type, one or more models were selected, according to their popularity in research and industrial applications. Factors also considered in the sensor selection process were mass, dimensions, power consumption, possible interactions between sensors, and cost.

The robotic platforms used to generate the datasets were equipped with a sonar belt for indoor data acquisition, including 12 Maxbotix EZ-2 ultrasound transducers positioned all around the robot. The sensors were activated in groups, to limit the possibility of crosstalk. Also an Inertial Measurement Unit (IMU) was considered and the XSens MTi was chosen. It provides 3-axis measurements of angular orientation, acceleration, rate-of-turn and Earth magnetic field data. The MTi used had a 1,7g full scale acceleration and 150 deg/s full scale rate of turn. Computer vision featured an important role, as several vision systems were included, among them stereo and trinocular setups with monochrome cameras. Additionally, standard perspective cameras with color and monochrome sensors were used. Omnidirectional vision was achieved with two cameras, one color and one monochrome, equipped with a hyperbolic mirror. Two classes of Laser Range Finders (LRF) were included. A couple of Hokuyo URG-04LX LRFs (4m range, less at low reflectivity) represented the cheap short-range class. Two high-performance LRFs with medium and long-range were mounted as well: the Sick LMS200 and LMS291, with respectively < 30m and < 100m range at 100% reflectivity.

The robotic platform used to record the data is shown in figure 1.1.

1.2.2   EISATS

The University of Auckland in collaboration with institutions like Daimler A.G., Humboldt-Universität zu Berlin among others, offers the .enpeda.. Image Sequence Analysis Test Site (EISATS [88]). It provides sets of image sequences of medium and high-resolution for the purpose of comparative performance evaluation of stereo vision, optical flow, motion analysis, or further techniques in computer vision.

To this date, they have made available 11 sets, each with one or more sequences. Most of the sets are targeted to perception and understanding of the environment for autonomous driving applications. A set for the specific problem of optical flow detection, recorded using a high frame rate, and another for the problem of driver monitoring under different lighting conditions are the only exceptions.

The sets aimed at autonomous driving comprise synthetic and real-world sequences captured in diverse scenarios under various weather and lighting conditions. Some of them are very challenging. The real-world scenes were recorded from moving vehicles using color and monochrome cameras in either monocular, stereo or trinocular configurations. Some sequences are complemented with one or more of the following: ego-motion data, laser scans from a LIDAR, ground truth disparity maps created from back projected laser scanner points, and hand-made scene labeling for testing segmentation algorithms. With the exception of set 3 and some of the monocular sequences, which have more than 1,000 frames, the sequences are short, with less than 400 frames. The pixel depth (i.e. the number of bits used to encode the grey or color levels) ranges from 8 to 12 bits per pixel in monochrome images, and is 24 bits per pixel in color images.

1.2.3   The HCI/Bosch Robust Vision Challenge

To extract depth and motion information from camera frames, computer vision algorithms make strong assumptions, such as brightness constancy between the left and right images from a stereo pair, or the supposition of single motion per pixel in the case of optical flow analysis. Thus, the algorithms diminish their performance when phenomena that violate these assumptions are present, for example reflective or transparent surfaces, lens flares, and dynamic illumination. The HCI/Bosch Robust Vision Challenge [51] provides eleven hand-picked challenging scenes captured in uncontrolled traffic scenarios that contain instances of these phenomena. The challenge was issued to the scientific community to stimulate the development of algorithms that can deal robustly and reliably with these issues, so that they can be applicable in industrial applications.

The ground truth for depth and motion of the objects visible in the scenes is not known, thus only a qualitative evaluation is possible. The scenes were selected from several million frames acquired with a carefully specified and characterized stereo pair. Each scene contains a different challenge, highlighting problems that befall commonly. The selected scenes feature a wide variety of weather conditions, motion and depth layers. They were recorded day and night in city and countryside environments.

The original data was recorded with a resolution of $1,312 \times 1,082$ pixels with a pixel depth of 12 bits, using a frame rate of 100 Hz. The baseline of the stereo camera was around 30 cm. The final images were both radiometrically and geometrically rectified (see chapter 3) and downscaled to the size of 656×541 by averaging each four pixels to a single one. Every fourth frame is delivered, yielding a sequence frame rate of 25 Hz. Example frames of one of the provided sequences is shown in figure 1.2.

Figure 1.2: Reference frames (Left/Right) from the sequence '*Wet Autobahn*' of the HCI/Bosch Vision Challenge. This sequence shows multiple adverse effects of rainy conditions. The film of water covering the street makes it reflective. Besides, the spray raised by passing cars is semitransparent, making the motion estimation on both the spray and occluded objects more complicated.

1.2.4 The Daimler Ground Truth Stixel Dataset

The Daimler Ground Truth Stixel Dataset [25] provides twelve stereo highway sequences annotated with ground truth for the obstacles in the environment, represented as Stixels. The Stixels are rectangular elements that allow to model the traffic scene compactly without relevant information loss. The image data is complemented with vehicle data, which includes velocity, yaw rate and timestamp. The ground truth was obtained from manual annotations and taking advantage of the known ego-motion in the case of non-moving structures. In the case of moving objects, a vehicle tracker based on an Extended Kalman Filter was used with supervision. The position, orientation, velocity, acceleration, and yaw rate of a perceived vehicle are estimated from a 3D point cloud tracked over time [7]. Some of the scenes were recorded with heavy rain. The images are provided in `PGM` format, with a resolution of $1{,}024 \times 333$, and with a pixel depth of 12 bits.

1.2.5 The annotated Leuven stereo data set

In [63] the authors present a method to jointly optimize the pixel labellings corresponding to both disparity and object class segmentation. To evaluate their method, the authors have augmented a subset of the Leuven stereo data set used in [64] with object class segmentation and dense disparity annotations. All image pairs were rectified and cropped to a size of 316×256 prior to including the new data. Then, 70 selected non-consecutive frames were manually annotated following a two-step procedure. First, each pixel was labeled with one of 7 object classes: *Building, Sky, Car, Road, Person, Bike* and *Sidewalk*. An 8th label, *Void*, was given to pixels that did not clearly belong to one of the considered classes. Next, by manually matching the corresponding planar polygons, the ground-truth disparity maps were obtained. The original stereo sequence was shot from a car driving around the streets of Leuven, Belgium, using a stereo camera with a baseline of 150 cm.

Figure 1.3: The vehicle platform used for the KITTI dataset. *Image from:* [41]

1.2.6   The KITTI Dataset

The KITTI Dataset [41] features 6 hours of diverse traffic scenarios recorded using two high-resolution color and grayscale stereo cameras with a baseline of approx. 54 cm. Accurate ground truth is provided by a Velodyne 3D LIDAR and a high-precision OXTS RT3003 GPS/IMU inertial navigation and localization system. Data provided by the LIDAR and the cameras was recorded at a frequency of 10 Hz.

The data was acquired while driving in the mid-size city of Karlsruhe, Germany and its surroundings. The different real-world traffic scenarios provided include situations in freeways, rural areas and inside the city, with many static and dynamic objects. The supplied data is calibrated, synchronized and timestamped, and the authors provide the rectified and raw image sequences. Also, for relevant objects in the camera's field of view, for example cars, cyclists or pedestrians, 3D bounding boxes are provided. The dataset was recorded from an equipped VW Passat station wagon shown in figure 1.3.

Besides providing all their data in raw format, the authors have created challenging computer vision benchmarks for evaluating state-of-the-art computer vision methods. The tasks of interest for which benchmarks are provided are stereo, optical flow, visual odometry, 3D object detection and 3D object tracking. For each of the benchmarks, an evaluation metric and an evaluation website is provided.

1.3   Thesis layout

In this chapter, the problems that autonomous driving technology aims to solve were introduced and the objectives of the thesis were described. Also, existing related work in the literature was presented.

The remaining of this document is structured as follows. In chapter 2, we offer a brief history of autonomous driving technology and a review of the sensing technologies used in autonomous vehicles. Then, in chapter 3, we describe the fundamental concepts of stereo imaging required for environment perception using this type of sensors. In chapter 4, we describe the equipment and the techniques used to acquire the dataset. The structure of the data files containing the recorded sequences we provide is described in chapter 5. Next, in chapter 6, we describe the work we have performed on selected stereo pairs of the dataset, featuring a qualitative analysis of the performance of several stereo matchers and data costs on the acquired pairs. Finally, in chapter 7 we summarize our contributions and conclusions. In this chapter we also propose topics for future work using the acquired dataset.

Chapter 2

Developments on the field of autonomous cars

The last two decades have seen tremendous advances in autonomous driving. However, steps towards self-driving cars have been taken for many decades. We present a non-exhaustive review of the most important events in the history of the technology in section 2.1. Then, in section 2.2, we talk about the most important sensors used in autonomous cars, cameras, laser scanners and radars, and compare their advantages and drawbacks. Also in this section, we talk about previous autonomous vehicles that have given stereo vision a significant role.

2.1 A brief history of autonomous driving technology

The idea of having autonomous cars is not new at all. In the 1939 World's Fair in New York, the "Futurama: Highways & Horizons" exhibit, created by Norman Bel Geddes and sponsored by General Motors, offered a vision of what the world of 1960 would be. The 35,738 square foot scale model featured cities with skyscrapers, countryside, and industrial areas [93]. The places in this imagined world were interconnected with superhighways where radio-guided cars obtain their power from circuitry embedded in the road [11]. Skyscrapers and superhighways have been with us for some time, but fully autonomous cars are yet to come.

The vision of autonomous cars in the 1950s can be seen in a newspaper ad from 1957, pictured in figure 2.1. It was sponsored by some electric power companies and shows an autonomous automobile piloting down the highway while the family members play a board game [94]. The car receives navigation instructions from magnets embedded in the road that encode binary patterns corresponding to the forthcoming state of the road, for example indicating to the car that it should slow down because there is a sharp curve ahead. This smart road system was simple and reliable, but building it on a large scale was never feasible.

The alternative to smart roads was then smart cars, more flexible but much more complex. Although some developments towards autonomous driving were introduced by car manufacturers like cruise control in the 1950s and the anti-lock braking system (ABS) in the 1970s, just to cite some examples, it was until the 1980s with the pioneering work of Ernst Dickmanns that the field started to flourish. In the beginning of that decade, Dickmanns and his group at Universität der Bundeswehr München (UniBwM) equipped a Mercedes-Benz van with cameras and other sensors. By 1986, the van

ELECTRICITY MAY BE THE DRIVER. One day your car may speed along an electric super-highway, its speed and steering automatically controlled by electronic devices embedded in the road. Highways will be made safe— by electricity! No traffic jams . . . no collisions . . . no driver fatigue.
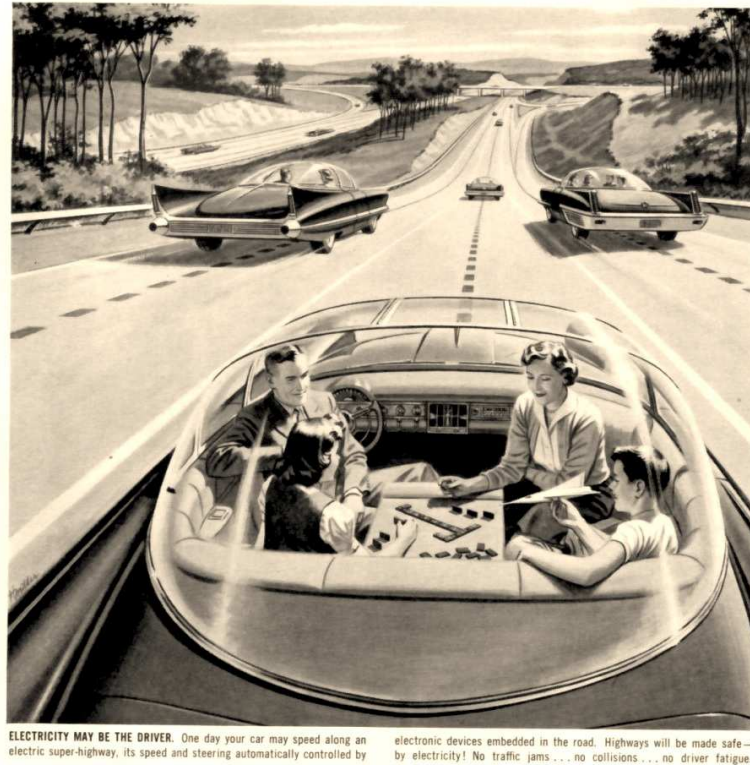
Figure 2.1: The vision of an autonomous car in the 1950s. *Image from:* [93]

"VaMoRs" managed to drive all by itself on streets without traffic [96].

In 1987, the largest R&D project ever in the field of autonomous cars started. The EUREKA PROMETHEUS Project (PROgraMme for an European Traffic of Highest Efficiency and Unprecedented Safety, 1987-1995) was funded with €749 million from the member countries of EUREKA [97]. The project defined the state of the art of autonomous vehicles. The project benefited from the participation of Dickmanns and his team at UniBwM, in cooperation with Daimler-Benz A.G. In 1994 a decisive point was achieved when the twin vehicles VITA-2 and VaMP, from Daimler-Benz and UniBwM respectively, drove more than 1,000 km on Autoroute A1, a multi-lane highway, in Paris. The cars maneuvered along normal heavy traffic at speeds up to 130 km/h, drove in convoy determining their distance to the preceding vehicle based on their current speed, changed lanes and performed autonomous passing. The twin cars were based on the Mercedes-Benz S-Class W140, and sixty transputers, a type of parallel computer, were used to handle the huge computational requirements. After that, a second completion point was reached in 1995 with a 1,758 km trip from Munich, Germany to Odense in Denmark and back to Munich. Most of the time, the throttle, brake and steering were controlled autonomously. On the German Autobahn, VaMP achieved speeds above 175 km/h. It was capable of driving up to 158 km without any human intervention despite not being designed for reliability over long distances. On average, the safety driver took control back every 9 km [30, 28, 98].

Around the same time, in July 1995, Dean Pomerleau, a roboticist at Carnegie Mellon University (CMU), and Todd Jochem, a robotics doctoral student, performed the "No hands across America" tour. They traveled 3,000 miles from Pittsburgh, PA to San Diego, CA in a 1990 Pontiac Trans Sport minivan, known as Navlab5. Running on the PANS (Portable Advanced Navigation Support) hardware platform, the Ralph (Rapidly Adapting Lateral Position Handler) vision system was able to determine the steering angle. Acceleration and braking were performed by the driver [74, 22].

ARGO, an autonomous car based on a Lancia Thema was developed around 1998 by the University of Parma. It was equipped with an automatic driving system with three modes of operation. In manual driving mode, the system simply monitored the driver activity and performed data logging. In Supervised Driving mode, the system alerted the driver when a potential unsafe situation was detected. Finally, in Automatic Driving mode, the system had total control of the vehicle and was also capable of platooning behind a leader car. To demonstrate ARGO's capabilities, the Mille Miglia in Automatico tour was performed. It lasted six days. The tour started from Parma on June 1, 1998, it passed through several Italian cities before heading back to Parma a week later, after driving for more than 2000 km in automatic mode. The car drove up to 95.1% a day in automatic mode and drove up to 54.3 km without human intervention [18, 90].

With all these impressive achievements, it seemed that the age of autonomous driving was around the corner, but it was not the case. All the mentioned tests drives were performed in highways and test tracks, which are essentially controlled environments where just keeping the car in the lane and at a safe distance of the preceding car may be enough. But, in more complicated scenarios like city streets, the cars would not behave correctly. The technology was not ready yet. Roboticist Sebastian Thrun is quoted as saying: "There was no way, before 2000, to make something interesting...The sensors weren't there, the computers weren't there, and the mapping wasn't there. Radar was a device on a hilltop that cost two hundred million dollars. It wasn't something you could buy at Radio Shack" [11].

In the beginnings of the 21st century, American military efforts on autonomous vehicles were not living up to expectations. The American Congress wanted a third of all ground combat vehicles to be autonomous by 2015, but military contractors, funded by the Defense Advanced Research Projects Agency (DARPA), weren't up to the challenge. It wasn't clear whether the lack of success was due to the technology still not being capable, or the inability of people to extract the most from it. To clarify the issue, people in DARPA thought it would be a good idea to have more people involved, so they convened a race.

The first DARPA Grand Challenge took place in the Mojave Desert on March 13, 2004. It offered a million-dolar prize for the robotic car that would complete the race course faster and with total autonomy. The road was rough, 142 miles from Barstow, California, to Primm, Nevada. None of the fifteen finalists drove more than eight miles. Sandstorm, a modified Humvee from CMU's Red Team was the participant which traveled the most, completing 7.4 miles of the course before getting stuck on an embankment after doing a sharp turn. No winner was declared, and consequently the cash prize was not awarded. [95].

Despite the failures in the first Grand Challenge, DARPA announced a second Grand Challenge, to be held on October, 2005. The robots that participated in the second challenge were much more successful than their counterparts in the first one. 22 of the 23 finalists traveled more than the 7.4 miles driven by the most successful robot in the first challenge. Outstandingly, 5 vehicles successfully completed the course. The winner of the two million-dollar prize was a modified Volkswagen Touareg nicknamed Stanley, from the Stanford Racing Team, led by Thrun. Stanley, pictured in figure 2.2, completed the requested 132 mile route on 6 hours, 53 minutes. H1ghlander and Sandstorm, both from CMU, finished within minutes of Stanley. [20, 86, 95].

The 2005 DARPA Grand Challenge proved that it was possible for an autonomous mobile robot to travel long distances in off-road terrain by itself. However, it did not prove that it was possible to do it in a urban environment with traffic, where moving robotic vehicles share the roads with vehicles driven by humans. This is why the DARPA Urban Challenge was born. The participating vehicles had to drive in a mock city environment installed in a former air force base in California. The cars had to perform simulated military supply missions. While executing the assigned duties, they had to

Figure 2.2: Stanley, the winning car in the second DARPA Grand Challenge.

follow all traffic rules, avoid other competitors and human-driven vehicles, and also deal with parking lots and closed roads. Pedestrians, cyclists or traffic lights were not considered. The 96 km course had to be completed in less than 6 hours. From 89 applications, 11 teams were selected to participate in the final event which took place in 2007. The Urban Challenge was won by a modified Chevrolet Tahoe named "Boss" developed by Tartan Racing, a team led by CMU. The course was completed successfully by 6 teams [21, 92, 95].

The Grand and Urban challenges started a wave of development in the field of autonomous cars, and after them several companies and universities continued the advancement of the technology. Also, many car manufacturers and Tier-1 automotive suppliers, like Continental and Bosch have established teams to develop the technology. A non-exhaustive review of some teams currently working on the field is presented below.

The RobotCar UK group [73], from the Department of Engineering Science of Oxford University, has modified a Nissan Leaf to make it an autonomous car. Their approach is not to rely on GPS for localization. Instead, the car learns previous routes driven manually, which are stored on the computer's memory and when the car recognizes a known route, it can take over the driving duties. Their approach is useful for common-driven routes like daily commuting but would be less suitable for driving long trips.

AutoNOMOS Labs [37] is part of the Artificial Intelligence Group of the Freie Universität Berlin. To date, they have developed two autonomous cars, Spirit of Berlin, a modified Dodge Caravan, and MadeInGermany, a modified Volkswagen Passat Wagon. MadeInGermany was the first car to get certified for autonomous driving in the state of Berlin in Germany. This made it possible to perform a fully autonomous drive in the traffic of Berlin. The car covered 4 times the 20 km route between the International Congress Center, the Bradenburg Gate and back to the Congress Center, in fully autonomous manner without any incident. According to their website, the team is now instrumenting an electric car. Prof. Raúl Rojas is the leader of the project.

The Artificial Vision and Intelligent Systems Laboratory (VisLab) [91] of Parma University (Italy) has worked continuously in the field of autonomous transportation starting from the ARGO project. Led by professor Alberto Broggi, the laboratory performs basic and advanced research in many disciplines such as machine vision, pattern recognition, robotics, real-time systems, among others, and

applies it to intelligent transportation systems and intelligent vehicles. In 2010, the laboratory organized a 13,000 km intercontinental trip from Parma, Italy to Shangai, China, with driverless electric vehicles, using a vehicle-follower approach [10]. The trip offered a wide variety of driving conditions. In some places, they were so challenging for the vehicles that it was impossible to drive in autonomous mode. On July 12, 2013, the laboratory performed the Public ROad Urban Driverless-Car Test 2013 (PROUD-Car Test 2013) with the autonomous car BRAiVE [89]. A remarkable point about the PROUD-Car test is the fact that, for the first time in history, an autonomous car traveled safely on open public roads of different types (rural, urban and freeways) with no person on the driver seat.

Google established their self-driving car project after the DARPA Grand Challenges, recruiting personnel from the most successful teams. Led initially by Sebastian Thrun, they first focused on driving on highways and country roads. By August 2012, they had already completed more than 300,000 miles of testing [45, 46]. After their success in highway driving, they shifted their focus to city driving, performing test drives in Mountain View, California — Google's hometown [44]. Now under the leadership of Chris Urmson, technical leader of the CMU team that won the 2007 Urban Challenge, their goal is to create a system that is capable of dealing with the complexity of driving in a busy urban environment. To do so, they have built software models to infer what might happen next given the current state of the surroundings and the previously acquired knowledge. Then they can determine the bests actions to take. They previously had been using modified conventional cars, namely Toyota Prius and Lexus RX450h, but in May 2014, they presented their own prototype for a self-driving car, in a major change of philosophy. Their prototype has no steering wheel, accelerator or brake pedals [43]. Thus, they are targeting to Full Self-Driving Autonomy, or Level 4 Autonomy, according to the automation levels specified by the NHTSA [70]. In this level, the driver is not expected to be available for vehicle control at any time during the trip. This includes both occupied and unoccupied vehicles. The first rolling prototypes to be tested in public roads will still have manual controls for safety and legal reasons, and are expected to be ready by Fall 2014.

Daimler A.G. has been working in Advanced Driving Assistance Systems (ADAS) and autonomous cars for a long time. They were very involved in the PROMETHEUS project (see above), and since then they have kept developing the technology. They have created several safety systems that are incorporated to many of their production vehicles, like Night View Assist Plus, which uses infrared lights and cameras to provide the driver a better perception at night of the road ahead, and DISTRONIC PLUS, which uses short, mid and long range radars to regulate automatically the vehicle's speed and distance to the preceding vehicle. In August 1888, the first long-distance journey in automotive history was performed by Bertha Benz and her two sons in the *Benz Patentmotorwagen Number 3*. To celebrate the 125th anniversary of this achievement, the latest prototype of autonomous vehicle from Daimler, the Mercedes Benz S-Class S 500 INTELLIGENT DRIVE, revisited the same route from Mannheim to Pforzheim, Germany using similar sensors to those mounted in production vehicles and driving almost all the route autonomously [109]. Notably, it was the first autonomous drive on public urban roads with real-world traffic that was performed without the use of laser scanners as sensors. We discuss more about the sensors used in autonomous cars in section 2.2.

Despite the fact that the previously presented achievements are very impressive, the problem of driving in urban environments with total reliability remains unsolved, mainly because urban environments in busy cities are very complex and unpredictable, and because they lack the structure that can be found in highway scenarios. For this reason, the field remains very active. Some companies like Nissan and Google feel optimistic and have ventured to say that they expect to have fully working autonomous cars before the end of this decade. Others remain more cautious, and while a steady increase in the integration of ADAS in production vehicles is expected, they think full autonomy will come only after several decades of development.

However, the challenges faced by autonomous cars are not only technical. Even if today someone would create a perfectly reliable and safe autonomous car capable of driving in all conditions, it wouldn't be possible to use it in public roads because the legal framework is not ready. One of the main concerns is determining the liability in case an autonomous car is involved in an accident. Who would be responsible? The car owner? The car manufacturer? The company that develops the software? Also, the laws in many countries do not consider autonomous cars at all, so it is not even clear if it is legal or not to use one. Until these and many other legal issues are solved, the use of the technology will be limited.

Some steps have already been taken in clarifying the legal status of self-driving cars and to get everything ready for their eventual introduction to the market. As of the second quarter of 2014, four U.S. states, (Nevada, Florida, California, and Michigan), along with the District of Columbia, have successfully promulgated laws addressing autonomous vehicles [94]. Also, in April 2014 an amendment to the United Nations Convention on Road Traffic was agreed that would let drivers take their hands off the wheel of autonomous cars [75]. If the new amendment is finally approved after all the bureaucratic procedures, it would allow a car to drive itself, as long as the system "can be overridden or switched off by the driver" [75]. A driver must be present and able to take the wheel at any time. The convention covers European countries, Mexico, Chile, Brazil and Russia, but not the United States, Japan or China, which are major automotive markets. The United Kingdom also has recently given green light to autonomous car testing in public roads. The tests will begin in January 2015 [84].

Before autonomous vehicles reach the roads and streets of the cities around the world, they will find applications in private land, for example, in airports or for material transportation in factories. In fact, autonomous vehicles are already in use in mines in Australia [5], where they have improved productivity and have increased the safety of the mining operations. Sooner than later we will see the benefits of the technology also in our daily lives.

2.2   Perception in autonomous cars

To be able to provide safe and reliable autonomous driving in real-world traffic, precise and comprehensive perception of the environment is fundamental. In autonomous vehicles, several types of sensors are often used together to provide a rich view of the surroundings and compensate for each other's weaknesses. The most used and studied sensors in autonomous driving applications for primary sensing are LIDARs, also known as laser scanners, and computer vision cameras [16]. Radar is also part of the equation as a good complement. Next we offer a brief description of these sensors, and describe their strengths and weaknesses.

2.2.1   A review on sensors for autonomous driving

LIDAR is a technology that allows to measure distance by emitting short pulses of laser light towards the objects of interest and measuring the time it takes to detect the reflection. Using this principle, LIDARs can estimate with great accuracy the distance to the target and its brightness as well. To provide 360 coverage, usually LIDARs are mounted in spinning platforms. A high-end LIDAR is pictured in figure 2.3. The advantages that LIDAR technology offers are the following:

- The result of a LIDAR scan is a 3-D map of the world around it, with no extra processing required.
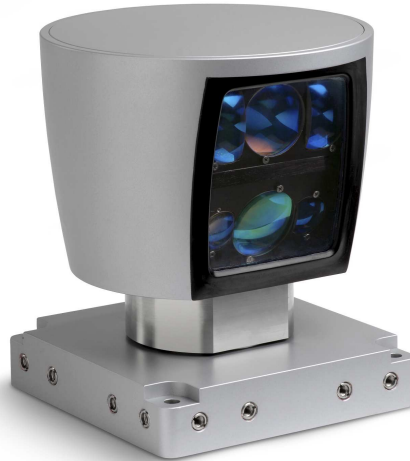
Figure 2.3: The Velodyne HDL-64E LIDAR. *Image credit:* Velodyne.

- They offer a very good accuracy. In high-end units it is less than 2 cm.

- LIDARs emit light, thus they are not dependent of ambient light. They offer about the same performance day or night, and are not affected by shadows or reflections.

- They are robust against interference, and have a much higher accuracy than radar, which works using a similar principle.

However, LIDAR has also disadvantages:

- LIDARs are very expensive. Today, the production of high-resolution LIDARs is limited. A high-end unit costs around 75,000 US dollars, which is more expensive that many premium cars.

- The resolution they offer is very moderate. High-end units provide a scan with only 64 elements in the vertical direction.

- The range in which LIDARs can work well is limited. Typical LIDARs can detect pavement from up to 50 meters. Objects with more reflectivity, like cars and foliage can be perceived from a distance of up 120 meters. More expensive 1.5 micron LIDARs have a higher working range.

- LIDARs update rates, which are typically between 5–15 Hz, are slower than those than can be attained with cameras and parallel processing hardware.

- Most LIDARs have moving parts. Moving parts are always a concern in terms of reliability.

- The majority of LIDARs scan a scene, thus the scene is distorted by the movement of the car in which the LIDAR is mounted and the movement of the objects being scanned. Flash LIDARs do not have moving parts and scan the scene in one instant but are currently even more expensive.

- LIDARs lower their performance in unfavorable weather conditions like heavy rain, snow and fog. Sometimes they can detect irrelevant things like car exhaust gas.

- LIDARs require much more power than cameras to work.

- It is preferred to install LIDARs on the outside of the vehicle, because they need to perceive the reflected light and installing them behind a tinted windshield may diminish their efficiency. This may have some aesthetic and integration implications.

On the other hand, computer vision cameras, like the human eye, can perceive a scene in an instant and provide lots of information. They provide information with low cost in acquisition, space and energy. With a calibrated camera and known motion, or with a multi camera setup, you can extract 3D information from the acquired 2D images. The advantages of using cameras are:

- Cameras are small and very cheap. Many of them can be placed in the car to provide 360° coverage of the surroundings.

- Conventional cameras that perceive visible light work with reflected radiation. In principle, with daylight they can see up to an arbitrary distance if equipped with the right optics.

- They can provide color and texture information. This makes them very useful for scene understanding.

- Cameras are available with very high resolutions.

- They consume little power.

The downsides of using cameras are the following:

- They need a strong light source to be able to work at night. In an autonomous car application, the light coming from the headlamps may not be enough.

- Cameras generate plenty of information, and for that reason a lot of bandwidth is needed.

- To obtain useful information from the images, very heavy processing is required.

- If using stereo for depth estimation, you can get very good resolution at short distances, but as the distance increases, the resolution decreases.

- The quality of the images obtained can be affected by changes of illumination, and shadowing.

A qualitative and quantitative comparison of the performance of computer vision systems and LIDARs in the context of intelligent vehicles is provided in [19].

The data obtained from LIDARs and/or cameras is often complemented with radar readings. Radar technology has important advantages [76]. It works in situations where other sensor alternatives struggle, like in adverse weather conditions such as snow and fog and even when the sensor has been covered with dirt. Also, radar readings return not only the distance to the detected object, but its relative velocity as well, because of the frequency shift caused by the Doppler effect. Radars are classified as short, mid and long-range depending on their detection range, speed detection range, angular resolution and angular width of view. Short-range radars provide a wide view of the surroundings, and have good range accuracy but their detection range is narrow and can detect relative speeds only up to moderate. They are suitable for implementing blind-spot detection, pre-collision alerts and systems for vehicle control in stop-and-go traffic. On the other hand, long-range radars work best in ranges larger than 30 m and can detect objects hundreds of meters away. The field of view they provide is narrow,

but they can operate even in high-speed situations (around 200 km/h). Their typical applications are active cruise control and the detection of crossing cars at intersections. Angular resolution is almost the same for both short and long-range radars. In general radar offers lower resolution than LIDARs or cameras and its performance can be affected by moisture and humidity because RF waves can be absorbed by the water in the environment.

Between the available options, we believe that the perception capabilities offered by stereo vision systems, combined with their relative low price, makes them a very good alternative. Most of the drawbacks associated with them can be overcome with algorithm improvements and with the increase of processing capabilities. The later is guaranteed by the advances in the semiconductor industry and the improvements on parallel computing platforms, while the former can only by attained by the work of the scientific community and the availability of relevant data to work with. This is the motivation of our work.

2.2.2 Previous works on computer vision applied to autonomous cars

Early efforts in the field of autonomous cars used only computer vision cameras to perceive the environment. The team around Dickmanns used 2 cameras with different focal lengths placed in a moving platform mounted behind the windshield. Thanks to the mobility of the cameras, it was possible to point them to the most important details of the viewed scene [29, 78]. Pomerleau and Jochen in the "No hands across America" tour used only a single color camera to perceive the road ahead [74]. The ARGO vehicle used a very modest sensor setup, incorporating just a stereo pair and a Hall effect-based speedometer. The stereo camera was made from two low-cost gray scale cameras that were synchronized to acquire the images simultaneously. The cameras were installed inside the vehicle, placed in the top corners of the windshield to maximize the distance between them [18].

The majority of successful teams in the DARPA Challenges used high-end LIDARs complemented with radars to sense their environment, and relegated computer vision to a secondary role, e.g. [86]. However, some teams used stereo vision to increase their perception capabilities, and we cite a couple of examples. Princeton University's entry in the 2005 DARPA Grand Challenge was Prospect Eleven [4]. It was the only finalist vehicle that depended only on stereo vision to detect obstacles and measure the distance to them. TerraMax [23], a modified Medium Tactical Vehicle Replacement Navy Tractor was Team Oshkosh's entry in the DARPA Urban Challenge. It used extensively vision systems for environment perception. It featured four vision systems: trinocular, stereo, monocular rearview, and monocular lateral. The forward-looking trinocular system offered 3 different baselines. One of them was selected depending on the current vehicle speed; at low speeds the shorter baseline was chosen, while at higher speeds the large baseline was more suitable for detection of distant obstacles. The trinocular system has a working range of 7-40 m. The two stereo systems, one forward-looking and one backward-looking enabled the perception of the environment closer to the vehicle. The use of fish-eye lenses, with an angle of view of about 160, allowed to sense an area of approximate dimensions of 10 × 10 meters. The lateral system, which consisted of two monocular cameras pointing towards each side of the vehicle, was used to detect approaching traffic at intersections. Finally, the rearview system, similar in architecture to the lateral system, was aimed at detecting overtaking vehicles to prevent collisions in passing maneuvers.

Among the current prototype autonomous cars, the latest vehicles form Vislab and Daimler are the ones that have put stereo vision, and computer vision in general, in the main role for perceiving the environment. Other teams such as Google's restrict the use of cameras to tasks like reading road signals and traffic lights.

Vislab's BRAiVE sensor suite is composed of 10 cameras and 5 laser scanners. Sensors are installed all around the car and provide 360° coverage. The sensor suite is designed to be comprehensive and redundant to allow the development and testing of algorithms for relevant tasks such as pedestrian recognition, traffic sign recognition, automated parking, among many others. Two stereo systems are mounted in the car, a monochromatic with a long baseline, and a color stereo pair with a short baseline. They look forward and are installed behind the front windshield. Lateral vision is provided by 2 cameras located in the front fenders. For blind-spot detection, a camera is installed inside each of the outside rearview mirrors. Two cameras installed in the top part of the trunk lid provide rear vision. Laser scanners mounted in the front corners on the vehicle, and in the front and rear bumpers, help detect obstacles and surrounding vehicles [47].

