STEREO AND COLOUR VISION TECHNIQUES FOR AUTONOMOUS VEHICLE GUIDANCE

Wannes van der Mark

STEREO AND COLOUR VISION TECHNIQUES FOR AUTONOMOUS VEHICLE GUIDANCE

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. J.W. Zwemmer
ten overstaan van een door het college voor
promoties ingestelde commissie, in het openbaar
te verdedigen in de Aula der Universiteit
op dinsdag 5 juni 2007, te 12:00 uur

door

Wannes van der Mark

geboren te Leiderdorp.

Advanced School for Computing and Imaging

This work was carried out in the ASCI graduate school.
ASCI dissertation series number 145.

The investigations were supported by the Netherlands Organisation for Applied Scientific Research (TNO). The research was performed at TNO Defence, Security and Safety, The Hague, The Netherlands and at the Intelligent Systems Laboratorium Amsterdam (ISLA) of the University of Amsterdam, The Netherlands.

ISBN-13: 978-90-5986-242-5

Author E-mail: `wannes.vandermark@tno.nl`

Voor mijn ouders

Contents

Chapter 1

Introduction

1.1 Autonomous mobile robotics

The main argument for the development of robotics technology is that it can be applied to do otherwise undesirable, uneconomical or dangerous tasks. In industries such as car manufacturing robotics has been successfully applied to automate steps that are too tedious or uneconomical to be performed manually. Because they can move around, mobile robots can transport payloads or perform useful tasks on different locations. They are therefore able to execute missions in areas that are either too dangerous or too inhospitable for humans.

Bomb disposal is a well known example of the application of mobile robotics in a dangerous situation. Explosive ordnance disposal squads often use a remotely controlled tracked vehicle that is equipped with a camera and a robotic gripper. By using the remote control, the robot can be moved near suspected explosive devices to inspect and, if necessary, handle them. Because these actions can all be done from a safe distance there is no risk for the robot operator.

Controlling a robot directly from a distance is also known as tele-operation. It requires communication between the controller and the robot. This is often implemented using a wireless radio connection. Unfortunately, tele-operation is not a suitable solution of all possible types of mobile robot missions.

An example of this can be found in space exploration, where robots have been send to investigate celestial bodies such as the planet Mars. When a mobile robot is exploring the planets surface, it must not crash into an obstacle or get stuck. Tele-operation from Earth cannot be used for this purpose, even when robot is driving at a relatively slow speed. This is due to the fact that the time needed to send a message from Earth to the robot on Mars is in the order of ten to twenty minutes. Therefore, any command to stop for an obstacle will not be received in time.

A (former) battlefield is also an example of a dangerous environment. High risk operations such as mine clearance [77], security, reconnaissance and resupply tasks could be performed by mobile robots in order to reduce casualties. Similar to the robots used by the explosive ordnance disposal squads, mine clearance can be achieved by tele-operation. However, the reconnaissance and resupply tasks usually involve travelling over longer distances. Tele-operation is problematic here because the radio signals can be blocked by mountainous terrain or can be disrupted on purpose by radio jamming.

A way around the problems associated with tele-operation is to equip a mobile robot with various degrees of autonomy. Generally, in order to complete a mission autonomously, a robot should be able to plan a safe route through the terrain by itself. This requires that the robot is aware of its position and can avoid obstacles.

Several navigation devices are available that can help a robot to locate its position. Shaft encoders that monitor wheel revolutions are often used to keep track of the driven distance by dead reckoning. Other devices, such as Internal Navigation Systems (INS) use accelerometers and gyroscopes to keep track of the position and orientation. A well known navigation system is the Global Positioning System (GPS). It is based on a constellation of satellites that serve as radio beacons. A robot with a GPS receiver can pin-point its absolute position on the Earth with a certain degree of accuracy.

Obstacle avoidance requires that a robot can detect possible hazards in its environment. The sensor systems that are often used for this goal can be divided into active and passive types. Active sensors emit signals to measure the proximity of obstacle surfaces. Different types of signals can be used for this purpose. Laser Imaging Detection and Ranging (LIDAR) device use laser pulses, while Radio Detection and Ranging (RADAR) is based on radio waves.

Stereo vision is a passive sensing technique for obtaining three-dimensional (3-D) measurements with imaging. It is based on using two or more cameras to observe the same scene. Each camera has an unique viewpoint of the scene because they are separated by a distance. Therefore, one point viewed by both cameras can have a different location in each of the camera images. The disparity between those locations is related to the distance between the point and the cameras. By recovering the disparity of stereo image points, their 3-D position can be estimated.

Sensor data needs to be interpreted before the robot can use it for navigating through an unknown environment. In man-made environments the presence of artificial structures can be exploited. For example, in urban areas a robot vehicle could follow the road infrastructure and obtain important information from traffic signs and lights. Several vision based techniques, suitable for this goal and other automotive applications are described by Franke et al. [24]. These resulted from a multi-year research effort at DaimlerChrysler.

Indoors, there are building floors that facilitate the access of mobile robots. Many indoor robot approaches detect the room walls and try to map them into a floor plan. This can then be used to navigate through the building. A detailed survey of robotic mapping approaches can be found in the work by Thrun [83].

However, there are also environments that do not feature a lot of man-made structures. Places like this are refereed to as “unstructured” or “uncooperative”. For example, there are large outdoor natural areas that do not contain any man-made structures at all. This does not mean that they are not cluttered with obstacles. There can be trees, thick bushes, steep sand dunes or ditches in the ground that can stop or damage the robot.

1.2 Recent developments

In this section we review some recent efforts to improve mobile robot autonomy in unstructured environments. It may not come as a surprise that most of them are related to space and defence applications. Since the end of the 1980's, NASA and JPL [55] have been developing technology for autonomous robot rovers that could explore Mars. The first result of the research was the Pathfinder mission that landed the Sojourner rover [82] on Mars in 1997. It was equipped with several sensors for autonomous operation. These included bumper sensors and a vision system for obstacle detection. It also included rate gyros, wheel revolution and steering position sensors for navigation by dead reckoning.

The Sojourner robot was relatively small and was only operated in the vicinity of the lander module. The technology needed for a more freely roaming vehicle was developed and demonstrated by the six wheeled NOMAD robot [99] in 1997. It was extensively tested in the Atacama desert in northern Chile. It was equipped with GPS, gyrocompass, and wheel encoders for navigation. Furthermore, two pairs of stereo cameras were aboard for both obstacle detection and navigation purposes. It managed to drive 223.5 km over 45 days, while it was partially tele-operated and driven autonomously.

In 2003, NASA sent two robots to Mars that were designed to explore the surface more freely. These robots arrived in January 2004 and were named "Spirit" and "Opportunity". Each robot was equipped with two "HazCam" stereo cameras [50] that are used to detect obstacles in front of or at the rear of the robot. The robots were designed to last 90 days to complete their mission. However, they far outlasted this period and are still operating at the time this is written. All together, they have travelled about 17.5 km on Martian surface in the three year time span.

Initial developments of autonomous ground robots for defence applications took place during the 1990's. In the PRIMUS project [78] conducted by DaimlerChrysler Aerospace, an unmanned robot was developed that can drive autonomously in open terrain. It is equipped with a LIDAR sensor that allows it to avoid obstacles at a speed of up to 25 km/h, while driving off-road. On-road, it uses a camera to follow the road contours and can reach speeds up to 50 km/h.

During the same period, the DEMO I, II and III projects were conducted in the United States. The starting point of the first project was the development of high speed tele-operation and simple autonomous capabilities such as automatic retro-traverse [79]. Technologies for autonomous cross county robot navigation while under the supervision of a human operator were developed and demonstrated in the later projects.

One of the results of DEMO III was a stereo vision system that used both daylight and infra-red image sensors for obstacle detection during day and night time [58, 5].

In order to stimulate the development of robotic technology further, the Defense Advanced Research Projects Agency organised the DARPA Grand Challenge. It called upon teams to develop a robot vehicle that is able to drive off- and on-road along an unknown course that takes it through the Californian dessert. Only unmanned robot vehicles were permitted to enter. During the race, remote control was forbidden and the robots had to follow a course set out by GPS way-points autonomously. The team with the robot that finished first, within a certain time period, would be awarded a price of one million U.S. dollars. The first edition of the Challenge was held on March 13, 2004. Robots from 14 different teams started that day on a course that was about 241 km long. Unfortunately, none of the robots completed the race. The best distance was achieved by the Red Team from of Carnegie Mellon University with their vehicle "Sandstorm". It travelled 11.9 km before it stuck on an obstacle.

Undaunted by the disappointing results, DARPA organised a second edition of the DAPRA Grand Challenge one and a half year later. It was held on October 8, 2005, on a course that was 211 km long. This time, five robots of the 23 teams that participated actually finished the race. The Stanford Racing Team from Stanford University won with their vehicle "Stanley" that finished first in 6 hours and 53 minutes [85]. For obstacle detection, five LIDAR units, a single colour camera and two RADAR sensors were used. Nearby obstacles were detected by the LIDARs, while the camera was used to find drivable road-like surfaces. The RADAR was used to detect large obstacles at further ranges.

1.3 Robotics research at TNO

Different applications of robotics are researched at the Netherlands Organisation for Applied Scientific Research (TNO). Among them is the development of autonomous robot technology for defence and security related tasks in outdoor and unstructured terrain.

The starting point of this development is the "RoboJeep" [89] robotic vehicle. It was build in-house by TNO for research into autonomous navigation through large unstructured outdoor environments. This research is conducted in collaboration with the University of Amsterdam. The RoboJeep has also proved to be a versatile and flexible testbed for experiments with obstacle detection technology, automotive driver assist systems and sensor fusion [90].

Two images of RoboJeep are shown in fig. 1.1. The basis is a Jeep Wrangler 4.0i all-terrain vehicle with an automatic gear box. To enable drive-by-wire it has been extensively modified. Throttle and brake control is provided by actuators that are normally used in vehicles customised for physically disabled drivers. Other actuators provide steering control and select the correct automatic gear box setting. The actuators themselves are controlled by real-time Programmable Logic Controller boards installed in one of two onboard computers. For navigation by dead reckoning, the vehicle has been fitted with sensors that monitor wheel revolutions and the steering angle. An OmniSTAR differential GPS (DGPS) receiver is also installed to provide accurate satellite based navigation.

Several sensor systems have been installed for obstacle detection. A LIDAR made by SICK is mounted on the front of the bull bar. Also, experimental RADAR and stereo vision systems have been installed on several occasions. The research of this thesis is based on image data collected by a stereo camera that is mounted on the RoboJeep.

Another robot, called the "EyeRobot", was developed by TNO and partners in a project for the Dutch armed forces. It is a demonstrator for reconnaissance and surveillance tasks in urban environments. The EyeRobot has three modes of operation; tele-operation, tele-presence and autonomous.

An image of the robot is shown in fig. 1.2. The robot platform type robuROC 4 [72] has been specially built for the project by the Robosoft company. Each of the four wheels is powered by an independent electric motor. This allows the robot to turn around its axis and drive by means of skid steering. Such manoeuvrability is important for operation in urban terrain.

For navigation, a Xsens inertial measurement unit (IMU) is onboard, to measure changes of the robots turn-rate and acceleration. Each wheel is fitted with an odometer to keep track of the driven distance. An OmniSTAR DGPS receiver is also onboard. As a safety measure, the rear and front bumper are sensors that stop the robot on contact. In the autonomous operation mode, obstacle detection is provided by the stereo camera system that is installed on the front of the robot.

The robot observation payload comprises of five cameras, 3-D microphones and a laser-range finder mounted on a pan-and-tilt unit. The set of cameras includes a stereo pair, a zoom camera, a multi-spectral camera and an infrared camera. During the tele-presence operation mode, the pan-and-tilt unit can be operated via a head mounded display, called the "Tele-Presence Binoculars", that tracks head motion. This is done in order to provide the operator with virtual human like vision and sound perception at the location of the robot. Two images of the Tele-Presence Binoculars are shown at the bottom of fig. 1.2.

Because of the high bandwidth data stream from the cameras, a special wireless video link is used to minimise latencies. For tele-operation, the robot is controlled by a low bandwidth radio link that provides a longer communication range.

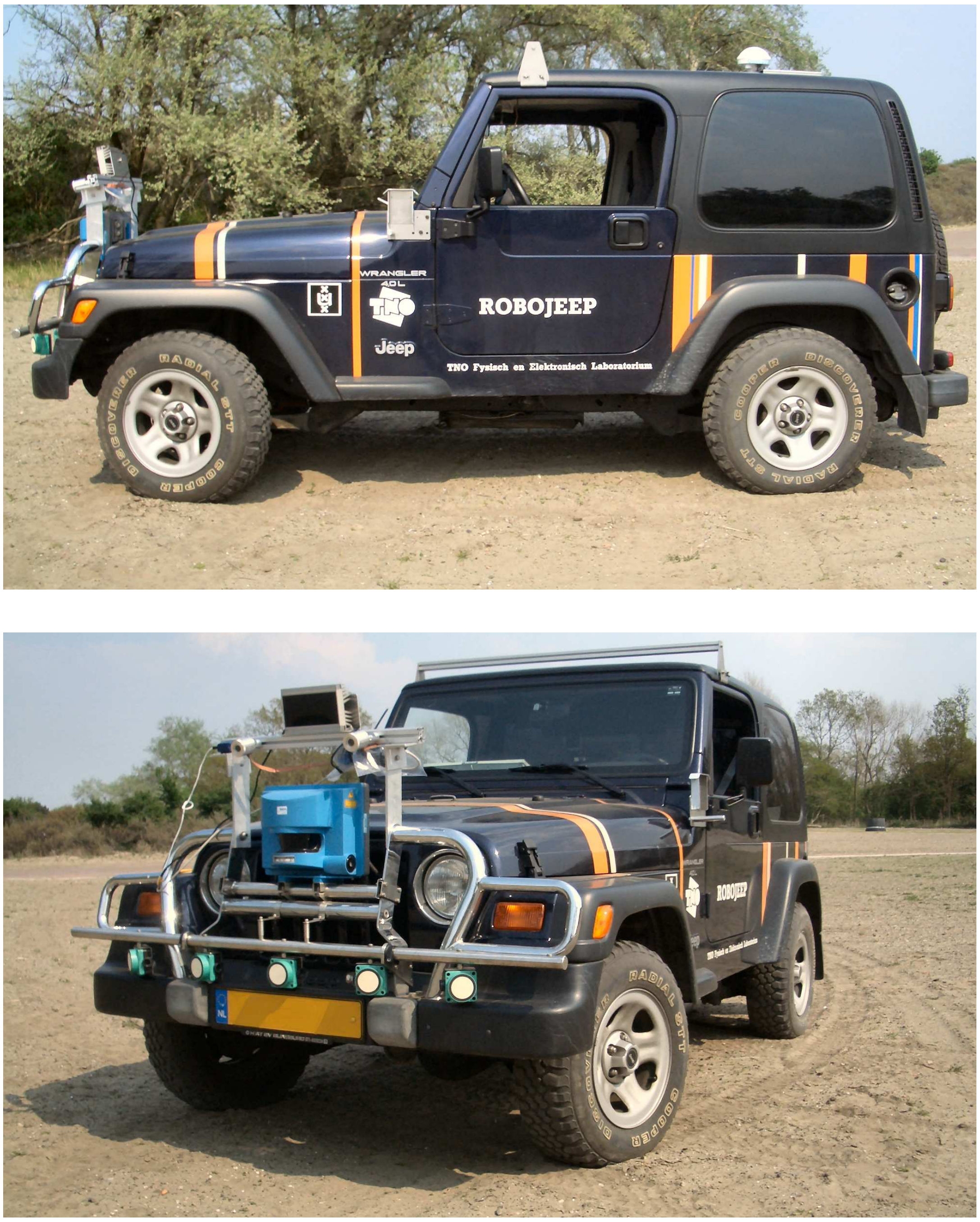


Figure 1.1: The RoboJeep robot vehicle developed by TNO. The top image shows a side view, where the DGPS unit is visible on the roof. In the bottom image, the front is visible with the LIDAR sensor and the stereo camera mounted on the bull bar.