The sensors mounted in the Mercedes-Benz S 500 INTELLIGENT DRIVE constitute a superset of the standard sensor setup already available in production S-Class vehicles, which comprises a forward-facing stereo camera, and mid and long-range radars in the front and back of the car. It was extended with four 150° short-range radars placed in the corners of the car to improve the detection of obstacles in intersections. Also two long-range radars mounted on the front fenders of the vehicle allow to monitor oncoming fast cross-traffic at intersections on rural roads. The baseline of the existing stereo camera system was widened to 35 cm to improve precision and working range (the stereo reconstruction resolution is proportional to the baseline, see chapter 3). To allow recognition of traffic lights and pedestrians in turning maneuvers, an extra monocular camera with a wide-angle lens was mounted on the dashboard. Another wide-angle camera pointing backwards was added to assist in map-relative self-localization [109]. The complete sensor setup and the vehicle are shown in figure 2.4.

For localizing themselves in the world, most of the mentioned cars use a high-precision GPS unit to obtain an estimate of their location in the world. The accuracy of this estimate is not enough to perform fundamental tasks, for example, lane keeping, so several strategies exploiting sensor readings are used to refine the obtained estimate. The most advanced cars like the S-Class INTELLIGENT DRIVE make use of a very high precision 3D map and localize themselves relative to that map for increased accuracy. The Google self-driving car also needs a highly detailed map to be able to navigate safely in urban environments, though it is not clear in the reviewed public information if they use it also for localization, although it is likely they do.
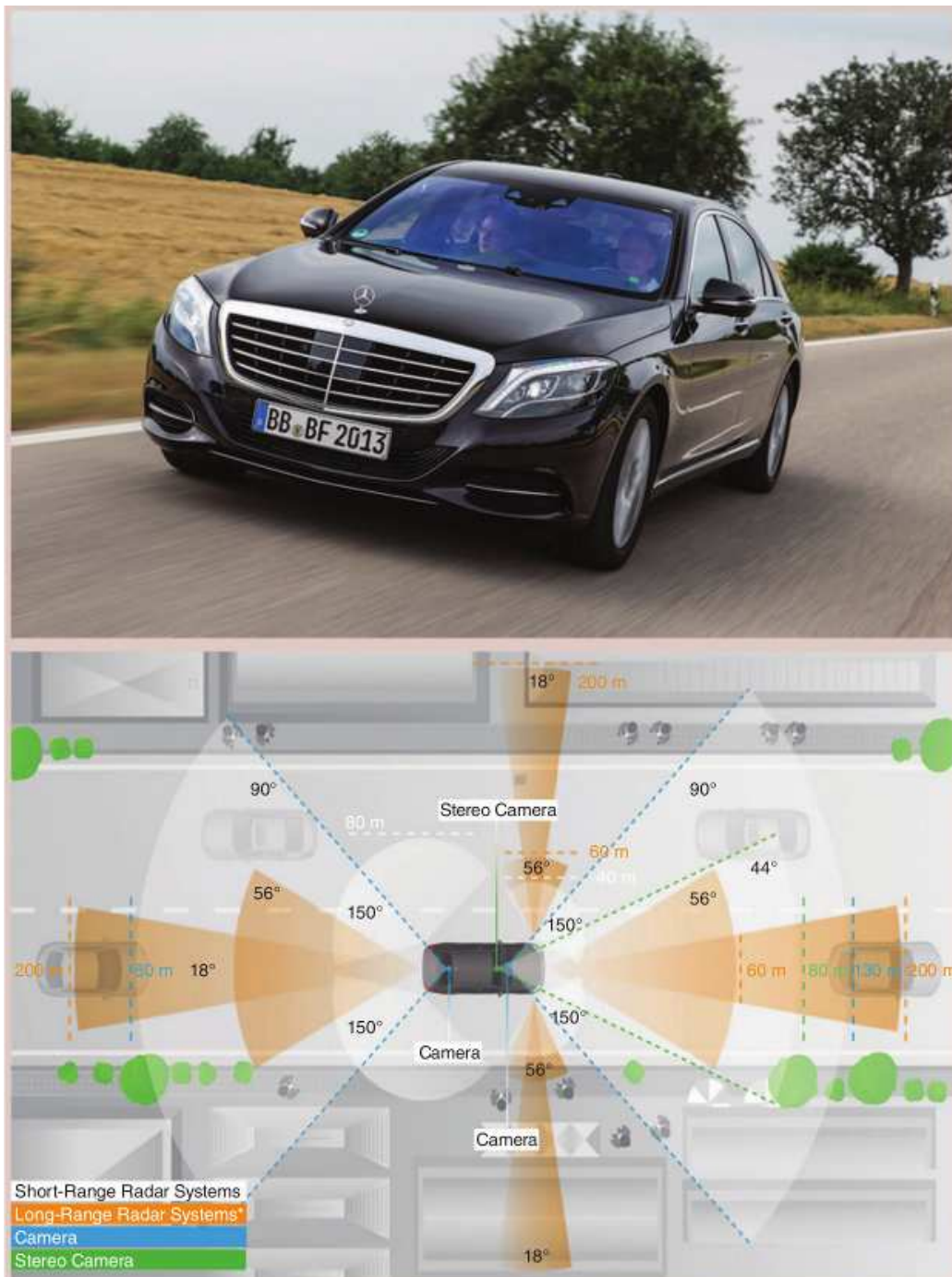
Figure 2.4: The Mercedes-Benz S 500 Intelligent Drive and its sensor setup. *Image from:* [109].

Chapter 3

Basics on stereo vision

In this chapter we briefly describe the fundamental concepts behind stereo imaging that allow to obtain a depth map from the acquired image pairs. We start by providing a description of the image formation process in cameras using the pinhole camera model in section 3.1. In section 3.2, we present important concepts associated with digital cameras. Then in section 3.3, we describe camera calibration, a process that allows to obtain the intrinsic parameters of a camera. Next in section 3.4, we discuss the geometry of a canonical stereo system, and how it allows to recover depth information from corresponding points in the acquired images using triangulation. In this same section, we discuss the geometry of a general stereo pair. After, in section 3.5, we describe how, using stereo rectification, one can obtain row-aligned images from an almost arbitrary stereo pair, making it easier to find correspondences. Finally, we talk about finding such correspondences in rectified image pairs in section 3.6.

3.1   The pinhole camera model

Visual perception starts with the detection of light rays coming from objects in the world. The light beams emitted by some source strike objects. Most of these beams get absorbed by the struck object, but some of them are reflected. This reflected light is then detected by the light-sensitive elements in cameras. A simple model that describes how this happens is the pinhole camera model. A *pinhole* is a "very small" aperture. Under the pinhole model it is assumed that, in a camera, light passes from the viewed scene through a pinhole and then to the imaging surface. Under this model, only a single ray enters the camera from any particular point, resulting in images that are always focused. Moreover, the size of an object in the scene and the size of its image in the camera are related only by a single parameter of the camera, its *focal length*. The focal length is the distance from the pinhole to the image plane. The axis perpendicular to the imaging surface, which passes through the pinhole is called *optical axis*. The point in the intersection between the optical axis and the image plane is the *principal point* (O) . The model is pictured in figure 3.1, *left*, where f is the focal length of the camera, Z is the distance from the camera to the object, X is the height of the object and x is the height of the object's image in the imaging plane. The pinhole is indicated with C.

It is more convenient to rearrange the pinhole model to make the image of the object appear with the same orientation as in the scene. To do so, the positions of the pinhole and the image plane are exchanged. This rearrangement is shown in figure 3.1, *right*. Now, the pinhole is reinterpreted as the *center of projection* (C). Under this model, the rays travel from the objects in the scene to the center of projection.
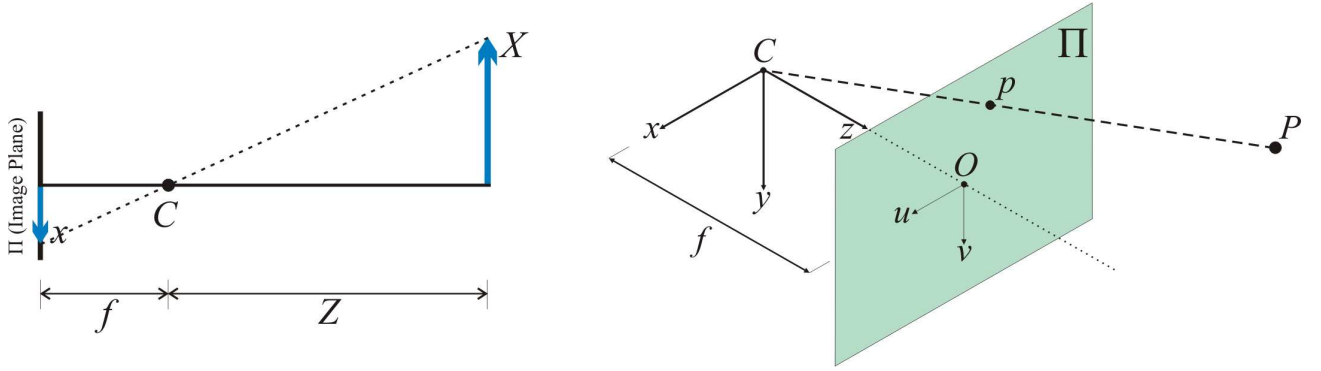
Figure 3.1: *Left:* The pinhole camera model. *Right:* The pinhole model is usually modified so that the image plane is located between the center of projection and the scene. This allows the image to keep the same orientation as the object.

To provide a mathematical description of the *perspective projection* performed by the camera, we introduce some reference coordinate systems in which to express the 3D coordinates of a scene point P and the coordinates of its projection p on the imaging surface. Let (x, y, z) be the *camera reference frame* with origin in C and a z-axis that is coincident with the optical axis. For now, it is assumed that the camera reference frame coincides with the *world reference frame* so that the coordinates of the scene point P are already expressed in the camera frame. We also introduce a 2D *image reference frame* (u, v) for the image plane Π with origin in O and the u and v axes aligned with x and y respectively as shown in figure 3.1, *right*. Let $P = [x, y, z]^T$ and $p = [u, v]^T$. By similarity of triangles we obtain:

$$u = f\frac{x}{z} \tag{3.1}$$

and

$$v = f\frac{y}{z} \ . \tag{3.2}$$

These are the perspective projection equations, which describe the mapping from 3D coordinates in the world to 2D coordinates in the image plane. The equations are clearly nonlinear, but using homogeneous coordinates it is possible to express the projection as a linear equation. Let $\tilde{P} = [x, y, z, 1]^T$ and $\tilde{p} = [u, v, 1]^T$ be the homogeneous coordinates of P and p respectively. The projection equation can then be expressed as:

$$\begin{bmatrix} \lambda u \\ \lambda v \\ \lambda \end{bmatrix} = \begin{bmatrix} fx \\ fy \\ z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} , \tag{3.3}$$

where λ corresponds to the third coordinate of P, which is precisely the distance from the scene point to the plane xy. From equation (3.3) it can be seen that every image point is the projection of all the points along the ray passing through the viewed world point and the center of projection. Consequently, using just a single pinhole camera it is not possible to estimate the distance to a viewed point.

3.1.1   Distortion

Real-world cameras do not use a pinhole to capture light. The amount of light that can pass through a pinhole is very small, thus, to get an acceptable image, a very long time of exposure would be
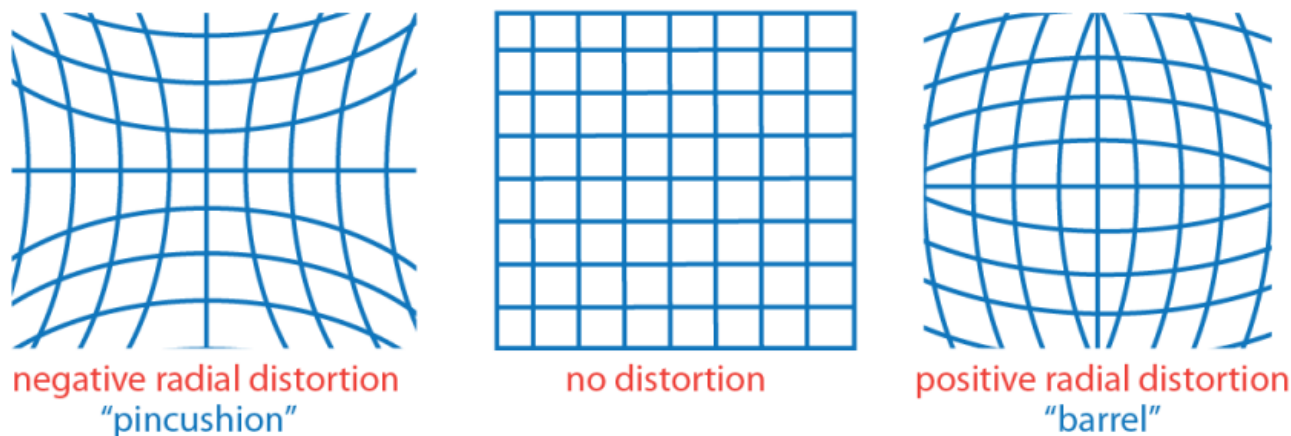
Figure 3.2: Radial distortion. *Left:* Pincushion distortion. *Center:* No distortion. *Right:* Barrel distortion. *Image credit:* MathWorks Inc.

needed and this is not practical in most situations. For this reason, cameras use lenses to increase their capability of gathering light. However, this comes with a cost, because the use of lenses introduces distortion of different types to the images that should be corrected before processing.

Radial distortion causes straight lines in the world to be projected as curves in the image. Depending on the type of radial distortion, the straight lines get curved towards the image border (*barrel* distortion, fig. 3.2, *right*) or towards the image center (*pincushion* distortion, fig. 3.2, *left*). The curvature of the lines is proportional to their radial distance. To model radial distortion, often low-order polynomials are used. More complete distortion models also consider *tangential distortion* (resulting from the lens not being exactly parallel to the image plane) and *decentering distortions*. More details on lens distortion can be found in [17], [80] and [82], for example.

3.2   Digital cameras

A digital camera has several components and properties that define the quality of the obtained image and the suitability of the camera itself to a particular task. Some of the most important aspects are described below.

3.2.1   Sensor

A digital camera uses a $m \times n$ matrix sensor of light-sensitive picture elements, or pixels, to record the viewed scene (figure 3.3, *left*). This matrix sensor is made either in *charge-coupled device* (CCD) or *complementary metal-oxide semiconductor* (CMOS) technology.

In a CCD, photons falling upon the sensor cause electrons to be released. During the exposure time, the loose electrons are accumulated in each pixel's "well". This principle is illustrated in figure 3.3, *center*. When the exposure has finished, the charges stored in the pixels are transferred in succession to a corner in the CCD where the sense amplifier is located. It converts each of the accumulated charges to a voltage. This voltage is afterwards converted to a digital value using an analog-to-digital converter (ADC).
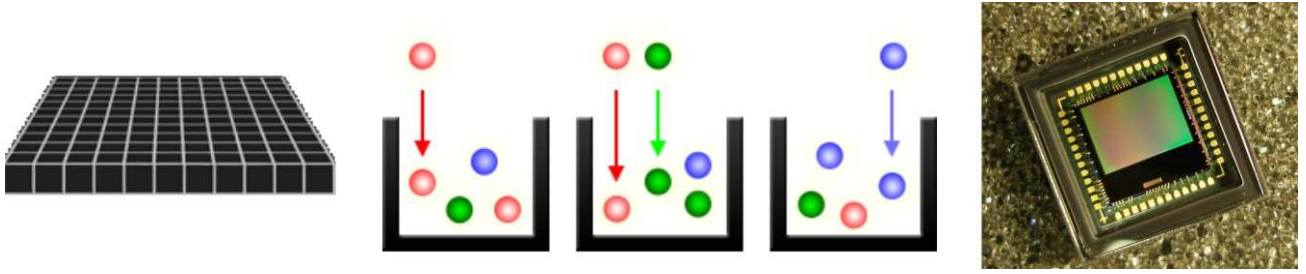
Figure 3.3: Digital image sensors. *Left:* Schematic of a matrix sensor. *Center:* An illustration of the working principle of a CCD. *Right:* A CMOS image sensor. *Image credit: Left, Center:* Cambridge in colour. *Right:* Wikimedia Commons.

Like CCD chips, pixels in CMOS sensors accumulate charge during the exposition, but CMOS sensors contain in each pixel a photodetector and an amplifier, thus no charge transfer is required. Photons hitting the sensor have a direct effect on the conductivity of the photodetector, affecting the charge flow. The output of the amplifier of each pixel is also digitized using the ADC of the camera. CMOS sensors, pictured in figure 3.3, *right*, are cheaper, because they do not require special manufacturing facilities as they can be manufactured in a traditional semiconductor factory. Also, they require much less power than CCD chips.

Traditionally CCD sensors offered better image quality than CMOS sensors. Also, CMOS sensors were affected by the effects of using a rolling shutter, in which the image is not acquired in an instant but progressively. This is specially important when capturing quickly-changing scenes, because the obtained image may appear distorted. Nowadays, good image quality can be obtained with either kind of sensor, and CMOS sensors are starting to include global shutters, which allow to acquire the acquisition of the entire scene in an instant.

Independently of the technology used to manufacture the image sensor, several factors related to it affect the quality of the obtained image. The *chip size* combined with the *spatial resolution* determines the pixel size. A larger pixel size is preferred, since each pixel can be more photo-sensitive, therefore increasing the signal-to-noise ratio (SNR). In an image sensor, square pixels are desired. *Sensor noise* is also an important factor to consider, and *dark current noise* is one of the main sources of it. Dark current is a small current that flows in the photosensitive elements when no photons are entering. Other aspects related to the camera sensor that affect image quality are dynamic range and color accuracy.

When color perception is required, there are several types of solutions available. They differ in the way they perform the separation of colors. Among the available options there is the Bayer filter, the Foveon X3 sensor or the use of a dichroic prism together with 3 separate image sensors. In our case, we will focus on monochromatic cameras.

3.2.2   ADC

In many cameras, before the analog-to-digital conversion takes place, the sensed signal is enlarged by an amplifier. The gain in the amplifier can be automatically controlled or specified by the user. The process of amplification can induce *amplifier noise*. The output of the amplifier is transferred to an ADC. The resolution of the ADC, i.e. the number of bits it delivers, is important. For computer vision applications, specially motion detection and stereo correspondence, it is often important to have an ADC with a bit depth higher than 8 bits. In addition, the digitizing process can generate *quantization*

noise.

3.2.3 Connectivity

For some applications it is important to synchronize the image acquisition to some other events, or to synchronize the acquisition between two or more cameras (i.e., for stereo vision). Cameras that among their connectivity options feature digital I/O lines allow this.

3.2.4 Data rate

The data rate of a digital camera is the amount of information that a camera is capable to acquire and transfer and it is dependent of the camera bus. For example a Firewire-B camera has a nominal data rate of 800 Mbps while a GigE camera has a nominal data rate of 1 Gb/s. Given a camera, spatial times temporal resolution is typically a constant defined by the maximum available data rate. You can select a region of interest (ROI) in the image sensor, allowing an increase in the framerate. For example, a camera which captures images with a resolution of 4,608 × 3,288 (15.1 Megapixels) at 10 frames per second (fps), records 151 Megapixels (Mpx) per second. The same camera may also support to record 2,304 × 1,644 (3.8 Mpx) at 40 fps, which is also 151 Mpx per second.

3.3 Camera Calibration

The perspective projection model introduced in section 3.1 does not consider the pixelization that takes place in the image sensor, so it needs to be further extended. First, in computer imaging, it is common practice to consider as the origin of the image coordinate system the upper left corner. In this new image coordinate system, the coordinates of the principal point are $[u_0, v_0]$. Also, because the coordinates of a point are measured in pixels, two scale factors are required, one for the horizontal and one for the vertical direction, to consider the possibility of the pixels not being perfect squares. We assume an image sensor in which the u and v axes are orthogonal, which is valid for good quality image sensors. This yields the following modified projection equations:

$$u = k_u f \frac{x}{z} + u_0 \ , \tag{3.4}$$

$$v = k_v f \frac{y}{z} + v_0 \ , \tag{3.5}$$

where $[u_0, v_0]$ are the coordinates of the principal point, and k_u, k_v are the scale factors in the horizontal and vertical directions, respectively. k_u is the inverse of the effective pixel size along the u direction. It has units $pixels \cdot m^{-1}$. The same applies for k_v along the v direction.

Using homogeneous coordinates, the updated projection equations turn into:

$$\begin{bmatrix} \lambda u \\ \lambda v \\ \lambda \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \ . \tag{3.6}$$

In the previous equation, $\alpha_u = f k_u$ and $\alpha_v = f k_v$ represent the focal lengths in pixels in the horizontal and vertical directions, respectively.
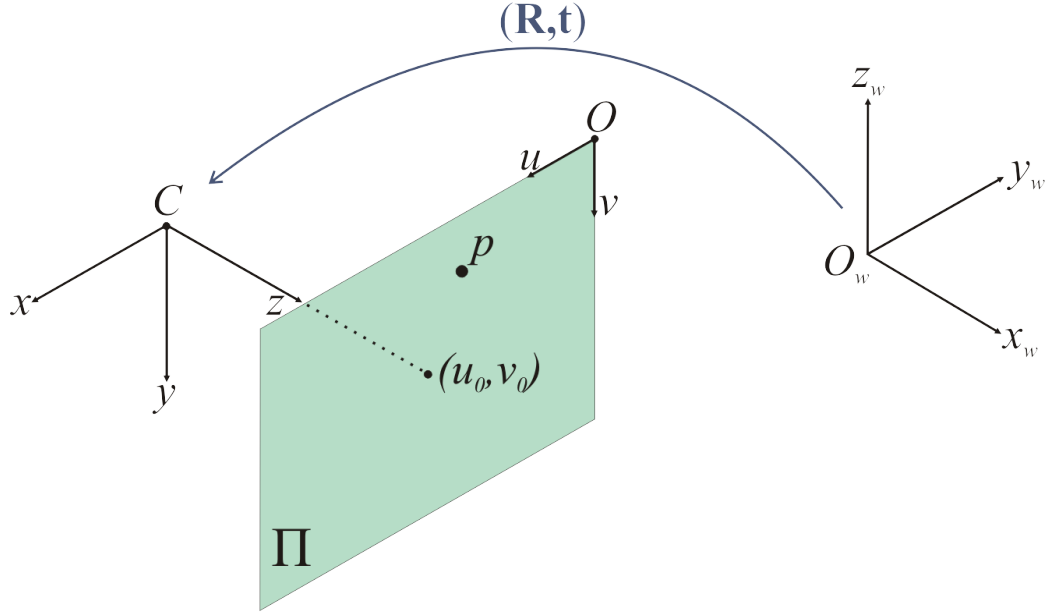
Figure 3.4: Camera and World coordinate systems

The model so far does not consider the fact that usually the camera coordinate system and the world coordinate system do not coincide. Thus, the rigid transformation between the two reference frames should be considered, as pictured in figure 3.4. This rigid transformation is composed of a rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{t}$ and is represented as:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{R} \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + \mathbf{t} \ . \tag{3.7}$$

Taking into account this transformation, equation (3.6) can be updated to:

$$\begin{bmatrix} \lambda u \\ \lambda v \\ \lambda \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \ , \tag{3.8}$$

which can be rewritten as:

$$\lambda \tilde{p} = \mathbf{K}[\mathbf{R}|\mathbf{t}]\tilde{P}_w \ , \tag{3.9}$$

where

$$\mathbf{K} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \ , \tag{3.10}$$

is known as the *camera intrinsics matrix*. $\alpha_u, \alpha_v, u_0, v_0$ are called *intrinsic parameters* and the rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t}$ together are called *extrinsic parameters*.

Camera calibration is a process that attempts to obtain accurate estimations of the intrinsic and extrinsic parameters of the camera model. Because these parameters define how a scene point is projected onto an image point, if both the pixel coordinates of some image points $\tilde{p}$ and the 3D

coordinates of the corresponding scene points $\tilde{P}$ are known, it is possible to obtain the unknown parameters $\mathbf{K}, \mathbf{R}$ and $\mathbf{t}$ by solving the perspective projection equation (3.9).

In 1987, Tsai [87] developed one of the first camera calibration techniques. It requires a few images of a planar pattern taken in different positions. The precise knowledge of the plane motion is needed. From the obtained images, one needs to obtain corresponding 3D point coordinates and 2D pixel coordinates. Tsai's technique has two stages. First, it computes the position and orientation of the camera, and, second, it determines the internal parameters.

More recently, Zhang proposed an alternative calibration technique [108]. To utilize this technique, a planar pattern featuring corners that are easy to detect in an image is required. Zhang's method is easy to use by both expert and novice users. To use the method, the user is required to take several pictures of the pattern in different positions and orientations. The motion doesn't need to be known. Then, the feature points in the images should be detected. By knowing the 2D position of the corners on the planar grid and their corresponding pixel coordinates on each image, the intrinsic and extrinsic parameters can be calculated. Then, an estimation of model parameters for radial and tangential distortion is obtained by solving a linear least-squares problem. Finally, the obtained estimates for all parameters are refined by solving a non-linear minimization problem, for example using the Levenberg-Marquardt algorithm.

3.4 Stereo Imaging

Stereo vision consists in recovering depth information from two images obtained from a pair of cameras which have a different view of the same scene. If one knows the camera intrinsics matrix for each camera and the geometry of the stereo pair – this is, the rotation matrix and translation vector between the cameras – the depth of a scene point can be obtained by finding the coordinates of the projection of this point in both images, i.e. finding *image correspondences*, and exploiting the knowledge of the geometry of the system.

The depth recovery process is simplified when we use a stereo setup in *standard stereo geometry*. This setup is also called *canonical stereo* or *frontal parallel*. In such arrangement, the two cameras are assumed to be identical, with the same spatial resolution, the same focal length and using ideal lenses that produce no distortion. Also, the optical axes of both cameras are assumed to be parallel. The cameras are placed in such a way that the image planes are coplanar and row-aligned. The distance between the centers of projection is a known quantity T, called *baseline*. Under these assumptions, depth estimation is easy using triangulation.

3.4.1 Triangulation

A canonical stereo pair is pictured in figure 3.5. A scene point P is observed in both cameras. The horizontal coordinates of the point's projection in the left and right camera images are u_l and u_r respectively. Note that in a canonical stereo setup, corresponding points are located in the same row. This will be further explained in section 3.5.

Using similar triangles, we can derive the following expression:

$$\frac{z}{T} = \frac{z - f}{T - (u_l - u_r)} \ ,$$

(3.11)
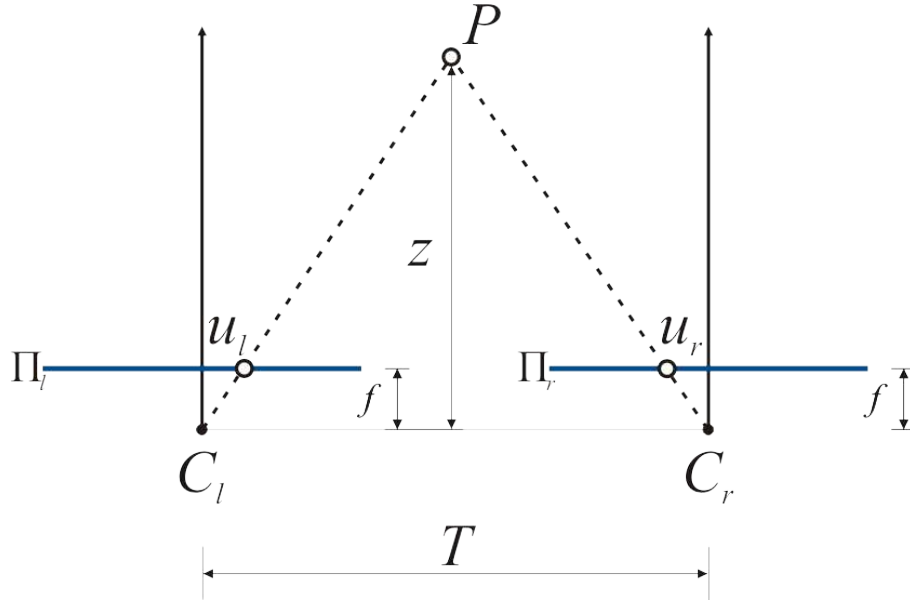
Figure 3.5: Depth estimation with a canonical stereo pair. Using triangulation, it is possible to determine the depth of a scene point P seen in both cameras in a canonical stereo pair.

from which we obtain:

$$z = \frac{fT}{d} \ , \tag{3.12}$$

where $d = u_l - u_r$ is defined as the *disparity*. From equation (3.12) it can be seen that depth is inversely proportional to disparity. This results in the fact that stereo vision systems have high depth resolution only for objects placed near the stereo camera. This effect is pictured in figure 3.6. It also results that the depth resolution is higher with larger baselines.

3.4.2   Epipolar Geometry

Manufacturing differences in lenses and cameras, added with the difficulty to align perfectly the cameras, make it almost impossible to obtain a canonical stereo pair in the real-world. Therefore, it is important to understand the geometry of a generic stereo pair. This geometry is commonly called *epipolar geometry* and is pictured in figure 3.7.

Both left and right cameras are modeled with the pinhole model. Each one has a center of projection, labeled C_l and C_r, and their projective planes are labeled Π_l and Π_r respectively. A point P in the scene is projected onto each image plane, resulting in points p_l and p_r. In epipolar geometry, an *epipole* is defined as the image of the center of projection of one camera into the other camera. Epipole e_l is the projection of C_r onto Π_l, for example. The lines that pass through the projections of P in the projective planes and the corresponding epipoles are called *epipolar lines*, which in this case are $p_l e_l$ and $p_r e_r$. The *epipolar plane* is the plane formed by the scene point P and the two centers of projection.

In figure 3.7 we can see that p_l is the projection of P in the left camera, but it could also be the projection of any of the points along the ray $C_l P$ located between P and p_l. This exhibits the fact that, with a single camera, we cannot determine the depth of the viewed point P. But it is interesting to note that, no matter the 3D location of a viewed point along the aforementioned projective ray, the
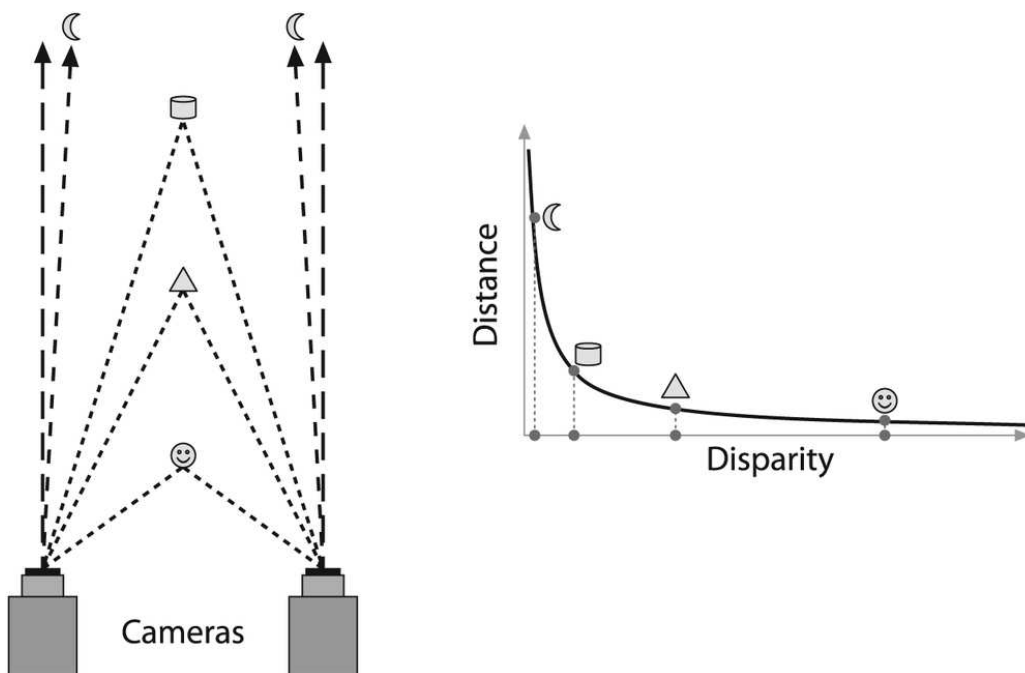
Figure 3.6: Relationship between depth and disparity. The relationship between depth and disparity is non-linear. This results in the fact that relatively precise depth measurements are only possible when the viewed objects are close to the stereo rig. *Image from:* [17].

corresponding pair in the other camera must lie in the corresponding epipolar line. This is known as the *epipolar constraint*. This constraint is of great help in reducing the complexity of the correspondence search problem. Instead of a 2D search problem, it becomes a simpler 1D search, because we no longer need to look for a corresponding point in the whole image, we only need to search for it on the associated epipolar line. However, the problem can be simplified even more, if we mathematically transform a general stereo pair into standard stereo geometry. This is known as *stereo rectification* and is briefly explained in next section.
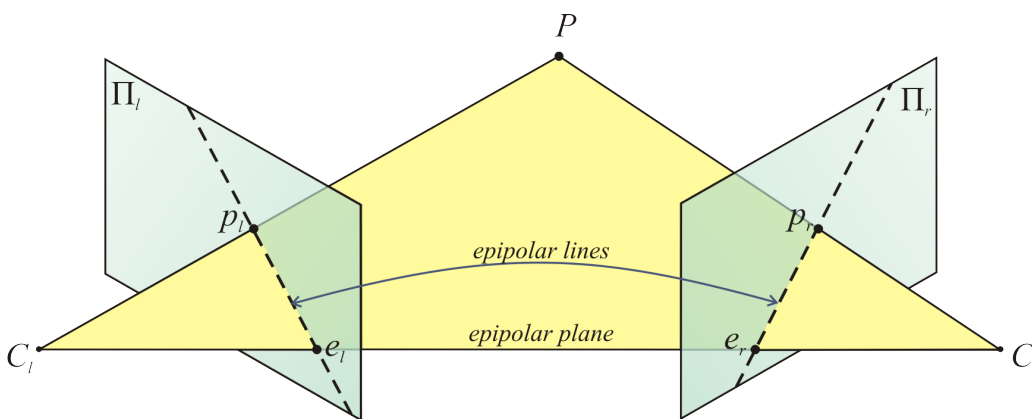
Figure 3.7: Epipolar Geometry in a general stereo pair.
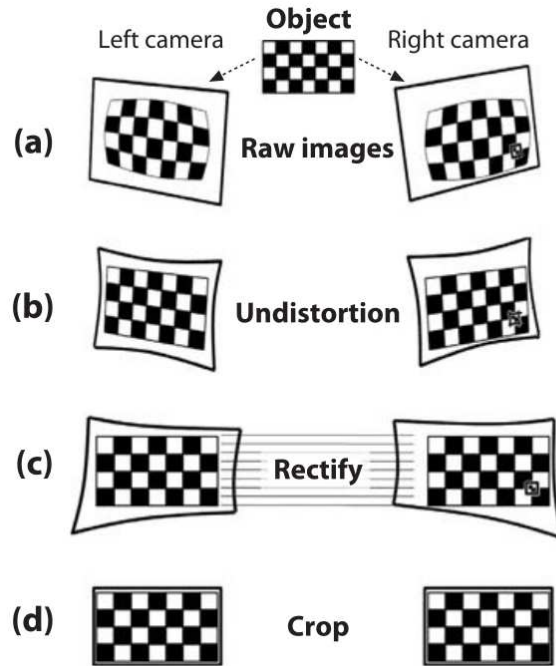
Figure 3.8: The image rectification process. *Image from:* [17]

3.5   Stereo Rectification

Stereo rectification consists in reprojecting mathematically the image planes of the two cameras in the stereo pair so that they become coplanar and row-aligned, that is, as if they had been acquired by a stereo pair in standard stereo geometry. After this process, the epipoles are located at infinity, and the corresponding epipolar lines become collinear and parallel. After rectification, the correspondence problem becomes simpler and computationally less expensive because the search for corresponding points is performed along the rows of the rectified images.

The process of stereo rectification is pictured in figure 3.8. After the images are acquired (**a**), it is important to remove the lens distortion (**b**). Then the actual rectification process takes place (**c**). Finally, the images can be cropped so that only the common areas of view and regions of interest are preserved (**d**).

Rectification algorithms need to have knowledge of either the *Fundamental Matrix* $\mathbf{F}$, a 3×3 matrix which relates corresponding points in image pixel coordinates between stereo images, or the rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t}$ that relate the coordinate systems of the cameras. $\mathbf{R}$ and $\mathbf{t}$ can be obtained by *stereo calibration*. The stereo calibration process makes use of individual camera calibration information and simultaneous views by both cameras of a known calibration pattern to obtain the desired values. More information about stereo calibration can be found in [17], for example.

There exist many algorithms that allow to obtain a rectified stereo pair, among them the ones proposed in [32], [38], [50] and [65]. Below we briefly describe Bouguet's algorithm for stereo rectification, which he uses on his Camera Calibration Toolbox for Matlab [56]. The goals that Bouguet pursued when he developed his algorithm was to make the common viewing area maximum and to minimize the amount of changes rectification introduces to the images. The steps performed by the algorithm are the following:

1. The rotation matrix **R** that makes the orientation of the right camera to be that of the left camera, previously obtained by stereo calibration, is split in two, and each camera performs half a rotation. After this step, both cameras have the same orientation w.r.t. the baseline vector, but are not yet row-aligned.

2. To obtain row-alignment between the cameras, they are rotated so that the u-axis of each camera is parallel to the baseline vector. This rotation is the same for both cameras.

More information about stereo rectification algorithms can be found in [17], [50], or [82], for example.

3.6   Stereo Correspondence

The stereo correspondence, or *stereo matching*, problem can be defined as follows: Given an image point in one of the images (the *base image*), find the corresponding image point in the other image (the *match image*). Two image points are in correspondence if they are projections of the same point in the viewed scene. Having the image coordinates of corresponding points allows us to obtain a depth estimate of the point in the scene.

When looking for corresponding points, it is assumed that an object in the scene looks similar in both images. Then, using a similarity measure, it is possible to pick one image point in the base image, and find the most similar point along the corresponding epipolar line in the match image. This point is signaled as the searched corresponding pair.

Although simple in concept, the search for correspondences is a challenging task that can be affected by several factors. Since the cameras perceive the scene from different viewpoints, *occlusions* can occur and parts of the scene could appear in one image, but not in the other. Because of this, a point in the base image may have no corresponding point in the match image. Also, *projective distortions* can make the search difficult, because a scene feature is projected in a different way in the two images. *Photometric distortion* also adds up to the complexity of the matching problem. Because of non-Lambertian surfaces, i.e. surfaces that do not reflect incident light uniformly in all directions, light reflections may be perceived differently by the two cameras. As a consequence of all these factors, the correspondence search based solely on the independent maximization of similarity measures may yield a wrong corresponding point.

However, some constraints make the search for matching points simpler. If the image pair is rectified, the epipolar lines are horizontal and a point lying in row y in the base image must have its corresponding point in row y on the match image. This is due to the epipolar constraint. By the *uniqueness constraint*, a point in the base image can have up to one corresponding point, or zero if an occlusion occurs. The *ordering constraint*, although not always satisfied, may be also of help. It says that if points P, Q and R appear in that order in the base image, their corresponding points in the match image should appear in that exact same order.

The result of performing stereo matching in a stereo pair is a *disparity map*. A disparity map is a grayscale image where the intensity of a pixel with a specific coordinate is proportional to the disparity between the image point in that same coordinate in the base image and its corresponding pair in the match image, see (3.12). Often, instead of gray levels, false color is used to present disparity maps so that they are better appreciated. An example disparity map is shown in figure 3.9.

Figure 3.9: *Right:* An example *ground truth* disparity map from the Tsukuba pair. The left and right frames of the scene are pictured in the *Left* and *Center* images, respectively. *Image credit:* University of Tsukuba.

3.6.1   Similarity Measures

Similarity measures can be either window or pixel-based. Typical pixel-based similarity measures include the absolute difference of intensity (AD), the squared difference of intensity (SD) and the absolute difference of gradient (GRAD). The matching cost between a pixel $\mathbf{p}_B$ with coordinates (u,v) in the base image I_B and a pixel $\mathbf{p}_M$ with coordinates $(u+d,v)$, where d is the disparity, in the match image I_M, is defined respectively for each of the similarity measures mentioned by:

$$E_{AD}(u,v,d) = |I_B(u,v) - I_M(u+d,v)| \tag{3.13}$$

$$E_{SD}(u,v,d) = (I_B(u,v) - I_M(u+d,v))^2 \tag{3.14}$$

$$E_{GRAD}(u,v,d) = |\nabla_x I_B(u,v) - \nabla_x I_M(u+d,v)| + |\nabla_y I_B(u,v) - \nabla_y I_M(u+d,v)| \ . \tag{3.15}$$

Another pixel-based matching cost we can mention is the hierarchical calculation of pixel-wise *Mutual Information* proposed by Hirschmüller [53]. Finally, we describe the sampling-insensitive similarity measure of Birchfield and Tomasi (BT) [12]. Instead of just comparing pixel values in integer steps looking for the best match, they compare each pixel in one image with extreme points of a linearly interpolated function obtained from the pixel intensities in the other image. This similarity measure deals better with sub-pixel disparity shifts. Denoting with $\hat{I}_B(w,v)$ the value at w of the linearly interpolated function corresponding to the v scanline in the base image and defining $\hat{I}_M(w,v)$ equivalently for the match image, the BT similarity measure is formally defined as:

$$E_{BT}(u,v,d) = \min \{ d_{BM}(u,v,d), d_{MB}(u,v,d) \} \ , \tag{3.16}$$

where

$$d_{BM}(u,v,d) = \min_{u+d-\frac{1}{2} \leq w \leq u+d+\frac{1}{2}} \left| I_B(u,v) - \hat{I}_M(w,v) \right| \ , \tag{3.17}$$

and

$$d_{MB}(u, v, d) = \min_{u-\frac{1}{2} \leq w \leq u+\frac{1}{2}} \left| \hat{I}_B(w, v) - I_M(u + d, v) \right| . \tag{3.18}$$

On the other hand, window-based similarity measures try to identify corresponding points by comparing neighborhoods around a given pixel. We define a pair of windows with size $(2a+1) \times (2b+1)$ centered around $\mathbf{p}_B$ and $\mathbf{p}_M$. The similarity is computed using the pixel intensities inside the two patches. Common-used examples of these kind of similarity measures are extensions of pixel-based measures such as the *Sum of Squared Differences* (SSD) and the *Sum of Absolute Differences* (SAD). These matching costs are particularly affected by image noise and differences in intensity, even small, between the base and match images. A more robust alternative to these measures are their respective *Zero-Mean* versions, where the mean value of the window is subtracted from all the elements before calculating the sum of differences. Denoting as $\overline{I_B}$ the mean value of the defined window in the base image and defining $\overline{I_M}$ equivalently in the match image, the Zero-Mean SSD (ZSSD) and the Zero-Mean SAD (ZSAD) data costs are defined respectively by:

$$E_{ZSSD}(u, v, d) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} \left[\left(I_B(u+i, v+j) - \overline{I_B} \right) - \left(I_M(u+d+i, v+j) - \overline{I_M} \right) \right]^2 \tag{3.19}$$

$$E_{ZSAD}(u, v, d) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} \left| \left(I_B(u+i, v+j) - \overline{I_B} \right) - \left(I_M(u+d+i, v+j) - \overline{I_M} \right) \right| . \tag{3.20}$$

The more similar the windows around the pixels $\mathbf{p}_B$ and $\mathbf{p}_M$, the smaller the matching cost. If the image patches match perfectly, the obtained cost will be zero.

The *Normalized Cross Correlation* (NCC) is also usually considered and is defined by:

$$E_{NCC}(u, v, d) = 1 - \frac{\sum_{i=-a}^{a} \sum_{j=-b}^{b} [I_B(u+i, v+j) - \overline{I_B}][I_M(u+d+i, v+j) - \overline{I_M}]}{\sqrt{\sigma^2_{I_B(u,v)} \cdot \sigma^2_{I_M(u+d,v)}}} , \tag{3.21}$$

where

$$\sigma^2_{I_B(u,v)} = \sum_{i=-a}^{a} \sum_{j=-b}^{b} [I_B(u+i, v+j) - \overline{I_B}]^2 \tag{3.22}$$

$$\sigma^2_{I_M(u+d,v)} = \sum_{i=-a}^{a} \sum_{j=-b}^{b} [I_M(u+i, v+j) - \overline{I_M}]^2 . \tag{3.23}$$

NCC is invariant to affine intensity changes. The output range of NCC is $[-1, 1]$, where 1 corresponds to the maximum similarity between the two windows.

The *Census* transform [107], a non-parametric measure which generates a bit vector containing values indicating whether a pixel in the window has a greater or smaller value than the central pixel, is commonly used as well. The resulting data cost is the *Hamming distance* between the bit vectors

generated from the windows in the base and match images. The Census transform is analytically defined as:

$$E_{CEN}(u,v,d) = \sum_{(u',v')\in\mathcal{N}(u,v)} \rho(u,v,u',v',d) \ , \tag{3.24}$$

with

$$\rho(u,v,u',v',d) = \begin{cases} 0 & I_B(u',v') \otimes I_B(u,v) \quad \text{and} \quad I_M(u'+d,v') \otimes I_M(u+d,v) \ , \\ 1 & \text{otherwise} \ . \end{cases} \tag{3.25}$$

with $\otimes$ being either $>$ or $<$ in both cases, and $\mathcal{N}(u,v)$ representing the set of all coordinate pairs associated to pixels in the neighborhood of the pixel $\mathbf{p}_B$ located at (u,v). This neighborhood is usually a window centered around $\mathbf{p}_B$, but not including it.

Comprehensive studies on the performance of different image similarity measures for stereo matching can be found in [54] and [77]. The latter paper deals particularly with the issue of matching images with radiometric differences. Among others, the census transform and the hierarchical mutual information [53] have performed well in the tests performed by the authors. More details about the image similarity measures mentioned in this section and others can be found in [57] and [82], for example.

3.6.2   Disparity Space Image

Given a stereo pair with base and match images of size $m \times n$ and having selected a similarity measure, we can now determine how likely are two pixels of being correspondent. We will denote as $E_{data}(\mathbf{p},d)$ the result of the similarity calculation between a pixel $\mathbf{p}$ with coordinates (u,v) in the base image and a pixel $\mathbf{q}$ with coordinates $(u+d,v)$ in the match image. Because the disparity is discretized, usually in pixel units but sub-pixel refinement is possible, the results of the calculations performed can be saved in a disparity space image (DSI) $\mathbf{C}(\mathbf{p},d)$ for further processing. The DSI is a 3D matrix with dimensions $m \times n \times d_{max}$, where d_{max} is the maximum possible disparity for the stereo pair.

The contents of the DSI comply with the following:

$$\mathbf{C}(\mathbf{p},d) = \begin{cases} E_{data}(\mathbf{p},d) & \text{if } 0 < d < min(d_{max}, n-u) \ , \\ -1 & \text{otherwise} \ . \end{cases} \tag{3.26}$$

In equation (3.26), the label -1 represents a non-calculated disparity or an invalid one.

3.6.3   Local Matching

Denoting with $\mathbf{p}$ a pixel with coordinates (u,v) in the base image, and considering that disparities are assigned by a disparity-assigning function $d = f(\mathbf{p})$, a local matcher attempts to find the function f that minimizes the following energy function:

$$E_D(f) = \sum_{\mathbf{p}} E_{data}\left(\mathbf{p}, f(\mathbf{p})\right) \ . \tag{3.27}$$

This energy function is minimized by calculating the DSI for a given similarity measure and selecting for every pair (u,v) the disparity d associated with the minimum matching cost, that is, choosing the
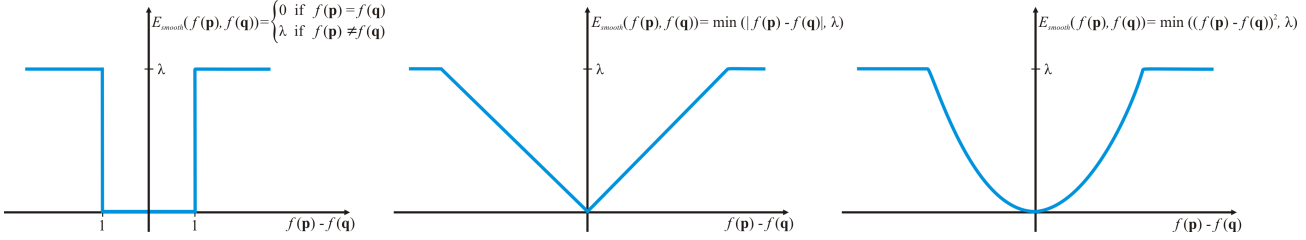
Figure 3.10: Discontinuity Preserving Smoothness Terms. *Left:* The Potts model. *Middle:* The linear truncated cost function. *Center:* The quadratic truncated cost function.

most similar pixel. This strategy is known as *the winner takes it all* (WTA) as is defined formally by:

$$\underset{0 \leq d \leq d_{max}}{\arg\min} \quad \{\mathbf{C}(\mathbf{p}, d) \geq 0\} \quad \forall \mathbf{p} \in I_B \ . \tag{3.28}$$

The disparity selection is locally optimal for every pixel. However, since no smoothness terms are involved and because of wrong matches caused by the factors described in section 3.6, the disparity maps created may have depth artifacts (such as a lot of discontinuities). Implementations of local matchers often use validity checks such as the *left-right consistency check* and *confidence measures* calculated on the selected disparities to keep only consistent and high-confidence disparities in the final disparity maps.

3.6.4 Global Matching

Global stereo matchers look for a disparity assignment that minimizes a global energy function that is made up of data terms *and* smoothness terms. The smoothness terms ensure that every assigned disparity has a global dependency from all the other calculated disparity values. The presence of the smoothness terms allows for a more accurate solution without the depth artifacts present in the solutions obtained from a local matcher. Examples of disparity maps obtained from local and semi-global matchers can be seen in figure 3.11. The global energy function to minimize has the following form:

$$\sum_{\mathbf{p}} \left[E_{data}(\mathbf{p}, f(\mathbf{p})) + \sum_{\mathbf{q} \in \mathcal{N}(\mathbf{p})} E_{smooth}(f(\mathbf{p}), f(\mathbf{q})) \right] \ , \tag{3.29}$$

where $\mathcal{N}(\mathbf{p})$ is a pixel neighborhood around $\mathbf{p}$. The smoothness term E_{smooth} can be any monotonically increasing function of the difference of disparity. However, the use of smoothness terms based on robust functions [14, 83] that allow the preservation of discontinuities is desirable. Examples of such functions are the Potts model, the linear truncated and the quadratic truncated cost functions, see figure 3.10.

The problem of minimizing the global energy function (3.29) using common smoothness terms that allow for discontinuity preservation is NP-Hard [15]. Thus, several strategies have been proposed in the literature to obtain an approximation of the solution limiting the area that influences the disparity assignment of a given pixel, making the problem tractable. *Dynamic Programming* [13] and *Scanline Optimization* [77] solutions perform the optimization along each of the epipolar lines individually in polynomial time. However the disparity maps obtained with these techniques often suffer from streaking effects, i.e. the lack of smoothness between disparities obtained in neighboring epipolar lines, because strong constraints are considered in the horizontal direction, but none on the vertical direction. In the following sections we describe some other approaches that attempt to obtain a minimum of the energy function (3.29), offering in general better results.
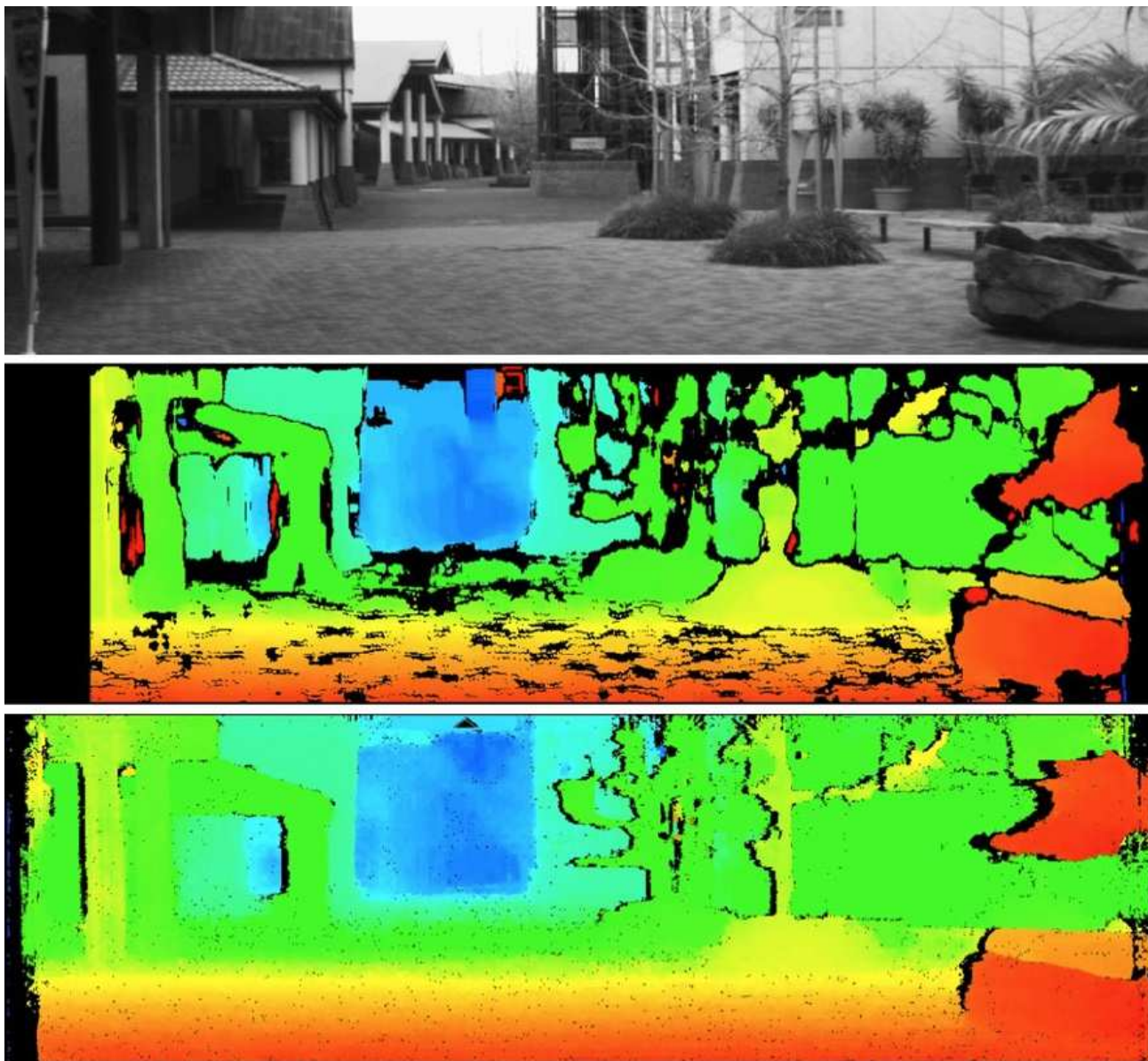
Figure 3.11: *Top:* A frame of a stereo sequence recorded in the University of Auckland. *Middle:* The obtained disparity map using Konolige's Block Matching algorithm [61] as implemented in OpenCV [72]. *Bottom:* A disparity map obtained using iSGM [52], a semi-global matcher. *Image from:* [57].

Semi-Global Matching

An approximation to the solution of the energy function (3.29) can be obtained by *Semi-Global Matching* (SGM) [53]. SGM uses a slightly modified energy defined as follows:

$$E(f) = \sum_{\mathbf{p}} \left[E_{data}(\mathbf{p}, f(\mathbf{p})) + \sum_{\mathbf{q} \in \mathcal{N}(\mathbf{p})} P_1 \mathbf{T} \left(|f(\mathbf{p}) - f(\mathbf{q})| = 1 \right) + \sum_{\mathbf{q} \in \mathcal{N}(\mathbf{p})} P_2 \mathbf{T} \left(|f(\mathbf{p}) - f(\mathbf{q})| > 1 \right) \right] ,$$

(3.30)

where $\mathbf{T}$ denotes the function that returns 1 when its argument is true, and returns 0 otherwise, P_1 is the penalty for disparity changes of 1, and P_2 is the penalty for all larger disparity changes. Also, $P_2 \geq P_1$. Assigning a lower penalty to small changes in disparity allows the adjustment to curved or slanted surfaces. Limiting the penalty for larger disparity changes allows the preservation of discontinuities.

The approach proposed to approximately minimize this energy is the resolution of many linear scanline optimization problems in the DSI. For several directions, the minimum cost path that ends in a given pixel $\mathbf{p}$ at disparity d is calculated, see figure 3.13, *left*, where 16 directions are considered. The cost $C_{\mathbf{r}}(\mathbf{p}, d)$ along a path in direction $\mathbf{r}$ to the pixel $\mathbf{p}$ at disparity d is recursively defined as follows:

$$C_{\mathbf{r}}(\mathbf{p}, d) = E_{data}(\mathbf{p}, d) + \min \begin{cases} C_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, d) \\ C_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, d - 1) + P_1 \\ C_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, d + 1) + P_1 \\ \min_i C_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, i) + P_2 . \end{cases}$$

(3.31)

In (3.31), the lowest cost of the previous pixel $\mathbf{p} - \mathbf{r}$ is added to the current data cost E_{data}, together with the corresponding smoothness penalty. The pixel-based data cost E_{data} can be E_{BT} or the mutual information cost described in [53]. For a pixel $\mathbf{p}$ and disparity d, the smoothed aggregated cost $S(\mathbf{p}, d)$ is obtained summing the costs of all paths, this is:

$$S(\mathbf{p}, d) = \sum_{\mathbf{r}} C_{\mathbf{r}}(\mathbf{p}, d) .$$

(3.32)

The calculation is performed for every possible disparity and the winning disparity for each pixel is selected by WTA. Disparity maps obtained using SGM are usually good and implementations in real-time exist [39, 40].

Belief Propagation

Belief Propagation (BP) [34, 81], described in this section, and *Graph Cuts* (GC) [15, 59], described in section 3.6.4 are techniques that have been developed to solve approximately the stereo correspondence problem using a *Markov Random Field* model. A Markov Random Field (MRF) is an undirected graphical model that can encode spatial dependencies. An MRF consists of nodes and links, and, in contrast to other graphical models, it can contain cycles. An MRF used to represent the stereo matching problem is shown in figure 3.12. The links between nodes in an MRF denote a dependency. Refering to figure 3.12, it can be seen that the value of a hidden node depends only on the values on the neighboring nodes. This assumption is called the *Markov assumption* and it is what allows to solve approximately for the values of the hidden nodes (disparities). Note that the MRF formulation of the stereo matching problem corresponds precisely to the energy function (3.29), when $\mathcal{N} = \mathcal{N}_4$.
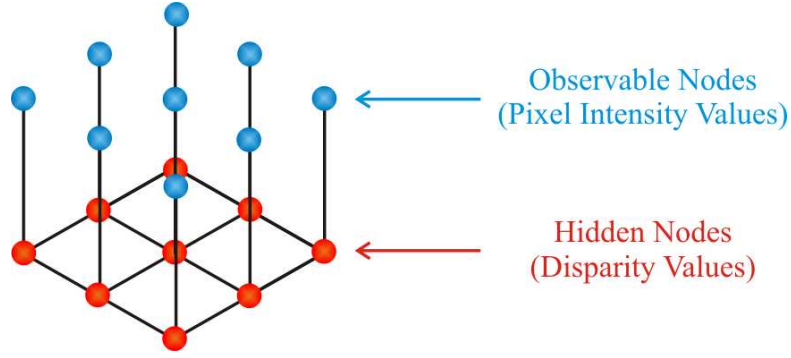
Figure 3.12: Markov Random Field for a 3×3 image. The blue nodes correspond to the observed variables, in this case, the pixel intensities. The red nodes correspond to the hidden variables. For stereo matching, they are the disparity values. A link between nodes denotes a dependency.

Belief Propagation was developed as an optimization method for graphical models with a tree structure. In such cases, it produces the global minimum and is guaranteed to converge. It is possible to apply BP to a graphical model with loops, such as an MRF, to obtain an approximation to the optimal solution. When BP is used to solve loopy graphical models, it is called Loopy BP (LBP). LBP is a message passing algorithm. The idea is that each node in the graphical model "tells" to each of its neighbors its "opinion" about every possible value they may have. The degree of certainty for each possible value is expressed in a quantity representing a likelihood. A node passes a message to a neighboring node only when it has received all incoming messages. To avoid a deadlock, prior to running the algorithm, all messages must be initialized to a suitable value. For the min-sum LBP algorithm we will discuss below, all messages are initialized to zero.

The min-sum LBP algorithm is iterative. The messages generated from all the nodes are passed in parallel. Each message is a vector of dimension given by the number of possible values a node may have. In the stereo matching case, the dimension of the message vector is the number of disparities. To make the notation simpler, we denote $f(\mathbf{p})$ as $f_{\mathbf{p}}$. Also, let $\mathbf{m}_{\mathbf{p} \to \mathbf{q}}^{t}$ be the message sent from node $\mathbf{p}$ to node $\mathbf{q}$ at iteration t. We have $\mathbf{m}_{\mathbf{p} \to \mathbf{q}}^{0} = \mathbf{0}$. The message update at each iteration is calculated with the following equation:

$$\mathbf{m}_{\mathbf{p} \to \mathbf{q}}^{t}[f_{\mathbf{p}}] = \min_{f_{\mathbf{p}}} \left(E_{data}\left(\mathbf{p}, f_{\mathbf{p}}\right) + E_{smooth}\left(f_{\mathbf{p}}, f_{\mathbf{q}}\right) + \sum_{\mathbf{s} \in \mathcal{N}(\mathbf{p}) \backslash \mathbf{q}} \mathbf{m}_{\mathbf{s} \to \mathbf{p}}^{t-1}[f_{\mathbf{p}}] \right) , \qquad (3.33)$$

where $\mathcal{N}(\mathbf{p}) \backslash \mathbf{q}$ denotes the neighbors of $\mathbf{p}$ in the MRF, excluding $\mathbf{q}$. After k iterations, a belief vector $\mathbf{b}$ is calculated for each node. For example, for node $\mathbf{q}$:

$$\mathbf{b}_{\mathbf{q}}\left[f_{\mathbf{q}}\right] = E_{data}\left(\mathbf{q}, f_{\mathbf{q}}\right) + \sum_{\mathbf{p} \in \mathcal{N}(\mathbf{q})} \mathbf{m}_{\mathbf{p} \to \mathbf{q}}^{k}[f_{\mathbf{q}}] . \qquad (3.34)$$

To obtain the desired disparity map, the disparity $f_{\mathbf{q}}^{*}$ that minimizes $\mathbf{b}_{\mathbf{q}}\left[f_{\mathbf{q}}\right]$ at each node is selected. The area of influence considered to obtain the disparity value of a pixel in LBP is increased every iteration, see figure 3.13,*right*.

Graph Cuts

In Graph Cut techniques, a directed flow graph is constructed to represent the energy function (3.29). The energy minimization is performed on the obtained graph using techniques from combinatorial
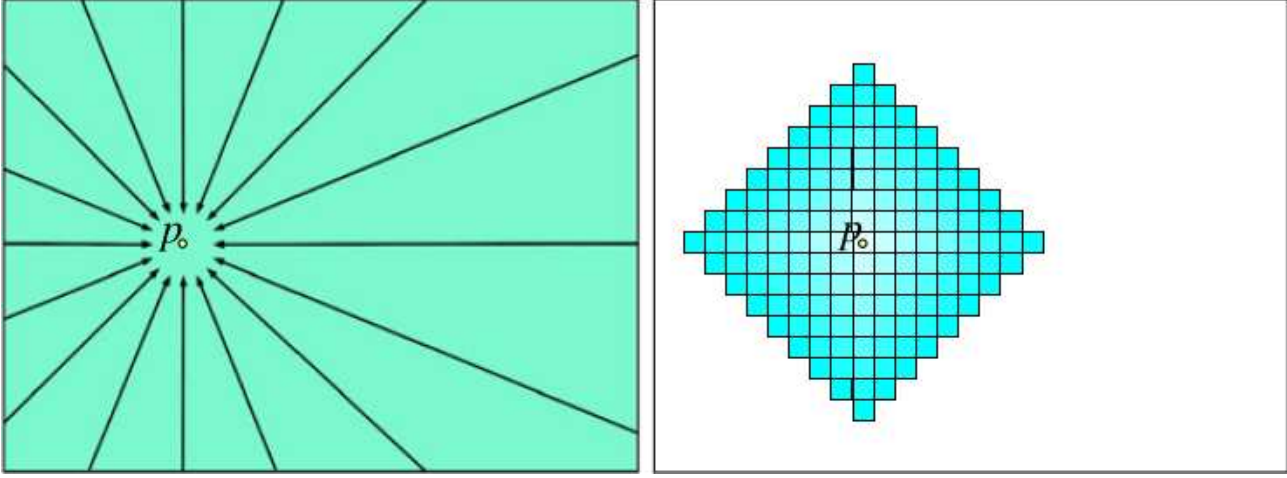
Figure 3.13: Areas of influence in semi-global matchers. *Left:* Area of influence for a pixel p made of 16 1D paths. *Right:* Area of influence created after 8 expansion iterations of adding the $\mathcal{N}_4$ adjacent pixels. *Image from:* [57].

optimization. Here we describe the approach presented in [59], where the authors propose an algorithm based on graph cuts to solve the stereo correspondence problem that handles specifically uniqueness and occlusions. The uniqueness constraint enforces that a pixel in one image should correspond to at most one pixel in the other image. A pixel which is found to have no corresponding pixel in the other image is labeled as occluded. Let $\mathbf{p}$ denote a pixel with coordinates $(\mathbf{p}_x, \mathbf{p}_y)$. Also let $\mathcal{A}$ be the set of pairs of pixels that may potentially correspond. In the case of a rectified stereo pair, $\mathcal{A}$ is defined as:

$$\mathcal{A} = \{\langle \mathbf{p}, \mathbf{q} \rangle | \mathbf{p}_y = \mathbf{q}_y \text{ and } 0 \leq \mathbf{q}_x - \mathbf{p}_x < d_{max}\} \ , \tag{3.35}$$

where $\mathbf{p}$ is a pixel in the left, base image and $\mathbf{q}$ is a pixel in the right, match image. An assignment $a = \langle \mathbf{p}, \mathbf{q} \rangle$ is an element of $\mathcal{A}$, i.e. $a \in \mathcal{A}$. A configuration $\mathcal{F}$ assigns to each assignment a a value $\mathcal{F}_a$, which is 1 if the pixels involved in the assignment are considered as corresponding, and 0 otherwise. The goal is to obtain an optimal configuration. An assignment with a value of 1 is called an *active* assignment.

Also, let $A(\mathcal{F})$ be the set of active assignments according to $\mathcal{F}$. Let $N_{\mathbf{p}}(\mathcal{F})$ be the set of active assignments in $\mathcal{F}$ involving pixel $\mathbf{p}$, this is, $N_{\mathbf{p}}(\mathcal{F}) = \{\langle \mathbf{p}, \mathbf{q} \rangle \in A(\mathcal{F})\}$. A configuration $\mathcal{F}$ is called *unique* only if:

$$\forall \mathbf{p} \in \mathcal{P} \qquad |N_{\mathbf{p}}(\mathcal{F})| \leq 1 \ , \tag{3.36}$$

where $\mathcal{P}$ represents the set of pixels in both the left and right images. Occluded pixels are those for which the expression $|N_{\mathbf{p}}(\mathcal{F})| = 0$ is satisfied.

The concept of an $\alpha - expansion$, which is essential to the approach proposed by the authors, is now defined. The disparity of an assignment $a = \langle \mathbf{p}, \mathbf{q} \rangle$ is defined as $\mathbf{d}(a) = (\mathbf{q}_x - \mathbf{p}_x, \mathbf{q}_y - \mathbf{p}_y)$, which in the case of a stereo rectified pair can be simplified to $d(a) = (\mathbf{q}_x - \mathbf{p}_x)$. Let also $\mathcal{A}^\alpha$ be the set of assignments in $\mathcal{A}$ which have $d = \alpha$. A configuration $\mathcal{F}'$ is said to be within a single $\alpha - expansion$ move of $\mathcal{F}$ if $A(\mathcal{F}')$ is a subset of $A(\mathcal{F}) \cup \mathcal{A}^\alpha$. Thus, in an $\alpha - expansion$, active assignments may become inactive, and inactive assignments whose disparity is α may become active.