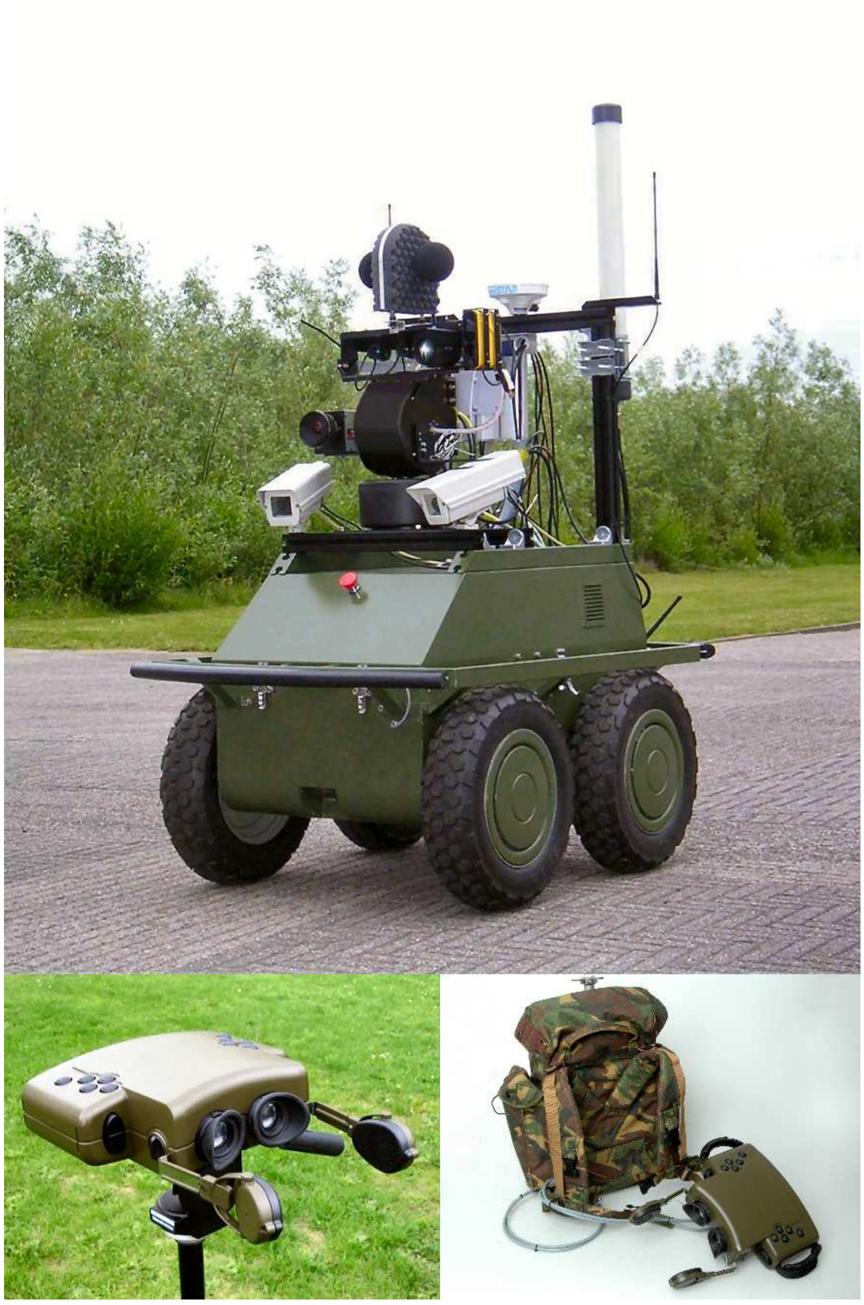


Figure 1.2: The top image shows the EyeRobot mobile robot demonstrator for reconnaissance and surveillance tasks. The two white cameras on the front are a stereo pair that is used for obstacle detection. In between these cameras is a pan-and-tilt unit that houses several cameras and microphones for observation functions. At the bottom, two images are shown of the Tele-Presence Binoculars.

1.4 Research questions

In defence related applications, the emphasis is often on low-observability or “stealth”. This is because an unit cannot be attacked if it is not detected first by the opponent. For example, many defence vehicles are painted in a camouflage pattern to let them blend in with the natural surroundings. However, a position can also be betrayed by emitting detectable signals. For robots that are intended for defence related roles it therefore undesirable to use active sensors, such as LIDAR. It is preferable to use passive sensing techniques, that do not emit detectable signals, instead.

The goal of this research is to use the passive (stereo) vision sensing technique to improve the autonomous capabilities of mobile robots. Several key capabilities have been identified for operation in outdoor terrain. The first one concerns the navigation. Robots often rely on DGPS to locate their position outdoors. Unfortunately, because it relies on radio signals, it is also susceptible to black-outs or interference. This means that good position information from DGPS is not always available. Therefore, readings from the wheel encoders or a INS are necessary to keep track of the robots position. An alternative would be to use stereo vision cameras. Vehicle motion can be estimated from camera image sequences by tracking fixed terrain points. An important question here is whether such methods are sufficiently accurate and robust for precision robot vehicle navigation tasks.

The second capability concerns the obstacle detection. Geometrical information from stereo can be used to distinguish between drivable and inaccessible areas in the terrain. This information should be sufficiently dense and accurate enough in order to detect all possible obstacles at a fair distance from the vehicle. Balancing this with the requirement of real-time processing, leads to finding a trade off between performance and computational cost for dense stereo distance estimation. A question here is how well suited real-time stereo vision methodologies are for obstacle detection. Obstacle detection based on stereo vision also has to deal with the fact that far-away distance estimates are less accurate than those for nearby distances. Another question is therefore; what are the consequences of this uncertainty and how can it be integrated to improve detection results?

Obstacle detection with stereo vision is based on geometric information such as terrain elevation. In natural terrain, not all objects that protrude out of the ground plane are potentially dangerous obstacles. For example; a vehicle can safely drive over tall grass. However, with range only information, tall grass is difficult to distinguish from real obstacles such as bushes. The disadvantage of not being able to recognise materials such as grass is, that the vehicle has to stop for, or drive around a lot of false obstacles. Terrain recognition is also important to recognise drivable areas such as dirt roads. Other features than range are needed to distinguish between the different materials in natural terrain. The question here is whether a vision based terrain type classifier can be developed, that uses features such as image texture and colour.

1.5 Thesis overview

The research questions mentioned above are addressed in this thesis. A short description of each chapter can be found below. In the sixth and final chapter, one will find a summary of the work, final conclusions and future directions for research.

Chapter 2, Stereo vision and Ego-motion estimation

Stereo camera ego-motion estimation is investigated as a means to obtain accurate three-dimensional pose and position estimates for navigation purposes. As a general introduction, the geometry involved in stereo image projection and reconstruction of three-dimensional points from stereo is explained first. We then turn our attention to the problem of estimating the motion of a moving stereo rig relative to fixed points in the environment. There are several methods for stereo ego-motion estimation. They range from a simple least-squares approach to more advanced techniques that include the uncertainty in the tracked feature measurements. Because we are interested in the accuracy of the three-dimensional position and orientation measurements, a novel landmark based technique is presented to obtain a ground truth ego-motion from a real stereo image sequence. This method is used to evaluate both the accuracy and robustness of the ego-motion methods discussed.

Chapter 3, Real-time dense stereo disparity estimation

In the third chapter, we present a framework of real-time dense stereo vision algorithms, that are all based on a Single Instruction Multiple Data (SIMD) architecture. We distinguish different methodical components and examine their performance-speed trade-off. We furthermore compare the resulting algorithmic variations with other publicly available algorithms. The evaluation is based on synthetic image data that depicts realistic urban traffic scenes with real world image perturbations such as lens vignetting and stereo calibration errors. The evaluation itself is not limited to only disparity estimation quality. It also considers the consequences of estimation artifacts for obstacle detection.

Chapter 4, Obstacle detection for unstructured environments

We discuss our approach to stereo based obstacle detection in off-road terrain in the fourth chapter. An efficient approach for detecting and clustering of positive obstacle pixels into objects is presented. It includes a novel method for dealing with the reduced range accuracy of stereo vision at larger distances. Our method also estimates the orientation difference between the cameras and the ground surface to compensate for uneven ground surface geometry. Unlike the majority of published research on the topic of stereo vision based obstacle detection in rough terrain, the evaluation of our method is not limited to qualitative images that demonstrate its capabilities. Several quantitative tests have been devised and conducted in order to evaluate the performance.

Chapter 5, Terrain classification by environment resemblance

Terrain type classification is investigated in the fifth chapter. Using image features, such as colour, for terrain recognition outdoors is problematic. Outdoors there is a large variation in apparent colour due to changes in environment conditions such as illumination. We present a novel approach that can segment a set of training images into smaller sets that contain images that share similar environment conditions. Classification is improved by modelling all the terrain types for each individual set. Experiments are conducted with different combinations of colour, texture and geometric features.

Chapter 2

Stereo Vision and Ego-Motion Estimation

2.1 Introduction

An autonomous mobile robot should be able to navigate successfully through an unknown environment. In off-road terrain, this requires that the robot keeps track of its current position and the direction in which it is heading (yaw-angle). Furthermore, the robot navigation should be sufficiently accurate enough to avoid nearby hazards such as obstacles. In rough terrain it is also important to monitor the roll and pitch angles of the vehicle. This can help to prevent the vehicle from tipping over, while negotiating a dangerous slope.

Several navigation systems and motion sensors can be installed in a mobile robot vehicle. Dead reckoning can be used to keep track of the covered distance as well as the turning angle. This is often based on readings from shaft encoders that monitor wheel revolutions as well as the steering angle. The vehicle inertia can also be used for this purpose. Originally developed for aviation, Internal Navigation Systems (INS) are based on accelerometers and gyroscopes to keep track of position and orientation.

An alternative to dead reckoning is to rely on external beacons. In case of a magnetic compass the naturally occurring Earth magnetic field is used. More advanced systems are based on radio beacons. An example is the Global Positing System (GPS) that consist of a constellation of satellites in order to provide world wide coverage.

There are some drawbacks associated with the listed navigation systems. Systems based on dead reckoning must accumulate motion measurements in order to keep track of position and orientation. Therefore, measurement errors will decrease the global position accuracy of these systems over time. In contrast to these systems, the accuracy of external beacon based systems will be more or less the same over time. Unfortunately, the dependance on external signals also has its drawbacks. The radio signals of GPS can be disrupted by atmospheric conditions or blocked by mountains or tall structures such as buildings. Therefore, an accurate position update is not always available from GPS. There are also systems that combine GPS with INS in order to improve accuracy and reliability. Unfortunately, such systems are costly.

An alternate would be to use computer vision for navigation tasks. In this approach, a camera system is fitted to the vehicle. Fixed reference points in the terrain are detected and tracked over time in its captured images. The changes in camera position and orientation caused by the vehicle motion could then be estimated from the tracked reference points. Camera motion is also referred to as ego-motion. Due to the vehicle's suspension and rough off-road terrain conditions, the ego-motion will have to be recovered for the full six degrees of freedom; three for rotation and three for translation.

As pointed out by Björkman et al. [7], the advantage of stereo ego-motion estimation is that it offers a direct insight in the depth of the scene. Most methods for ego-motion estimation based on stereo, are therefore either reconstruction based or feature tracking approaches. An example of the first type can be found in the work by Zhang [101]. In this approach, stereo is used to obtain reconstructions of the world from different viewpoints. Motion between viewpoints can be estimated by fitting the different reconstructions onto each other.

Only a limited set of world points is used to estimate motion in the feature tracking approach. Numerous examples of this approach are known from literature [7, 51, 59, 66, 73, 91, 95, 94]. All these methods basically consist of three phases for motion estimation between two succeeding frames. In the first phase, a limited number of features is selected in the first stereo pair. Tracking is done in the second phase to determine the new positions of the features in the second stereo pair. Finally, ego-motion is estimated from the displacement of the tracked features.

The main advantage of this approach is the reduced complexity, because only a limited set of features is used. However, the central problem for these methods is the accuracy of motion estimation itself. The motion of a stereo rig has up to six degrees of freedom; three for rotation and three for translation. This three-dimensional (3-D) motion has to be estimated correctly from two-dimensional (2-D) tracked image features. Unfortunately, the tracked feature locations will deviate from their true positions due to measurement errors. The estimation can also be disrupted by outliers. These can be caused by feature mismatches during tracking. Another source can be the presence of objects, such as pedestrians, that have a different motion pattern than the ego-motion. It is therefore important to use estimation techniques that can deliver optimal estimates and are robust to the various noise influences.

In order to keep track of the absolute position and orientation, estimates from ego-motion based approaches have to be combined over time. Therefore, they too suffer from the same drawback as dead-reckoning and INS; a progressive decline in absolute position accuracy due to error accumulation. It order to assess the consequences for vehicle navigation it is important to know how much error is accumulated by ego-motion methods over a certain distance.

In this chapter we investigate several methods for stereo vision based ego-motion estimation. They range from a simple least-squares approach to more advanced techniques that include the uncertainty in the tracked point measurements. Also robust

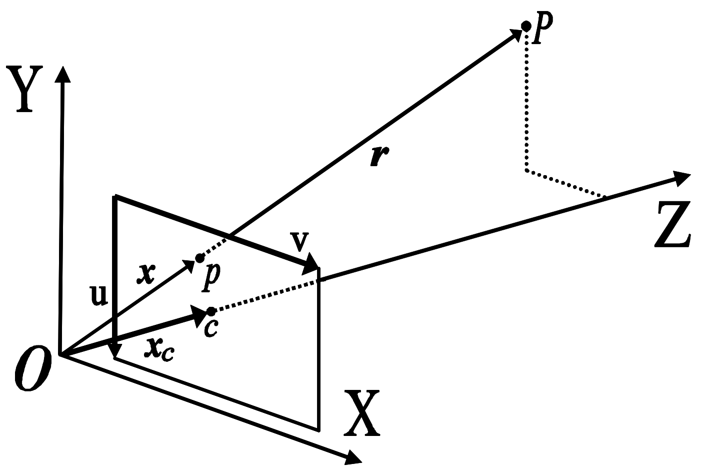


Figure 2.1: Geometric model for pin-hole projection with image in front of the origin O. Point P, indicated by vector $\boldsymbol{r}$, is projected onto the image point p, indicated by vector $\boldsymbol{x}$.

variants are considered that can handle large amounts of outliers. All methods are evaluated on stereo data recorded in off-road terrain. The test sequences contain realistic amounts of outliers. These are caused by both tracking errors and an independently moving pedestrian. A landmark based technique is applied to obtain an outlier free ground truth to compare with the estimator results. Statistical bootstrap analysis is used to analyse its accuracy.