The energy of a configuration $\mathcal{F}$ is defined as:

$$E(\mathcal{F}) = E_{data}(\mathcal{F}) + E_{occ}(\mathcal{F}) + E_{smooth}(\mathcal{F}) \ , \tag{3.37}$$

where E_{data} is the penalty obtained from dissimilarities of corresponding pixels, E_{occ} imposes a penalty for labeling a pixel occluded, and E_{smooth} tries to ensure that neighboring pixels have similar disparities. An infinite energy is considered for a non-unique configuration in order to impose the uniqueness constraint. The E_{data} term is defined as follows:

$$E_{data}(\mathcal{F}) = \sum_{a \in A(\mathcal{F})} D(a) \ , \tag{3.38}$$

where $D(a)$ is a pixel dissimilarity measure such as the ones described in section 3.6.1. The occlusion term E_{smooth} applies a penalty C_o for labeling a pixel as occluded and is defined as:

$$E_{occ}(\mathcal{F}) = \sum_{\mathbf{p} \in \mathcal{P}} C_o \ \mathbf{T} \left(|N_{\mathbf{p}}(\mathcal{F})| = 0 \right) \ . \tag{3.39}$$

The smoothness term E_{smooth} is defined as follows:

$$E_{smooth}(\mathcal{F}) = \sum_{\{a1,a2\} \in \mathcal{N}} V_{a1,a2} \ \mathbf{T} \left(\mathcal{F}(a1) \neq \mathcal{F}(a2) \right) \ , \tag{3.40}$$

where $\mathcal{N}$ represents a neighborhood system where $\{a1, a2\}$ have the same disparities. A penalty $V_{a1,a2}$ is imposed when an assignment is active in one configuration, but a neighbor is not. This is, it applies a penalty when two adjacent pixels have been assigned a different disparity. It represents a Potts model applied to assignments.

A strong local minimum of the energy (3.37) is calculated using algorithm 1.

Algorithm 1 Stereo Correspondence by Kolmogorov and Zabih

Require: An initial unique configuration $\mathcal{F}$, $t = 0$
Ensure: A strong local minimum of $E(\mathcal{F})$ is obtained
 1: **while** $t < k$ || CONVERGENCE **do**
 2:    **for all** α (in a random permutation) **do**
 3:       Find a unique configuration $\mathcal{F}'$ within one $\alpha - expansion$ of $\mathcal{F}$
 4:       **if** $E(\mathcal{F}) < E(\mathcal{F}')$ **then**
 5:          $\mathcal{F} = \mathcal{F}'$
 6:       **end if**
 7:       $t = t + 1$
 8:    **end for**
 9: **end while**

In each iteration, the configuration $\mathcal{F}'$ within one $\alpha - expansion$ of $\mathcal{F}$ is obtained using graph cuts over a graph constructed to represent the energy function (3.37). Below we offer more details about graph cuts and the constructed graph.

Let $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ be a weighted graph with two special vertexes s, t called source and sink respectively. A *cut* $\mathcal{C} = \mathcal{V}^s, \mathcal{V}^t$ is a partition of the graph in two sets that satisfy $s \in \mathcal{V}^s$, and $t \in \mathcal{V}^t$. The cost of the cut, denoted $|\mathcal{C}|$, is equal to the sum of the cost of the edges that go from a vertex in $\mathcal{V}^s$ to a vertex in $\mathcal{V}^t$. A minimum cut is a cut with the smallest cost. The minimum cut in a graph can be solved efficiently by computing the maximum flow between the terminals. This is usually accomplished with Ford-Fulkerson style algorithms [36].

For constructing the graph, suppose there is an initial configuration $\mathcal{F}^0$. The active assignments for a new configuration within one $\alpha - expansion$ of $\mathcal{F}^0$ will be a subset of $\tilde{A} = \mathcal{A}^0 \cup \mathcal{A}^\alpha$, where

$\mathcal{A}^0 = \{a \in A(\mathcal{F}^0)|d(a) \neq \alpha\}$ and $\mathcal{A}^\alpha = \{a \in \mathcal{A}|d(a) = \alpha\}$. The configuration $\tilde{\mathcal{F}}$ is the configuration that satisfies $A(\tilde{\mathcal{F}}) = \tilde{A}$. In general, $\tilde{\mathcal{F}}$ is not unique. The constructed graph $\mathcal{G}$ has vertices that correspond to assignments. For every assignment in $\tilde{A}$ there is a vertex. It also has the source s and the sink t. The edges in $\mathcal{G}$ are defined as follows. For every vertex $a \in \tilde{A}$, there are edges (s, a) and (a, t). Also, if $\{a1, a2\} \in \mathcal{N}$, edges $(a1, a2)$ and $(a2, a1)$ are created. Finally if vertices $a1$ and $a2$ share a common pixel $\mathbf{p}$, i.e. $a1 = \langle \mathbf{p}, \mathbf{q} \rangle$ and $a2 = \langle \mathbf{p}, \mathbf{r} \rangle$, edges between every such pairs of vertices are created. To define the weights of the edges first the costs D_{occ} and D_{smooth} are defined as follows:

$$D_{smooth}(a1) = \sum_{\{a1,a2\}\in\mathcal{N}, a2\notin\tilde{A}} V_{a1,a2} \ , \tag{3.41}$$

$$D_{occ}(\langle \mathbf{p}, \mathbf{q} \rangle) = D_{occ}(\mathbf{p}) + D_{occ}(\mathbf{q}) \ , \tag{3.42}$$

where $D_{occ}(\mathbf{p}) = C_o$ if $\tilde{A}$ has only one edge entering $\mathbf{p}$, and 0 otherwise. The weights for the edges are defined as in table 3.1.

edge	weight	for
(s, a)	$D_{occ}(a)$	$a \in \mathcal{A}^0$
(a, t)	$D_{occ}(a)$	$a \in \mathcal{A}^\alpha$
(a, t)	$D(a) + D_{smooth}(a)$	$a \in \mathcal{A}^\alpha$
(s, a)	$D(a)$	$a \in \mathcal{A}^\alpha$
$(a1, a2), (a2, a1)$	$V_{a1,a2}$	$\{a1, a2\} \in \mathcal{N}, a1, a2 \in \tilde{A}$
$(a1, a2)$	∞	$\mathbf{p} \in \mathcal{P}, a1 \in \mathcal{A}^0, a2 \in \mathcal{A}^\alpha$ $a1, a2 \in N_\mathbf{p}(\tilde{\mathcal{F}})$
$(a2, a1)$	C_o	$\mathbf{p} \in \mathcal{P}, a1 \in \mathcal{A}^0, a2 \in \mathcal{A}^\alpha$ $a1, a2 \in N_\mathbf{p}(\tilde{\mathcal{F}})$

Table 3.1: Edge weights in the GC stereo matching approach proposed in [59].

Finally, consider a cut $\mathcal{C} = \mathcal{V}^s, \mathcal{V}^t$ on $\mathcal{G}$. The configuration $\mathcal{F}^\mathcal{C}$ corresponding to $\mathcal{C}$ is defined by:

$$\forall a \in \mathcal{A}^0 \qquad \mathcal{F}^\mathcal{C}_a = \begin{cases} 1 & \text{if } a \in \mathcal{V}^s \\ 0 & \text{if } a \in \mathcal{V}^t \ , \end{cases} \tag{3.43}$$

$$\forall a \in \mathcal{A}^\alpha \qquad \mathcal{F}^\mathcal{C}_a = \begin{cases} 1 & \text{if } a \in \mathcal{V}^t \\ 0 & \text{if } a \in \mathcal{V}^s \ , \end{cases} \tag{3.44}$$

$\mathcal{F}^\mathcal{C}$ is within one $\alpha - expansion$ of $\mathcal{F}^0$. Also, it can be proven that if $\mathcal{C}$ is the minimum cut on $\mathcal{G}$, then $\mathcal{F}^\mathcal{C}$ is the configuration that minimizes the energy $E(\mathcal{F})$ over unique configurations.

3.6.5 Efficient Large-Scale Stereo Matching

In [42] the authors propose a novel approach to the stereo matching problem. Their solution is based on the observation that, while some corresponding points in a stereo pair are very hard to find, for example in texture-less regions, others can be matched easily and with confidence. Example of such points are corners or any scene point that is highly distinctive. These reliable matched points, which the authors call *support points* provide valuable and trust-worthy disparity information that is used

to create a prior on the possible disparities around the support points, allowing to reconstruct a dense disparity map with accuracy and without the need of performing global optimization. The estimation of the disparity for points not robustly matched is performed using probabilistic inference. The prior is obtained by generating a 2D mesh via Delaunay triangulation using the support points as vertices and then generating a piecewise linear function which interpolates the obtained disparities of the support points.

In section 6.2 we discuss the qualitative performance of some of the similarity measures described in section 3.6.1, combined with several of the stereo matchers mentioned in the previous sections, when used to obtain disparity maps from the stereo pairs we have recorded.

Chapter 4

Dataset generation

This chapter describes the hardware and software utilized for generating the dataset. To do so, we have equipped a vehicle with a stereo pair of high-resolution grayscale cameras. We have complemented the image data with acceleration and orientation data from an Xsens MTi IMU, GPS data obtained from a smartphone, and vehicle speed and RPM readings from the car computer.

The hardware components we used and their interconnections are described in section 4.1. In section 4.2, we describe the data acquisition software we developed. Finally, in section 4.3 we describe some post-processing steps performed after the data acquisition.

4.1 Hardware selection and description

To acquire the dataset we have developed an in-vehicle data acquisition platform. To keep the implementing cost low, it was desirable to use as many components already available in CIMAT as possible. In the end, the only purchased components were the cameras, lenses and the car computer interface. In this section we briefly describe the chosen components, and explain the reasons behind our selections.

4.1.1 Computer System

A suitable computer system for our purposes had to satisfy certain requirements and restrictions, which are listed below:

- I/O Availability. The computer system had to provide the following ports to be able to interact with the data acquisition devices:

  - 2 Firewire ports, each of them with an available bandwidth of 400 Megabits per second (Mbps), to acquire image data from the cameras. See section 4.1.4 for more details.
  - Bluetooth capabilities to communicate with the selected car computer interface.
  - 1 digital output, with voltage levels between 0–24 V, to provide triggering pulses to the cameras.
  - 1 USB port to communicate with the smartphone to retrieve GPS data.
  - 1 USB or RS-232 port to read the output of the Xsens MTi.

- Display. The capability to visualize the current status of the system was very important. This would help in camera adjusting, software debugging and monitoring the performance of the system.

- Low energy consumption. Modifying the electric system of the car to provide more electric power, for example installing a second alternator and battery or adding a super capacitor, was not an option, so we had to work with the stock electric system of the car. This limited the available power to supply the data acquisition system to 300 W.

- Real-Time capabilities. We wanted to acquire the images with a very precise frame rate, thus, a real-time system was necessary to provide the camera triggering pulses with the desired accuracy.

- Physical constraints. The dimensions of the selected system should fit the free interior space of the car. Also, it should be possible to mount it safely with no modification to the car.

- Processing and storage capabilities. The selected computer system should be able to acquire and save all the data streams with no loss of data, and provide a responsive user interface.

We had several options for choosing a computer system that would satisfy all the aforementioned requirements. Among them was a desktop PC, a National Instruments (NI) EVS-1464RT embedded vision system, a NI sbRIO-9632 embedded real-time controller and a Dell Precision M4600 mobile workstation.

The desktop PC had lots of non-volatile memory and good processing power. Also it was the most customizable of all the options; with several USB ports and expansion card slots available, it would have been relatively easy to meet all the I/O requirements. However, it would need an external monitor, and the electric supply of both devices would exceed the available power. This, combined with the difficulties of mounting them in the car made them unsuitable. The EVS-1464RT is an embedded controller targeted to computer vision applications that combines a floating-point processor and an FPGA. It offers several digital I/O lines and two Firewire-B ports, with no expansion options, and the real-time (RT) capability for generating the triggering pulses using the FPGA and an integrated high-precision clock. However, it lacked Bluetooth capabilities, an integrated display and enough non-volatile memory for storing the image streams. The sbRIO-9632 is similar in concept to the EVS-1464RT, but it is not designed as an off-the-shelf vision system. It is more a general purpose embedded system suitable for prototyping. It lacks a protective enclosure and has no Firewire ports. It also shares the same weaknesses with the EVS-1464RT, so it could not handle all the requirements by its own. Finally, the mobile workstation, as the PC, provided good processing and storage capabilities and a good set of I/O expansion possibilities, plus an integrated display. However, it lacked real-time capabilities and digital I/O.

So, no device could handle all the requirements by itself, but a combination of two of them would do. We decided that the best option was to use a combination of the mobile workstation and the sbRIO-9632 embedded real-time controller, and split the workload between both devices. A combination of the mobile workstation with the EVS-1464RT would have been also possible, but the sbRIO-9632 is smaller and requires less electric power. The mobile workstation was responsible for running the user interface, acquiring and saving data streams from the cameras, GPS, and car ECU data while the sbRIO-9632 triggered the cameras in a timely fashion and communicated with the XSens MTi IMU through a RS-232 link.

4.1.2 Stereo camera

Commercial stereo cameras feature small baselines, which do not allow to obtain a good depth estimate for objects at long range. For that reason, we decided that the best option was to build our own stereo camera from two separate cameras. For the camera selection, several requirements had to be satisfied in order to acquire useful stereo images from a moving vehicle. The requirements were the following:

- A spatial resolution greater or equal than 1 Megapixel.

- A pixel depth of at least 10 bits.

- A frame rate of at least 20 fps.

- Perfect synchronization between both cameras.

At the time when the cameras were selected, 3 digital camera buses were available, Camera Link, Firewire and Gigabit Ethernet (GigE). Camera Link is targeted to high-throughput applications, using high-resolution cameras at even hundreds of frames per second, and the equipment is very expensive. Since our requirements were more modest, Camera Link was discarded for this application. The nominal bandwidths of Firewire (400 Mbps for FirewireA and 800 Mbps for FirewireB) and GigE (1 Gbps) were both acceptable. Since we had worked a bit previously with Firewire cameras, and had already available a couple of Firewire framegrabbers we decided to choose Firewire over GigE.

A word should be given regarding the synchronization requirement. Since a very precise synchronization was required, software synchronization was not acceptable, specially working with a general-purpose operating system, where it is difficult to get a time resolution lower than 1 ms. Using a real-time kernel may permit a better time resolution but it still may be difficult to obtain the desired results. So, the best option was hardware-based triggering, using cameras with an external trigger line. A single digital signal triggers both cameras, allowing excellent synchronization. This is the option we have chosen.

Given the mentioned requirements, a market research was done to find Firewire cameras available in Mexico. We looked for distributors from leading digital camera manufacturers such as Basler, Point Grey Research, Unibrain and Allied Vision Technologies (AVT). We did not obtain a response from Unibrain; Point Grey does not have a distributor in Mexico but they can ship to Mexico from Canada. Since we wanted to avoid customs issues, we sticked with brands with local distributors. AVT and Basler do have distributors in Mexico. Table 4.1 summarizes the features of cameras from AVT and Basler that have a resolution close to the required one and that meet all the other requirements. The technical data was obtained from the datasheets of the cameras and the prices come from quotes obtained directly from the distributors.

Maker	Series	Model	Resolution	Sensor	Pixel Depth	FPS	Price (MX$)
Basler	Scout	scA1300-32fm	1296 x 966	ICX445	12 bit	33	$12,735.00
Basler	Scout	scA1300-32fc	1296 x 966	ICX445	12 bit	33	$13,445.00
Basler	Scout	scA1000-30fm	1034 x 779	ICX204	12 bit	30	$16,420.00
Basler	Scout	scA1000-30fc	1034 x 779	ICX204	12 bit	30	$16,845.00
AVT	Stingray	F-125B	1292 x 964	ICX445	14 bit	30	$18,280.00
AVT	Stingray	F-125C	1292 x 964	ICX445	14 bit	30	$18,280.00

Table 4.1: Selected requirements-compliant cameras available in Mexico

Figure 4.1: Stereo setup tests. *Left:* Cameras mounted using off-the-shelf camera supports. *Center:* Cameras mounted in the interior of the vehicle. *Right:* Cameras mounted using custom-made camera supports with a degree of freedom.

All cameras listed in Table 1 are Firewire, use a CCD with progressive scan and global shutter as a sensor and have a sensor size of $\frac{1}{3}$ inch. From the models listed, we chose the scA-1300-32fm model from Basler [9]. This model is chosen over the scA1000-30fm because the camera is cheaper and also because we were told by a technical representative of Graftek Inc., a Basler distributor in the U.S., that the ICX445 sensor it holds is newer and provides better image quality than the ICX204 sensor from the scA1000-30fm. This is confirmed in [68], where the ICX445 shows better performance in categories such as quantum efficiency and temporal dark noise. We selected the Basler camera over the AVT Stingray F-125 because of the cost advantage (they share the same CCD) and because we have used Basler cameras in the past and they have proven to be very reliable.

Before arriving to the final mechanical setup for the stereo camera, we tried several configurations. First, we tried to mount the pair of cameras in a bar which originally pertained to a roof rack. We attached the cameras to the bar using off-the-shelf camera supports, see figure 4.1, *left*, but the supports were simply not good enough in keeping the cameras static. We also tried mounting the cameras inside the car, using supports fitted with suction cups, as can be seen in figure 4.1, *center*, but the alignment of the cameras was difficult and some of the support joints would change slightly their angles while driving. Then we tried custom-made camera supports to attach the cameras to the bar, see figure 4.1, *right*. In a first stage, the custom-made supports allowed to modify the cameras angles with respect to the road surface, but the fixation was not stiff enough and the angle would present tiny variations after driving in non-smooth surfaces, which would void the calibration of the stereo pair. For this reason, the supports were modified and the angle was fixed, as pictured in figure 4.2, *bottom*. The orientation of the cameras was chosen so that the optical axis would be parallel to the ground surface when the bar was placed on the floor. When mounted on the car, the cameras were slightly tilted towards the road surface because of the mounting position of the bar on the car's roof, and the shape of the roof itself. To reduce the effects of vibration on the cameras, we have placed a small neoprene sheet under them.

The baseline was selected to be as wide as possible to provide a better depth resolution, without sacrificing too much the common-viewing area of the two cameras. Initially we attempted to use a baseline of 60 cm. but the common-viewing area between the left and right camera was small, resulting in a depth map with limited coverage. The reduction of the baseline to 50 cm. helped to obtain depth maps with increased usability. The camera setup is pictured in figure 4.2.

4.1.3   Lenses

Once the cameras had been chosen, we searched for suitable and compatible lenses. The selected cameras are designed to work with C-Mount lenses, and feature an image sensor with a size of $\frac{1}{3}$ inch.
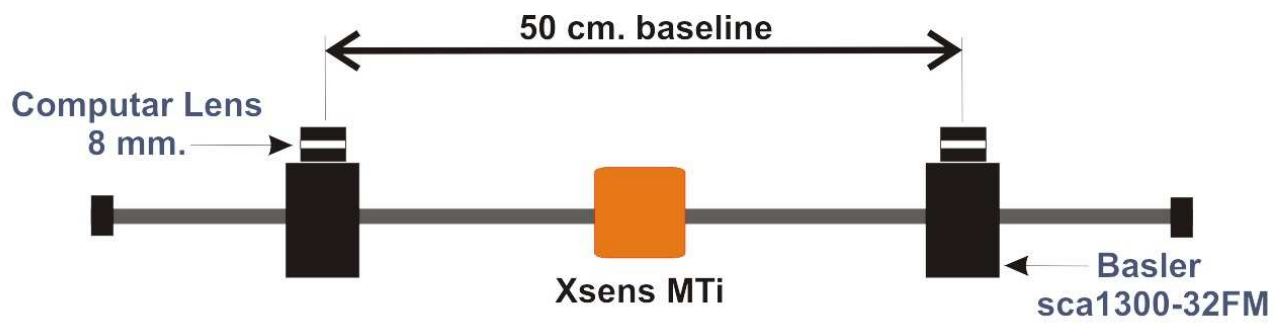
Figure 4.2: *Top:* A sketch of the sensor setup. *Bottom:*. The sensor setup fixed to the vehicle. The two cameras and the Xsens MTi IMU are mounted on an bar which is then rigidly attached to the car.

Therefore, we needed C-Mount lenses with a format ideally of $\frac{1}{3}$ inch or higher. A smaller format would produce an image that would not fill the entire image sensor, and that is undesirable. Other desired features for the lenses were low distortion and being appropriate for mega-pixel cameras.

Regarding the field of view (FOV), we did some research on the FOV used in similar applications. We found typical values range between 45° and 100°, e.g. [10, 41, 47, 109]. A FOV within that range would have been desirable. Given an image sensor, the focal length f and the field of view α are related by the following expression:

$$\alpha = 2 \arctan \frac{d}{2f} \ , \tag{4.1}$$

where d is the dimension of the image sensor in the direction of interest. With the knowledge of the dimensions of the image sensor we can resolve for the focal length given a desired FOV angle. From (4.1) we obtain:

$$f = \frac{d}{2 \tan(\frac{\alpha}{2})} \ . \tag{4.2}$$

The dimensions of an $\frac{1}{3}$ inch image sensor like the one mounted on the selected cameras are 4.8×3.6 mm in the horizontal and vertical directions, respectively. Thus, for a desired horizontal FOV of 45°, the required focal length we obtain is 5.79 mm. The closest focal length that is commercially available is 6 mm. The ideal lens would have been then a high-quality C-Mount lens, with a format of $\frac{1}{3}$ inch or higher and a focal length of 6 mm. Unfortunately, no distributor in Mexico had available high-quality lenses with the required characteristics. The closest model we found was the M0814-MP2 from Computar. It has a focal length of 8 mm, a $\frac{2}{3}$ inch format and is C-Mount. It is designed to be a low distortion lens for mega-pixel applications. The horizontal FOV we obtain with the lens is $\sim 34°$, which is smaller than the desired FOV, but it still yields useful images.

4.1.4   Framegrabbers

A framegrabber is a device that allows to capture the image data streams coming from computer vision cameras. In the case of Firewire framegrabbers, there are two variants. Firewire-A allows data transfers at a speed up to 50 MBps (MegaBytes per second, equivalent to 400 Mbps), while Firewire-B offers a nominal data rate of 100 MBps (800 Mbps). To select the quantity and type of framegrabbers needed, we calculated the required data rate for each camera using the following equation:

Required data rate = Number of pixels $\times$ Bytes per pixel $\times$ Frames per second .

For a camera with a resolution of 1 Mpx, using 12 bits per pixel (1.5 Bytes) and a framerate of 20 fps the calculation yields a required data rate of 30 MBps, which grows to 40 MBps if the camera drivers do not allow to split a Byte in two sections to store information corresponding to different pixels. In either case, a Firewire-A adapter per camera or a single Firewire-B adapter for the two cameras would be enough. We opted to use two Firewire-A adapters, since we already had available one Firewire-A ExpressCard adapter. The second adapter was obtained from the integrated Firewire bus of the mobile workstation.

4.1.5   Inertial Measurement Unit

For obtaining 3D orientation and acceleration data, we had available an XSens MTi [104] miniature inertial measurement unit. It integrates 3D accelerometers and gyroscopes to perform the measurement

estimates. It also features a 3D magnetometer to read Earth magnetic field data. The integrated low-power signal processor allows to obtain the data in real-time. The use of only gyros to calculate the orientation estimate results in the slow, but steady, accumulation of error. For this reason, the MTi uses a sensor fusion algorithm based on a Kalman Filter to obtain a high-accuracy estimate of the orientation, using both the gravity vector and the Earth magnetic north as reference vectors to compensate for drift. This capability makes the MTi an Attitude and Heading Reference System (AHRS). The most important specifications of the sensor are listed in table 4.2. To communicate with the MTi we used a RS-232 link. The steps we followed to establish the communication are described in section 4.2.2.

Angular Rate	
Category	Units: $°/s$
Range (Roll, Pitch and Yaw):	±300
Bias (Roll, Pitch and Yaw):	±0.5 (Roll, Pitch), ±1.0 (Yaw)
Resolution:	0.05°
Acceleration	
Category	Units: m/s^2
Range (X,Y,Z):	±50
Bias (X,Y,Z):	±0.2
Resolution:	0.01
Communication and Performance	
Update rate:	Max. 256 Hz for full data
Communication interface:	RS-232/422/485 & USB (with external converter)

Table 4.2: Xsens MTi specifications [104]

The MTi was mounted on the same bar where the cameras were placed. A custom-made base was needed to attach it to the bar. The MTi is approximately placed in the middle point between the cameras, as can be seen in figure 4.2.

4.1.6 Car Computer Interface

Modern cars are equipped with one or more networked computers that control several components of the car, like the engine, brakes, transmission, among others. To interact with the computer network in the car, first we had to determine both the communication protocol and the type of port used to access the computer network. After performing research on the topic, we found out that it all depends on the manufacturing year, the type of fuel used and the target market of the vehicle. The vehicle we used was manufactured in Spain in 2007 under European specifications and uses gasoline as fuel. The applicable regulations of the European Union regarding car data access (European On-Board Diagnostics, EOBD) made mandatory for this kind of vehicle to provide a SAE-J1962 female port to communicate with the computers of the car.

Regarding the communication protocol, according to EOBD, the manufacturer is allowed to choose among 5 protocols: SAE-J1850-PWM, SAE-J1850-VPW, ISO-9141-2, ISO-14230 (KWP2000) and ISO-15765 (CAN). We could not find reliable information about the communication protocol used in the car, so we had to determine it by ourselves. Each protocol uses different pins on the connector, so, as a first approach to detect the used protocol, we performed a visual inspection on the car connector to find the existing pins. This analysis was non-conclusive because the present pins would fit more than one

protocol. Then, we tried to establish communication with the car using a NI USB-8473 CAN interface we had available, but the attempt was unsuccessful, so the CAN protocol was discarded. Finally, we acquired a Bluetooth OBD-II (OBD-II, On-Board Diagnostics-II, the American equivalent of EOBD) communication interface based on the ELM-327 chip [31]. This chip is a car communications interface with support of all the OBD-II protocols. It also has the capability to auto-detect the protocol used in the vehicle. Using the interface, we discovered that the protocol used was ISO-9141-2. Details on how to establish communication with the car computer network using the ELM-327-based interface can be found in section 4.2.3.

4.1.7   GPS

Although initially the acquisition of GPS data was not considered, we decided in the end to incorporate it to enrich the dataset. Our first attempt to obtain GPS data was using a high-quality GPS unit intended for in-car use as a navigation assistant. Unfortunately, since the unit was not designed to be a GPS data source, it did not offer any option to transmit the position data. We then attempted successfully to obtain GPS data from an Android-based smartphone using an USB link. The measurements it provides are not the most accurate, but they give a good estimate of the vehicle's position. The process we followed to obtain the data from the smartphone is detailed in section 4.2.4.

4.1.8   Power Supply

The electric system of the car we used makes available to the user about 300 W at 12 VDC. However, several of the devices, mainly the mobile workstation, needed AC voltage to work. For this reason, we used a 300 W power inverter to convert the 12 VDC supply of the car to the 110 VAC supply required by the power adapters of the different devices.

4.1.9   Hardware System Architecture

The hardware interconnections and wiring of the created data acquisition system are shown in figure 4.3. The cameras are equipped with Firewire-B ports, so, in order to connect the cameras to the framegrabbers, we used Firewire-B-to-A cables. Besides, the cameras are designed to be powered through the Firewire cables, thus the framegrabbers must supply the electric power. Natively, none of the selected adapters is powered, but the ExpressCard adapter offers the option to be supplied with 12 V coming from an AC adapter. To provide power to the camera connected to the integrated Firewire bus, an intermediate power supply model PS-1FW from Kramer Electronics was necessary to inject the required energy.

To communicate with the Xsens MTi, we manufactured a custom cable. The Xsens came from factory with a USB communication cable, but it did not fit our needs because the acquisition of inertial and orientation data needed to be performed by the real-time controller, which does not feature a USB port, but does have an RS-232 port, so we built an appropriate cable. For assembling the cable, we also took into account the fact that the Xsens requires a supply of 4.5 – 30 VDC through the communication cable. The custom cable we made featured in one end the Xsens port and in the other end two ports: a RS-232 port for data communication with the real-time controller and a USB port used to obtain the required voltage from a USB port in the mobile workstation.

The real-time (RT) controller was also responsible for generating the triggering pulses to the ca-
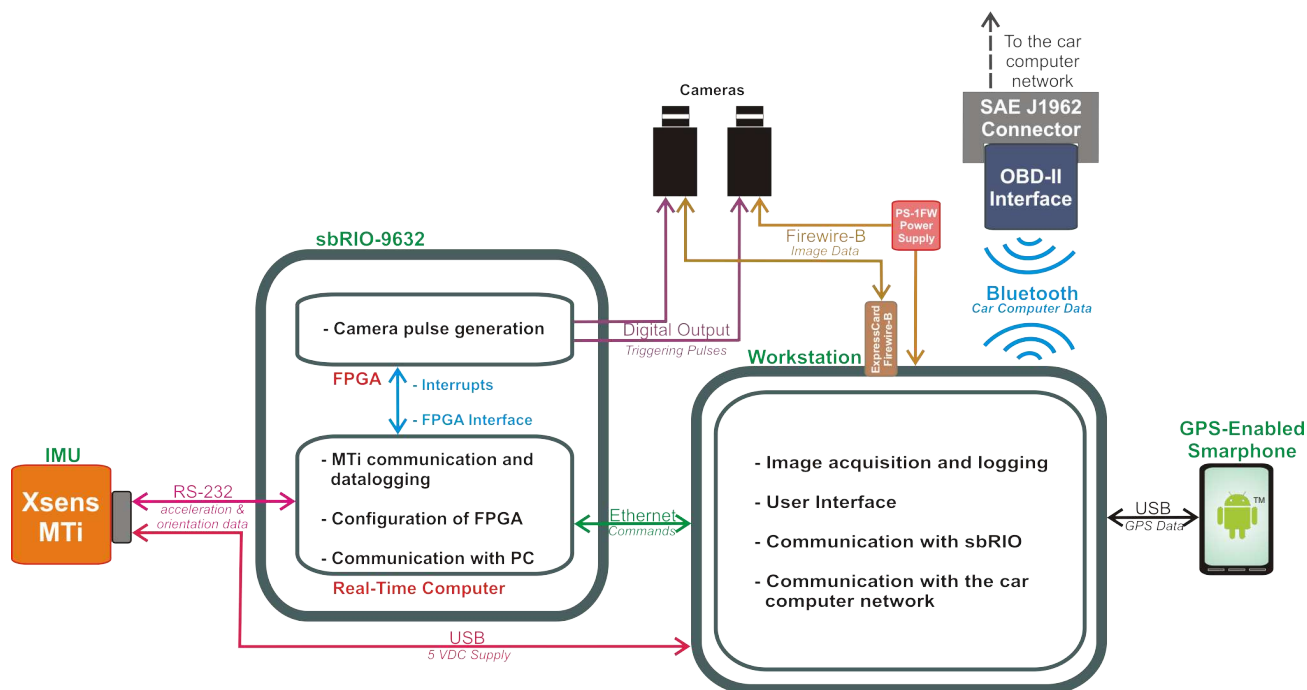
Figure 4.3: The figure shows the interconnections of all the hardware used for the dataset acquisition.

meras. One of the digital output lines was used to send the pulses to both cameras. The connection between the RT controller and the triggering lines of the cameras was made using the recommended I/O cables provided by the camera manufacturer. The communication between the RT controller and the mobile workstation was performed using a standard Ethernet cable. The smartphone was connected to the mobile workstation using the USB cable that came with the phone. Finally, the communication between the ODB-II interface and the mobile workstation was performed using a wireless Bluetooth link, emulating a serial port.

4.2   Data Acquisition Software

The data-acquisition platform we developed, illustrated in figure 4.3, is a distributed computing system where the two processing nodes, the mobile workstation and the real-time controller, are each responsible of certain tasks.

The mobile workstation is responsible of providing the user interface, acquiring and saving the image streams from the cameras, communicating with the car computer and sending commands to the sbRIO-9632 using NI's Network Shared Variables API. All the software for the workstation was developed in Visual C++ and runs on Windows. We had initially planned to develop the software on a Linux platform, but were forced to switch to Windows because of the need to use Basler's Firewire Pylon API, only available for Windows, see section 4.2.1 for details. The software running on the workstation is a multi-threaded application that performs in parallel the data acquisition from several sources. We developed a hierarchy of specialized thread classes based on Vic Hargrave's `Thread` class [48], which provides a nice object oriented wrapper around the `pthreads` API.

On the other hand, the real-time controller is responsible for performing acquisition and logging of the incoming MTi data. The real-time controller has also the important task of triggering the
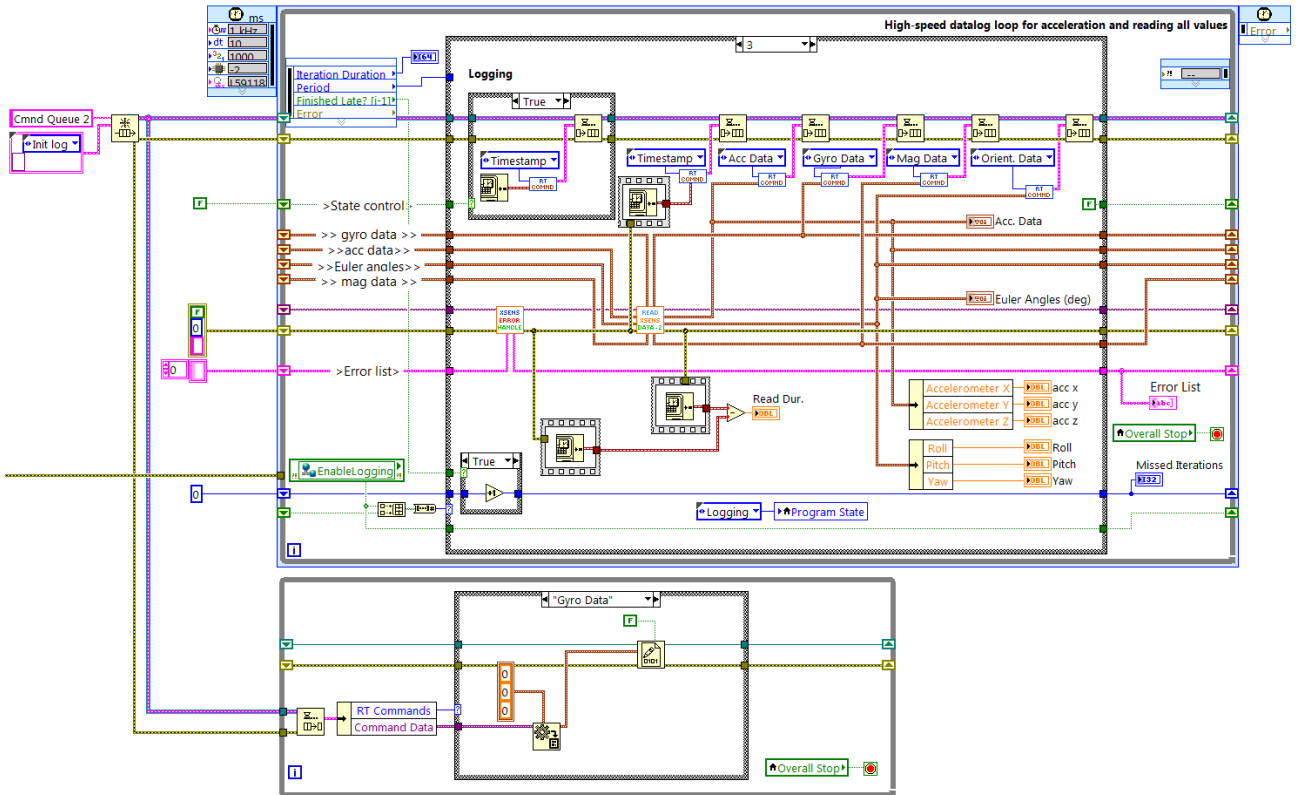
Figure 4.4: A section of the real-time controller block diagram implementing a producer-consumer architecture.

cameras. For the triggering to be very precise time-wise, it is performed using a digital output line linked to the FPGA of the sbRIO-9632, which works with a high-accuracy clock of 40 MHz. The software running in the real-time controller was programmed using LabVIEW, with the Real-Time and FPGA modules. The software running on the floating-point processor of the real-time controller is designed under a producer-consumer architecture. A screenshot showing a section of the block diagram (source code) of the RT program is shown in figure 4.4. Communication from the software on the floating-point processor to the FPGA is performed using the front-panel FPGA interface. It is used to send configuration data, such as the frequency of the camera-triggering pulses. The notification from the FPGA back to the floating point software that a pulse has been generated, is sent via one of the interrupt lines.