2.2 Geometry

The geometry involved in modelling a moving stereo rig is explained in this section. The starting point is the well known pin-hole model for camera image projection. We continue with explaining the intrinsic parameters needed to transform pixel positions between an ordinary camera and this model. Then we move onto the topic of stereo vision, where the epipolar geometry is covered. Subsequently, we explain how distance can be recovered for matched stereo image points. Finally, we describe the geometry of the ego-motion estimation problem. The geometry used in this chapter is largely based on the methodology described in the book by Kanatani [40]. A clarification of the used notation for vectors and matrices can be found in appendix A. It also contains a definition of the linear algebra operations and their notation.

2.2.1 Pin-hole camera geometry

The pin-hole camera model describes an ideal camera. Its projection does not suffer from arbitrations such as lens distortion. In fig. 2.1 the geometry of a pin-hole model is shown. Its origin *0* is the "hole", where all the projection rays coincide. Note that the image plane, spanned by $\boldsymbol{u}$ and $\boldsymbol{v}$, is located *in front of* the optical centre. This position is preferred, because it prevents image points from becoming mirrored projections of real points.

A vector $\boldsymbol{x}_c$ indicates the focal distance $\|\boldsymbol{x}_c\| = f$ and orientation of the image plane. The vector is perpendicular to the plane and indicates the image optical centre c. For simplicity, $\boldsymbol{x}_c$ is chosen to be located on the Z-axis:

$$\boldsymbol{x}_c = \begin{pmatrix} 0 \\ 0 \\ f \end{pmatrix} . \tag{2.1}$$

The figure also shows the projection of a 3-D point P onto a image point p. The vector $\boldsymbol{x} = (x\, y\, f)^\top$, which indicates image point p, can be obtained from the pin-hole projection of vector $\boldsymbol{r} = (r_X\, r_Y\, r_Z)^\top$, which indicates point P:

$$\boldsymbol{x} = \frac{\boldsymbol{r}}{\boldsymbol{x}_c \cdot \boldsymbol{r}} . \tag{2.2}$$

If the focal length $f = 1.0$, the pin-hole camera is called normalised. The normalised pin-hole projection of $\boldsymbol{r}$ is equal to:

$$\boldsymbol{x} = \frac{\boldsymbol{r}}{\lambda} , \tag{2.3}$$

where the scalar λ is equal to the distance r_Z between the point P and the camera.

2.2.2 Intrinsic camera parameters

The normalised pin-hole camera model does not consider lens distortion and the magnitude of physical properties such as focal length and pixel size. These properties do play a role in the formation of real camera images. Images taken with a normal camera can be transformed in such a way that the projection equals that of the pin-hole model. This transformation requires a set of intrinsic camera parameters.

The first set of these parameters describe the radial and tangential distortions of the camera lens. Fig. 2.2 shows a raster of an undistorted (pin-hole) image of the left type. The middle image shows the effect of radial distortion. It can be seen that points with a larger radial distance $r = \sqrt{x^2 + y^2}$ are more effected by this distortion. In Heikkilä et al. [33], the radial displacement vector $\Delta\boldsymbol{x}_{rad}$ is approximated by:

$$\Delta\boldsymbol{x}_{rad} = \sum_{i=1}^{K} k_i r^{2i} \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} . \tag{2.4}$$

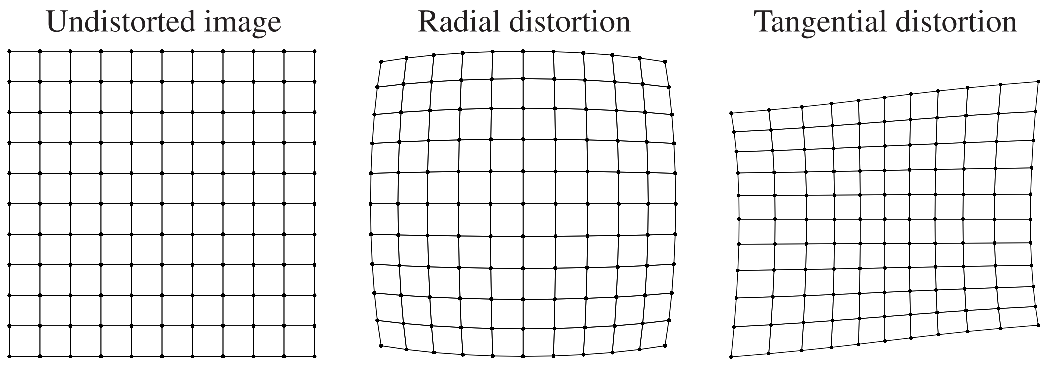


Figure 2.2: The influence of radial and tangential lens distortion effects.

Here, K indicates the number of k_i coefficients that are used to approximate the distortion. For most applications, $K = 3$ is sufficient.

If the lens is asymmetrical, tangential distortion occurs. The effect of this distortion is shown on the right side of fig. 2.2. The displacement vector $\Delta\boldsymbol{x}_{tan}$ caused by this distortion can be modelled by:

$$\Delta\boldsymbol{x}_{tan} = \begin{pmatrix} 2\rho_1 xy + p_2(r^2 + 2x^2) \\ 2\rho_2 xy + p_1(r^2 + 2y^2) \\ 0 \end{pmatrix} \quad . \tag{2.5}$$

The coefficients ρ_1 and ρ_2 determine respectively, the horizontal and vertical tangential distortion. The distortion vectors $\Delta\boldsymbol{x}_{rad}$ and $\Delta\boldsymbol{x}_{tan}$ can be added to the normalised image vector $\boldsymbol{x}$ to form the distorted image vector $\widehat{\boldsymbol{x}}$:

$$\widehat{\boldsymbol{x}} = \boldsymbol{x} + \Delta\boldsymbol{x}_{rad} + \Delta\boldsymbol{x}_{tan} \quad . \tag{2.6}$$

The second set of intrinsic parameters is needed to relate pixel measurements to actual physical distance measures. In order to relate a normalised projection to pixel coordinates in a pin-camera image, a projection matrix $\boldsymbol{P}$ can be used. It describes the transformation needed to map a distorted image vector onto point $\boldsymbol{p}$ in actual image pixel coordinates:

$$\boldsymbol{p} = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} s_u & \gamma & c_u \\ 0 & s_v & c_v \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \widehat{x} \\ \widehat{y} \\ 1 \end{pmatrix} = \boldsymbol{P}\widehat{\boldsymbol{x}} \quad . \tag{2.7}$$

Here, s_u and s_v are ratios between the focal length in pixels and the actual pixel dimensions. If the skew parameter $\gamma \neq 0$, the image axis are non-orthogonal. This allows for cameras where the pixels do not have a rectangular shape or where the chip is not mounted orthogonally to the optical axis. The projection of optical centre in the image is indicated by parameters c_u and c_v. The inverses of Eq. 2.6 and 2.7 allows transformation of image pixel coordinates to normalised pin-hole image vectors.

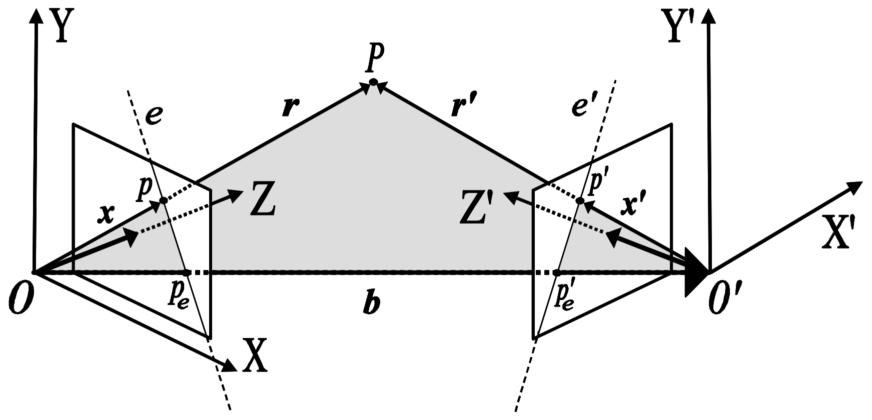


Figure 2.3: Geometry of a stereo camera where both cameras observe a point P. The colour gray is used to indicate the epipolar plane.

Various camera calibration techniques have been developed to estimate intrinsic parameters s_u, s_v, γ, c_u, c_v, $k_{i=1..K}$, ρ_1 and ρ_2 of lens type cameras. The calibration techniques of Heikkilä et al. [33] and Zhang [102] use images taken of planar surfaces with patterns of known dimensions as input. Recently, Zhang [103] also showed that camera calibration is possible with images of simpler one-dimensional (1-D) calibration objects.

2.2.3 Non-parallel stereo camera geometry

A stereo camera consists of two cameras that view the same scene. Fig. 2.3 shows the geometry of two pin-hole cameras that observe that same 3-D point P. This point is indicated by vector $\boldsymbol{r}$ in the left cameras reference frame, while it is indicated by vector $\boldsymbol{r'}$ in the right camera reference frame. The difference in orientation and translation between the cameras is defined by rotation matrix $\boldsymbol{R}$ and vector $\boldsymbol{b}$. The latter is also referred to as the stereo camera baseline. Vector $\boldsymbol{r'}$ can be transformed into vector $\boldsymbol{r}$ with the Euclidean motion:

$$\boldsymbol{r} = \boldsymbol{R}\boldsymbol{r'} + \boldsymbol{b} \ . \tag{2.8}$$

Note that this operation indicates a transformation from the right camera reference frame to that of the left camera.

The plane defined by the point P and the optical centres *0* and *0'* is known as the *epipolar* plane. In the figure, the epipolar surface between the vectors $\boldsymbol{r}$, $\boldsymbol{r'}$ and $\boldsymbol{b}$ is coloured gray. A vector that is perpendicular to this surface can be computed by taking the cross product of $\boldsymbol{x}$ and $\boldsymbol{b}$. If the dot product of the perpendicular vector $\boldsymbol{x} \times \boldsymbol{b}$ with $\boldsymbol{Rx'}$ is equal to zero, the vectors $\boldsymbol{x}$, $\boldsymbol{Rx'}$ and $\boldsymbol{b}$ must be coplanar. This is known as the *epipolar constraint*:

$$(\boldsymbol{x} \times \boldsymbol{b}) \cdot \boldsymbol{Rx'} = (\boldsymbol{b} \times \boldsymbol{Rx'}) \cdot \boldsymbol{x} = (\boldsymbol{Rx'} \times \boldsymbol{x}) \cdot \boldsymbol{b} = 0 \ . \tag{2.9}$$

The point where the line between the two optical centres intersects an image plane is called the *epipole*. By applying pin-hole projection of Eq. 2.2 to $\boldsymbol{b}$ and $\boldsymbol{Rb}$, the left image epipole p_e and the right image epipole p'_e can be found. Together with projection points p and p' they form a left image line e and a right image line e'. These image lines are called *epipolar lines* and correspond to the intersections of the epipolar plane with the image planes. The epipolar lines of an image will always run trough its epipole, regardless of the position of P. However, for images of a parallel stereo camera the epipole is located at infinity.

For ranging applications of stereo vision, the goal is to recover the 3-D location indicated by $\boldsymbol{r}$ for arbitrary image points. A problem here is that the correspondences between points in the left and right images are unknown. For each point $\boldsymbol{p}$ in the left image a corresponding point $\boldsymbol{p'}$ has to be searched in the right image before $\boldsymbol{r}$ can be reconstructed. Fortunately, the search space for $\boldsymbol{p'}$ can be reduced to a single epipolar line if the epipolar geometry is known.

The epipolar line in the left image can be defined by the support vector $\boldsymbol{x}$ and normal vector $\boldsymbol{n}$:

$$\boldsymbol{n} = \frac{\boldsymbol{b} \times \boldsymbol{Rx'}}{||\boldsymbol{b} \times \boldsymbol{Rx'}||} \quad . \tag{2.10}$$

The cross product of rotation matrix $\boldsymbol{R}$ and translation $\boldsymbol{b}$ defines the *Essential*[1] matrix [34] $\boldsymbol{E}$:

$$\boldsymbol{E} = \boldsymbol{b} \times \boldsymbol{R} \quad . \tag{2.11}$$

This matrix is a compact definition of the epipolar constraint. Substitution of $\boldsymbol{E}$ into Eq. 2.9 leads to:

$$\boldsymbol{x} \cdot \boldsymbol{Ex'} = 0 \quad . \tag{2.12}$$

Given a point p, indicated by $\boldsymbol{x}$ in the left image, the essential matrix can be used to compute the normal $\boldsymbol{n'}$ of the epipolar line in the right image.

$$\boldsymbol{n'} = \boldsymbol{E}^\top \boldsymbol{x} \tag{2.13}$$

The vector $\boldsymbol{Rx'}$ of the right image, indicates a point on this line because it lies in the same epipolar plane. Similarly, the epipole normal $\boldsymbol{n}$ in the left image can be computed with the essential matrix for a right image point indicated by vector $\boldsymbol{x'}$.

$$\boldsymbol{n} = \boldsymbol{Ex'} \quad . \tag{2.14}$$

When the corresponding $\boldsymbol{x}$ and $\boldsymbol{x'}$ pairs have been found, space vectors $\boldsymbol{r}$ and $\boldsymbol{r'}$ can be obtained. Because both $\boldsymbol{r}$ and $\boldsymbol{r'}$ indicate the same point in space, the distances λ and λ' to this point can also be related to each other:

$$\boldsymbol{r} = \boldsymbol{Rr'} + \boldsymbol{b} \quad \rightarrow \quad \lambda \boldsymbol{x} = \lambda' \boldsymbol{Rx'} + \boldsymbol{b} \quad . \tag{2.15}$$

[1]The Essential matrix $\boldsymbol{E}$ should not be confused with the *Fundamental* matrix $\boldsymbol{F}$ [22]. The main difference between them is that the geometry behind the Fundamental matrix does not assume that pin-hole projection is normalised: $\boldsymbol{F} = (\boldsymbol{P'}^{-1})^\top \boldsymbol{EP}^{-1}$.