The clocks of the two computing nodes are synchronized by using the Simple Network Time Protocol (SNTP). The workstation acts as a server and the real-time controller is the client. To enable SNTP synchronization on the sbRIO-9632, the "TIME SYNC" section of the ni-rt.ini configuration file (located in the root of the controller's file system) should look similar to the following:

```
[TIME SYNC]
source.rtc.enable=true
source.sntp.enable=true
source.priority=sntp;rtc;
source.sntp.address=10.0.0.100
source.sntp.interval=60
source.sntp.port=123
source.sntp.verbose=false
source.sntp.log=true
```

Also, the server must implement `SNTP` version 4. The time server was mounted on the mobile workstation using Meinberg's NTP Time Server version 4.2.6 [67]. A firewall exception had to be added so that network traffic using the User Datagram Protocol (UDP) was allowed through port 123.

4.2.1 Image Acquisition

As explained in section 4.1.4, we thought the Firewire-A nominal data rate of 50 MBps would be enough for our purposes. However, we discovered through testing that the real, effective data rate was not 50 MBps, but only $\sim$ 31MBps. This restriction forced us to use very efficiently the available bandwidth in order to achieve the desired framerate. Initially, we had planned to use a standard IIDC-compliant (The IIDC is the 1394 Trade Association Instrumentation and Industrial Control Working Group, Digital Camera Sub Working Group) library such as `libdc1394`, or the more friendly `camwire`, a higher-level library built around `libdc1394`. However, these libraries only support the acquisition of images using the video modes specified by the IIDC, and none of them considers the possibility of one byte to carry information from more than one pixel. Thus, the storage of a pixel with 12 bits of depth would take 2 bytes, wasting the 4 upper bits of the 16-bit word. This represents an unnecessary use of valuable bandwidth that we could not afford. Therefore, the image acquisition is performed using the C++ Pylon API [8] from the camera manufacturer, which allows an efficient use of the available Firewire bandwidth. Just the 12 effective bits/pixel are transmitted, and 3 bytes can store the information from two pixels. This type of pixel storage is called `MONO12_PACKED`. For Firewire cameras, the Pylon API is only available for the Windows platform, thus we had to develop the data acquisition software under Windows.

The used mobile workstation is equipped with a standard 5,400 RPM platter-based hard-disk drive. Thus, the disk is not very fast, and using the highest possible frame rate, the data rate of the image streams easily surpasses the saving capabilities of the disk. To deal with this issue, and avoid data loss, we have implemented a multi-threaded image logger. For each camera, one image acquisition thread retrieves the image data from the camera and quickly buffers it in RAM memory, on a thread-safe queue based on the one proposed in [49]. A corresponding image logging thread handles the more time-consuming tasks, allowing the image acquisition process to work on an timely fashion. This strategy, represented in figure 4.5, takes advantage of the high-speed of volatile memory. The length of the sequences was limited by the available RAM. The maximum possible sequence length with the 8 GB of total memory available was around 3,000 stereo pairs. The use of this strategy implied that after recording an image sequence, we had to wait several minutes while the images in memory were saved to disk, before we could start to record another sequence.

Prior to the image acquisition loop, the camera object representing the physical camera is opened and configured. The triggering is set up so that the camera starts the exposition when a rising edge is detected in the physical input line 1. The Pylon API provides an internal image buffer to save the
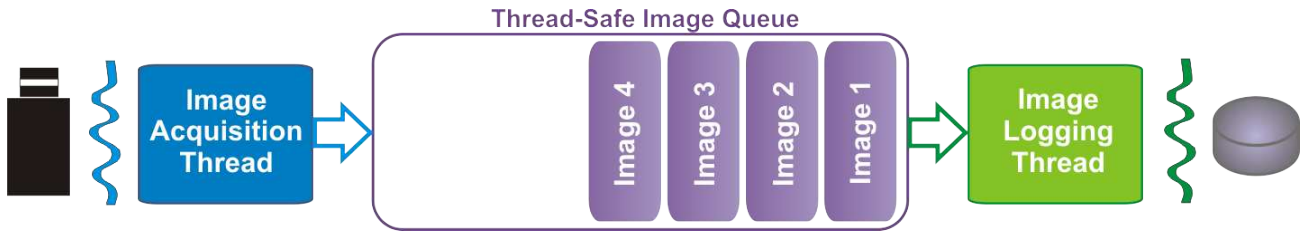
Figure 4.5: Multi-thread image acquisition.

images from the camera and avoid data loss in case the image acquisition thread fails to retrieve the image data on time. We have configured this buffer to hold up to 40 images. Because of the bandwidth restrictions, we were unable to use the full CCD of the camera, so we defined a region of interest (ROI) as big as possible to keep the maximum framerate at 20 fps. The selected ROI has dimensions of 1,096 × 822 and is centered in the CCD. We then configure the camera object to use the `MONO12_PACKED` data format. Finally, the image is sent through the Firewire bus in packets, and we have set the packet size to be 4,096 bytes. The packet size determines how much of the available bandwidth the camera uses. The selection of 4,096, the maximum possible value, ensures that the camera uses all available bandwidth. Since in our configuration each camera has its own framegrabber, this represents no problem. However, if two cameras would share the same Firewire bus, the packet size should be adjusted so that each of them has a portion of the available bandwidth.

When the user has started the logging process, we continuously retrieve the front element from the image buffer using the `RetrieveResult` method from the camera object and place it in the image queue. The obtained data is converted from the `MONO12_PACKED` format to a standard `MONO16` format used in most image processing libraries, which uses two bytes per pixel. The encoding of the image data to `PNG` format was initially performed by the image acquisition thread, but it proved to be a costly operation that would slow down the data acquisition, resulting in reduced framerates or data loss. For this reason, we switched the encoding task to the image logging thread. This thread takes the image data out of the queue, encodes it in `PNG` loss-less format and finally writes it down to disk. For encoding the images we use the `OpenCV`'s `imencode` function and for saving them we use the standard C++ streams API.

Before each recording day, we carefully focused each camera and ensured that the iris aperture of both lenses was approximately the same. These adjustments are important to obtain an image pair well suited for finding stereo correspondences. Although both lenses were of the same model, selecting an aperture taking as reference only the scale printed on the lenses would yield a slightly different aperture in the lenses diaphragm. We compensated for this difference manually. For recording sequences at night, we had to open totally the iris aperture to maximize the amount of light entering the image sensor, which at night is very limited. The big aperture caused a slight loss of focus, which is perceived in the acquired images. Also, we increased the exposure time to gather as much light as possible. The maximum exposure time we used was 20 ms. This value was determined experimentally as the highest exposure value that would not cause significant motion blur while traveling at speeds up to 40 km/h.

4.2.2   IMU Data Acquisition

The communication with the MTi was performed using the Xsens LabVIEW instrument driver [103] over an RS-232 link. The instrument driver provides a set of virtual instruments (VIs, program functions in the LabVIEW context) that encapsulate the Xsens communication protocol and release the user from having to program all the communication functions. In a first stage we acquired only
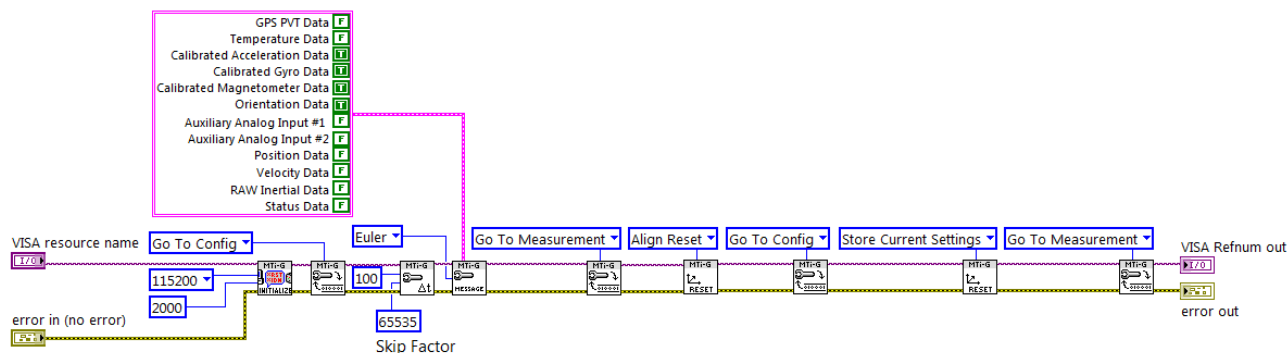
Figure 4.6: MTi IMU initialization sequence.

acceleration and orientation data at 50 Hz using a connection speed of 57,600 bauds per second (bps). After some improvements on the software and the wiring, we increased the connection speed to 115,200 bps, the acquisition rate was upgraded to 100 Hz and also started to acquire data from the 3D gyro and the 3D magnetometer.

The Xsens MTi has two modes of operation: *measurement* and *configuration*. In measurement mode, lectures from the integrated sensors can be retrieved but the sensor configuration cannot be modified. The configuration mode exhibits the opposite behavior. We developed the software for IMU data acquisition taking this into account. The acquisition process can be divided in three stages: Initialization, data acquisition and closure. The steps performed in the initialization stage are enlisted below:

1. Open the connection to the MTi using the `Initialize` VI.

2. Activate configuration mode, using the `Change State` VI. Instruct the MTi to transmit data on request by calling the `Configure Sampling Rate` VI.

3. Indicate the MTi, with a call to the `Configure Message` VI, which of the available data fields it should send. We select acceleration, gyro, magnetometer and orientation data. The orientation data is transmitted using Euler angles.

4. Switch to measurement mode. Perform an *align reset* by calling the `Reset Orientation` VI. An align reset changes the sensor coordinate system so that the Z axis is along the local gravity vector, pointing upwards, and the X axis is parallel to the horizontal plane, along the direction pointed by the factory sensor coordinate system, which is marked in the sensor enclosure. The Y axis is determined so that a right-hand coordinate frame results.

5. Go back to *Configuration* mode. Save the coordinate system defined in the previous step.

6. Activate measurement mode. The MTi is ready to acquire data.

All the mentioned VIs in the initialization sequence are contained in the LabVIEW library in which the instrument driver is provided. A LabVIEW library is conceptually equivalent to a Java package or a C++ namespace. The code used to perform the MTi initialization is shown in figure 4.6. After a successful initialization, the data acquisition begins. To retrieve the latest data from the sensor, we send a data request using the `Read Single Data` VI. After the request, we read acceleration, gyro, magnetic field and orientation data, in that precise order. The order is defined by the Xsens
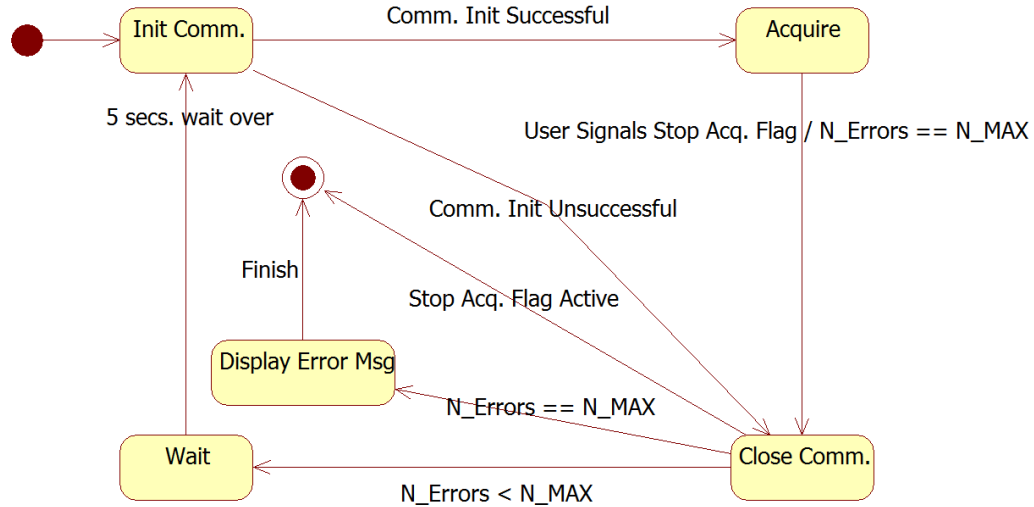
Figure 4.7: State machine of the communication with the car computer.

communication protocol. Finally, upon user request, the communication is closed. All the acquired data is saved in the non-volatile memory available in the sbRIO-9632. After the data acquisition is finished, we retrieve the data by using an FTP connection.

While developing the necessary routines for communicating with the MTi, we found a bug in one of the VIs of the instrument driver. The VI that performs the align reset did not append the checksum to the package sent to the sensor. As a result, the sensor coordinate frame would not change, and some errors in the communication would occur. We corrected the bug and reported it to the driver supplier.

4.2.3   Car Computer Data Acquisition

To communicate with the OBD-II interface, we first set up a virtual COM port associated with the interface. The communication is then performed by opening the port, sending commands and requests, and reading the device response. In initial tests, we detected some erratic behavior in the communication, specially in the initialization steps preceding the talk to the car computer network. Unfortunately, we found out that the most likely explanation to this was that we got an interface equipped with a non-genuine ELM-327 chip, i.e. a clone. The device claims to be version 1.5, but the manufacturer of the ELM-327 chip never produced such a version. The clones of the chip are known to present some communication issues. This made it difficult to obtain a reliable initialization sequence, but after much trial and error, we managed to create a procedure that would reliably establish the connection. The whole communication process modeled as a state machine is represented in figure 4.7.

The connection to the virtual port is performed using the following settings: 2,400 bps, 8 data bits, no parity, 1 stop bit and no flow control enabled. Once the port has been opened successfully, several AT commands are sent to the OBD-II interface, and their response is read. The AT commands, where AT is the abbreviation of ATTENTION, are text strings beginning with "AT" that modify the configuration of the interface. The AT commands where originally designed to send instructions to modems but their use has been extended to other devices. The initialization sequence is described in the following code block:

```
sendCommandReadResponse("ATWS");
sendCommandReadResponse("ATSS");
sendCommandReadResponse("ATSPA3");
sendCommandReadResponse("ATSI");
```

The command `ATWS` (Warm Start) requests the interface to perform a complete reset, except for the power on LED test. Then, sending `ATSS` (Standard Search) commands the interface to perform the protocol search in the order indicated by SAE J1978. Next, the command `ATSP3` (Search Protocol) starts the protocol discovery phase, using protocol 3 as the first protocol to test. Protocol 3 is precisely ISO-9141-2, the protocol used in the data acquisition vehicle. Finally, the command ATSI (Slow Initiation) instructs the interface to perform the initialization procedure to communicate with the car computer network. Analyzing the response obtained from this last command we can detect if the initialization sequence was successful or not. If successful, then we proceed to the data acquisition, or "Acquire" state.

The request of data to the car computer network is performed by issuing OBD commands, which are strings formed by hexadecimal characters. A typical OBD command has the form "010C". The first two characters represent the data mode. Mode 01 represents the "Show Current Data Mode". As its name implies, it allows to obtain the current value of the requested data field. The second pair of characters, "0C", are together called PID, from Parameter IDentification. The standard PIDs are specified in norms such as SAE J1979 or ISO 15031-5. Car manufacturers also use their own PIDs for data fields non contemplated by the standards, which are focused in data relevant to car maintenance and emissions control. Despite our best efforts, we could not find documentation about non standard PIDs for the utilized car, so we had to restrain the data gathering to standard PIDs. After examination of the list of standard PIDs, we found two that were relevant to our application: "0C" and "0D" which correspond respectively to the revolutions per minute (RPM) of the engine, and the current speed of the vehicle. The data request sequence is shown in the code block below.

```
sendCommandReadResponse("010D1");
sendCommandReadResponse("010C1");
```

First, we request the speed and then the RPM. Notice we have appended an extra '1' to the OBD commands. This indicates to the OBD interface that we expect the response to have a length of only 1 line, so after it receives 1 line of data from the car, it forwards it to us immediately. If the '1' is omitted, the interface waits 200 ms for more data. Thus, the inclusion of the extra '1' allows to increase the communication speed. On average, we are able to obtain 3 updates per second for each of the two data fields. The responses to the speed and RPM requests have the format 410CXX and 410DXXXX respectively, where X represents a hexadecimal digit. The speed is returned as an integer value encoded in the two hexadecimal digits. Similarly, the RPM data is encoded in the last four hex digits of the response. To obtain the real RPM value, the hex value should be converted to decimal and then divided by four.

The implementation of the low-level communication was done using the C++ serial communication library proposed in [27], as it offers a clearer interface than the standard Windows API. The communication logic runs on its own specialized thread.

Figure 4.8: A successful call to the `abd` program.

4.2.4   GPS Data Acquisition

To obtain GPS data from the Android-based smartphone, we relied on the ShareGPS application [79]. The application retrieves the GPS data from the phone and publishes it using the Transport Control Protocol (TCP) on port 50,000 of the phone. Then, using the USB line, we forward the data from the TCP port in the smartphone to a TCP port in the workstation. To perform the data forwarding, it was necessary to enable USB Debugging on the Android phone. Also, a copy of the Android SDK was obtained and installed in the mobile workstation, so that the program `adb` (Android Debug Bridge) was available. After connecting the smartphone to the workstation for the first time, the smartphone driver should be installed automatically in the workstation. If not, one has to manually look for the driver and install it. After all these steps have been performed successfully, the data forwarding is performed as follows:

1. Run the ShareGPS application in the phone. Make sure that the "Use USB" option is checked on the ShareGPS application settings.

2. Run the `abd` program. In recent versions of the SDK, the executable is located in the following path relative to the base directory of the SDK: `Android/android-sdk/platform-tools`. The call to `adb` should be of the form:

   <div align="center">

   `adb forward tcp:50000 tcp:50000`

   </div>

   This calls instructs `adb` to forward mobile TCP port 50,000 to PC TCP port 50,000. If desired, a different TCP port can be chosen on the workstation side. After the call, a message should appear in the command line indicating that the `adb` daemon process has started. See figure 4.8.

After the above steps have been performed, a TCP server should be listening to TCP port 50,000 on the workstation, ready to forward the GPS data to a client. The data is published in the form of NMEA strings. The NMEA is the National Marine Electronics Association and it has created a specification that describes how the communication interface between marine electronic devices should be. The specification, among other things, defines the standard data format used by most of the computer programs that work with real-time GPS position data. According to the specification, the GPS data should be transmitted in self-contained and independent ASCII character sequences called NMEA sentences. A NMEA sentence begins with a '$' symbol and ends with a carriage return/line feed sequence. It can have a length of up to 80 characters, not including the termination chars. After the '$' symbol, a NMEA sentence contains five identification characters. The first two indicate the

type of device and the remaining three indicate the sentence type. The prefix corresponding to GPS receivers is "GP". Each type of NMEA sentence contains different data fields, and not all of them contain position data. For more information on NMEA sentences, refer to [55], for example.

To receive the NMEA sentences in the data acquisition application, we developed a TCP client using the `Winsock2` API. This client connects to the TCP server established by the `abd` program. Our client, which runs in its own thread, retrieves data from the buffer, splits the data into NMEA sentences and parses them to obtain the values of the relevant data fields, which in this case are the latitude and the longitude. The parsing of the sentences is performed using the function `nmea_parse` from the NMEA library [1]. Other functions of the library are also used in the processing of the sentences. The latitude and longitude are updated about once a second. While recording the dataset, we noticed the smartphone would overheat after about two hours of continuous GPS data acquisition. This would result in a slower update rate for the GPS data. For this reason we tried to limit the length of the acquisition sessions to less than 2 hours, or to provide pauses so that the phone could cool down.

4.2.5   Stereo Camera calibration

We used Zhang's technique [108] as implemented in `OpenCV` [72] to calibrate the cameras. As described in section 3.3, the procedure proposed by Zhang requires the use of a planar calibration pattern viewed in different positions and orientations. In the case of stereo calibration, it is also needed that the pattern is fully visible by the two cameras, so that the rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t}$ between the cameras can be estimated.

A chessboard-like calibration pattern is used traditionally, and we tried it as our first option. We used a printed pattern of 9 columns $\times$ 7 rows glued to a flat piece of wood. The effective pattern dimensions were approximately 630×490 mm. After several calibration sessions we concluded that both the type of pattern and the material it was attached to were not the best options. The corners in the pattern were sometimes detected with poor accuracy. For this reason, when using this type of pattern, we had to carefully check the corner detection in each of the calibration images to remove all images in which one or more corners were improperly detected. This process was painstaking and time-consuming. Also, we found it difficult to work with wood because it requires much care to keep it flat, and often it would bend slightly anyway. The use of a bent pattern is undesirable, so we switched to another pattern, best suited for our purposes.

We generated a symmetric circle grid pattern using `OpenCV`'s pattern generator. The generator is a `Python` script, located on the path `doc/pattern_tools/gen_pattern.py` relative to the installation directory of `OpenCV`. It allows to generate an `SVG` vector file of the desired pattern. The fact that the file is vector-based allows to obtain a high-quality printing when printed in different sizes, in contrast to raster images that may introduce distortion. We created a bigger pattern, with the intention of covering as much of the image as possible, in both cameras simultaneously. The generated pattern was a regular circle grid of 9×6, with 120 mm of separation between circle centers. The printed pattern was glued to a 6 mm-thick glass sheet using spray adhesive. The dimensions of the glass sheet were approx. $1{,}200 \times 900$ mm. The results obtained using this pattern were much better. The detection of the circles centers performed in general very well so no exhaustive verification of each calibration frame was necessary. Also, because of its rigidity, the glass sheet would not bend, keeping the pattern flat. However, it had also a few downsides. The glass required careful handling and was very heavy. Maybe a better, but more expensive option would be the use of polycarbonate or some similar material, which is lighter, but still rigid. Also we noticed that the glued printed pattern had a tendency to wrinkle in cold weather, maybe because of a less-than-ideal glue work. For that reason, we tried to perform the
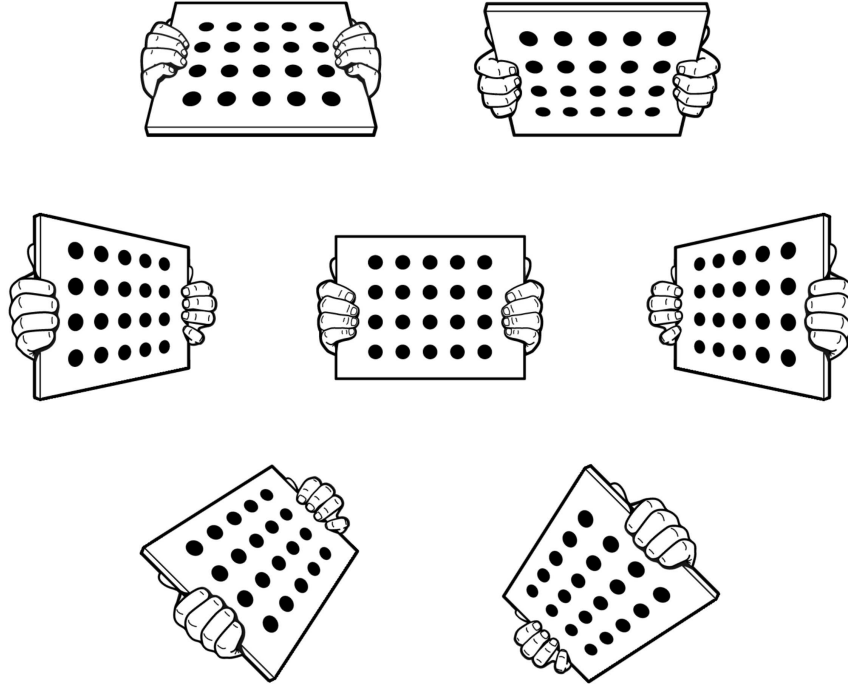
Figure 4.9: Base orientations of the calibration pattern. *Partial Image Credit:* National Instruments

calibration procedure in sunny weather, after the pattern had received a good sun bath.

To obtain the intrinsic parameters of each camera and the rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t}$ between cameras, one can obtain first the intrinsics for each camera by calibrating them individually, and use them afterwards complemented with stereo calibration pairs (in which the calibration pattern appears complete simultaneously in both cameras) to estimate $\mathbf{R}$ and $\mathbf{t}$. Or, one can estimate all the unknown parameters from a single set of stereo calibration pairs. Also, in camera calibration it is good practice to take pictures of the calibration pattern covering most of the image, so that the estimated parameters are calculated from most of the image sensor. In contrast to stereo calibration, calibrating the cameras individually allows this. We tried both strategies, calibrating cameras individually and as a pair, and found that in practice, besides requiring more time, individual calibration yields no benefits. We even obtained a slightly less accurate calibration (based on reprojection error) by using individual calibration followed by the estimation of $\mathbf{R}$ and $\mathbf{t}$ than by performing the stereo calibration in a single step. Thus, all the calibration parameters for the dataset were obtained using the latter technique.

For acquiring the stereo calibration pairs we used a predefined set of orientations of the calibration pattern, as shown in figure 4.9. We first took a picture of the pattern in each of the selected orientations, placing the pattern as close as possible to the cameras so that it would cover most of the image, while still appearing complete in both cameras. After, we would move the pattern away and take pictures of it in several of the selected orientations, while also moving it around so that it would appear in different parts of the image. We tried to cover most of the image area with the far shots. For improving the vertical area coverage, we took pictures of the pattern flipped around, as shown in figure 4.10, *Top left*, but the software would not recognize the pattern. It was interpreted instead as a 6×9 pattern.
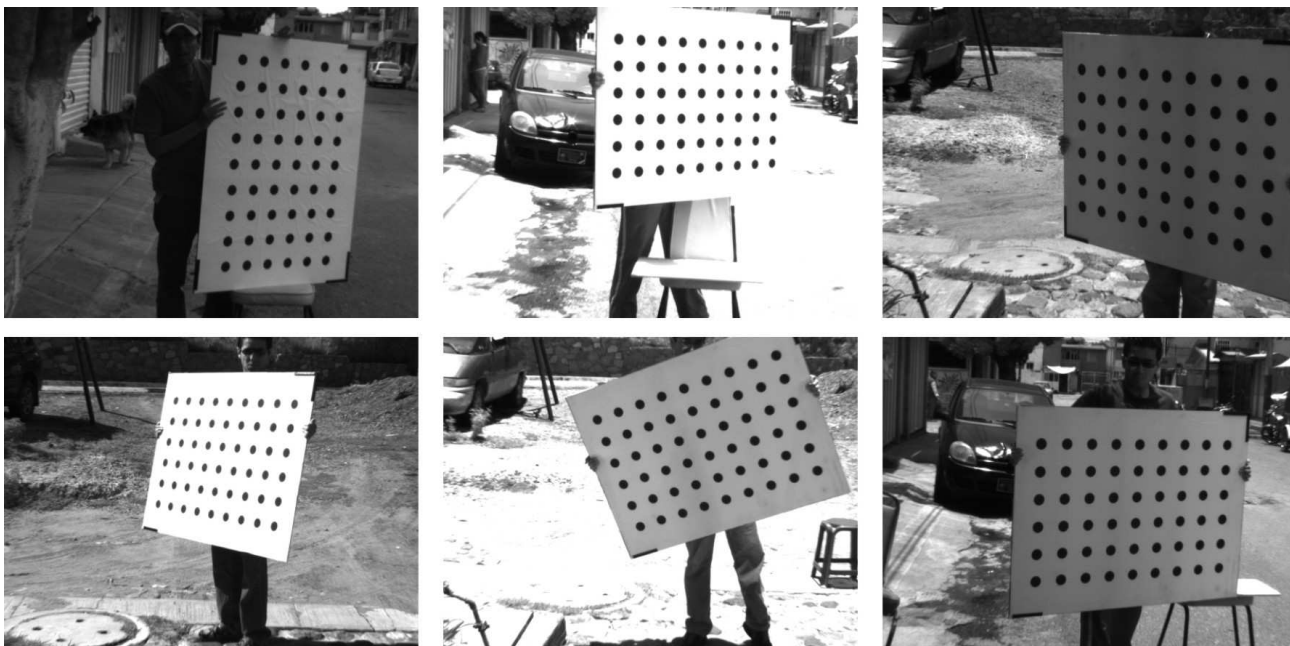
Figure 4.10: Example calibration photos. *Top Left:* The software would not recognize the pattern when oriented vertically. *Rest:* The calibration pattern shown in different positions and orientations. In all cases, the left frame is shown.

Initially we thought that calibrating the stereo pair would be needed only once, since the cameras were mounted rigidly to the bar, and the bar itself is attached rigidly to the car. However, we discovered that the calibration would not stay valid after unmounting the bar from the car and mounting it again. Rectifying images from the stereo pair using the parameters obtained in a previous calibration session would result in pairs not rectified correctly as corresponding points may not appear in the same row in both images. The row difference was small, one or two pixels, but significant enough to invalidate the calibration. We believe this was caused by a slight torsion in the bar derived from minor differences in the mounting torque applied to the supports of the bar. Because of this, we had to calibrate the stereo pair before each recording session. Given that the number of sessions was moderate, this was feasible. However, in a real-world application where the calibration may be affected constantly by many factors, such as vibration, dilatation caused by high temperatures and more, a self-calibration technique such as the one proposed in [26] should be used. Some examples of calibration images corresponding to the left camera are shown in figure 4.10.

To obtain the calibration parameters we relied on the `stereoCalibrate` function of `OpenCV`. To reduce the number of parameters to estimate, we assumed that the pixels are perfect squares, as stated by the camera datasheet, and that tangential distortion can be neglected. Both assumptions are supported by the fact that we used high-quality vision equipment.

4.3 Data Postprocessing

After the image sequences were acquired, some post-processing steps were required to make them better suited for stereo correspondence. We rectified them geometrically and reduced the effect of noise on the image sequences acquired at night. The details are described below.
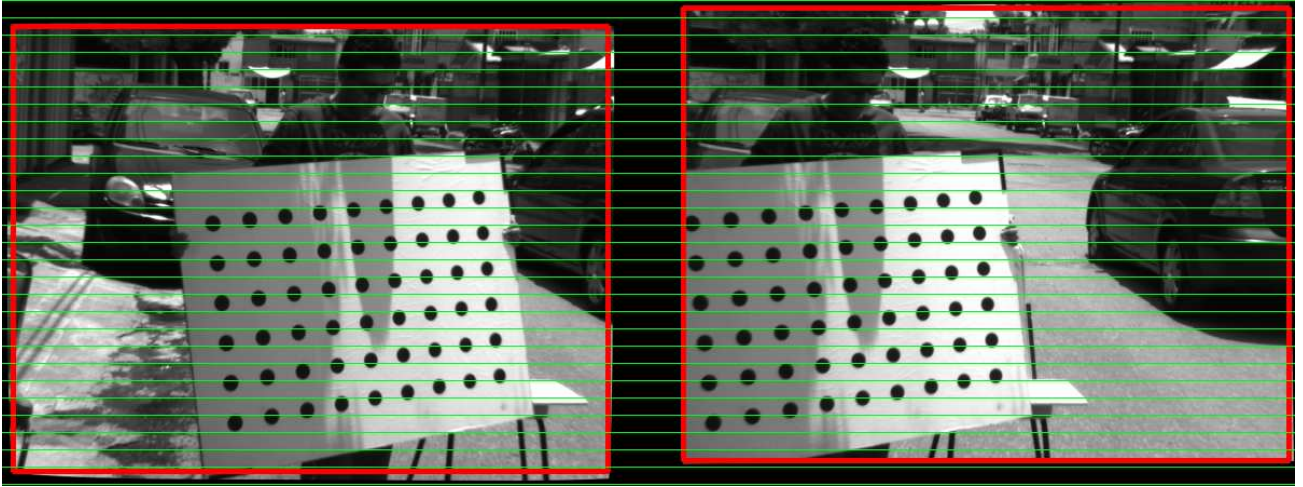
Figure 4.11: A rectified stereo calibration pair.

4.3.1   Image rectification

Because of slight mechanical misalignment caused by manufacturing differences of the mounting parts, after the images sequences were acquired it was necessary to rectify them, to make each of the image pairs row-aligned. To rectify the images, we have passed the calibration data obtained previously to the function `stereoRectify` from `OpenCV`, that implements the rectification algorithm proposed by Bouguet, see section 3.5. The function returns rectification maps that are used afterwards to warp the images. The resulting rectified images are as if they would have been taken by a canonical stereo pair. The coordinates of the principal point in both cameras are the same, and the focal length is the same in both u and v directions because of the square pixels. Because of the aforementioned misalignment and the effects of lens distortion, the rectified image pairs are slightly smaller than the original frames. A rectified stereo calibration pair is pictured in figure 4.11.

Some of the image sequences were recorded at night. In low-light situations like this, where the pixel intensities are low and in consequence the signal-to-noise ratio lowers, it is important to take measures to reduce the amount of noise affecting the image. Therefore, right before we started to record the night sequences, for each camera we acquired 100 images with the camera lens cap closed, and averaged them to obtain what is usually called a *dark frame*, an average image in which the pixel intensities are generated only by noise. Then, we subtracted from the acquired images the corresponding dark frame, in order to reduce the effects of noise in the images. A histogram of intensities obtained from one of the closed-cap images is shown in figure 4.12, *top*. The intensities correspond to a pixel depth of 12 bit, where the maximum value for a pixel would be 4,095. It can be seen that the observed intensities are low, but still significant. Figure 4.12, *bottom* shows a plot of the pixel intensities of the dark frame obtained for the left camera. Most of the intensities are low, but in some pixels, which are usually called *hot pixels*, the intensity is clearly above the average value. One such pixel is particularly visible in the figure. The observed intensity levels caused by noise are specially relevant when working with images with the full pixel depth. When working with 8-bit images, only the intensities of the hot pixels have a significant impact, because after the conversion to 8 bits, which implies a division by 16, most of the other pixel intensities become 0.
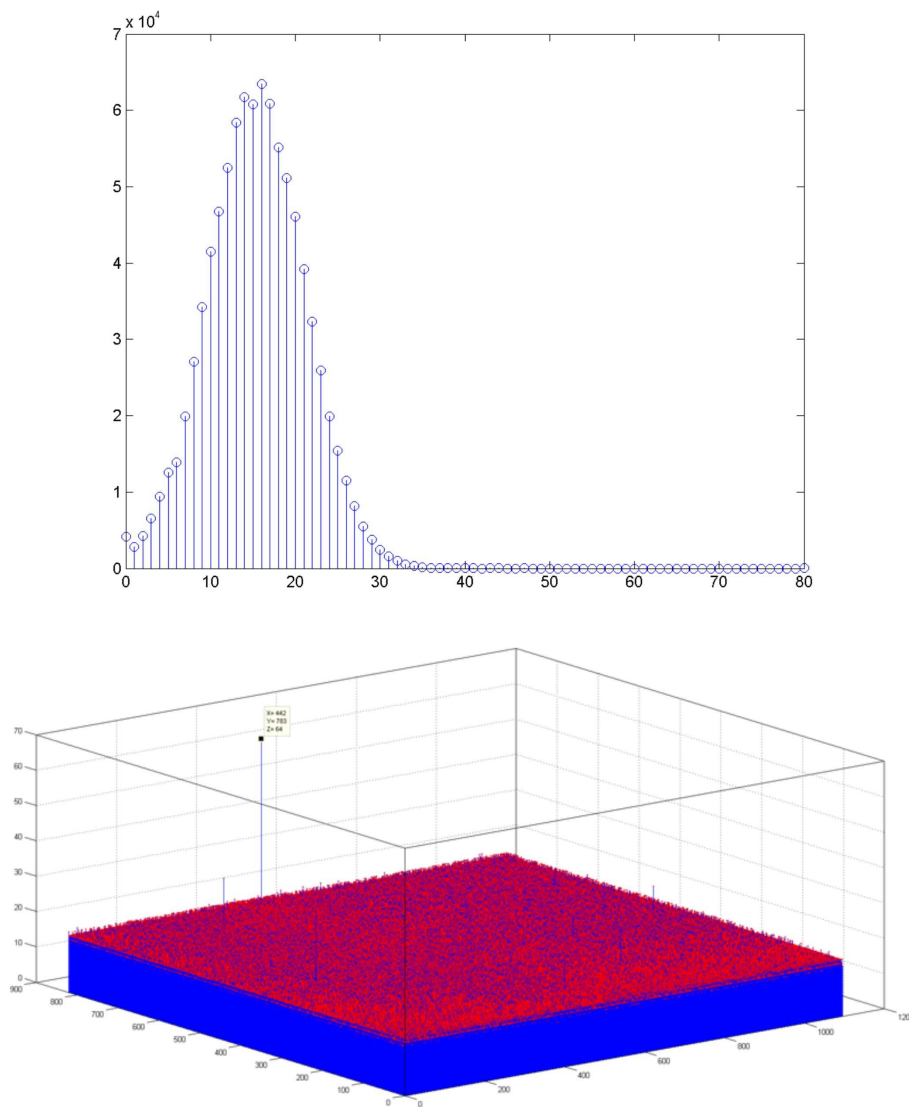
Figure 4.12: Dark Frames. *Left:* A histogram of intensities from a closed-cap image. *Right:* A plot of pixel intensities corresponding to the left camera dark frame. A hot pixel can be observed

Chapter 5

Dataset Description

In this chapter we describe the generated dataset. First, in section 5.1, we provide some general information about the dataset and the layout of the provided files. Then, in sections 5.2, 5.3, 5.4 and 5.5 we describe respectively the image, IMU, GPS and car data files that we provide. In section 5.6 the camera calibration data we obtained and make available is described. After, in section 5.7, we describe the calibration images we supply, so that the dataset users can perform their own camera calibration, if they wish to do so. Then, in section 5.8, we describe the contents of additional files we provide. Finally, in section 5.9, we summarize the created dataset and compare it with other similar datasets existing in the literature.

5.1   Introduction

We have acquired more than 96,000 stereo pairs distributed in 42 sequences. The longest sequence comprises 3,162 pairs whereas the shortest one is made of 1,201 pairs. Some of the sequences were recorded in urban environments located in the metropolitan area of Mexico City. The main environment challenges that can be found in these sequences are the poor quality of the road, as potholes and speed bumpers are very common. Also, the road signaling is sometimes insufficient, specially in places were road work is being performed. Besides, lane marking may not be clear and sometimes it is absent. To further enrich our dataset, besides the urban scenarios just described, we have recorded some sequences in the colonial city of Guanajuato. They offer different environment challenges such as very narrow streets, steep roads, an unstructured and abundant flow of pedestrians and a subterranean tunnel network.

The sequences we provide were recorded in three recording sessions. The first recording session was performed in the metropolitan area of Mexico City whereas the remaining two were performed in Guanajuato. For each recording session we performed the calibration of the stereo setup as described in section 4.2.5. We have classified the recorded sequences in four categories: '*Colonial Town Streets*', '*Urban Streets*', '*Avenues and Small Roads*', and '*Tunnel Network*'. Some example frames of each category are shown in figure 5.1. All sequences were recorded in daytime with varying lighting conditions, except for the sequences corresponding to the '*Tunnel network*' category, which were recorded at night. Upon examination of the images in the sequences, one can observe that they offer challenges for stereo-matching algorithms typically present in real-world scenarios like shadowing, transparency, and solar/lens flares.

| Urban Streets | Avenues & Small Roads | Colonial Town Streets | Tunnel Network |

Figure 5.1: Dataset image samples. The figure shows four examples for each category (along each column). The images shown correspond to the left camera.
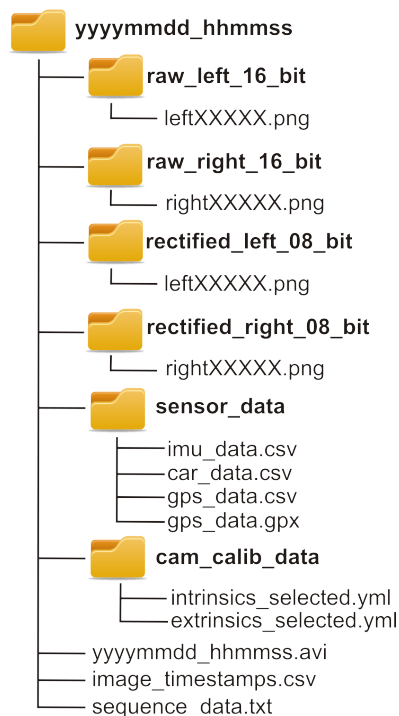
Figure 5.2: Structure of a recording file. The figure shows the structure and contents of a recording Zip file. Here 'yyyymmdd_hhmmss' represents the date and time of the recording, and 'XXXXX' represents the sequential frame number starting from zero.

For each recorded sequence, we provide a `ZIP` file with the folder structure described in figure 5.2. The file contains, besides all the acquired data, a video in compressed `AVI` format of the full sequence for quick visualization of its contents. All the provided timestamps correspond to the seconds elapsed since the start of the recording day, with a resolution up to milliseconds.

The coordinate frame in which the IMU data is expressed is shown in figure 5.3. In this figure, the right camera frame is also pictured. The frame of the left camera is not shown for clarity, but it has the same orientation as the one for the right camera. In the following sections, we describe in detail the data we provide.

5.2   Image data

We have acquired the stereo pairs at 20 fps, using a radiometric depth of 12 bits per pixel. The image resolution is 1,096 × 822. The raw images are saved with 16 bits per pixel, thus the 4 most significant bits are set to zero. The raw images corresponding to the left and right cameras can be found in the subdirectories `raw_left_16_bit` and `raw_right_16_bit`, respectively. For convenience, we also provide rectified images with 8-bit pixel depth in the folders `rectified_left_08_bit` and `rectified_right_08_bit`. Because of the effect of lens distortion and mechanical misalignment, the rectified images are slightly smaller than the raw ones. All images are saved in `PNG` files with lossless compression. The timestamps corresponding to the image pairs are provided in the text file `image_timestamps.csv`. Each line of the file contains one timestamp.

Prior to rectifying the images pertaining to the '*Tunnel Network*' category, we have subtracted the

Figure 5.3: Sensor coordinate frames. The IMU coordinate frame is pictured in green, whereas the right camera coordinate frame is shown in blue.

dark frames from the raw images. See section 4.3.1 for more details. The dark frames are provided in the calibration data we offer, which is described in section 5.7.

5.3   IMU data

The IMU we used can provide data for acceleration (m/s^2), orientation (*degrees*), rate of turn (*degrees/s*) and Earth magnetic field (*a.u.*, arbitrary units, normalized to Earth field strength). As mentioned in section 4.1.5 the orientation data is filtered to compensate for gyro drift, allowing to obtain a statistical optimal 3D orientation estimate of high accuracy. The rest of the fields are provided unfiltered.

The IMU data we have acquired is provided in a *comma-separated values* (CSV) text file called imu_data.csv. The data is stored in a tabular format, where each row corresponds to an IMU reading and the individual fields are separated by commas. Initially, we acquired data from the IMU at 50 Hz and acquired only acceleration and orientation data. Later improvements of the software and the wiring have allowed us to increase the acquisition rate to 100 Hz, and to include also the rate of turn and Earth magnetic field data.

The fields for each record are the following, and are delivered in the order specified: 'Timestamp', 'Roll', 'Pitch', 'Yaw', 'MagX', 'MagY', 'MagZ', 'GyroX', 'GyroY', 'GyroZ','AccX', 'AccY', 'AccZ'. When only orientation and acceleration data is provided, the data fields come in the following order: 'Timestamp', 'Roll', 'Pitch', 'Yaw', 'AccX', 'AccY', 'AccZ'.

As with all accelerometers, the ones included in the Xsens MTi measure all accelerations, including

Figure 5.4: View of a recorded GPS data file obtained using the software Google Earth.

the acceleration due to gravity. Actually, the accelerometers do not measure gravity (which would be negative in the IMU coordinate system) but the opposite force that keeps the IMU in place in the bar where it is mounted. Thus, to estimate the acceleration due to vehicle motion, gravity must be subtracted from the readings.

5.4 GPS data

The GPS data was recorded with an approximate frequency of 1 Hz and is provided in GPX format [2] in the file gps_data.txt. The use of the GPX format allows a quick review of the data in a compatible mapping application. A single track called 'Car Path' is used to save all the recorded track points. A view of one of the recorded GPS data files using the software Google Earth is shown in figure 5.4.

To keep data uniformity, the GPS data is also provided in a CSV file, called gps_data.csv. To convert the data form the GPX format to the CSV format we developed a small XML parser based on the Xerces library. The provided fields in the CSV file are 'Timestamp', 'Latitude' and 'Longitude' and come in that precise order.

5.5 Car data

The data acquired from the car computer is provided in the file car_data.csv which shares the basic format with the IMU and GPS data files. The car data was recorded with an approximate frequency of 3 Hz. The provided fields are the following: 'Timestamp', 'Speed' and 'RPM'. The speed units are km/h.
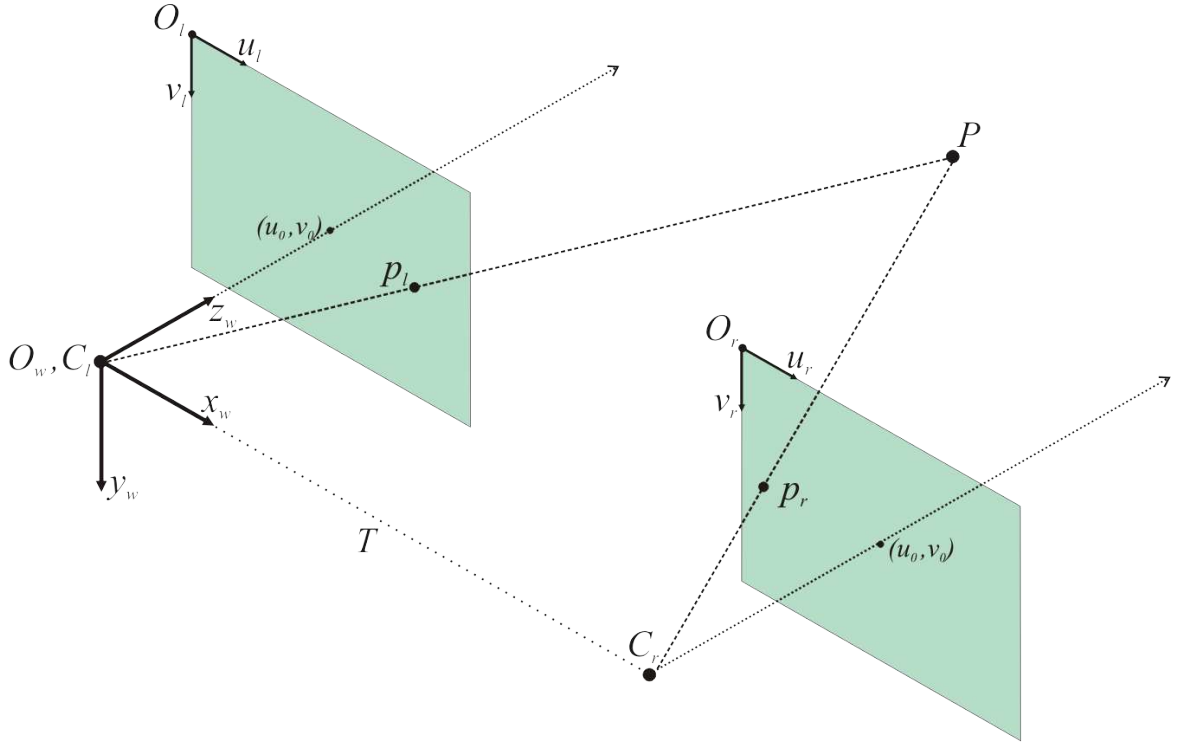
Figure 5.5: Rectified Coordinate Systems. The world coordinate system is placed in such a way that it is aligned with the camera coordinate systems and centered in the left camera's center of projection.

5.6   Camera calibration data

For each recording session, we acquired a set of calibration pairs. We used a selected subset of these calibration pairs to perform the calibration and rectification procedures described in sections 4.2.5 and 4.3.1. The use of a subset of the calibration pairs allows a shorter processing time without sacrificing the precision of the calibration. The calibration and rectification results are delivered in the text files `intrinsics_selected.yml` and `extrinsics_selected.yml`. The suffix 'selected' highlights the fact that the results were obtained from the selected subset of the calibration pairs, and not using all of them. Both files are compliant with `OpenCV`'s `FileStorage` format, which is based on `YAML`. In the file descriptions below, $i \in \{0, 1\}$, where 0 represents the left image and 1 represents the right image. The provided calibration data is calculated considering that 3D points in the world are expressed in a coordinate system O_w which is aligned with the rectified left camera coordinate system, but centered in the left camera's center of projection, see figure 5.5. The contents of the file `intrinsics_selected.yml` are the following:

- $\mathbf{K}_i \in \mathbb{R}^{3 \times 3}$: Camera calibration matrix for the i-th camera.
- $\mathbf{D}_i \in \mathbb{R}^8$:   Distortion coefficients for the i-th camera.

It is worth mentioning that $\mathbf{K}_i$ and $\mathbf{D}_i$ correspond to the unrectified images. When working with the rectified pairs, one should only use the matrices provided in the file `extrinsics_selected.yml`. The contents of this file are described below:

- $\mathbf{R} \in \mathbb{R}^{3\times3}$: Rotation matrix that, before rectification, would align the right-camera coordinate system with the left one.
- $\mathbf{T} \in \mathbb{R}^{3}$: Translation vector that, before rectification, would bring the origin of the right-camera coordinate system into the left one. Thus, $\mathbf{R}$ and $\mathbf{T}$ are the rotation matrix and translation vector relating the right and left camera coordinate systems before rectification.
- $\mathbf{R}_i \in \mathbb{R}^{3\times3}$: Rectifying rotation matrix for the i-th camera. These rotation matrices are the result of Bouguet's rectification algorithm, see section 3.5.
- $\mathbf{P}_i \in \mathbb{R}^{3\times4}$: Projection matrix for the rectified i-th camera. It maps 3D points expressed in O_w to 2D points expressed in the corresponding camera frame. See below for more details.
- $\mathbf{Q} \in \mathbb{R}^{3\times4}$: Reprojection matrix for points viewed in the rectified left camera. It maps an image point, together with its disparity, to a 3D point expressed in O_w. See below for more details.

The projection matrices $\mathbf{P}_i$ have the form:

$$\mathbf{P}_i = \begin{pmatrix} \alpha & 0 & u_0 & \alpha T_i \\ 0 & \alpha & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} , \tag{5.1}$$

where α is the focal length in pixels and T_i is the baseline with respect to the left camera, expressed in the i-th camera coordinate frame. They map 3D points $\mathbf{X} = \begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T$ in homogeneous coordinates to 2D points $\mathbf{x} = \begin{bmatrix} x & y & w \end{bmatrix}^T$ in homogeneous coordinates as follows:

$$\mathbf{P}_i \mathbf{X} = \mathbf{x} . \tag{5.2}$$

The reprojection matrix $\mathbf{Q}$ has the form:

$$\mathbf{Q} = \begin{pmatrix} 1 & 0 & 0 & -u_0 \\ 0 & 1 & 0 & -v_0 \\ 0 & 0 & 0 & \alpha \\ 0 & 0 & \frac{1}{T} & 0 \end{pmatrix} , \tag{5.3}$$

and can be used to obtain the 3D coordinates expressed in O_w of an image point $[u, v]$ in the rectified left camera, given its associated disparity d. The depth z of the point can be estimated using equation (3.12). The equations for obtaining the x and y coordinates can be derived from equations (3.4) and (3.5). The derivation yields:

$$x = z \frac{u - u_0}{\alpha} , \tag{5.4}$$

$$y = z \frac{v - v_0}{\alpha} . \tag{5.5}$$

Substituting equation (3.12) in (5.4) and (5.5) we finally obtain:
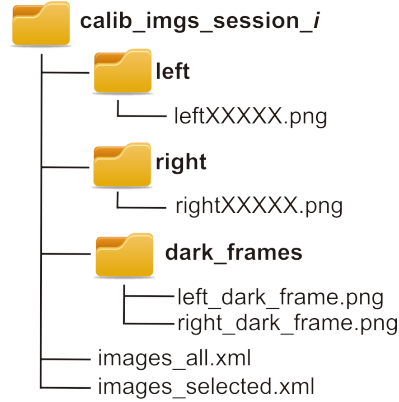
$$x = T \frac{u - u_0}{d} , \tag{5.6}$$

Figure 5.6: Structure of a session calibration data file.

$$y = T\frac{v - v_0}{d} \ .$$ (5.7)

Thus, given the coordinates $[u, v]$ of an image point and its disparity d, an homogeneous point $[u, v, d, 1]^T$ can be constructed. Then, it can be mapped to 3 dimensions using the following equation:

$$\mathbf{Q} \begin{bmatrix} u \\ v \\ d \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \ .$$ (5.8)

The 3D coordinates of the point are then $[\frac{x}{w}, \frac{y}{w}, \frac{z}{w}]$.

5.7   Calibration Images

We provide for each recording session all the acquired stereo calibration pairs in case the dataset user wants to perform the stereo calibration by him/herself. They are stored in a `ZIP` file named `calib_imgs_session_i.zip`, $i \in 1, 2, 3$, whose contents follow the distribution shown in figure 5.6. Besides the image pairs, we provide a couple of `XML` files called `images_all.xml` and `images_selected.xml`, which respectively enlist all the available calibration images and the ones we used to perform the calibration. The stereo calibration pairs are `PNG` files saved with a bit depth of 8 bits.

In the case of recording session 3, the session in which the night sequences were recorded, we also provide the obtained dark frames in the subdirectory 'dark_frames'. They are `PNG` image files with 16 bits of pixel depth.

5.8   Additional Data

In the file `sequence_data.txt`, we provide additional data about the sequence. In particular, we provide the number of pairs the sequence is made of, the recording session in which it was recorded, and the category we have assigned it to. An example file is shown below.

```
Number of pairs:  1998
Recording Session:  3
Category:  Colonial Town Streets
```

5.9   Dataset summary

We have acquired a novel stereo dataset, targeted to the task of environment perception in the context of autonomous driving. It is unique among existing datasets because it includes scenes featuring particular challenges that are mainly present in roads and streets of developing countries, and thus are not considered in the currently available datasets, which are recorded in developed countries.

The dataset comprises more than 96,000 monochrome stereo pairs, distributed in 42 sequences, ranging from 1,201 to 3,162 pairs in length. The sequences are classified in 4 categories: '*Colonial Town Streets*', '*Urban Streets*', '*Avenues and Small Roads*', and '*Tunnel Network*'. A diversity of challenges typically present in real-world, outdoor environments such as shadowing, reflections and lens flares are present in the acquired images. The stereo pairs were acquired with a custom-made stereo camera with a baseline of 50 cm. The pairs feature a wide range of disparities, 240 or greater for a typical pair. The images were acquired at 20 frames per second, with a resolution of $1,096 \times 822$ and a pixel depth of 12 bits. Rectified 8-bit images are also provided. The image data is complemented with GPS data and with readings from an IMU. All data is timestamped. Monocular and stereo camera calibration information is also provided.

A comparison of the main characteristics of our dataset, compared to those of the datasets mentioned in section 1.2 is shown in table 5.1. In the table, G.T. stands for Ground Truth and N/A stands for Not Available. Some of the featured datasets include different types of data, but the values shown in the table correspond exclusively to stereo data. The values of the Number of Disparities field were obtained from a typical pair, and may vary depending on the captured scene.

Dataset	Resolution	Framerate	Pixel Depth	No. of disp.	Color/Mono	G.T.
Our dataset	$1,096 \times 822$	20 fps	12 bits	~ 240	Mono	No
The Rawseeds project	640×480	15 fps	8 bits	N/A	Mono	No
EISATS	Up to $1,024 \times 1,024$	25 fps	Up to 12 bits	~ 110	Color & Mono	Some sequences
The HCI/Bosch Robust Vision Challenge	656×541	25 fps	12 bits	~ 70	Mono	No
The Daimler Ground Truth Stixel Dataset	$1,024 \times 333$	24 fps	12 bits	~ 160	Mono	Sparse
The annotated Leuven stereo data set	316×256	-	8 bits	~ 125	Color	Yes
The KITTI Dataset	$1,392 \times 512$	10 fps	8 bits	~ 128	Color & Mono	Yes

Table 5.1: Characteristics of our dataset compared with those of other similar datasets.

The dataset will be published at the following web address: `http://www.cimat.mx/~jbhayet/`

Chapter 6

Performance evaluation of selected stereo vision algorithms

Several of the stereo pairs that constitute the recorded dataset offer different challenges that are not easily handled by current stereo matching algorithms. Lens flares, reflections, motion blur, and projective distortion resulting from a wide baseline are some of the challenges that can be found in the acquired images. We have evaluated the performance of several stereo matchers under our real-world imagery and the results of the performed evaluation are presented in this chapter. First, in section 6.1, we present the selected stereo pairs for which we have obtained disparity maps. In this section, we also enlist the stereo matching algorithms we have used, together with the associated data costs. Then, in section 6.2, we present the obtained disparity maps and perform a qualitative analysis of them. Finally, in section 6.3, we perform a runtime comparison between the selected stereo matchers and evaluate their viability to be used in the context of autonomous driving.

6.1   Disparity Map Generation

To evaluate the performance of the selected stereo matching algorithms, we have chosen 12 pairs from the dataset. Our goal in choosing the pairs was to put together a set of pairs featuring different challenges that could potentially show the weaknesses of the stereo matching algorithms. The obtained disparity maps are shown in section 6.2. The selected pairs, together with a brief description of their content, are presented below.

- Stereo pair *"Big avenue"*. The pair features a big avenue with a relatively untextured road surface. This lack of texture makes the matching process more complicated. The lane markers are not clearly visible in some regions, and the roadwork ahead in the construction of a bus stop is poorly signaled.

Figure 6.1: The stereo pair *"Big avenue"*

- Stereo pair *"Street with potholes 1"*. A urban street with a very poor road surface is shown. Potholes are abundant, and besides they are filled with water from a recent downpour. The reflections in the accumulated water may cause errors in the matching process.

Figure 6.2: The stereo pair *"Street with potholes 1"*

- Stereo pair *"Juárez street"*. In the pair, a complex urban scene with many elements can be seen. Reflections on bright surfaces and saturation are the main challenges in this pair.

Figure 6.3: The stereo pair *"Juárez street"*

- Stereo pair *"Pocitos street 1"*. A urban scene where the shadow from a tall building makes it more difficult to distinguish corresponding pixels. The slope of the road magnifies the projective distortion caused by the horizontal displacement of the cameras. Also, some pixels on the road and sidewalk are saturated.

Figure 6.4: The stereo pair "*Pocitos street 1*"

- Stereo pair "*Lens flare 1*". The pair shows a scene which has been affected by a lens flare. The angle of the sunlight entering the camera lens caused reflections and the visualization of the iris aperture (seen as a set of concentric hexagons in the pictures). This results in a violation of the assumption of constant intensity of corresponding pixels, and potentially complicates the matching process.

Figure 6.5: The stereo pair "*Lens flare 1*"

- Stereo pair "*Tunnel crossing*". A crossing in the subterranean tunnel network we recorded is shown in this pair. The challenges of this scene are the low-light conditions and reflections caused by water on the road surface.
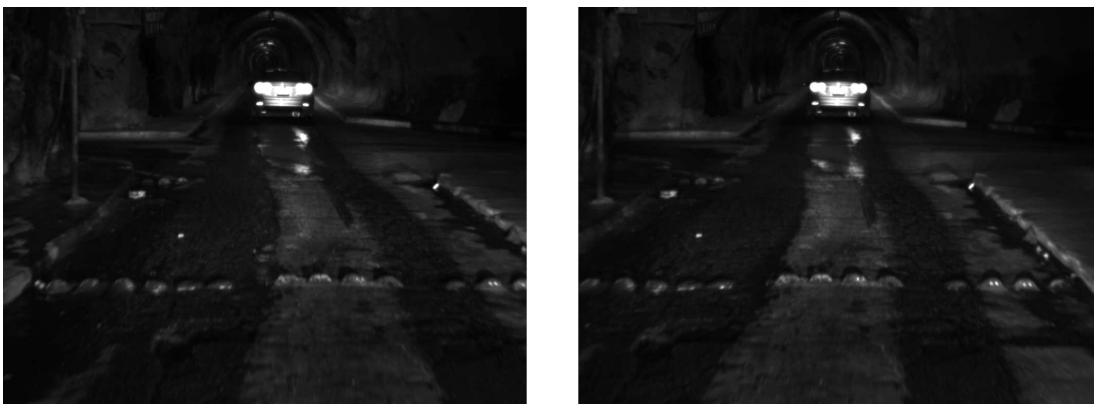
Figure 6.6: The stereo pair "*Tunnel crossing*"

- Stereo pair "*Tunnel speed bumper*". The pair shows a transition point between the exit of a

tunnel and the entrance of another. Low-light conditions, together with shadows, and projective distortions constitute the main challenges offered by the pair.

Figure 6.7: The stereo pair "*Tunnel speed bumper*"

- Stereo pair "*Street with potholes 2*". Another urban street with deplorable road surface is shown. The road, excluding the potholes, looks relatively untextured. Also, there is a speed bumper in the lower part of the image which in some places is difficult to distinguish from the main road surface.

Figure 6.8: The stereo pair "*Street with potholes 2*"

- Stereo pair "*Lens flare 2*". This pair shows another example of a lens flare affecting the scene. In this case, besides the presence of a reflection of the iris aperture, it can be seen that the left frame has more artifacts caused by the lens flare than the right frame, representing a significant violation of the intensity constancy assumption. Because of the curve on the road, the number of occluded pixels is significant.

Figure 6.9: The stereo pair "*Lens flare 2*"

- Stereo pair "*Top of a hill*". Acquired while driving on a road that goes up and down on a hill, the frame features shadows and several depth levels. Also, because of the white color of the visible houses and the sun position, many pixels are saturated, making it very difficult to find corresponding pixels. The sky in the horizon is also saturated.
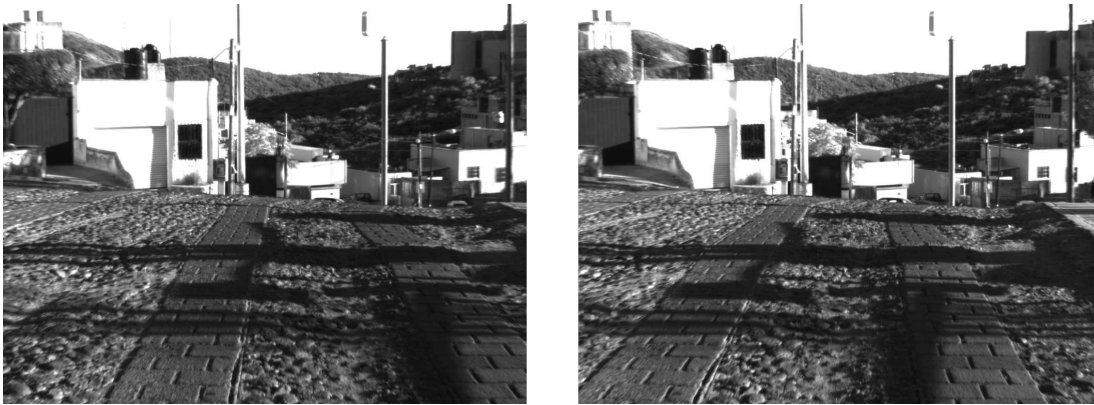

Figure 6.10: The stereo pair "*Top of a hill*"

- Stereo pair "*Columns in tunnel*". The pair, also recorded in the tunnel network, is affected by low-light conditions, and also by motion-blur, which is caused by the increased exposure time needed to gather enough light to perceive the scene with reasonable quality.
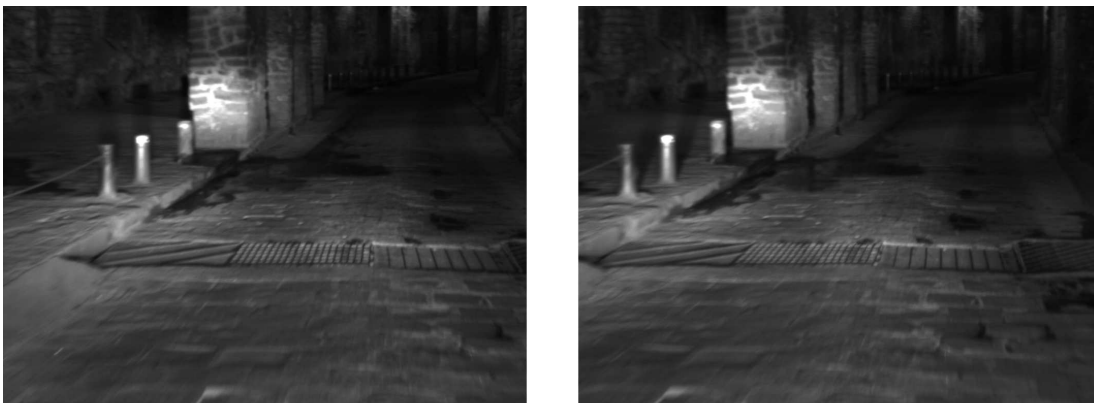

Figure 6.11: The stereo pair "*Columns in tunnel*"

- Stereo pair "*Pocitos street 2*". A typical colonial street is shown in the pair. The shadow of the building affects the left portion of the images, reducing the available texture information to determine corresponding pixels. Some pixels are saturated.

Figure 6.12: The stereo pair "*Pocitos street 2*"

6.1.1   Selected stereo matching algorithms

We have selected five stereo matching algorithms for evaluation. We have considered *block matching*, which is a local matcher (see section 3.6.3). Also, *semi-global matching*, an approximation to global matching (see section 3.6.4) is considered. We have included also two global matchers, *belief propagation matching* and *graph cut matching* (see sections 3.6.4 and 3.6.4 respectively). Finally, the ELAS stereo matching approach described in section 3.6.5 is included.

In the case of block-matching (BM), we have used the implementation of the algorithm included in `OpenCV` [72]. The algorithm is similar to the approach presented in [61]. In the implementation, prior to the cost calculation, a pre-filtering stage is performed to normalize image brightness and improve image texture. Then the DSI (see section 3.6.2) is calculated over the filtered image, using as data cost E_{SAD}. The winning disparities are calculated by WTA using equation (3.28). A post-filtering stage is performed on the obtained disparity map where a left-right consistency check is performed, and non-confident disparities are removed. Also, a "*speckle*" filter is optionally run over the disparity map to remove small patches of disparity values inconsistent with its surroundings. The values we used for all the required parameters are shown in table 6.1.

For semi-global matching, we have also used the implementation included in `OpenCV`. The algorithm is based on the original proposal described in [53], however, there are several differences, enlisted below:

- The used data cost is E_{BT} (see section 3.6.1) instead of the original mutual information metric proposed by Hirschmüller.

- The data cost is aggregated in a window, in contrast to the pixel-wise data cost calculation in the original paper. Because of this, the algorithm is called *Semi-Global Block Matching* (SGBM). However, if the parameter `SADWindowSize` is set to 1, the data cost is pixel-based.

- The number of directions used is normally 5, but can be increased to 8 if the `fullDP` flag is set. The original paper uses 16 directions.

- The pre and post-filtering steps used in `OpenCV`'s implementation of block matching are also included in the implementation of SGBM.

The corresponding parameter values used to generate our disparity maps are shown in table 6.1.