Because vector $\lambda' \boldsymbol{Rx'}$ points towards the same direction of $\boldsymbol{Rx'}$, the cross product of the previous equation with $\boldsymbol{Rx'}$ yields:

$$\lambda \boldsymbol{x} \times \boldsymbol{Rx'} = \boldsymbol{b} \times \boldsymbol{Rx'} \ . \tag{2.16}$$

Therefore, left distance λ can be calculated with the following equation:

$$\lambda = \frac{\|\boldsymbol{b} \times \boldsymbol{Rx'}\|}{\|\boldsymbol{x} \times \boldsymbol{Rx'}\|} \ . \tag{2.17}$$

The right distance λ' can be calculated by taking the cross product of Eq. 2.15 with $\boldsymbol{x}$:

$$\lambda' = \frac{\|\boldsymbol{b} \times \boldsymbol{x}\|}{\|\boldsymbol{x} \times \boldsymbol{Rx'}\|} \ . \tag{2.18}$$

Reconstruction of $\boldsymbol{r}$ or $\boldsymbol{r'}$ is now possible by simply multiplying $\boldsymbol{x}$ or $\boldsymbol{x'}$ with the scalar distance λ or λ' (Eq. 2.3):

$$\boldsymbol{r} = \lambda \boldsymbol{x} \qquad \text{and} \qquad \boldsymbol{r'} = \lambda' \boldsymbol{x'} \ . \tag{2.19}$$

2.2.4 Parallel stereo camera geometry

Fig. 2.4 shows an ideal parallel stereo camera, where both image planes lie in the same plane. The translation between them is limited to the X-axis. Therefore the rotation matrix $\boldsymbol{R}$ is equal to the identity matrix $\boldsymbol{I}$ and translation vector $\boldsymbol{b}$ only contains the scalar b as X value.

$$\boldsymbol{R} = \boldsymbol{I} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \boldsymbol{b} = \begin{pmatrix} b \\ 0 \\ 0 \end{pmatrix} \tag{2.20}$$

Because both image planes are parallel to the translation vector, their epipole points will be infinitely far away in the translation (X-axis) direction. As a consequence, the epipolar lines in both images will coincide. Therefore corresponding left and right points will lie on the same horizontal images lines. In this case, essential matrix $\boldsymbol{E}$ is simply:

$$\boldsymbol{E} = \boldsymbol{b} \times \boldsymbol{I} = b \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} \ . \tag{2.21}$$

It is clear that only the x value of the left vector $\boldsymbol{x} = (x\, y\, 1)^\top$ differs from the x' value of the right vector $\boldsymbol{x'} = (x'\, y'\, 1)^\top$. This difference is known as the *disparity* d:

$$d = x - x' \ . \tag{2.22}$$

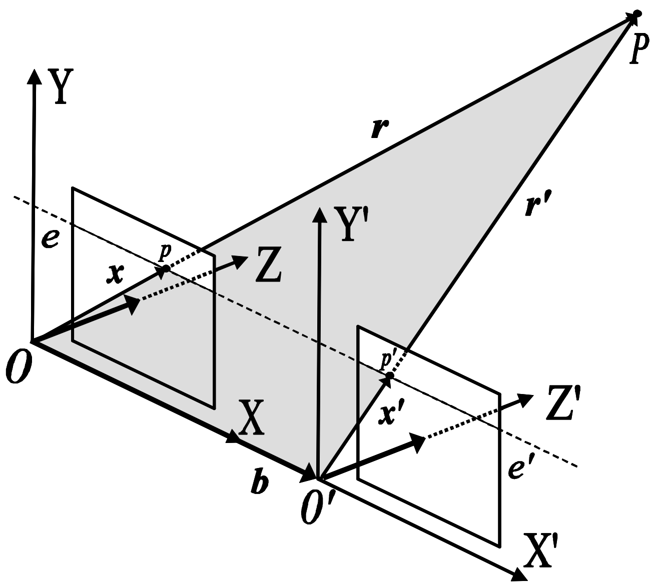


Figure 2.4: Geometry of a rectified stereo camera

The distances λ and λ', needed for stereo reconstruction, are equal to each other because of the parallel camera setup. Given disparity d, their value is easily computed:

$$\lambda = \lambda' = \frac{b}{d} \ . \tag{2.23}$$

The parallel stereo camera configuration has some advantages over other configurations. Because the epipolar lines are parallel with the image lines, the image data is correctly aligned for searching stereo correspondences. Furthermore, depth reconstruction can be achieved by a simple scalar division with the disparity between the matched stereo points. The correct data alignment and the simplified reconstruction computation reduce the computational requirements for the stereo matching and reconstruction.

If the rotation $\boldsymbol{R}$ and the translation $\boldsymbol{b}$ between the cameras of a non-parallel stereo setup are known, their images can be "rectified" to a parallel setup. Techniques for computing the required transformations can be found in the work by Fusiello et al. [27] and Hartley [32].

2.2.5 Euclidean ego-motion geometry

The rotation and translation between stereo cameras have already been introduced as an Euclidean motion. The general function M for applying an Euclidean motion to a point, indicated by vector $\boldsymbol{r}$, to obtain the moved point position indicated by $\boldsymbol{s}$ is:

$$\boldsymbol{s} = M(\boldsymbol{r}, R, \boldsymbol{t}) = R(\boldsymbol{r}) + \boldsymbol{t} \ . \tag{2.24}$$

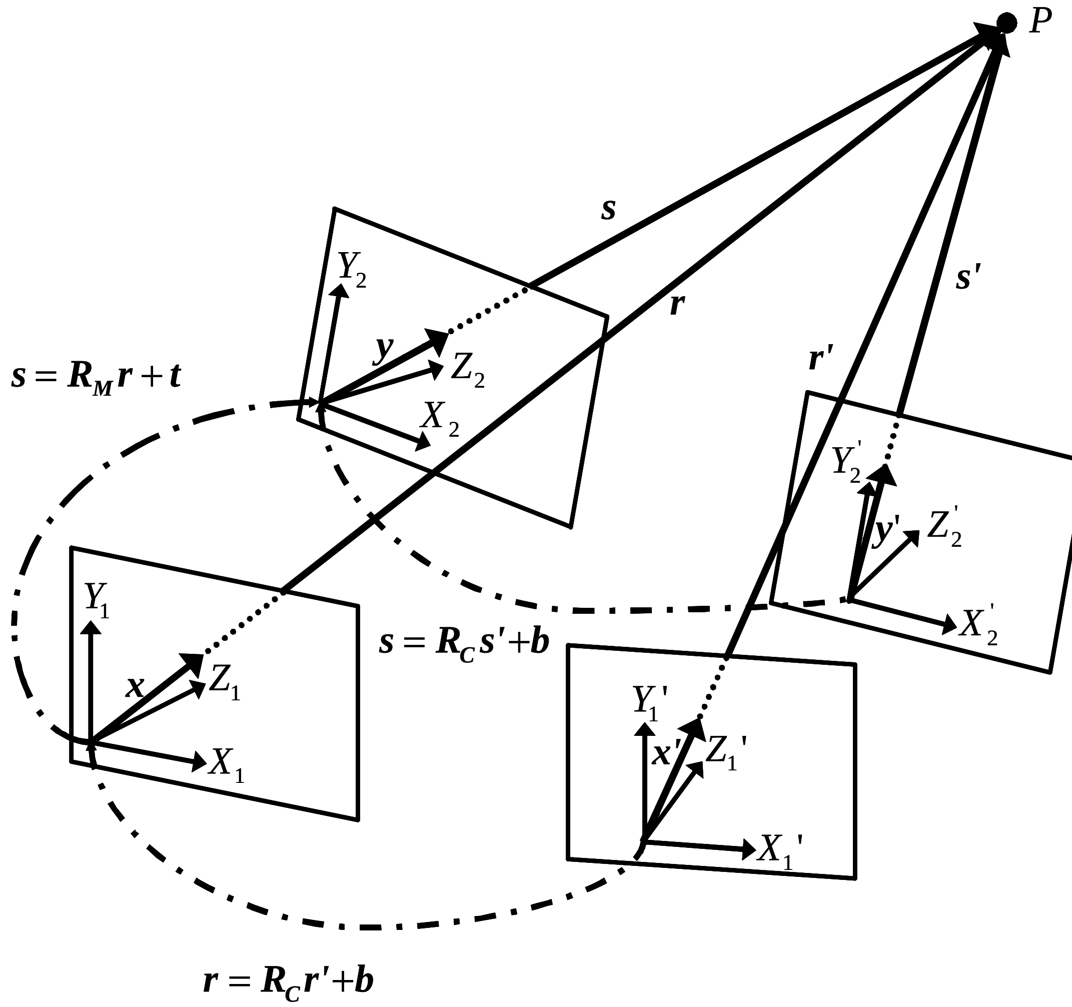


Figure 2.5: Geometry of a moving stereo camera.

The vector $\boldsymbol{t}$ defines the translation. Function R is the general function for rotating a 3-D vector $\boldsymbol{r}$. The function can carry out the rotation by either using Euler angles, a rotation matrix, an unit quaternion or a rotation vector. Quaternions are discussed in appendix B. A good overview of all these rotation representations can be found in the work by Diebel [20].

Suppose that the result of applying function M to $\boldsymbol{r}$ is $\boldsymbol{s}$. Then the inverted Euclidean motion function M^- can be used to transform $\boldsymbol{s}$ back to $\boldsymbol{r}$, by use of the inverted rotation function R^-:

$$\boldsymbol{r} = M^-(\boldsymbol{s}, R, \boldsymbol{t}) = R^-(\boldsymbol{s} - \boldsymbol{t}) \ . \qquad (2.25)$$

The Euclidean motion function can also be used to describe the ego-motion of a stereo camera pair. It is assumed that the cameras of the stereo pair have been mounted rigidly so that the rotation R_C and translation $\boldsymbol{b}$ inside the stereo rig does not change over time. Fig. 2.5 shows a rigid stereo camera before and after a motion through space.

The point P is observed by the stereo pair in both positions. This point is indicated by the space vectors $\boldsymbol{r}$ and $\boldsymbol{r'}$ in the first position and by the vectors $\boldsymbol{s}$ and $\boldsymbol{s'}$ in the second position.

Both vector pairs are separated by the Euclidean motion, defined by rotation R_M and translation $\boldsymbol{t}$:

$$\boldsymbol{s} = M(\boldsymbol{r}, R_M, \boldsymbol{t}) \quad \text{and} \quad \boldsymbol{s'} = M(\boldsymbol{r'}, R_M, \boldsymbol{t}) \ . \tag{2.26}$$

In section 2.4 we will explain several methods for estimating this motion from stereo reconstructed space vectors.

2.3 Uncertainty models for geometry

If a stereo camera pair observes a set of points that undergo a Euclidean motion, this motion could also be recovered from the tracked features. However, due to influences such as the image resolution and sensor noise, the tracked feature locations will contain inaccuracies. It is therefore important to understand how these errors propagate in estimation problems such as stereo reconstruction and ego-motion estimation. A statistical model can be used to model and study the effects of the noise influences.

2.3.1 Gaussian noise distributions

Assume the general case of a n-dimensional vector that indicates a point. If the vector indicates the precise unperturbed position of the point it is indicated with $\hat{\boldsymbol{x}}$. A perturbation from this position is indicated with the error vector $\Delta\boldsymbol{x}$. Vector $\boldsymbol{x}$ now indicates the perturbed (measured) position of the point:

$$\boldsymbol{x} = \hat{\boldsymbol{x}} + \Delta\boldsymbol{x} \ . \tag{2.27}$$

It is clear that when $\Delta\boldsymbol{x}$ is known, $\boldsymbol{x}$ can be optimally corrected into $\hat{\boldsymbol{x}}$. If $\Delta\boldsymbol{x}$ is the same for each observed $\boldsymbol{x}$, the error is said to be systematic. However, errors caused by influences such as noise are random. In this situation $\Delta\boldsymbol{x}$ is a random variable with zero mean. Therefore, $\boldsymbol{x}$ is also a random variable with mean $\hat{\boldsymbol{x}}$. We define that the expectation for $\boldsymbol{x}$ is $\hat{\boldsymbol{x}}$:

$$E[\boldsymbol{x}] = \hat{\boldsymbol{x}} \ . \tag{2.28}$$

If it is assumed that the distribution of $\boldsymbol{x}$ is Gaussian, than its multivariate probability density function is:

$$\phi(\boldsymbol{x}, \hat{\boldsymbol{x}}, \boldsymbol{C}[\boldsymbol{x}]) = \frac{\exp\left(-\frac{1}{2}(\boldsymbol{x}-\hat{\boldsymbol{x}})^\top C[\boldsymbol{x}]^{-1}(\boldsymbol{x}-\hat{\boldsymbol{x}})\right)}{(2\pi)^{n/2}\sqrt{\det(\boldsymbol{C}[\boldsymbol{x}])}} \ . \tag{2.29}$$

Here, $\boldsymbol{C}[\boldsymbol{x}]$ is the covariance of the random variable vector $\Delta\boldsymbol{x}$. The probability density of $\phi(\boldsymbol{x}, \hat{\boldsymbol{x}}, \boldsymbol{C}[\boldsymbol{x}])$ is also referred to as the *likelihood* of $\boldsymbol{x}$.

An indication of how well $\boldsymbol{x}$ statistically coincides with the optimum $\hat{\boldsymbol{x}}$ can be obtained by calculating their Mahalanobis distance:

$$d_M(\boldsymbol{x},\hat{\boldsymbol{x}}) = \sqrt{(\boldsymbol{x}-\hat{\boldsymbol{x}}) \cdot \boldsymbol{C}[\hat{\boldsymbol{x}}]^- (\boldsymbol{x}-\hat{\boldsymbol{x}})} \ . \tag{2.30}$$

In some geometric representations, such as homogeneous coordinates, the associated covariance matrices are not of full rank. The inverse of the covariance matrix ($\boldsymbol{C}[\hat{\boldsymbol{x}}]^-$) is therefore computed as the Moore-Penrose generalised inverse based on Single Value Decomposition [70].

A constant c, equal to the squared Mahalanobis distance $c = d_M(\boldsymbol{x},\hat{\boldsymbol{x}})^2$, corresponds with a surface in the n-dimensional space where the probability density is equal. A vector $\boldsymbol{x}$ located on this surface satisfies:

$$(\boldsymbol{x}-\hat{\boldsymbol{x}}) \cdot \boldsymbol{C}[\boldsymbol{x}]^-(\boldsymbol{x}-\hat{\boldsymbol{x}}) = c \ . \tag{2.31}$$

The shape of this equiprobable surface is defined by the covariance matrix. The covariance matrix can be decomposed into its eigenvectors $\boldsymbol{u}_1$, $\boldsymbol{u}_2$, $\boldsymbol{u}_3$ and the associated eigenvalues σ_1, σ_2, σ_3:

$$\boldsymbol{C}[\boldsymbol{x}] = \sum_{i=1}^{n} \sigma_i \boldsymbol{u}_i \boldsymbol{u}_i^\top \quad \text{where} \quad \sigma_1 \geq \sigma_2 \geq \sigma_3 \geq 0 \ . \tag{2.32}$$

Because the eigenvectors are unit length and orthogonal, they can be combined to form a rotation matrix. This rotation defines the orientation of the equiprobable surface. The eigenvalues define the elongation of the surface along the direction of the eigenvectors. If $\sigma_1 = \sigma_2 = \sigma_3$, the shape is a n-dimensional sphere and the distribution is called *isotropic*. In this case, the error $\Delta\boldsymbol{x}$ occurs equally in every direction. If the eigenvalues are not equal, the shape is a n-dimensional ellipsoid and the distribution is called *anisotropic*. Here, there are more errors in a specific direction than in the other directions.

The surface envelops a region for which there is a certain probability that vector $\boldsymbol{x}$ is located within it. The region enveloped when $c = 1$, is called the standard confidence region. In two dimensions, the standard region becomes an area enclosed by a circle in case of isotropic noise or an ellipse in case of anisotropic noise. For a single dimension, the standard region is the interval bound by the standard deviation.

Because estimation problems involve more than one point, the standard region at each point can be used to visualise the effects of different noise types. It is assumed that the errors of the points are uncorrelated and therefore independently distributed. Using the nomenclature from Kanazawa et al. [41], noise is called *isotropic homogeneous* if the amount of noise is equal for each point and its distribution is isotropic. This case is known more generally as *independently and identically distributed* (i.i.d.) noise. The resulting standard regions for several points are shown in fig. 2.6a. If the distribution remains isotropic and the amount of noise varies for each point, the noise is called *isotropic inhomogeneous*, which is shown in fig. 2.6b. The noise distribution can also be anisotropic. The standard regions of *anisotropic homogeneous* noise are shown in fig. 2.6c, while those of *anisotropic inhomogeneous* noise are shown in fig. 2.6d.