Parameter	Block Matching	Semi-Global Block Matching
SADWindowSize	11	3
minDisparity	0	0
numberOfDisparities	240	240
preFilterSize	11	N/A
preFilterCap	31	30
textureThreshold	10	N/A
uniquenessRatio	10	10
P1	N/A	50
P2	N/A	1000
speckleWindowSize	60	0
speckleRange	10	0
disp12MaxDiff	1	1
fullDP	N/A	true

Table 6.1: Parameter values considered for the used BM and SGBM implementations.

In the case of ELAS, we have used the implementation made available by the authors at [3]. We have used the parameter set `ROBOTICS` as specified by the authors. We have modified the original code to enable it to read `PNG` files, and to generate disparity maps with no scaling, in both gray scale and with a color code applied for better visualization.

For Belief Propagation (BP) matching, we have used the implementation of the approach presented in [34], which is available at [33]. The authors present several improvements over the usual implementations of belief propagation algorithms to improve runtime efficiency, including a hierarchical approach to perform BP in several levels, in a coarse-to-fine manner. In this approach, BP is run at a low level of resolution for some iterations. The resulting messages are then used to initialize the messages at the next finer level, accelerating convergence and enabling a quicker propagation of information. The authors originally use a scaled and truncated modified version of the E_{AD} data cost (see section 3.6.1), this is:

$$E_{data}(u, v, d) = \lambda \min(|E_{AD}(u, v, d)|, \tau) \ , \tag{6.1}$$

where λ is the scaling factor and τ is the truncation limit. In practice, this data cost proved to be too simple, and obtaining good quality disparity maps was not possible. For this reason, we have modified the provided code to use different data costs that replace E_{AD} in equation (6.1). We integrated the E_{SAD}, E_{ZSAD} and E_{CEN} data costs into the existing code, but only obtained good results when using E_{ZSAD} and E_{CEN}. The smoothness term is a linear truncated cost of the form:

$$E_{smooth}(f(\mathbf{p}), f(\mathbf{q})) = \min(|f(\mathbf{p}) - f(\mathbf{q})|, \gamma) \ . \tag{6.2}$$

The actual values used for λ, τ and γ, together with the window sizes are presented in table 6.2

Data Cost	λ	τ	γ	Window Size
E_{ZSAD}	0.1	50	100	3×3
E_{CEN}	0.2	10	100	3×3

Table 6.2: Parameter values used in the BP implementation.

For both used data costs, we have used 7 iterations of belief propagation and 7 scale levels. As

with the ELAS implementation, we have also added support to `PNG` files, and the generation of colored disparity maps.

In the case of graph cut matching, we have used the implementation of the approach described in section 3.6.4, which is available at [58]. The data cost integrated by the authors is E_{BT}, but extended to include also the vertical direction. In our tests, this data cost did not perform well, even when aggregated on a window, so we adapted the E_{SAD}, E_{ZSAD} and E_{CEN} data costs to the original implementation. In practice, we could obtain only good results with the E_{CEN} data cost. If $a1 = \langle \mathbf{p}, \mathbf{q} \rangle$ and $a2 = \langle \mathbf{r}, \mathbf{s} \rangle$, the penalty $V_{a1,a2}$ in equation (3.40) is defined as follows:

$$V_{a1,a2} = \begin{cases} 3\lambda & \text{if } \max(|I(\mathbf{p}) - I(\mathbf{r})|, |I(\mathbf{q}) - I(\mathbf{s})|) < 5 \\ \lambda & \text{otherwise ,} \end{cases} \tag{6.3}$$

where $I(\mathbf{p})$ represents the intensity at pixel $\mathbf{p}$. The penalty for not having the same disparity is small (λ) if the difference of intensity between adjacent assignments is significant, otherwise the penalty is big (3λ). We have defined the occlusion penalty C_o in equation (3.39) as $C_o = 10\lambda$, so that all the penalties are expressed in terms of a single parameter, λ.

Our best results were obtained using $\lambda = 3$, a window with dimensions 7×7, and setting the number of iterations to $t = 6$ in algorithm 1. No significant improvement on the quality of the disparity maps was noticeable using more iterations. In the used implementation, the pixels that end up labeled as occluded, are filled with the disparity from the left-closest non-occluded pixel along the same epipolar line. Similarly to the implementations of the other algorithms, we have included the capabilities of working with `PNG` files and generating disparity maps with a color code applied.

For all the stereo matchers but ELAS, the maximum disparity d_{max} was defined as 240, which is suitable for all the considered pairs. ELAS automatically detects the disparity range.

6.2   Qualitative evaluation

Unfortunately, we have no disparity ground-truth for the selected stereo pairs, so we cannot offer a quantitative analysis on the performance of the algorithms. Nevertheless, we can offer an analysis over the perceived quality of the disparity maps. Attributes such as density, presence of noise, coherence with scene information, smoothness, object border definition are the main qualities we have considered in our evaluation. The disparity maps we obtained are shown in figures 6.14, 6.15 and 6.16.

We have applied a color code to the obtained disparity maps so that they are better appreciated. The used color code is shown in figure 6.13. Red is applied to the closest objects and violet is for objects far away. The disparity maps obtained with BM, SGBM and ELAS show some pixels in black. They correspond to pixels for which a disparity has not been assigned, because either the left-right consistency check failed or the confidence on the assigned disparity was not high enough.

The block matching algorithm produced in general more sparse disparity maps than the rest of their counterparts. This is due to the fact that it failed to find disparity values for image zones with low texture, where image information alone is not enough to find a reliable disparity, see for example the BM disparity maps for the pairs *"Big avenue"* and *"Tunnel speed bumper"* in figures 6.14 and 6.15 respectively. It is in this situation that the lack of smoothness constraints penalizes the performance of the algorithm.

The results obtained by semi-global block matching are a clear improvement over the observed performance of block matching. This is clear while comparing the disparity maps for the pairs *"Big*
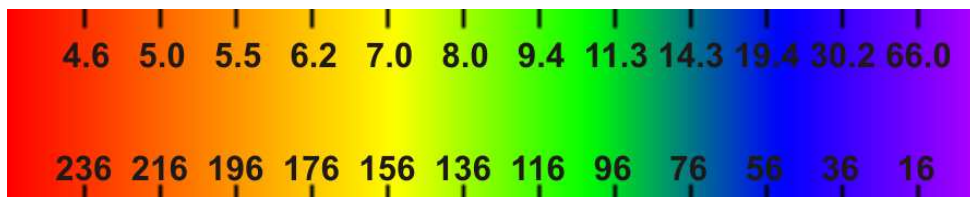
Figure 6.13: Color code applied to the obtained disparity maps. The upper scale shows the depth in meters. The lower scale shows the disparity.

avenue" and "*Tunnel speed bumper*" obtained with BM and SGBM, for example. Although SGBM also fails to determine disparities with high confidence on some low texture regions, the disparity maps it delivers are denser and provide more information that would allow a better scene understanding. As we show in section 6.3, the performance of SGBM is by far the fastest among global and semi-global matchers.

The disparity maps obtained by ELAS have a density that is slightly higher than the one obtained with SGBM. Also, they look "edgy", this is, the borders have in some cases a polygonal appearance, caused by the triangulation of the support points. In some pairs it failed to determine the disparities in the regions of the image that correspond to close objects, see for example the disparity maps for the pairs "*Big avenue*" and "*Pocitos street 1*" in figure 6.14. The performance of the algorithm was hampered, as in the case of the previous two, by the lack of texture. In some cases, such as in the far road section of the pair "*Columns in tunnel*", it manages to estimate the disparity, when BM and SGBM fail to do so.

BP allowed to obtain the disparity maps with the highest quality, specially using the E_{CEN} data cost. Even in low-texture situations, it found reasonable disparity estimates, generating dense and coherent disparity maps. No post-filtering step, like speckle filtering, was considered for the disparity maps obtained in this method, but, even if it was included, it seems that the obtained disparity maps would be the most dense and accurate of all. The definition of the borders of objects looks a bit over-smoothed in some pairs. In some cases, it "creates" non-existent structures, as in the the background of the pair "*Top of a hill*", in figure 6.16. Comparing the performance of the E_{ZSAD} and E_{CEN} data costs, it can be seen that, in general the disparity maps obtained using the E_{CEN} data cost are of better quality than the ones obtained using E_{ZSAD}. For example, for the pair "*Big avenue*", an artifact in the road surface can be seen in the BP disparity map obtained with E_{ZSAD} that is not present in the one obtained using E_{CEN}.

GC in general allowed to obtain disparity maps with reasonable quality. Similarly to BP, it was capable of determining disparities in low-texture situations. However, in contrast to the disparity maps obtained with BP, the smooth changes in depth, such as the ones corresponding to the road surface, sometimes do not show as smooth changes in disparity, see for example the disparity obtained by GC for the pair "*Tunnel crossing*" in figure 6.15. Also, the disparity maps obtained with GC are affected by noise, and several small patches of miscalculated disparities can be seen. The GC approach failed to assign correct disparities to the close region of the road in the pair "*Big avenue*" and also struggled with the background of the pair "*Top of a hill*".

Among the challenges present in the selected pairs, the reflections of water present on the road surface were the ones causing more trouble to the stereo correspondence algorithms. All of them were affected by this situation to a greater or lesser extent. BP with E_{CEN} data cost was the best dealing with this situation, see the disparity maps corresponding to the pairs "*Street with potholes 1*" and "*Tunnel crossing*" in figures 6.14 and 6.15 respectively. Rather surprisingly, GC and specially BP

were able to deal with the artifacts introduced by lens flares, as it can be seen in the disparity maps obtained for the pairs "*Lens flare 1*" and "*Lens flare 2*". ELAS also delivered reasonably good results, although some effects caused by the lens flares can be observed in the obtained disparity maps. BM and SGBM struggled more with this condition. Regarding the dark conditions present in pairs "*Tunnel crossing*", "*Tunnel speed bumper*" and "*Columns in tunnel*", BP was also the best matcher dealing with them, followed by GC, although the maps obtained with the latter technique are more affected with noise. The performance of ELAS was slightly better than the one offered by SGBM, both offering disparity maps with an acceptable density, but ELAS dealt in general better with the low-texture regions, which are common in this situation.

We conclude that BP offers the best overall performance for the selected stereo pairs. It was able to deal with most of the challenges presented in the pairs and delivers the disparity maps with the highest quality among the selected matchers. GC comes second in terms of the quality of the obtained disparity maps. However, they are affected by some noise and the lack of desirable smoothness in certain situations. ELAS and SGBM offer similar performances. The disparity maps they deliver are far from perfect, but seem useful enough to allow the basic understanding of the viewed scene. Their performance is low in regions where the image texture is scarce and cannot handle with ease the existing challenges in the pairs. Finally, the disparity yielded by BM are the ones with the lowest quality. This was expected because of the simplicity of the algorithm and its implicit lack of smoothness terms. However, in most cases the obtained disparity maps are still capable of offering valuable information which would allow to infer the scene structure.

6.3   Runtime comparison

To perform a comparison of the runtimes of the selected stereo matchers, we have calculated the disparity maps over 30 pairs, including the ones discussed in the previous section. We have used the frames at full resolution. The calculations were done on a mobile workstation running Linux Mint 64 Bit, equipped with a Intel Core i7 processor with 8 cores, each running at 2.20 Ghz, and 8 GB of RAM.

It is worth mentioning that `OpenCV`'s implementation of block matching and semi-global block matching takes advantage of the SSE2 (Streaming SIMD Extensions 2) instruction set when supported by the computer, enhancing the performance. Similarly, the implementation of ELAS exploits the SSE3 (Streaming SIMD Extensions 3) instruction set when the hardware supports it, reducing the calculation time. The used implementations of BP and GC are very efficient from the algorithmic point of view, but do not exploit any performance-enhancing library. The used computer supports both SSE2 and SSE3.

The running times for all the pairs are shown in figures 6.17, 6.18, 6.19, 6.20, 6.21 and 6.22 for BM, SGBM, ELAS, BP with ZSAD data cost, BP with CEN data cost and GC respectively. From the charts it can be seen that, in general, the content of a given stereo pair does not affect significantly the running time of the algorithms, as they remain relatively unchanged from pair to pair. The running times are expressed in milliseconds.

For an application like autonomous driving, it is very important to update as often as possible the representation of the environment ahead, to navigate safely. An update rate of several times a second is the minimum update rate required. For example, while traveling at 100 km/h, a car travels more than 27 meters each second. Also, at this speed, an average modern car equipped with ABS requires about 40 meters to stop completely in emergency braking. Because of this, it is of utmost importance to use a very fast algorithm.
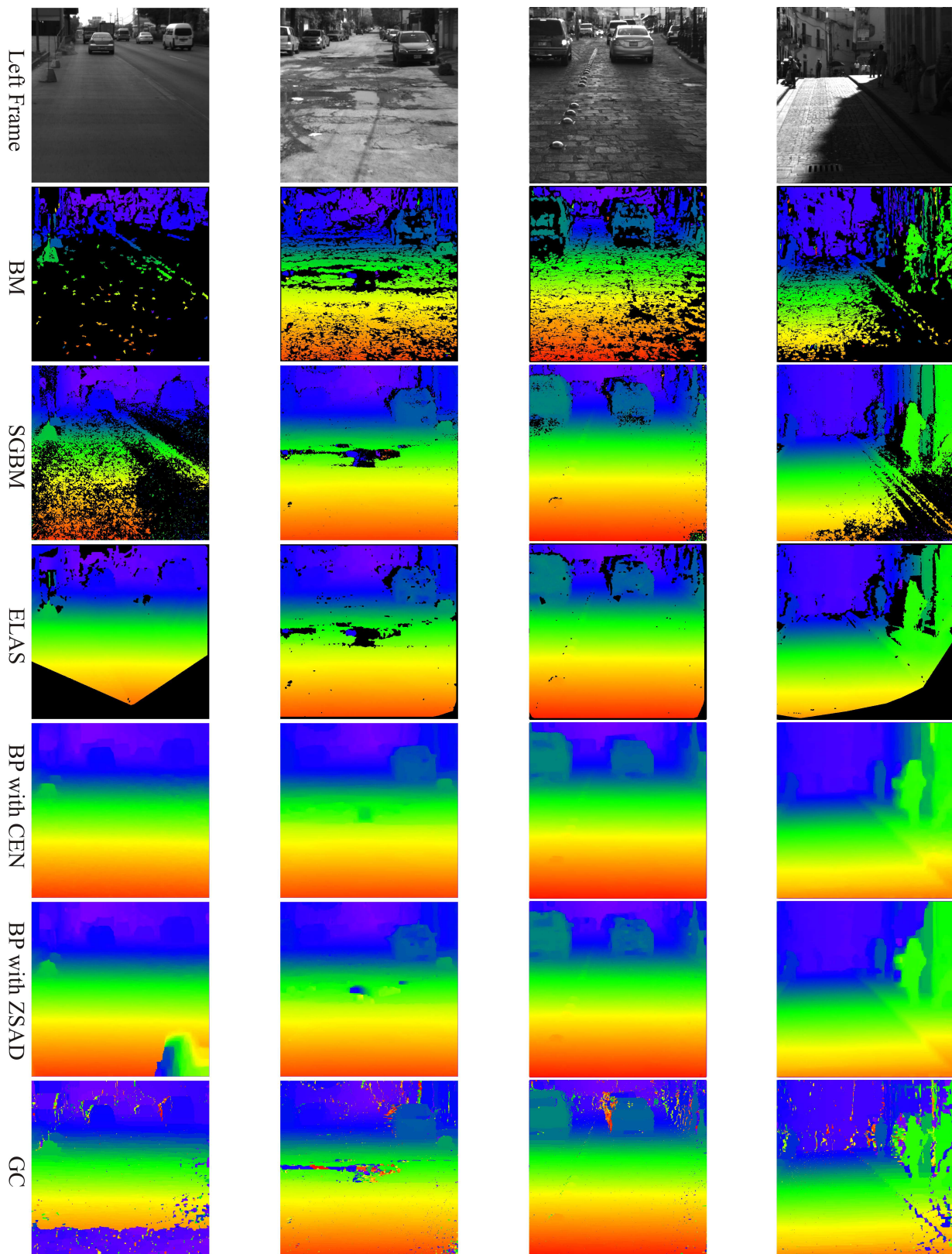
Figure 6.14: Disparity maps for the pairs *"Big avenue"*,*"Street with potholes 1"*,*"Juarez street"* and *"Pocitos street 1"* are shown respectively in the columns from left to right. In the first row, the left frame of the stereo pair is pictured for reference. From the second row and on, the disparity maps obtained using BM, SGBM, ELAS, BP with CEN data cost, BP with ZSAD data cost and GC are shown.
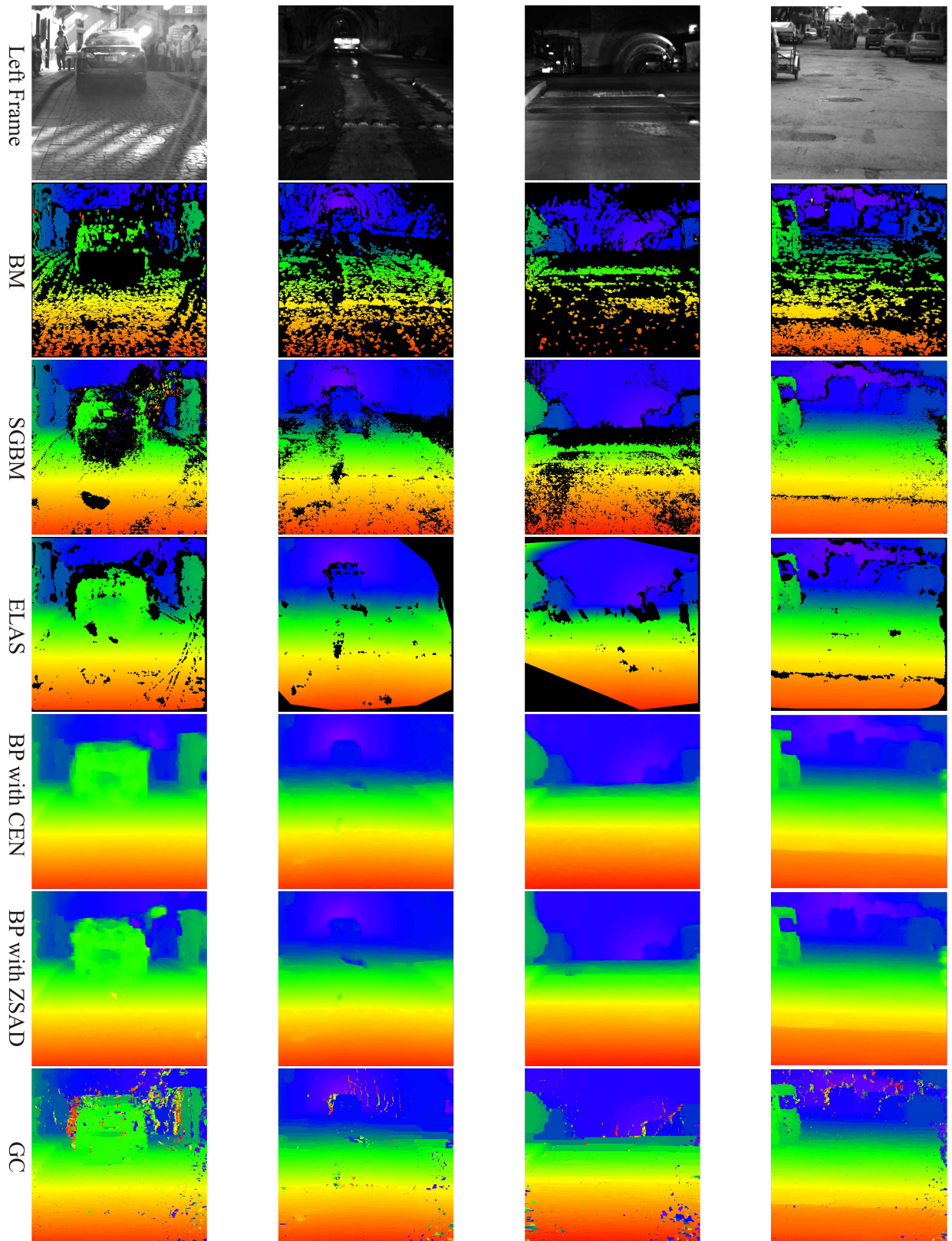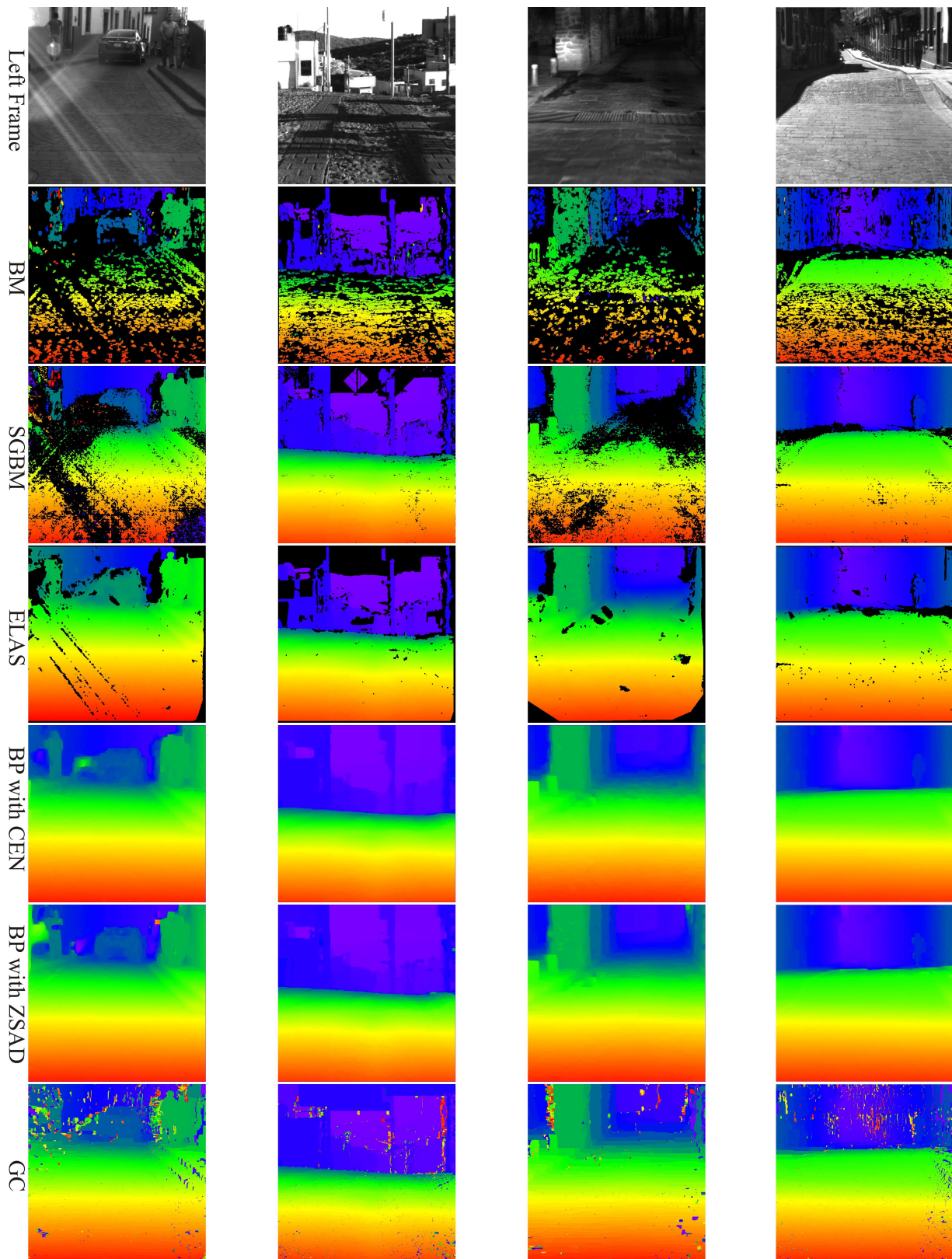
Figure 6.15: Disparity maps for the pairs "*Lens flare 1*","*Tunnel crossing*","*Tunnel speed bumper*" and "*Street with potholes 2*" are shown respectively in the columns from left to right. In the first row, the left frame of the stereo pair is pictured for reference. From the second row and on, the disparity maps obtained using BM, SGBM, ELAS, BP with CEN data cost, BP with ZSAD data cost and GC are shown.

Figure 6.16: Disparity maps for the pairs *"Lens flare 2"*,*"Top of a hill"*,*"Columns in tunnel"* and *"Pocitos street 2"* are shown respectively in the columns from left to right. In the first row, the left frame of the stereo pair is pictured for reference. From the second row and on, the disparity maps obtained using BM, SGBM, ELAS, BP with CEN data cost, BP with ZSAD data cost and GC are shown.
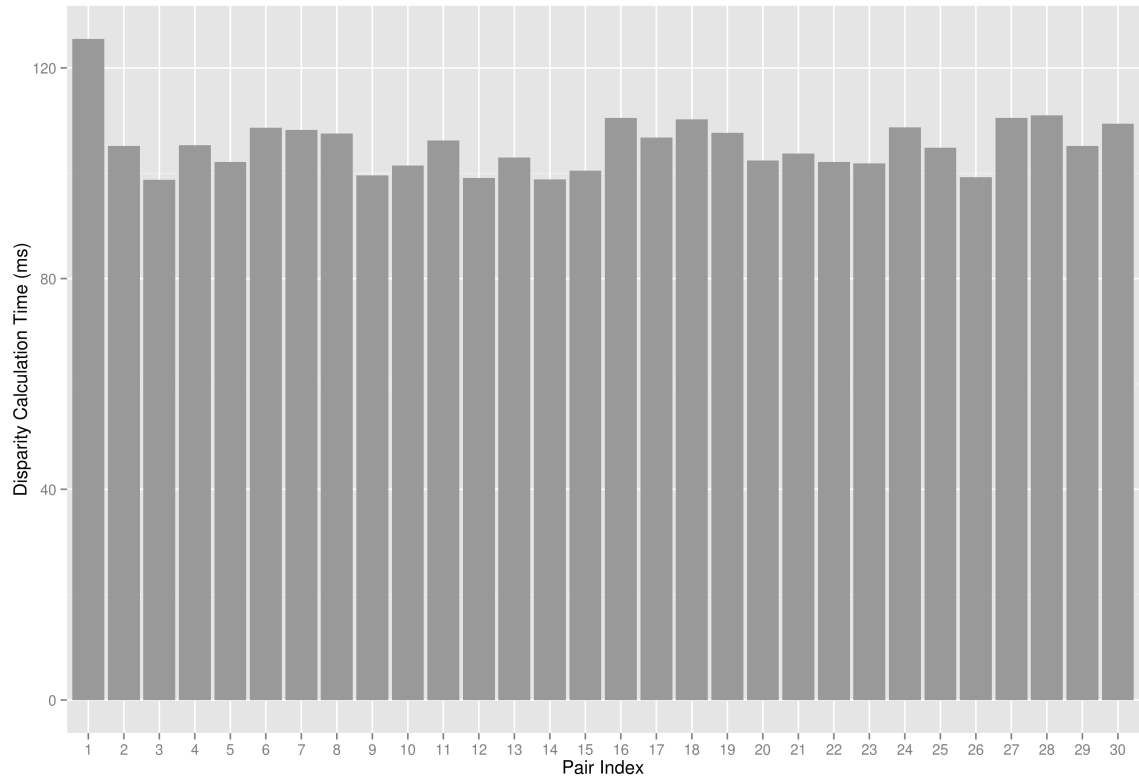
Figure 6.17: Disparity calculation run-times for block-matching.
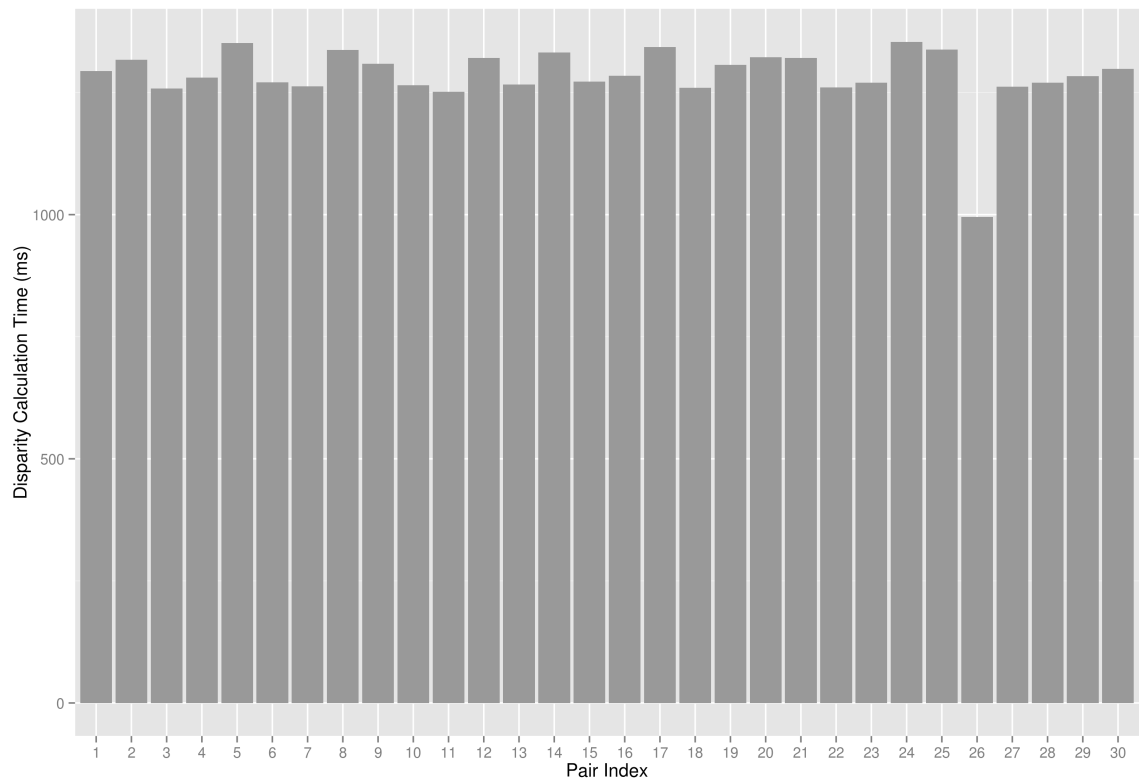

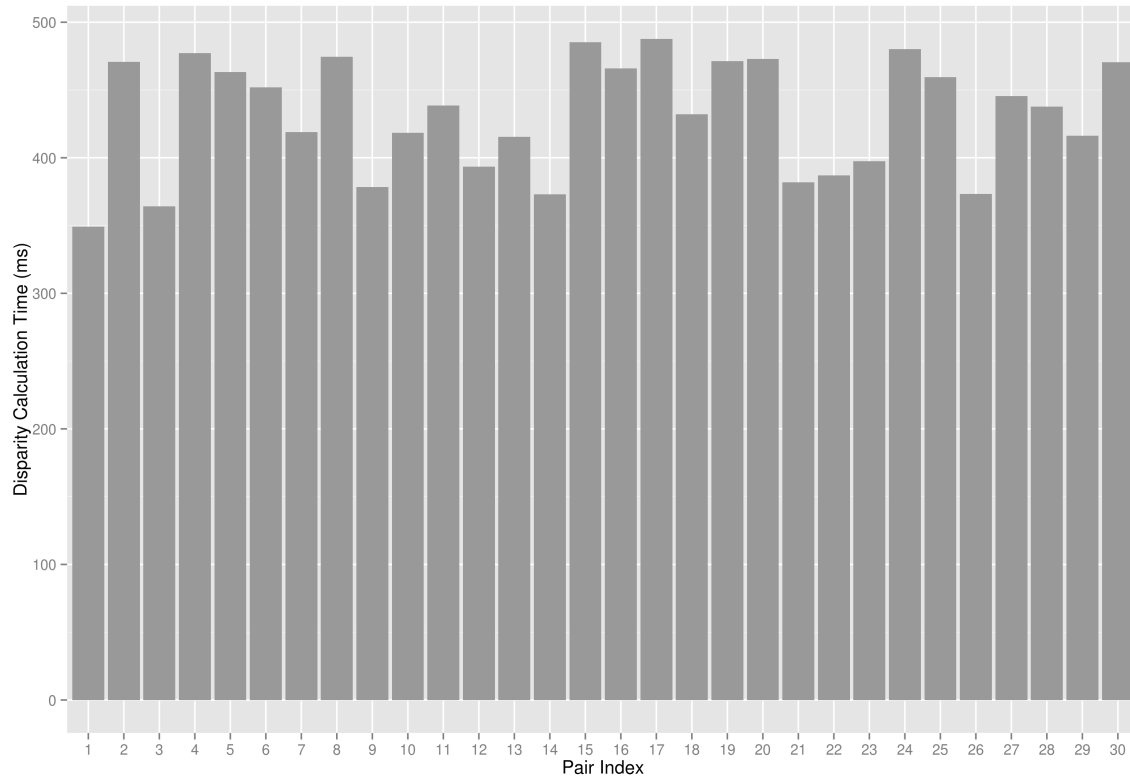Figure 6.18: Disparity calculation times for semi-global block-matching.

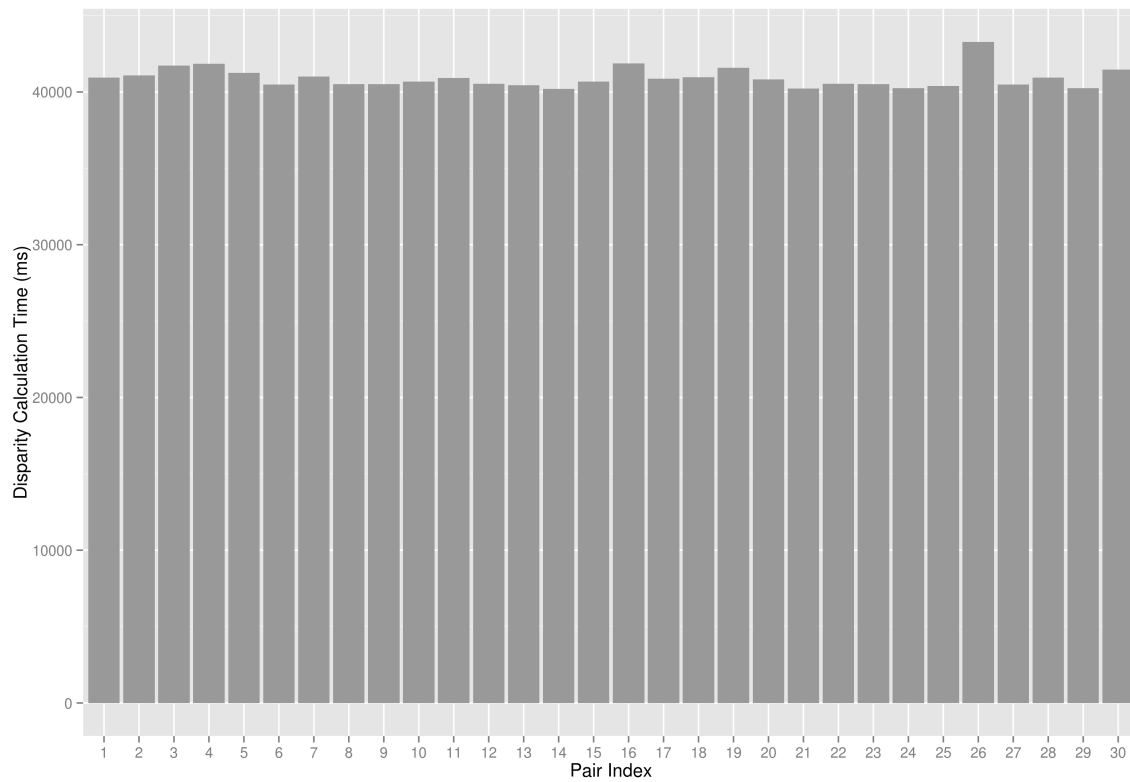Figure 6.19: Disparity calculation times for ELAS matching.

Figure 6.20: Disparity calculation times for BP matching with ZSAD data cost.
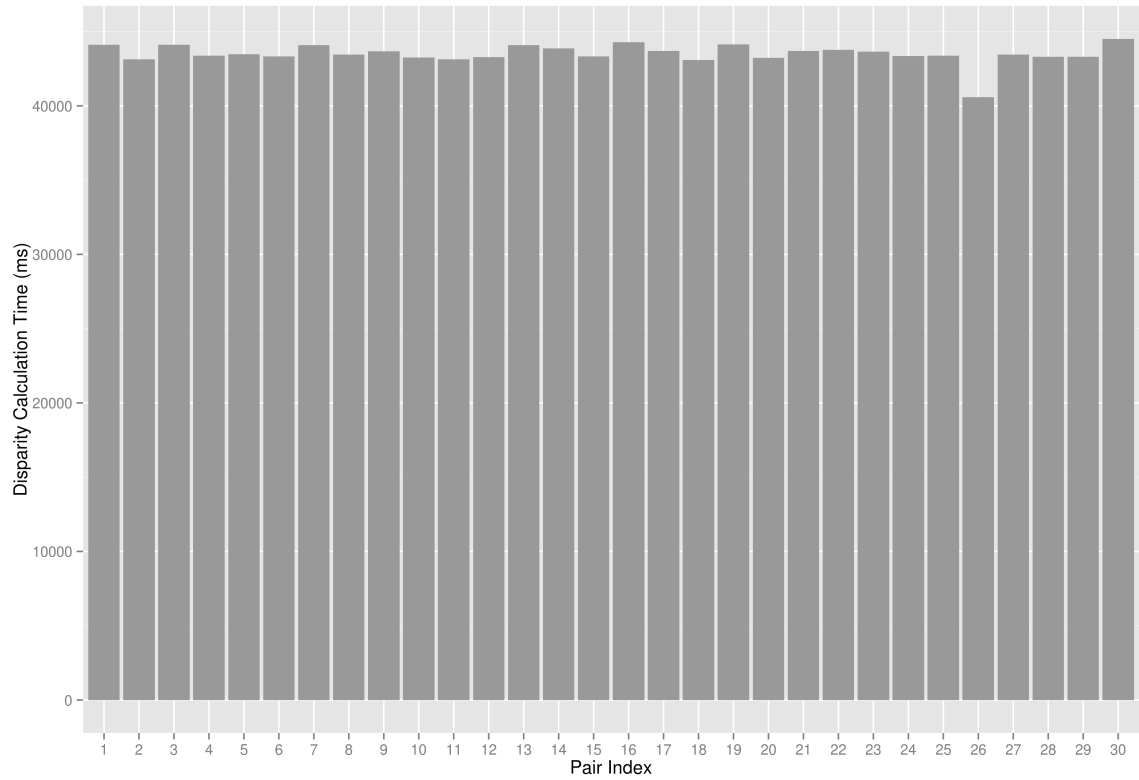
Figure 6.21: Disparity calculation times for BP matching with CEN data cost.
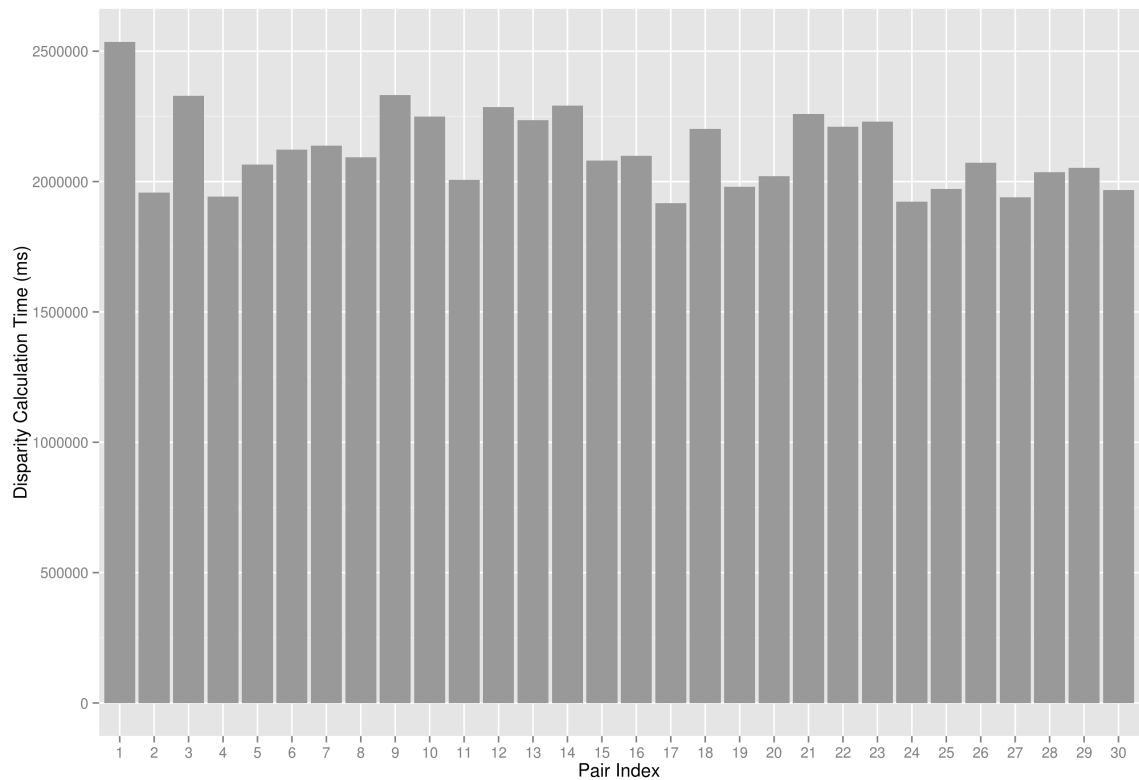

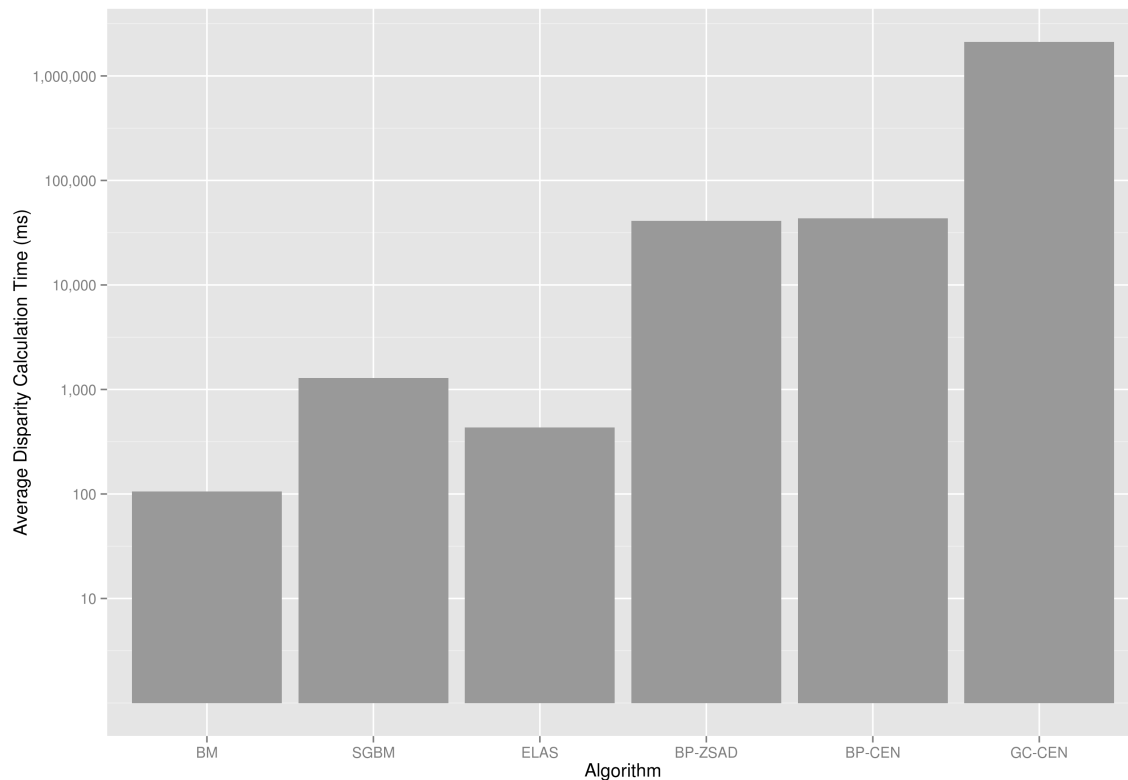Figure 6.22: Disparity calculation times for GC matching.

Figure 6.23: Runtime comparison of the selected stereo matchers. The $y-$scale is logarithmic to allow a better visualization of the data.

A side by side comparison of the average running time of each algorithm is presented in figure 6.23. In the bar plot shown, it can be seen that, from the used implementations, only the one corresponding to block matching allows a reasonable update time, with an average running time close to 100 ms. Furthermore, the block-matching algorithm is highly parallelizable and can be implemented very efficiently in parallel hardware. The runtimes of ELAS and SGBM are within the same order of magnitude, with average running times around 500 and 1,000 ms respectively, and, could potentially be used in autonomous driving after some algorithmic improvement or after their adaptation to run on specialized hardware, such as FPGAs or GPUs. Actually, for semi-global matching, several real-time implementations have already been developed ([6], [39], [40]). The execution times for BP, and specially GC, for which the runtime for some pairs was more than half an hour, are orders of magnitude above the desired runtime, so it will require more work from the scientific community before they can be used in real-time applications. In the case of BP, there are some works in the literature, for example [102] and [105], where the use of a GPU has made it possible to reach close to real-time performance on low-resolution imagery. It is worth mentioning that while generating the disparity maps for our images, we observed a memory usage of more than 4 GB in the used implementation of BP, which is a lot, specially considering that in autonomous driving the hardware platforms should be ideally low-power embedded systems, where memory is a constrained resource.

Chapter 7

Conclusions and Future Work

In this thesis we approached the problem of environment perception for autonomous driving using stereo vision. We focused particularly in the difficulties that exist in the roads and streets of developing countries, such as poor road conditions, abundant speed bumpers, insufficient signaling, among others, that add even more complexity to the task of autonomous vehicle navigation. To enable the testing, and subsequent improvement, of the existing environment perception algorithms in these conditions, we instrumented a vehicle to acquire stereo sequences featuring the mentioned challenges. Using the acquired pairs, we performed an evaluation on the performance of a few selected stereo correspondence algorithms, to determine their suitability for this task. We hope that this work is found useful and helps in further development of autonomous driving technology.

7.1 Main Contributions

- We have performed a review on the history and state of the art of autonomous driving technology, focusing particularly in environment perception. We discuss the main types of sensors used to perceive the environment surrounding the vehicles and expose their advantages and drawbacks.

- An in-vehicle data acquisition platform was developed from scratch to acquire our dataset. The devised platform is a distributed computing system that allows to acquire and save information obtained from multiple sensors, including a custom-made stereo camera and an inertial measurement unit.

- We provide a novel stereo dataset acquired in Mexican streets and roads. The recorded dataset features scenes where the road conditions are less than ideal, with many challenges for the task of autonomous driving that are not present in existing datasets, which have been recorded in developed countries. The dataset shows a variety of scenarios and environment conditions, that have proven to be challenging for traditional stereo matching algorithms.

- We have performed a qualitative evaluation of the performance of selected stereo matchers on selected pairs of the dataset. We have found suitable parameters and data cost functions that allow to obtain disparity maps with good quality in general. This evaluation may serve as a good starting point for future work in the dataset.

7.2   Future Work

Below we enlist a list of possible topics that may be developed as a follow-up of the work presented in this thesis.

- Extend the implementations of the selected stereo matching algorithms to allow the use of images with their full pixel depth of 12-bit. In preliminary tests we have seen that the increase in the bit depth allows to obtain better disparity maps.

- A few of the sequences we recorded lack rectifying calibration pairs, because of mechanical variations that result from unmounting and mounting again the sensor setup. Since the variations are minor, it may be possible to obtain the calibration information for these pairs with the self-calibration technique described in [26]. This would allow to extend the dataset.

- Test the road obstacle detection algorithms proposed in [62] and [106], which work in the disparity space. It would be important to see if they are capable of detecting road irregularities such as potholes and speed bumpers.

- Use the acquired sequences to test the performance of techniques for disparity calculation that take into account temporal consistency and/or motion information, such as the one presented in [69].

- Test existing object recognition algorithms on the acquired sequences for relevant object categories such as cars, motorcycles and pedestrians.

Bibliography

[1] NMEA Library.
http://nmea.sourceforge.net/.

[2] The GPS Exchange Format.
http://www.topografix.com/gpx.asp.

[3] Andreas Geiger. LIBELAS: Library for Efficient Large-scale Stereo Matching.
http://www.cvlibs.net/software/libelas/.

[4] Atreya, Anand R. and Cattle, Bryan C. and Collins, Brendan M. and Essenburg, Benjamin and Franken, Gordon H. and Saxe, Andrew M. and Schiffres, Scott N. and Kornhauser, Alain L. Prospect Eleven: Princeton University's entry in the 2005 DARPA Grand Challenge. *Journal of Field Robotics*, 23(9):745–753, 2006.

[5] Australian Broadcasting Corporation. Robotic trucks taking over Pilbara mining operations in shift to automation.
www.abc.net.au/news/2014-04-25/computer-controlled-trucks-taking-over-in-pilbara-mining-wa/5412642.

[6] Banz, Christian and Hesselbarth, Sebastian and Flatt, Holger and Blume, Holger and Pirsch, Peter. Real-time stereo vision system using semi-global matching disparity estimation: Architecture and FPGA-implementation. In Kurdahi, Fadi J. and Takala, Jarmo, editor, *International Conference on Embedded Computer Systems (SAMOS)*, pages 93–101. IEEE, 2010.

[7] Barth, Alexander and Franke, Uwe. Where Will the Oncoming Vehicle be the Next Second? In *IEEE Intelligent Vehicles Symposium*, pages 1068 – 1073, 2008.

[8] Basler. Basler Pylon Software.
http://www.baslerweb.com/Software_pylon-15606.html.

[9] Basler A.G. Basler scout Series.
http://www.baslerweb.com/products/scout.html/.

[10] Massimo Bertozzi, Alberto Broggi, Alessandro Coati, and Rean Isabelli Fedriga. A 13,000 km Intercontinental Trip with Driverless Vehicles: The VIAC Experiment. *IEEE Intelligent Transportation Systems Magazine*, 5(1):28–41, 2013.

[11] Bilger, Burkhard. AUTO CORRECT. Has the self-driving car at last arrived?
http://www.newyorker.com/reporting/2013/11/25/131125fa_fact_bilger.

[12] Birchfield, Stan and Tomasi, Carlo. A Pixel Dissimilarity Measure That Is Insensitive to Image Sampling. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(4):401–406, Apr 1998.

[13] Birchfield, Stan and Tomasi, Carlo. Depth Discontinuities by Pixel-to-Pixel Stereo. *Int. Journal Computer Vision*, 35(3):269–293, Dec 1999.

[14] Black, Michael J. and Rangarajan, Anand. On the Unification of Line Processes, Outlier Rejection, and Robust Statistics with Applications in Early Vision. *Int. Journal Computer Vision*, 19(1):57–91, Jul 1996.

[15] Boykov, Yuri and Veksler, Olga and Zabih, Ramin. Fast Approximate Energy Minimization via Graph Cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1222–1239, Nov 2001.

[16] Brad Templeton. Cameras or Lasers?
http://www.templetons.com/brad/robocars/cameras-lasers.html.

[17] Bradski,Gary R. and Kaehler, Adrian. *Learning OpenCV - computer vision with the OpenCV library: software that sees.* O'Reilly, 2008.

[18] Alberto Broggi, Massimo Bertozzi, Alessandra Fascioli, Corrando Guariono Lo Bianco, and Aurelio Piazzi. The ARGO autonomous vehicle's vision and control systems. *International Journal of Intelligent Control and Systems*, 3(4), 1999.

[19] Alberto Broggi, Paolo Grisleri, and Paolo Zani. Sensors Technologies for Intelligent Vehicles Perception Systems: a Comparison between Vision and 3D-LIDAR. In *IEEE Intl. Conf. on Intelligent Transportation Systems*, pages 887–892, 2013.

[20] Martin Buehler, Karl Iagnemma, and Sanjiv Singh, editors. *The 2005 DARPA Grand Challenge. The Great Robot Race*, volume 3 of *Springer Tracks in Advanced Robotics*. Springer, 2007.

[21] Martin Buehler, Karl Iagnemma, and Sanjiv Singh, editors. *The DARPA Urban Challenge. Autonomous Vehicles in City Traffic*, volume 56 of *Springer Tracks in Advanced Robotics*. Springer, 2009.

[22] Carnegie Mellon University. No hands across America official press release.
http://www.cs.cmu.edu/afs/cs/usr/tjochem/www/nhaa/official_press_release.html.

[23] Chen, Yi-Liang and Sundareswaran, Venkataraman and Anderson, Craig and Broggi, Alberto and Grisleri, Paolo and Porta, Pier Paolo and Zani, Paolo and Beck, John. TerraMax: Team OshKosh urban robot. *Journal of Field Robotics*, 25(10):841–860, 2008.

[24] CONAPRA. Seguridad Vial.
http://conapra.salud.gob.mx/Programas/Seguridad_Vial.html.

[25] Daimler A.G. Ground Truth Stixel Dataset. Available at:
http://www.6d-vision.com/ground-truth-stixel-dataset.

[26] Dang,Thao and Hoffmann,Christian and Stiller,Christoph. Continuous Stereo Self-calibration by Camera Parameter Tracking. *IEEE Transactions on Image Processing*, 18:1536–1550, 2009.

[27] de Klein, Ramon. Serial Library for C++.
http://www.codeproject.com/Articles/992/Serial-library-for-C.

[28] Ernst D. Dickmanns. *Dynamic Vision for Perception and Control of Motion.* Springer, 2007.

[29] Dickmanns, E. D. and Behringer, R. and Dickmanns, D. and Hildebrandt, T. and Maurer, M. and Thomanek, F. and Schiehlen, J. The Seeing Passenger Car 'VaMoRs-P'. In *Int. Symp. on Intelligent Vehicles*, pages 68–73, 1994.

[30] Dickmanns, Ernst. Dynamic machine vision.
`http://www.dyna-vision.de`.

[31] Elm Electronics. ELM-327 OBD Interpreter.
`http://elmelectronics.com/obdic.html#ELM327`.

[32] Faugeras, Olivier and Luong, Quang-Tuan and Papadopoulou, T. *The Geometry of Multiple Images: The Laws That Govern The Formation of Images of A Scene and Some of Their Applications.* MIT Press, Cambridge, MA, USA, 2001.

[33] Felzenszwalb, Pedro. Efficient Belief Propagation for Early Vision website.
`http://cs.brown.edu/~pff/bp/`.

[34] Felzenszwalb, Pedro F. and Huttenlocher, Daniel P. Efficient Belief Propagation for Early Vision. *Int. J. Comput. Vision*, 70(1):41–54, Oct 2006.

[35] Fontana,Giulio and Matteucci,Matteo and Sorrenti,Domenico G. *Methods and Experimental Techniques in Computer Engineering*, chapter 4. Rawseeds: Building a Benchmarking Toolkit for Autonomous Robotics, pages 55–68. Springer Briefs in Applied Sciences and Technology. Springer International Publishing, 2014.

[36] L.R. Ford and Fulkerson D.R. *Flows in Networks*. Princeton University Press, 2010.

[37] Freie Universität Berlin. AutoNOMOS Labs, the Freie Universität Berlin.
`http://autonomos.inf.fu-berlin.de`.

[38] Fusiello, Andrea and Trucco, Emanuele and Verri, Alessandro. A Compact Algorithm for Rectification of Stereo Pairs. *Machine Vision and Applications*, 12(1):16–22, July 2000.

[39] Gehrig, Stefan K. and Eberli, Felix and Meyer, Thomas. A Real-Time Low-Power Stereo Vision Engine Using Semi-Global Matching. In Fritz, Mario and Schiele, Bernt and Piater, Justus, editor, *7th International Conference on Computer Vision Systems: Computer Vision Systems*, volume 5815 of *Lecture Notes in Computer Science*, pages 134–143. Springer, 2009.

[40] Gehrig, Stefan K. and Rabe, Clemens. Real-Time Semi-Global Matching on the CPU. In *IEEE Computer Vision and Pattern Recognition Workshops*, pages 85–92, San Francisco, CA, USA, June 2010.

[41] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets Robotics: The KITTI Dataset. *International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[42] Geiger,Andreas and Roser,Martin and Urtasun,Raquel. Efficient Large-Scale Stereo Matching. In *Asian Conference on Computer Vision (ACCV)*, pages 25–38. Springer-Verlag, 2010.

[43] Google. Just Press Go: designing a self-driving vehicle.
`http://googleblog.blogspot.mx/2014/05/just-press-go-designing-self-driving.html`.

[44] Google. The latest chapter for the self-driving car.
`http://googleblog.blogspot.mx/2014/04/the-latest-chapter-for-self-driving-car.html`.

[45] Google. The self-driving car logs more miles on new wheels.
`http://googleblog.blogspot.mx/2012/08/the-self-driving-car-logs-more-miles-on.html`.

[46] Google. What we're driving at.
     http://googleblog.blogspot.com.es/2010/10/what-were-driving-at.html.

[47] Grisleri, Paolo and Fedriga, Isabella. The BRAiVE platform. In *7th IFAC Symposium on Intelligent Autonomous Vehicles*, pages 497–502, Lecce, Italy, 2010.

[48] Hargrave, Vic. Java-Style thread class in C++.
     http://vichargrave.com/java-style-thread-class-in-c/.

[49] Hargrave, Vic. Multi-threaded work queue in C++.
     http://vichargrave.com/multithreaded-work-queue-in-c/.

[50] Hartley, R. I. and Zisserman, A. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2nd edition, 2004.

[51] Heidelberg Collaboratory for Image Processing. Robust Vision Challenge. Available at:
     http://hci.iwr.uni-heidelberg.de/Static/challenge2012/.

[52] Hermann, Simon and Klette, Reinhard. Iterative Semi-Global Matching for Robust Driver Assistance Systems. In *Asian Conference on Computer Vision*, pages 465–478, 2012.

[53] Hirschmuller, Heiko. Stereo Processing by Semiglobal Matching and Mutual Information. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(2):328–341, February 2008.

[54] Hirschmuller, Heiko and Scharstein, Daniel. Evaluation of Stereo Matching Costs on Images with Radiometric Differences. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(9):1582–1599, Sep 2009.

[55] GPS Information. NMEA Data.
     http://www.gpsinformation.org/dale/nmea.htm.

[56] Jean Yves Bouguet. Camera Calibration Toolbox for Matlab.
     http://www.vision.caltech.edu/bouguetj/calib_doc/.

[57] Klette, Reinhard. *Concise Computer Vision - An Introduction into Theory and Algorithms*. Undergraduate Topics in Computer Science. Springer, 2014.

[58] Kolmogorov, Vladimir. Stereo Match Software.
     http://pub.ist.ac.at/~vnk/software.html#MATCH.

[59] Kolmogorov, Vladimir and Zabih, Ramin. Computing Visual Correspondence with Occlusions using Graph Cuts. In *International Conference of Computer Vision*, pages 508–515, 2001.

[60] KPMG and the Center for Automotive Research. Self-driving cars: The next revolution. Technical report, KPMG and the Center for Automotive Research, 2012.

[61] Kurt Konolige. Small vision system: Hardware and implementation. In *International Symposium of Robotics Research (ISRR)*, pages 111–116, 1997.

[62] Labayrade,Raphael and Aubert,Didier and Tarel,Jean-Philippe. Real Time Obstacle Detection in Stereovision on Non Flat Road Geometry Through "V-disparity" Representation. *Intelligent Vehicle Symposium*, 2:646–651, 2002.

[63] Ladicky, Lubor and Sturgess, Paul and Russell, Christopher and Sengupta, Sunando and Bastanlar, Yalin and Clocksin, William F. and Torr, Philip H. S. Joint Optimization for Object Class Segmentation and Dense Stereo Reconstruction. *International Journal of Computer Vision*, 100(2):122–133, 2012.

[64] Leibe, Bastian and Cornelis, Nico and Cornelis, Kurt and Gool, Luc J. Van. Dynamic 3D Scene Analysis from a Moving Vehicle. In *Conference on Computer Vision and Pattern Recognition*, 2007.

[65] Loop, C. and Zhang, Z. Computing rectifying homographies for stereo vision. In *Procs. IEEE Conference on Computer Vision and Pattern Recognition*, pages 125–131, 1999.

[66] Maddox, John. Improving Driving Safety Through Automation. Technical report, National Highway Traffic Safety Administration, 2012.

[67] Meinberg Radio Clocks GmbH. Meinberg NTP Software.
`http://www.meinberg.de/english/sw/ntp.htm`.

[68] Michael Jacoby. Mono Camera Sensor Review Q2 2014. Technical report, Point Grey Research Inc., 2014.

[69] Morales, Sandino and Klette, Reinhard. Spatio-Temporal Stereo Disparity Integration. In Real, Pedro and Diaz-Pernil, Daniel and Molina-Abril, Helena and Berciano, Ainhoa and Kropatsch, Walter, editor, *Computer Analysis of Images and Patterns*, volume 6855 of *Lecture Notes in Computer Science*, pages 540–547. Springer Berlin Heidelberg, 2011.

[70] National Highway Traffic Safety Administration. U.S. Department of Transportation Releases Policy on Automated Vehicle Development.
`http://www.nhtsa.gov/About+NHTSA/Press+Releases/U.S.+Department+of+`
`Transportation+Releases+Policy+on+Automated+Vehicle+Development`.

[71] National Highway Traffic Safety Administration. Traffic Safety Facts. Technical report, National Highway Traffic Safety Administration, 2012.

[72] OpenCV. Open Source Computer Vision Library.
`http://www.opencv.org`.

[73] Oxford University. The RobotCar UK group, Oxford University.
`http://mrg.robots.ox.ac.uk/robotcar`.

[74] Pomerleau, Dean and Jochem, Todd. Rapidly Adapting Machine Vision for Automated Vehicle Steering. *IEEE Expert: Intelligent Systems and Their Applications*, 11(2):19–27, April 1996.

[75] REUTERS. Europe Moves Ahead with Self-Driving Cars in Reponse to Google.
`http://www.insurancejournal.com/news/international/2014/05/19/329448.htm`.

[76] SaberTek. Automotive Radar.
`http://www.sabertek.com/automotive-radar.html`.

[77] Scharstein, Daniel and Szeliski, Richard. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47:7–42, 2002.

[78] Schiehlen, J. and Dickmanns, E.D. A camera platform for intelligent vehicles. In *Int. Symp. on Intelligent Vehicles*, pages 393–398, 1994.

[79] ShareDroid. ShareGPS. Android App for sharing GPS data.
`http://sharedroid.jillybunch.com/user.html`.

[80] Siegwart, Roland and Nourbakhsh, Illah R. and Scaramuzza, Davide. *Introduction to Autonomous Mobile Robots*. MIT press, 2nd edition, 2011.

[81] Sun, Jian and Zheng, Nan-Ning and Shum, Heung-Yeung. Stereo Matching Using Belief Propagation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(7):787–800, Jul 2003.

[82] Szeliski, Richard. *Computer Vision - Algorithms and Applications.* Texts in Computer Science. Springer, 2011.

[83] Terzopoulos, Demetri. Regularization of Inverse Visual Problems Involving Discontinuities. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(4), Jun 1986.

[84] The Guardian. Driverless cars get green light for testing on public roads in UK.
`http://theguardian.com/technology/2014/jul/30/government-driverless-car-self-driving-car`.

[85] The Rawseeds Project. The Rawseeds Project. Available at:
`http://www.rawseeds.org/home`.

[86] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, Kenny Lau, Celia Oakley, Mark Palatucci, Vaughan Pratt, Pascal Stang, Sven Strohband, Cedric Dupont, Lars-Erik Jendrossek, Christian Koelen, Charles Markey, Carlo Rummel, Joe van Niekerk, Eric Jensen, Philippe Alessandrini, Gary Bradski, Bob Davies, Scott Ettinger, Adrian Kaehler, Ara Nefian, and Pamela Mahoney. Stanley: The Robot That Won the DARPA Grand Challenge. *J. Robot. Syst.*, 23(9):661–692, September 2006.

[87] Tsai, Roger Y. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal on Robotics and Automation*, 3(4):323–344, 1987.

[88] University of Auckland. The .enpeda.. Image Sequence Analysis Test Site (EISATS). Available at:
`http://ccv.wordpress.fos.auckland.ac.nz/eisats`.

[89] University of Parma. Public ROad Urban Driverless-Car Test 2013.
`http://vislab.it/proud-en`.

[90] University of Parma. The ARGO project.
`http://www.argo.ce.unipr.it/argo/english/index.html`.

[91] University of Parma. The Artificial Vision and Intelligent Systems Laboratory (VisLab) of Parma University.
`http://vislab.it`.

[92] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, M. N. Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, Michele Gittleman, Sam Harbaugh, Martial Hebert, Thomas M. Howard, Sascha Kolski, Alonzo Kelly, Maxim Likhachev, Matt McNaughton, Nick Miller, Kevin Peterson, Brian Pilnick, Raj Rajkumar, Paul Rybski, Bryan Salesky, Young-Woo Seo, Sanjiv Singh, Jarrod Snider, Anthony Stentz, William "Red" Whittaker, Ziv Wolkowicki, Jason Ziglar, Hong Bae, Thomas Brown, Daniel Demitrish, Bakhtiar Litkouhi, Jim Nickolaou, Varsha Sadekar, Wende Zhang, Joshua Struble, Michael Taylor, Michael Darms, and Dave Ferguson. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.

[93] Weber, Marc. Where to? A History of Autonomous Vehicles.
`http://www.computerhistory.org/atchm/where-to-a-history-of-autonomous-vehicles`.

[94] Wikipedia. Autonomous car.
http://en.wikipedia.org/wiki/Driverless_car.

[95] Wikipedia. DARPA Grand Challenge.
http://en.wikipedia.org/wiki/DARPA_Grand_Challenge.

[96] Wikipedia. Ernst Dickmanns.
http://en.wikipedia.org/wiki/Ernst_Dickmanns.

[97] Wikipedia. Eureka Prometheus Project.
http://en.wikipedia.org/wiki/EUREKA_Prometheus_Project.

[98] Wikipedia. VaMP.
http://en.wikipedia.org/wiki/VaMP.

[99] World Bank. World Bank Data: Motor vehicles (per 1,000 people).
http://data.worldbank.org/indicator/IS.VEH.NVEH.P3.

[100] World Health Organization. World Report On Traffic Injury Prevention. Technical report, World Health Organization, 2004.

[101] World Health Organization. Global Status Report on Road Safety. Technical report, World Health Organization, 2013.

[102] Xiang, Xueqin and Zhang, Mingmin and Li, Guangxia and He, Yuyong and Pan, Zhigeng. Real-time stereo matching based on fast belief propagation. *Machine Vision and Applications*, 23(6):1219–1227, 2012.

[103] Xsens Inc. Xsens MTi-MTi-G LabVIEW driver.
http://sine.ni.com/apps/utf8/niid_web_display.download_page?p_id_guid=
D1248F88E5C82923E0440021287E65E6.

[104] Xsens Technologies B.V. Xsens MTi.
http://www.xsens.com/products/mti/.

[105] Yang, Qingxiong and Wang, Liang and Yang, Ruigang and Wang, Shengnan and Liao, Miao and Nister, David. Real-time Global Stereo Matching Using Hierarchical Belief Propagation. In *BMVC*, volume 6, pages 989–998, 2006.

[106] Yu, Qian and Araujo, Helder and Wang, Hong. Stereo-Vision Based Real time Obstacle Detection for Urban Environments. *International Conference on Advanced Robotics*, pages 1671–1676, 2003.

[107] Zabih, Ramin and Woodfill, John. Non-parametric Local Transforms for Computing Visual Correspondence. In *Proceedings of the Third European Conference on Computer Vision (Vol. II)*, ECCV '94, pages 151–158, 1994.

[108] Zhang, Zhengyou. A Flexible New Technique for Camera Calibration. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(11):1330–1334, 2000.

[109] Ziegler, Julius and Bender, Philipp and Schreiber, Markus and Lategahn, Henning and Strauss, Tobias and Stiller, Christoph and Dang, Thao and Franke, Uwe and Appenrodt, Nils and Keller, Christoph Gustav and Kaus, Eberhard and Herrtwich, Ralf G. and Rabe, Clemens and Pfeiffer, David and Lindner, Frank and Stein, Fridtjof and Erbs, Friedrich and Enzweiler, Markus and Knöppel, Carsten and Hipp, Jochen and Haueis, Martin and Trepte, Maximilian and Brenk,

Carsten and Tamke, Andreas and Ghanaat, Mohammad and Braun, Markus and Joos, Armin and Fritz, Hans and Mock, Horst and Hein, Martin and Zeeb, Eberhard. Making Bertha Drive - An Autonomous Journey on a Historic Route. *IEEE Intell. Transport. Syst. Mag.*, 6(2):8–20, 2014.