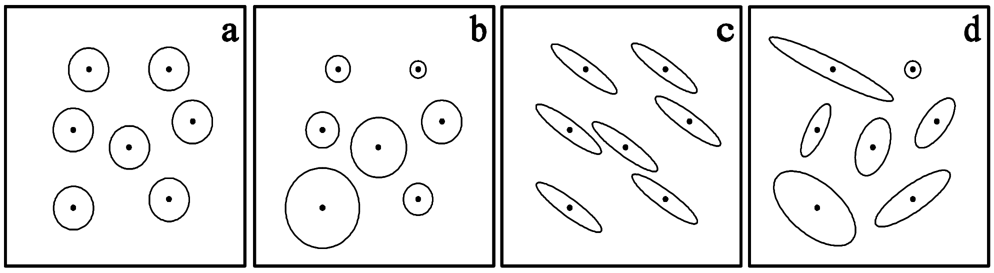


Figure 2.6: The four images show the standard confidence regions of different noise models; a) isotropic homogeneous, b) isotropic inhomogeneous, c) anisotropic homogeneous, d) anisotropic inhomogeneous.

2.3.2 Optimal correction theory

Often, the magnitude of the error in image position is not known or cannot be determined exactly a priori. However, for features such as corners and edges it is possible to measure the covariance in 2D image position up to a scale. For corners, the standard deviation ellipse of the covariance matrix is mostly isotropic. And for edge features the ellipse is elongated along the length of the edge.

In many geometry problems, the solution is based on a set of data points that must satisfy a constraint. A simple example of this is the optimal correction of a point to a line l that passes through the origin. The point is indicated by the 2-D vector $\boldsymbol{x}$ and the normal of the line l is indicated by the 2-D unit vector $\boldsymbol{n}$. Because the point should lie on the line, its correct position, indicated by vector $\hat{\boldsymbol{x}}$, should satisfy the constraint:

$$\boldsymbol{n} \cdot \hat{\boldsymbol{x}} = 0 \ . \tag{2.33}$$

This situation is shown on the left side of fig. 2.7. The vector that optimally corrects $\boldsymbol{x}$ into $\hat{\boldsymbol{x}}$ is indicated by $\Delta\boldsymbol{x}$. Suppose that the covariance of $\boldsymbol{x}$ is the matrix $\boldsymbol{C}[\boldsymbol{x}]$. The vector $\boldsymbol{x}$ that maximises Eq. 2.29 is called the *maximum likelihood estimate*. This can be achieved by finding the shortest Mahalanobis distance for $\Delta\boldsymbol{x}$. If $\Delta\boldsymbol{x}$ is a random variable in Eq. 2.31, the constant value c will become the residual J:

$$J = \Delta\boldsymbol{x} \cdot \left(\boldsymbol{C}[\boldsymbol{x}]^{-1}\Delta\boldsymbol{x}\right) \ . \tag{2.34}$$

By minimising the residual J, the maximum likelihood estimate can be obtained.

The covariance matrix can be decomposed into a scalar ε^2 which represents the variance and a normalised covariance matrix $\boldsymbol{C}_0[\boldsymbol{x}]$:

$$\boldsymbol{C}[\boldsymbol{x}] = \varepsilon^2 \boldsymbol{C}_0[\boldsymbol{x}] \ . \tag{2.35}$$

If the noise is isotropic, $\boldsymbol{C}_0$ can be chosen to be equal to the identity matrix $\boldsymbol{I}$. The residual computed with this matrix is equal to:

$$J = \Delta\boldsymbol{x} \cdot \left(\frac{1}{\varepsilon^2}\boldsymbol{I}\Delta\boldsymbol{x}\right) = \frac{\Delta\boldsymbol{x} \cdot \Delta\boldsymbol{x}}{\varepsilon^2} = \frac{||\Delta\boldsymbol{x}||}{\varepsilon} \ . \tag{2.36}$$

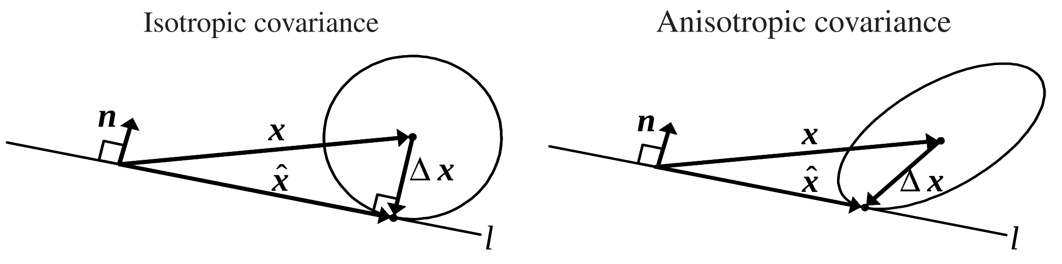


Figure 2.7: Two examples of correcting a point to a line. On the left the correction is shown in case of isotropic covariance. The example on the right shows the correction with anisotropic covariance.

Therefore, in isotropic noise conditions the residual is equal to the Euclidean distance divided by ε. Value ε is also referred to as the noise level. In this case, the optimal correction vector $\Delta \boldsymbol{x}$ is the orthogonal shortest vector from $\boldsymbol{x}$ to the line. The correction vector can be computed with:

$$\Delta \boldsymbol{x} = (\boldsymbol{n} \cdot \boldsymbol{x})\boldsymbol{n} \ . \tag{2.37}$$

On the right of fig. 2.7, the same estimation problem is shown for a situation where the noise is anisotropic. The shortest distance is now governed by the anisotropic covariance. In his book, Kanatani [40] explains a general approach to geometric minimisation with the Mahalanobis distance. Using this approach, a first order approximation of $\Delta \boldsymbol{x}$ can be obtained with the following equation:

$$\Delta \boldsymbol{x} = \frac{(\boldsymbol{n} \cdot \boldsymbol{x})\boldsymbol{C}[\boldsymbol{x}]\boldsymbol{n}}{\boldsymbol{n} \cdot (\boldsymbol{C}[\boldsymbol{x}]\boldsymbol{n})} \ . \tag{2.38}$$

Note that this equation is invariant to scalar multiplication of the covariance matrix. If the scalar is defined by ε^2, the normalised covariance matrix $\boldsymbol{C}_0[\boldsymbol{x}]$ is equal to:

$$\boldsymbol{C}_0[\boldsymbol{x}] = \frac{\boldsymbol{C}[\boldsymbol{x}]}{\varepsilon^2} \ . \tag{2.39}$$

This means that the covariance matrix $\boldsymbol{C}[\boldsymbol{x}]$ only has to be known up to a scale ε^2 in order to correct $\boldsymbol{x}$.

2.3.3 The renormalisation method for optimal estimation

Geometric models for stereo vision and ego-motion have been explained in earlier sections. The problem of obtaining model parameters from noisy data is called data-fitting. An example is the estimation of a 2-D line from multiple noisy data points. For simplicity, it is assumed that the line passes through the origin.

Given a set of N image points, indicated by vectors $\boldsymbol{x}_i$, we want to estimate the direction of the line. If the direction is defined by a normal vector, we only have to obtain the unit vector $\boldsymbol{n}$ that perpendicular to the direction of the line. If the points $\hat{\boldsymbol{x}}_i$ are without any errors they will reside on the line:

$$\sum_{i=1}^{N} \boldsymbol{n} \cdot \hat{\boldsymbol{x}}_i = 0 \ . \tag{2.40}$$

If noise is present, the vectors $\boldsymbol{x}_i$ are the result of a perturbation of $\hat{\boldsymbol{x}}_i$ by the error $\Delta\boldsymbol{x}_i$:

$$\boldsymbol{x}_i = \hat{\boldsymbol{x}}_i + \Delta\boldsymbol{x}_i \ . \tag{2.41}$$

It is assumed that $\Delta\boldsymbol{x}_i$ is a Gaussian random variable with mean 0. The covariance of each point is indicated by $\boldsymbol{C}[\boldsymbol{x}_i]$. In the case that the absolute covariances are available, Kanatani [40] proposes the minimisation of the following residual J in order to find the optimal estimate for $\boldsymbol{n}$:

$$J = \sum_{i=1}^{N} \frac{(\boldsymbol{n} \cdot \boldsymbol{x}_i)^2}{\boldsymbol{n} \cdot (\boldsymbol{C}[\boldsymbol{x}_i]\boldsymbol{n})} \ . \tag{2.42}$$

Often, the absolute values of the point covariances are not available before the estimation process. Just like the example of the previous section, it is more likely that the covariance is only known up to scale. In this case the normalised covariance matrix $\boldsymbol{C}_0$ and the noise level ε can be used:

$$\boldsymbol{C}[\boldsymbol{x}_i] = \varepsilon^2 \boldsymbol{C}_0[\boldsymbol{x}_i] \ . \tag{2.43}$$

The above fitting problem can now be approximated by a least-squares solution. This solution uses a moment matrix $\boldsymbol{M}$:

$$\boldsymbol{M} = \frac{1}{N}\sum_{i=1}^{N} w_i \boldsymbol{x}_i \boldsymbol{x}_i^\top \quad \text{with weights} \quad w_i = \frac{1}{\boldsymbol{n}^* \cdot (\boldsymbol{C}_0[\boldsymbol{x}_i]\boldsymbol{n}^*)} \ , \tag{2.44}$$

where $\boldsymbol{n}^*$ is an approximate estimate of $\hat{\boldsymbol{n}}$. The solution should be found by minimising:

$$J = \boldsymbol{n} \cdot (\boldsymbol{M}\boldsymbol{n}) \ . \tag{2.45}$$

Let $\boldsymbol{u}_1..\boldsymbol{u}_N$ be the eigenvectors and be the corresponding eigenvalues $\sigma_1..\sigma_N$ of the matrix $\boldsymbol{M}$. The best estimate is equal to the unit eigenvector with the smallest eigenvalue σ_i. The expectation E of matrix $\boldsymbol{M}$ is equal to:

$$E[\boldsymbol{M}] = \frac{1}{N}\sum_{i=1}^{N} w_i E\left[(\hat{\boldsymbol{x}}_i + \Delta\boldsymbol{x}_i)(\hat{\boldsymbol{x}}_i + \Delta\boldsymbol{x}_i)^\top\right] = \frac{1}{N}\sum_{i=1}^{N} w_i E[\boldsymbol{x}_i \boldsymbol{x}_i^\top] \ . \tag{2.46}$$

It is clear that the expectation of the moment matrix $\boldsymbol{M}$ is perturbed from matrix $\hat{\boldsymbol{M}}$, which is based on the real point locations, by a matrix $\boldsymbol{N}$ with scalar ε^2:

$$E[\boldsymbol{M}] = \hat{\boldsymbol{M}} + \varepsilon^2 \boldsymbol{N} \quad \text{with} \quad \boldsymbol{N} = \frac{1}{N}\sum_{i=1}^{N} w_i \boldsymbol{C}_0[\boldsymbol{x}_i] \ . \tag{2.47}$$

If $\boldsymbol{u'}_1..\boldsymbol{u'}_N$ and $\sigma'_1..\sigma'_N$ are the eigenvectors and values of $E[\boldsymbol{M}]$ and $\hat{\boldsymbol{u}}_1..\hat{\boldsymbol{u}}_N$ as well as $\hat{\sigma}_1..\hat{\sigma}_N$ are those of $\hat{\boldsymbol{M}}$, the perturbation theory can be applied:

$$\begin{aligned} \sigma'_i &= \hat{\sigma}_i + \varepsilon^2 (\hat{\boldsymbol{u}}_i \cdot \boldsymbol{N}\hat{\boldsymbol{u}}_i) + O\left(\varepsilon^2\right) \ , \\ \boldsymbol{u'}_i &= \hat{\boldsymbol{u}}_i + \varepsilon^2 \sum_{i \neq j} \delta_{i,j} \frac{(\hat{\boldsymbol{u}}_j \cdot \boldsymbol{N}\hat{\boldsymbol{u}}_i)\hat{\boldsymbol{u}}_j}{\hat{\sigma}_i - \hat{\sigma}_j} + O\left(\varepsilon^2\right) \ . \end{aligned} \tag{2.48}$$

Here, $O\left(\varepsilon^2\right)$ stands for terms with the same order or higher than ε^2. In spite of the used weights, the estimate for $\boldsymbol{n}$ will be statically biased by $O\left(\varepsilon^2\right)$, while the solution to $\hat{\boldsymbol{M}}\boldsymbol{n}$ should be 0, because $\boldsymbol{n}$ is the unit eigenvector with eigenvalue 0 for matrix $\hat{\boldsymbol{M}}$.

Kanatani [40] proposes to use his “renormalisation” method for correcting these types of statistical bias. It uses matrix $\boldsymbol{N}$ for compensating the bias. With this matrix an unbiased moment matrix $\tilde{\boldsymbol{M}}$ can be defined as:

$$\tilde{\boldsymbol{M}} = \boldsymbol{M} - \varepsilon^2 \boldsymbol{N} \ . \tag{2.49}$$

This matrix does have an expectation with $E[\tilde{\boldsymbol{M}}] = \hat{\boldsymbol{M}}$. The unbiased least-squares estimator is now equal to minimising:

$$\tilde{J} = \boldsymbol{n} \cdot \left(\tilde{\boldsymbol{M}}\boldsymbol{n}\right) \ . \tag{2.50}$$

If the noise level ε is known, the solution is the unit eigenvector of $\tilde{\boldsymbol{M}}$ with the smallest eigenvalue. The following iterative scheme can be used to estimate $\tilde{\boldsymbol{M}}$ if the noise level is unknown:

1. Initialise $s = 0$ and $w_i = 1$ for all $i = 1..N$.
2. Calculate matrices $\boldsymbol{M}$ and $\boldsymbol{N}$.
3. Find the eigenvector $\boldsymbol{n}$ of $\tilde{\boldsymbol{M}} = \boldsymbol{M} - s\boldsymbol{N}$ with the smallest eigenvalue σ.
4. Stop the iteration, if the eigenvalue σ is close to 0. If not, update the values of c and w_i for all $i = 1..N$ with:

$$s = s + \frac{\sigma}{\boldsymbol{n} \cdot (\boldsymbol{N}\boldsymbol{n})} \quad \text{and} \quad \mathrm{w_i} = \frac{1}{\boldsymbol{n} \cdot (\boldsymbol{C}_0[\boldsymbol{x}_\mathrm{i}]\boldsymbol{n})} \ . \tag{2.51}$$

5. Goto step 2.

2.3.4 Optimal correction applied to stereo vision

The geometrical correction technique, discussed in section 2.3.2, can also be applied to the problem of stereo reconstruction. Kanatani [40] proposes the following optimal correction method for this purpose.

It is assumed that for a point pair the noisy image locations $\boldsymbol{x}$ and $\boldsymbol{x'}$ are known. These deviate from the true image locations $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{x'}}$ by $\Delta\boldsymbol{x}$ and $\Delta\boldsymbol{x}'$. This can be expanded to cover N point pairs:

$$\boldsymbol{x}_i = \hat{\boldsymbol{x}}_i + \Delta\boldsymbol{x}_i \ , \qquad \boldsymbol{x'}_i = \hat{\boldsymbol{x'}}_i + \Delta\boldsymbol{x'}_i \quad \text{for all} \quad i = 1..N \ . \tag{2.52}$$

The uncertainty in the positions $\boldsymbol{x}_i$ and $\boldsymbol{x'}_i$ is indicated by the covariance matrices $\boldsymbol{C}[\boldsymbol{x}_i]$ and $\boldsymbol{C}[\boldsymbol{x'}_i]$.

$$\boldsymbol{C}[\boldsymbol{x}_i] = E[\Delta\boldsymbol{x}_i\Delta\boldsymbol{x}_i^\top] \qquad \boldsymbol{C}[\boldsymbol{x'}_i] = E[\Delta\boldsymbol{x'}_i\Delta\boldsymbol{x'}_i^\top] \tag{2.53}$$

According to the epipolar constraint of Eq. 2.12, the true image location vectors satisfy:

$$\sum_{i=1}^{N} \hat{\boldsymbol{x}}_i \cdot \boldsymbol{E}\hat{\boldsymbol{x'}}_i = 0 \ . \tag{2.54}$$

In order to understand this constraint a space is defined where all possible instances of $\boldsymbol{x}_i$ and $\boldsymbol{x'}_i$ exist. Each $\boldsymbol{x}_i$ and $\boldsymbol{x'}_i$ pair is one vector $\boldsymbol{p}_i$ in this four-dimensional (4-D) space. Because of the epipolar constraint the space contains a 3-D manifold Ψ. Vectors that belong to this manifold satisfy the epipolar constraint:

$$\forall \hat{\boldsymbol{p}}_i \in \Psi, \hat{\boldsymbol{p}}_i = \left(x_i\, y_i\, x'_i\, y'_i\right)^\top \rightarrow (x_i\, y_i\, 1)^\top \cdot \boldsymbol{E} \left(x'_i\, y'_i\, 1\right)^\top = 0 \ . \tag{2.55}$$

The optimal correction for $\boldsymbol{x}_i$ and $\boldsymbol{x'}_i$ is a translation in the direction of the shortest Mahalanobis tangential distance to this manifold. This corresponds to the minimisation of the following residual:

$$J_i = \Delta\boldsymbol{x}_i \cdot (\boldsymbol{C}[\boldsymbol{x}_i]^- \Delta\boldsymbol{x}_i) + \Delta\boldsymbol{x'}_i \cdot (\boldsymbol{C}[\boldsymbol{x'}_i]^- \Delta\boldsymbol{x'}_i) \ . \tag{2.56}$$

In most situations, the covariances matrices $\boldsymbol{C}[\boldsymbol{x}_i]$ and $\boldsymbol{C}[\boldsymbol{x'}_i]$ can only be determined up to an unknown scale. For stereo, it can be assumed that the noise level ε_i in both cameras is the same, while each left and right point has an individual normalised covariance matrix $\boldsymbol{C}_0[\boldsymbol{x}_i]$ and $\boldsymbol{C}_0[\boldsymbol{x'}_i]$:

$$\boldsymbol{C}[\boldsymbol{x}_i] = \varepsilon_i^2 \boldsymbol{C}_0[\boldsymbol{x}_i] \ , \qquad \boldsymbol{C}[\boldsymbol{x'}_i] = \varepsilon_i^2 \boldsymbol{C}_0[\boldsymbol{x'}_i] \ . \tag{2.57}$$

Kanatani [40] proposes the following first order approximation of $\Delta\boldsymbol{x}_i$ and $\Delta\boldsymbol{x'}$ in order to correct $\boldsymbol{x}_i$ and $\boldsymbol{x'}_i$ to the epipolar lines:

$$\Delta\boldsymbol{x}_i = \frac{1}{\alpha}(\boldsymbol{x}_i \cdot \boldsymbol{E}\boldsymbol{x'}_i)\boldsymbol{C}_0[\boldsymbol{x}_i]\boldsymbol{E}\boldsymbol{x'}_i \quad \text{and} \quad \Delta\boldsymbol{x'}_{\mathrm{i}} = \frac{1}{\alpha}(\boldsymbol{x}_{\mathrm{i}} \cdot \boldsymbol{E}\boldsymbol{x'}_{\mathrm{i}})\boldsymbol{C}_0[\boldsymbol{x'}_{\mathrm{i}}]\boldsymbol{E}^\top\boldsymbol{x}_{\mathrm{i}}$$

$$\text{with:} \quad \alpha = \boldsymbol{x'}_{\mathrm{i}} \cdot (\boldsymbol{E}^\top\boldsymbol{C}_0[\boldsymbol{x}_{\mathrm{i}}]\boldsymbol{E}\boldsymbol{x'}_{\mathrm{i}}) + \boldsymbol{x}_{\mathrm{i}} \cdot (\boldsymbol{E}\boldsymbol{C}_0[\boldsymbol{x'}_{\mathrm{i}}]\boldsymbol{E}^\top\boldsymbol{x}_{\mathrm{i}}) \ . \tag{2.58}$$

Because the correction is only an approximation, it needs to be repeated for $\Delta\boldsymbol{x}_i$ and $\Delta\boldsymbol{x'}_i$ until the epipolar constraint of Eq. 2.54 is approximately zero. After convergence, the vectors found can be used to compute the corrected image positions $\tilde{\boldsymbol{x}}_i$ and $\tilde{\boldsymbol{x'}}_i$.

$$\tilde{\boldsymbol{x}}_i = \boldsymbol{x}_i - \Delta\boldsymbol{x}_i \quad \text{and} \quad \tilde{\boldsymbol{x'}}_i = \boldsymbol{x'}_i - \Delta\boldsymbol{x'}_i \tag{2.59}$$

By using the method of Kanatani [40] the residual of Eq. 2.56 can be rewritten to include the covariances $\boldsymbol{C}[\boldsymbol{x}_i]$ and $\boldsymbol{C}[\boldsymbol{x'}_i]$, which can be substituted by $\boldsymbol{C}_0[\boldsymbol{x}_i]$ and $\boldsymbol{C}_0[\boldsymbol{x'}_i]$ from Eq. 2.57:

$$\tilde{J}_i = \frac{(\boldsymbol{x}_i \cdot \boldsymbol{E}\boldsymbol{x'}_i)^2}{\left(\tilde{\boldsymbol{x}}_i \cdot \boldsymbol{E}\,(\varepsilon^2 \boldsymbol{C}_0[\boldsymbol{x'}_i])\,\boldsymbol{E}^\top \tilde{\boldsymbol{x}}_i\right) + \left(\tilde{\boldsymbol{x'}}_i \cdot \boldsymbol{E}^\top\,(\varepsilon^2 \boldsymbol{C}_0[\boldsymbol{x}_i])\,\boldsymbol{E}\tilde{\boldsymbol{x'}}_i\right)} \ . \tag{2.60}$$

With this residual the variance ε^2 can be estimated. Because $\tilde{J}_i$ is a χ^2 variable with one degree of freedom the variance ε_i^2 is equal to:

$$\varepsilon_i^2 = \frac{(\boldsymbol{x}_i \cdot \boldsymbol{E}\boldsymbol{x'}_i)^2}{\left(\tilde{\boldsymbol{x}}_i \cdot \boldsymbol{E}\boldsymbol{C}_0[\boldsymbol{x'}_i]\boldsymbol{E}^\top \tilde{\boldsymbol{x}}_i\right) + \left(\tilde{\boldsymbol{x'}}_i \cdot \boldsymbol{E}^\top \boldsymbol{C}_0[\boldsymbol{x}_i]\boldsymbol{E}\tilde{\boldsymbol{x'}}_i\right)} \ . \tag{2.61}$$

Using the equations above, stereo point pairs can be corrected optimally to the epipolar lines, without knowing the scale of the original covariances matrices $\boldsymbol{C}[\boldsymbol{x}_i]$. The corrected positions can be used to obtain optimal estimates for stereo reconstruction vectors $\boldsymbol{r}_i$ and $\boldsymbol{r'}_i$ (Eq. 2.19). After the correction, the noise level ε can be estimated which also enables the approximation of the original feature location covariances.

In our experiments we will only use rectified stereo images. Therefore only the parallel stereo camera setup of fig. 2.8 is considered. Suppose that in both cameras a corner feature is detected that is the projection of space point $\boldsymbol{P}$. The research of Kanazawa et al. [41] shows that the location covariance of such image features is mostly isotropic. It can therefore be assumed that the covariance matrices $\boldsymbol{C}_0[\boldsymbol{x}_i]$ and $\boldsymbol{C}_0[\boldsymbol{x'}_i]$ are equal to the identity matrix $\boldsymbol{I}$. The figure shows the feature locations indicated by vectors $\boldsymbol{x}_i$, $\boldsymbol{x'}_i$ and their standard confidence regions.

Minimisation is now based on the Euclidean distance because of the isotropic noise. Also, the epipolar lines are parallel to the scan lines due to parallel stereo camera setup. Therefore, it can be shown that the correction of Eq. 2.58 only involves the relative vertical offsets [40]:

$$\tilde{\boldsymbol{x}}_i = \begin{pmatrix} x_i \\ \frac{1}{2}(y_i + y_i') \\ 1 \end{pmatrix} \quad \text{and} \quad \tilde{\boldsymbol{x'}}_i = \begin{pmatrix} x_i' \\ \frac{1}{2}(y_i + y_i') \\ 1 \end{pmatrix} \ . \tag{2.62}$$

Furthermore, Eq. 2.61 for the variance ε^2 estimate reduces to:

$$\varepsilon_i^2 = \frac{1}{2}\left(y_i - y_i'\right)^2 \ . \tag{2.63}$$

For applications of stereo vision it is important to know what the reliability is of the 3-D reconstructed points. Kanatani [40] provides a method for calculating both the distance variance of Eq. 2.23 and the 3-D point location covariance $\boldsymbol{C}[\boldsymbol{r}_i]$ of Eq. 2.19. The following equation is used to propagate noise level ε_i into the variances $V[\lambda_i]$ and $V[\lambda_i']$ of the distance estimates λ_i and λ_i':

$$V[\lambda_i] = V[\lambda_i'] = \frac{2\varepsilon_i^2 \lambda_i^4}{b^2} \ . \tag{2.64}$$

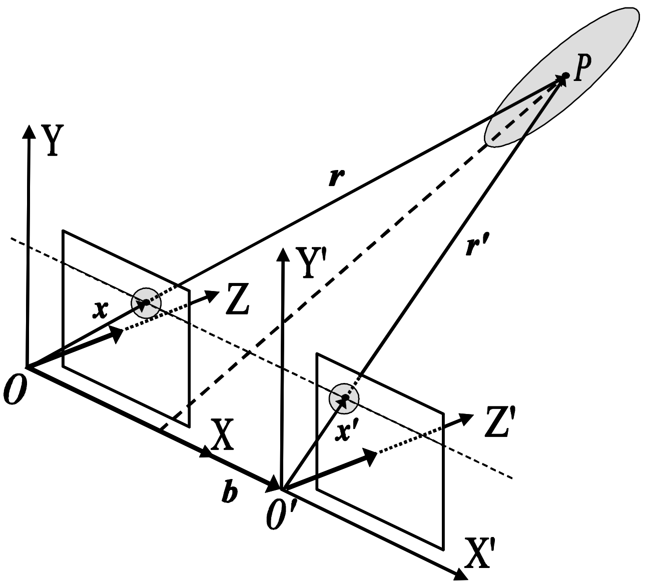


Figure 2.8: Covariance propagation in parallel stereo reconstruction.

This equation shows that the variance is larger for distant points. It also shows that stereo cameras with a small baseline will have a larger variation in their distance estimates than stereo cameras with a large baseline.

The propagation of noise level ε_i into the covariance $\boldsymbol{C}[\boldsymbol{r}_i]$ of reconstruction vector $\boldsymbol{r}_i$ is defined as:

$$\boldsymbol{C}[\boldsymbol{r}_i] = \frac{\varepsilon_i^2\lambda_i^2}{2}\left(\boldsymbol{P}_0 + \frac{4\boldsymbol{x}_i^*\boldsymbol{x}_i^{*\top}}{d^2}\right) \text{ with } \boldsymbol{x}_i^* = \frac{\tilde{\boldsymbol{x}}_i + \tilde{\boldsymbol{x}}'_i}{2} \text{ and } \boldsymbol{P}_0 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \quad (2.65)$$

Here, the vector $\boldsymbol{x}_i^*$ is the midpoint between vectors $\tilde{\boldsymbol{x}}_i$ and $\tilde{\boldsymbol{x}}'_i$. It defines the orientation of the standard confidence region for $\boldsymbol{r}_i$. The disparity d influences the elongation of the standard confidence region in the direction of $\boldsymbol{x}_i^*$. The elongation is larger for distant points because of the smaller disparity value d. The magnitude of the standard region depends on the distance λ_i and the noise level ε_i.

The resulting noise of the reconstructed stereo points is anisotropic and inhomogeneous. This is even the case if the noise of the image features is assumed to be isotropic and homogeneous. Ego-motion estimation based on these points is therefore susceptible to anisotropic and inhomogeneous noise.

2.4 Techniques for ego-motion estimation

Stereo vision based ego-motion estimation uses stereo reconstructed feature points in order to estimate changes in camera position and orientation between consecutive stereo images. In this section, it is assumed that the input for the ego-motion estimation are 3-D space vectors which have been reconstructed from stereo image features. There are N space points before and after motion. The estimated positions of the space points of before and after the motion are indicated with vectors $\boldsymbol{r}_i$ and $\boldsymbol{s}_i$. The reconstruction vectors $\boldsymbol{r}_i$ and $\boldsymbol{s}_i$ are estimations of the true positions $\hat{\boldsymbol{r}}_i$ and $\hat{\boldsymbol{s}}_i$:

$$\boldsymbol{r}_i = \hat{\boldsymbol{r}}_i + \Delta\boldsymbol{r}_i \ , \qquad \boldsymbol{s}_i = \hat{\boldsymbol{s}}_i + \Delta\boldsymbol{s}_i \ . \tag{2.66}$$

The true ego-motion, defined by rotation operation $\hat{R}_M$ and translation $\hat{\boldsymbol{t}}$, puts the following constraint on the true positions:

$$\forall_{i=1\ldots N} \rightarrow \hat{R}_M(\hat{\boldsymbol{r}}_i) + \hat{\boldsymbol{t}} = \hat{\boldsymbol{s}}_i \ . \tag{2.67}$$

Each estimated vector has an error covariance matrix indicated by $\boldsymbol{C}[\boldsymbol{r}_i]$ and $\boldsymbol{C}[\boldsymbol{s}_i]$:

$$\boldsymbol{C}[\boldsymbol{r}_i] = E[\Delta\boldsymbol{r}_i\Delta\boldsymbol{r}_i^\top] \ , \qquad \boldsymbol{C}[\boldsymbol{s}_i] = E[\Delta\boldsymbol{s}_i\Delta\boldsymbol{s}_i^\top] \ . \tag{2.68}$$

In the remainder of this section we first discuss an estimator that assumes isotropic and homogeneous noise in the reconstructed feature points. This is followed by more advanced methods that are capable of dealing with anisotropic and inhomogeneous noise.

2.4.1 Least-squares ego-motion estimation

If it is assumed that the noise of the reconstructed features is isotropic and homogeneous the covariances $\boldsymbol{C}[\boldsymbol{r}_i]$ and $\boldsymbol{C}[\boldsymbol{s}_i]$ are equal to the identity matrix $\boldsymbol{I}$ up to a scale ε^2.

$$\boldsymbol{C}[\boldsymbol{r}_i] = \varepsilon^2\boldsymbol{I} \qquad \boldsymbol{C}[\boldsymbol{s}_i] = \varepsilon^2\boldsymbol{I} \tag{2.69}$$

In this case, the Euclidean distance measure can be used as a error measure J for the ego-motion estimation:

$$J = \sum_{i=1}^{N} ||\boldsymbol{r}_i - \hat{\boldsymbol{r}}_i||^2 + \sum_{i=1}^{N} ||\boldsymbol{s}_i - \hat{\boldsymbol{s}}_i||^2 \ . \tag{2.70}$$

Given the constraint of Eq. 2.67, the estimated rotation R_M and translation $\boldsymbol{t}$ should minimise measure J. This leads to the following least-squares minimisation estimation problem:

$$\arg\min_{R_M,\boldsymbol{t}} \sum_{i=1}^{N} ||\boldsymbol{s}_i - R_M(\boldsymbol{r}_i) + \boldsymbol{t}||^2 \ . \tag{2.71}$$

A straightforward approach to this minimisation is to first eliminate the translation difference by centering both vector sets around their centroids:

$$\bar{\boldsymbol{r}}_i = \boldsymbol{r}_i - \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{r}_i \ , \qquad \bar{\boldsymbol{s}}_i = \boldsymbol{s}_i - \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{s}_i \ . \tag{2.72}$$

The problem is now to estimate the rotation R_M between the sets $\bar{\boldsymbol{r}}_i$ and $\bar{\boldsymbol{s}}_i$. Obtaining a least-squares estimate involves computing the cross correlation matrix $\boldsymbol{K}$ of the centered vector sets:

$$\boldsymbol{K} = \sum_{i=1}^{N} \bar{\boldsymbol{s}}_i \bar{\boldsymbol{r}}_i^\top \ . \tag{2.73}$$

If the function R_M uses the rotation matrix $\boldsymbol{R}$, a least-squares estimate can be obtained with the Single Value Decomposition (SVD) [70] of the cross correlation matrix:

$$\boldsymbol{K} = \boldsymbol{V}\boldsymbol{D}\boldsymbol{U}^\top \ \rightarrow \ \boldsymbol{R} = \boldsymbol{U}\boldsymbol{V}^\top \ . \tag{2.74}$$

After the rotation has been estimated, it can be used to estimate the translation. Given the estimated rotation $\boldsymbol{R}$, this estimate becomes:

$$\boldsymbol{t} = \frac{1}{N}\left(\sum_{i=1}^{N} \boldsymbol{s}_i - \boldsymbol{R}\left(\sum_{i=1}^{N} \boldsymbol{r}_i\right)\right) \ . \tag{2.75}$$

Alternatively, function R_M could be based on rotation with an unit quaternion vector $\boldsymbol{q}$. Quaternions and their application to rotation are explained in appendix B. Representing rotations with quaternions has some advantages over rotation matrices. Firstly, the representation of a quaternion is more compact with four values instead of nine. Another advantage is that it is less cumbersome to extract the rotation axis and angle from an unit quaternion or to compose a new one from it. There is however, one drawback of using quaternion rotation. Because they describe half angles, rotations with angles close to $\pm\pi$ radians can lead to a numerically instable rotation plane.

In the approach by Horn [38] the least-squares estimation problem of Eq. 2.71 is restated as:

$$\arg \min_{\boldsymbol{q},\boldsymbol{t}} \sum_{i=1}^{N} \|\boldsymbol{Z}_i \boldsymbol{q} - \boldsymbol{Q}\boldsymbol{t}\|^2 \ . \tag{2.76}$$

Here, the translation vector $\boldsymbol{t}$ is multiplied with a matrix $\boldsymbol{Q}$ that is based on the quaternion vector $\boldsymbol{q} = (w \ [x\, y\, z])^\top$:

$$\boldsymbol{Q} = \begin{pmatrix} -w & -z & y \\ z & -w & -x \\ -y & x & w \end{pmatrix} \ . \tag{2.77}$$

The 3 by 4 matrix $\boldsymbol{Z}_i$ can be referred to as the measurement matrix because it is assembled from the $\boldsymbol{r}_i$ and $\boldsymbol{s}_i$ vectors:

$$\boldsymbol{Z}_i = \left[\ \boldsymbol{s}_i - \boldsymbol{r}_i \quad (\boldsymbol{s}_i + \boldsymbol{r}_i) \times \boldsymbol{I} \ \right] \ . \tag{2.78}$$

Similar to the rotation matrix based method, the least-squares estimate for $\boldsymbol{q}$ can be obtained by first centering $\boldsymbol{r}_i$ and $\boldsymbol{s}_i$ around their centroids with Eq. 2.72. This allows for computation of the matrices $\bar{\boldsymbol{Z}}_i$ using the previous equation based on $\bar{\boldsymbol{r}}_i$ and $\bar{\boldsymbol{s}}_i$. These matrices are obtained for all vector pairs and used to compute matrix $\boldsymbol{Y}$:

$$\boldsymbol{Y} = \sum_{i=1}^{N} \bar{\boldsymbol{Z}}_i^{\top} \bar{\boldsymbol{Z}}_i \ . \tag{2.79}$$

The eigenvector $\boldsymbol{u}_j$ with the smallest eigenvalue σ_j of matrix $\boldsymbol{Y}$ is the least-squares estimate for the quaternion vector $\boldsymbol{q}$. Finally, the translation vector can be computed with:

$$\boldsymbol{t} = \boldsymbol{Q}^{-} \left(\sum_{i=1}^{N} \boldsymbol{Z}_i \boldsymbol{q} \right) \ . \tag{2.80}$$

2.4.2 Iterative optimisation estimator

The ego-motion estimation from the previous section is based on 3-D vectors which are obtained by stereo reconstruction. The stereo reconstruction itself will use image locations of features such as corners. It can be expected that the positions will contain unknown inaccuracies caused by such factors as sensor noise and resolution. Any stereo reconstruction based on these locations will therefore also contain errors. The least-squares approach of the previous action incorrectly assumes isotropic homogeneous noise.

Before the estimators are discussed that use the more appropriate Mahalanobis distance instead of the Euclidean distance, we will first discuss an approach that includes the stereo reconstruction into the estimation process. As mentioned earlier, the largest error of a stereo reconstructed point is in its projection direction. The distance needed along this direction in order to extend feature vectors $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$, into space vectors $\boldsymbol{r}_i$ and $\boldsymbol{s}_i$, can be defined by scalars λ_i and μ_i:

$$\lambda_i \boldsymbol{x}_i = \boldsymbol{r}_i \ , \qquad \mu_i \boldsymbol{y}_i = \boldsymbol{s}_i \ . \tag{2.81}$$

The least-squares problem of Eq. 2.71 can be expanded to include the distance scalars:

$$\underset{\lambda_{i=1..N},\, \mu_{i=1..N},\, R_M,\, \boldsymbol{t}}{\arg\min} \ \sum_{i=1}^{N} \|\mu_i \boldsymbol{y}_i - R_M(\lambda_i \boldsymbol{x}_i) + \boldsymbol{t}\|^2 \ . \tag{2.82}$$

At first sight, the additional scalars λ and μ do not seem to help, they just expand the number of variables to estimate. However, in the paper of Lasenby et al. [49] an approach is given for estimating camera positions relative to each other. This approach is used to calibrate a set of cameras for motion analysis of points in space. It assumes that there is a single set space points indicated by $\boldsymbol{v}_i$ in the reference frame for the first camera, so that $\lambda_i \boldsymbol{x}_i = \boldsymbol{v}_i$ and $\mu_i \boldsymbol{y}_i = R_M(\boldsymbol{v}_i) + \boldsymbol{t}$. Iteration is used to update the different entities, which describe the relative camera positions. Not only does the approach try to optimise estimates for R_M and $\boldsymbol{t}$ between cameras, it also optimises estimates for λ_i,μ_i and $\boldsymbol{v}_i$ in the different cameras.

We have adopted this method to estimate ego-motion between the stereo reconstructed points $\boldsymbol{r}_i$ and $\boldsymbol{s}_i$. By projecting these onto the stereo cameras, starting values for scalars λ_i and μ_i are computed. Initially, vectors $\boldsymbol{v}_i$ are made equal to $\boldsymbol{r}_i$. The estimator also needs initial guess for the rotation function R_M and translation vector $\boldsymbol{t}$. This can be obtained with the least-squares rotation matrix or a quaternion vector estimation method of the previous section. In an iteration loop different steps are undertaken to optimise the estimated motion, vectors $\boldsymbol{v}_i$ and the scalars λ_i, μ_i. This iteration is repeated until the error-measure of Eq. 2.82 does not show a significant decrease.

Update of the translation

The first step of the optimisation iteration is re-estimating the translation ego-motion vector $\boldsymbol{t}$. This is based on the reconstruction products $\mu \boldsymbol{y}_i$ and the space vectors $\boldsymbol{v}_i$. Using the current estimate for rotation R_M, the update can be based on Eq. 2.75:

$$\boldsymbol{t} = \frac{1}{N} \sum_{i=1}^{N} \left(\mu_i \boldsymbol{y}_i - R_M(\boldsymbol{v}_i) \right) \ . \tag{2.83}$$

Update of the rotation

The second step is the rotation update. This consists out of finding the orientation difference between the reconstruction products $\mu_i \boldsymbol{y}_i$ and the space vectors $\boldsymbol{v}_i$. As discussed in the previous section, the translation can be removed by centering the vectors sets around their centroid. This enables the least-squares estimation of a rotation matrix or quaternion vector as update for rotation R_M.

Update of the scalars

In the next step, scalars λ_i and μ_i are both updated from the space vectors $\boldsymbol{v}_i$ and the feature image vectors $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$:

$$\lambda_i = \frac{\boldsymbol{v}_i \cdot \boldsymbol{x}_i}{\boldsymbol{x}_i \cdot \boldsymbol{x}_i} \ , \qquad \mu_i = \frac{(R_M(\boldsymbol{v}_i) + \boldsymbol{t}) \cdot \boldsymbol{y}_i}{\boldsymbol{y}_i \cdot \boldsymbol{y}_i} \qquad \text{for} \quad i = 1..N \ . \tag{2.84}$$

Update of the space vectors

Finally, the space vector set $\boldsymbol{v}_i$ is updated based on an average of $\lambda_i \boldsymbol{x}_i$ and $\mu_i \boldsymbol{y}_i$:

$$\boldsymbol{v}_i = \frac{\lambda_i \boldsymbol{x}_i + R_M^{-1}(\mu_i \boldsymbol{y}_i - \boldsymbol{t})}{2} \qquad \text{for} \quad i = 1..N \ . \tag{2.85}$$

2.4.3 Optimal correction applied to ego-motion estimation

Section 2.3.4 showed that the noise of reconstructed stereo features is anisotropic and inhomogeneous. This is even the case if the reconstruction is based on stereo image features with isotropic and homogeneous noise. Therefore, in order to estimate the optimal ego-motion between stereo reconstructed features $\boldsymbol{r}_i$ and $\boldsymbol{s}_i$, the Mahalanobis distance is the appropriate error measure J:

$$J = \sum_{i=1}^{N} (\boldsymbol{r}_i - \hat{\boldsymbol{r}}_i) \cdot \boldsymbol{C}[\boldsymbol{r}_i]^{-1} (\boldsymbol{r}_i - \hat{\boldsymbol{r}}_i) + \sum_{i=1}^{N} (\boldsymbol{s}_i - \hat{\boldsymbol{s}}_i) \cdot \boldsymbol{C}[\boldsymbol{s}_i]^{-1} (\boldsymbol{s}_i - \hat{\boldsymbol{s}}_i) \ . \tag{2.86}$$

Here, the individual covariances $\boldsymbol{C}[\boldsymbol{r}_i]$ and $\boldsymbol{C}[\boldsymbol{s}_i]$ can be obtained with the optimal stereo reconstruction method of Kanatani [40] discussed in section 2.3.4.

2.4.4 Optimal rotation estimator

A rotation estimator based on the Mahalanobis distance was developed by Ohta et al. [65]. Their technique is based on the quaternion vector $\boldsymbol{q}$ representation for rotation.

$$\boldsymbol{q} = (w\, \boldsymbol{u})^\top = (w\, [x\, y\, z])^\top \tag{2.87}$$

If applied to ego-motion estimation, the translation needs to be removed first by centering $\boldsymbol{r}_i$ and $\boldsymbol{s}_i$ around their centroids. Similar to the least-squares approach of section 2.4.1, this leads to measurement matrices $\bar{\boldsymbol{Z}}_i$ (Eq. 2.78 after removing translation) that should satisfy:

$$\forall i = 1..N \rightarrow \hat{\bar{\boldsymbol{Z}}}_i \boldsymbol{q} = (0\,0\,0)^\top \ . \tag{2.88}$$

This approach applies the Kanatani renormalisation approach introduced in section 2.3.3. It uses a moment matrix $\boldsymbol{M}$ that is defined as:

$$\boldsymbol{M} = \sum_{i=1}^{N} \bar{\boldsymbol{Z}}_i \boldsymbol{W}_i \bar{\boldsymbol{Z}}_i^\top \ . \tag{2.89}$$

The weight matrix $\boldsymbol{W}_i$ is computed as follows:

$$\boldsymbol{W}_i = \left(w^2 \boldsymbol{A}_i - w\,(\boldsymbol{u} \times \boldsymbol{B}_i) - w\,(\boldsymbol{u} \times \boldsymbol{B}_i)^\top + \boldsymbol{u} \times \boldsymbol{A}_i \times \boldsymbol{u} \right)^{-1} \ . \tag{2.90}$$

Here, matrices $\boldsymbol{A}_i = \boldsymbol{C}[\boldsymbol{r}_i] + \boldsymbol{C}[\boldsymbol{s}_i]$ and $\boldsymbol{B}_i = \boldsymbol{C}[\boldsymbol{r}_i] - \boldsymbol{C}[\boldsymbol{s}_i]$. The unbiased estimate $\tilde{\boldsymbol{M}}$ of $\boldsymbol{M}$ requires matrix $\boldsymbol{N}$.

$$\tilde{\boldsymbol{M}} = \boldsymbol{M} - s\boldsymbol{N} \qquad \boldsymbol{N} = \begin{pmatrix} n_0 & \boldsymbol{n}^\top \\ \boldsymbol{n} & \boldsymbol{N'} \end{pmatrix} \tag{2.91}$$

The matrix $\boldsymbol{N}$ is a 4 by 4 matrix consisting out of the scalar n_0, vector $\boldsymbol{n}$ and matrix $\boldsymbol{N'}$:

$$n_0 = \sum_{i=1}^{N} \boldsymbol{W}_i \vee \boldsymbol{A}_i \ , \tag{2.92}$$

$$\boldsymbol{n} = -2\sum_{i=1}^{N} t_3 \left[\tfrac{1}{2}\boldsymbol{W}_i\boldsymbol{B}_i - \tfrac{1}{2}(\boldsymbol{W}_i\boldsymbol{B}_i)^\top\right] \ , \tag{2.93}$$

$$\boldsymbol{N'} = \sum_{i=1}^{N} \boldsymbol{W}_i \wedge \boldsymbol{A}_i \ . \tag{2.94}$$

The definition of the matrix inner product $\vee$, exterior product $\wedge$ and selection operation t_3 can be found in appendix A.

Iteration is applied during the renormalisation procedure to estimate $\boldsymbol{q}$. One iteration consists out of five steps:

1. Compute all the $\bar{\boldsymbol{Z}}_i$ matrices from $\bar{\boldsymbol{r}}_i$ and $\bar{\boldsymbol{s}}_i$.
2. Initialise $s = 0$ and set matrices $\boldsymbol{W}_i = \boldsymbol{I}$ for all $i = 1..N$.
3. Compute matrices $\boldsymbol{M}$ and $\boldsymbol{N}$.
4. Find the smallest eigenvalue σ of matrix $\tilde{\boldsymbol{M}}$ and its eigenvector $\boldsymbol{q}$.
$$\tilde{\boldsymbol{M}} = \boldsymbol{M} - s\boldsymbol{N} \tag{2.95}$$
5. Stop the iteration if $|\sigma|$ is zero. Otherwise update s, each $\boldsymbol{W}_i$ matrix (Eq. 2.90) and goto step 3.
$$\tilde{s} = s + \frac{\sigma}{\boldsymbol{q}\cdot\boldsymbol{N}\boldsymbol{q}} \tag{2.96}$$

A drawback of this estimator is that it does not consider translation. Just like the least-squares method it is possible to first centre the vector sets $\boldsymbol{r}_i$ and $\boldsymbol{s}_i$ around their centroid to enable a rotation estimate. However, if only a simple mean is used to compute the centroid, their position will be biased due to the anisotropic and inhomogeneous noise. This will lead to inaccurate rotation and translation estimates.

2.4.5 Optimal rotation and translation estimator: HEIV

The Heteroscedastic Error-In-Variables (HEIV) approach was developed by Matei [54] for ego-motion estimation with data that contains anisotropic and inhomogeneous noise. It uses the same quaternion representation as the previous approach. However, this technique does not need to remove translation beforehand and can therefore also provide an optimal estimate for translation. The three 4-dimensional row vectors $\boldsymbol{b}_{ij=1..3}$ of measurement matrix $\boldsymbol{Z}_i$ (Eq. 2.78) can be combined to form a single 12-dimensional column vector z_i:

$$z_i = \mathrm{rows}(\boldsymbol{Z}_i) = \mathrm{rows}\left([\boldsymbol{b}_{i1}\ \boldsymbol{b}_{i2}\ \boldsymbol{b}_{i3}]^\top\right) = \left(\boldsymbol{b}_{i1}^\top\ \boldsymbol{b}_{i2}^\top\ \boldsymbol{b}_{i3}^\top\right)^\top \ . \tag{2.97}$$

The vector z_i is equal to the unperturbed vector $\hat{z}_i$ plus the error Δz_i:

$$z_i = \hat{z}_i + \Delta z_i \ . \tag{2.98}$$

The expectation of covariance matrix $\boldsymbol{C}[z]_i$ of z_i is equal to:

$$\boldsymbol{C}[z_i] = E\left[\Delta z_i \Delta z_i^\top\right] \ . \tag{2.99}$$

In Matei [54] it is shown how these covariances can be computed from the covariance matrices $\boldsymbol{C}[\boldsymbol{r}_i]$ and $\boldsymbol{C}[\boldsymbol{s}_i]$. Note that the dimensions of the covariance matrix $\boldsymbol{C}[z_i]$ are 12 by 12 and that:

$$\boldsymbol{C}[z_i] = \begin{pmatrix} \boldsymbol{C}[\boldsymbol{b}_{i1},\boldsymbol{b}_{i1}] & \boldsymbol{C}[\boldsymbol{b}_{i1},\boldsymbol{b}_{i2}] & \boldsymbol{C}[\boldsymbol{b}_{i1},\boldsymbol{b}_{i3}] \\ \boldsymbol{C}[\boldsymbol{b}_{i2},\boldsymbol{b}_{i1}] & \boldsymbol{C}[\boldsymbol{b}_{i2},\boldsymbol{b}_{i2}] & \boldsymbol{C}[\boldsymbol{b}_{i2},\boldsymbol{b}_{i3}] \\ \boldsymbol{C}[\boldsymbol{b}_{i3},\boldsymbol{b}_{i1}] & \boldsymbol{C}[\boldsymbol{b}_{i3},\boldsymbol{b}_{i2}] & \boldsymbol{C}[\boldsymbol{b}_{i3},\boldsymbol{b}_{i3}] \end{pmatrix} \ . \tag{2.100}$$

The problem now is to minimise the residual J:

$$J = \sum_{i=1}^{n} (z_i - \hat{z}_i)^\top \boldsymbol{C}[z]_i^- (z_i - \hat{z}_i) \ . \tag{2.101}$$

An iterative procedure has been developed for this estimation problem by Matei [54]. It requires an initial estimate for $\boldsymbol{q}$ and $\boldsymbol{t}$, which can be obtained with the least-squares technique of section 2.4. The following steps are proposed to refine this estimate:

1. First compute the $\boldsymbol{L}_i$ matrices from the current estimate for $\boldsymbol{q}$ and the covariance $\boldsymbol{C}[z_i]$ for all i:

$$\boldsymbol{L}_i = \boldsymbol{K}^\top \boldsymbol{C}[z_i]\boldsymbol{K} \quad \text{with} \quad \boldsymbol{K} = \begin{pmatrix} \boldsymbol{q}^\top & \cdots & \cdots & 0\,0\,0\,0 \\ \cdots & \boldsymbol{q}^\top & \cdots & \cdots \\ \cdots & \cdots & \boldsymbol{q}^\top & \cdots \\ 0\,0\,0\,0 & \cdots & \cdots & \boldsymbol{q}^\top \end{pmatrix} \ . \tag{2.102}$$

2. Because the translation has not been removed from the measurement matrices $\boldsymbol{Z}_i$ an optimum estimate for the centroid matrix $\bar{\boldsymbol{Z}}$ is computed next.

$$\bar{\boldsymbol{Z}} = \left(\sum_{i=1}^{n} \boldsymbol{L}_i^-\right)^- \left(\sum_{i=1}^{n} \boldsymbol{L}_i^- \boldsymbol{Z}_i\right) \tag{2.103}$$

3. The matrix $\boldsymbol{S}$ describes the scatter of the measurements $\boldsymbol{Z}_i$ relative to their mean $\bar{\boldsymbol{Z}}$.

$$\boldsymbol{S} = \sum_{i=1}^{n} \left(\boldsymbol{Z}_i - \bar{\boldsymbol{Z}}\right)^\top \boldsymbol{L}_i^- \left(\boldsymbol{Z}_i - \bar{\boldsymbol{Z}}\right) \tag{2.104}$$

4. The weighted covariance matrix $\boldsymbol{C}$ is computed next.

$$\boldsymbol{C} = \sum_{i=1}^{n} \sum_{k,l=1}^{3} l_{ik} l_{il} \boldsymbol{C}[\boldsymbol{b}_{ik}, \boldsymbol{b}_{il}] \tag{2.105}$$

$$\boldsymbol{l}_i = \boldsymbol{L}_i^{-} \left(\boldsymbol{Z}_i - \bar{\boldsymbol{Z}}\right) \boldsymbol{q} = (l_{i1}\ l_{i2}\ l_{i3})^{\top} \tag{2.106}$$

5. The HEIV problem is now to solve the following equation for $\tilde{\boldsymbol{q}}$.

$$\boldsymbol{S}\tilde{\boldsymbol{q}} = \boldsymbol{C}\tilde{\boldsymbol{q}} \tag{2.107}$$

   In Matei [54] it is explained how this can be achieved with the Choleskey decomposition and the Generalized Single Value Decomposition.

6. The Steps 1 until 5 can be repeated by setting $\boldsymbol{q} = \tilde{\boldsymbol{q}}$ until the smallest eigenvalue, encountered in the Generalised Single Value Decomposition, becomes 1.0.

After convergence in the iteration for $\tilde{\boldsymbol{q}}$, the translation $\boldsymbol{t}$ can be estimated with:

$$\boldsymbol{t} = -\boldsymbol{Q}^{-1}\bar{\boldsymbol{Z}}\tilde{\boldsymbol{q}} \ . \tag{2.108}$$

2.5 Robust estimation

In spite of the noise influences, it has been assumed that the point sets used in ego-motion estimation are all separated by a single Euclidean motion. In reality, it is possible that the motion of some points is not consistent with the ego-motion. Such points are called outliers and can be caused by various sources. Errors can occur during stereo matching or feature tracking over time. If features are mismatched in stereo, they are reconstructed with erroneous distances. Tracking errors lead to the association of wrong point pairs. Both types of errors cause point pairs that are separated by a different motion than the ego-motion. Other outlier sources can be independently moving objects such as other vehicles and people. Points tracked on these objects will have different motion than the camera ego-motion. The presence of outliers can degrade the performance of an estimator.

Robust estimation methods can be applied to reduce the influence of outliers. Methods that can handle large numbers of outliers are the Random Sample Consensus (RANSAC) [23] and the Least Median of Squares (LmedS) [86].

Both techniques draw a number of small subsets that contain a random selection of points from the original data set. If the number of subsets is sufficiently high, there will be sets that do not contain outliers. Any of the previously discussed techniques can be used to estimate the motion parameters based on only the points in each subset. This leads to a number of different solutions for the motion parameters. Each solution is evaluated with the whole data set. The solution that fits the most data points is selected as the best solution.

An important consideration is the probability P_{subset} that one of the random point selections is actually outlier free. If this probability is too low, there will be insufficient estimation robustness. In the work by Demirdjian et al. [17] an equation is provided where P_{subset} depends on the probability of selecting an outlier (P_{outlier}), the number of points in a subset (N_{subset}) and the total number of repeated selections R. Solving for R, the following function can be determined:

$$R = \frac{\log\left(1.0 - P_{\text{subset}}\right)}{\log\left(1.0 - \left(1.0 - P_{\text{outlier}}\right)^{N_{\text{subset}}}\right)} \quad . \tag{2.109}$$

In order to obtain a stable ego-motion estimate with all of the ego-motion methods discussed, a minimum of six points is needed. Therefore, N_{subset} is set to this number. Because the percentage of outliers is often unknown, it can set to a default value such as 50% which corresponds with $P_{\text{outlier}} = 0.5$. Suppose that we desire probability $P_{\text{subset}} = 0.95$. According to the equation, a minimum of 191 selections are required to achieve this probability.

An error measure is needed in order to select a solution that fits the most data points. For ego-motion estimation this should indicate how well the points indicated by $\boldsymbol{r}_i$ coincide with $\boldsymbol{s}_i$ after applying the motion parameters. Suppose that $\boldsymbol{R}$ and $\boldsymbol{t}$ are motion parameters obtained from a subset. These parameters can be applied to the vectors $\boldsymbol{r}_i$ and their covariances to bring them in the same reference frame as $\boldsymbol{s}_i$.

$$\boldsymbol{s'}_i = \boldsymbol{R}\boldsymbol{r}_i + \boldsymbol{t} \qquad \boldsymbol{C}[\boldsymbol{s'}_i] = \boldsymbol{R}\boldsymbol{C}[\boldsymbol{r}_i]\boldsymbol{R}^\top \tag{2.110}$$

Now, vector pairs $\boldsymbol{s}_i$ and $\boldsymbol{s'}_i$ should coincide in vector $\hat{\boldsymbol{s}}_i$. However, due to the noise influences, the vectors will be perturbed from this position. An equation for computing the optimal coincide vector $\hat{\boldsymbol{s}}_i$ is given in Kanatani [40]:

$$\hat{\boldsymbol{s}}_i = \frac{\boldsymbol{C}[\boldsymbol{s}_i]^- \boldsymbol{s}_i + \boldsymbol{C}[\boldsymbol{s'}_i]^- \boldsymbol{s'}_i}{\boldsymbol{C}[\boldsymbol{s}_i]^- + \boldsymbol{C}[\boldsymbol{s'}_i]^-} \quad . \tag{2.111}$$

The residual J_i of this estimated position with $\boldsymbol{s}_i$ and $\boldsymbol{s'}_i$ can be used to measure how well the subset motion $\boldsymbol{R}$ and $\boldsymbol{t}$ fits the point pairs:

$$J_i = (\boldsymbol{s}_i - \hat{\boldsymbol{s}}_i) \cdot \boldsymbol{C}[\boldsymbol{s}_i]^- (\boldsymbol{s}_i - \hat{\boldsymbol{s}}_i) + (\boldsymbol{s'}_i - \hat{\boldsymbol{s}}_i) \cdot \boldsymbol{C}[\boldsymbol{s'}_i]^- (\boldsymbol{s'}_i - \hat{\boldsymbol{s}}_i) \quad . \tag{2.112}$$

During the evaluation of a subset motion estimate, J_i is calculated for each point pair. As error measure, the average of these values is computed in RANSAC, while in LmedS the median value is selected. Both approaches then proceed to select the subset motion that has the lowest error score as best solution. Only LmedS will be used in this research, because it has been shown by Torr et al. [86] that LmedS provides more accurate results than RANSAC if there are less than 50% outliers.

2.6 Evaluation of stereo ego-motion methods

Several techniques for stereo based ego-motion estimation have been described in the previous sections. In the introduction, stereo based ego-motion was presented as a possible alternative for navigation systems such as odometry or INS. Because the described methods only allow for motion estimation between successive stereo image pairs, motion estimates have to be combined over time to keep track of the vehicles position and orientation. Unfortunately, estimation errors will also accumulate and reduce the localisation accuracy over time. It is therefore important to use methods that provide the most accurate and reliable ego-motion estimates.

The accuracy of the methods presented has to be measured in order to find out their localisation capabilities. The magnitude of the measured errors can be used to determine for which navigational systems or tasks stereo vision based ego-motion can act as a viable alternative.

Our evaluation of the different estimation methods is based on real stereo data. It was recorded using a forward looking stereo camera that is rigidly mounted on our robotic test vehicle. During the recordings, the vehicle was driven over a test track in off-road terrain. This is shown in fig. 2.9. Due to the terrain roughness, there are several roll and pitch vehicle motions observed in the resulting stereo image sequences.

2.6.1 Feature selection, matching and tracking

It was only briefly mentioned in the introduction that ego-motion methods base their estimates on features extracted from stereo images. Most methods known from literature [59, 73, 17, 91, 95, 94] use tracked corner like features.

We emulate these approaches by using the Harris and Stephens [31] corner detector to detect features. A detected feature in the left image is indicated by a vector $\boldsymbol{x}$ and by vector $\boldsymbol{x'}$ in the right image. The same features are indicated by $\boldsymbol{y}$ and $\boldsymbol{y'}$ in the next stereo pair (also see fig. 2.5). Only detected feature locations are searched and compared to each other to find correspondences between them, since they have sufficiently rich and stable image gradients.

To enable motion estimation, feature points need to be matched in stereo $\boldsymbol{x}_i \leftrightarrow \boldsymbol{x'}_i$, and between succeeding images in a sequence $\boldsymbol{x}_i \leftrightarrow \boldsymbol{y}_i$. A maximum distance, equal to maximum disparity in stereo and maximum displacement expected by motion, is imposed to limit the number of possible matches. For stereo, corresponding features must also lie on the same epipolar lines. Normalised Cross Correlation [69] is used as a similarly measure to compare windows of pixels around the features. Two features from different images are only matched if their correlation value is sufficiently high and different from other possible matches. This prevents many bad and ambiguous matches, however it does not remove all errors. Therefore, the set of tracked features will contain a limited amount of outliers.

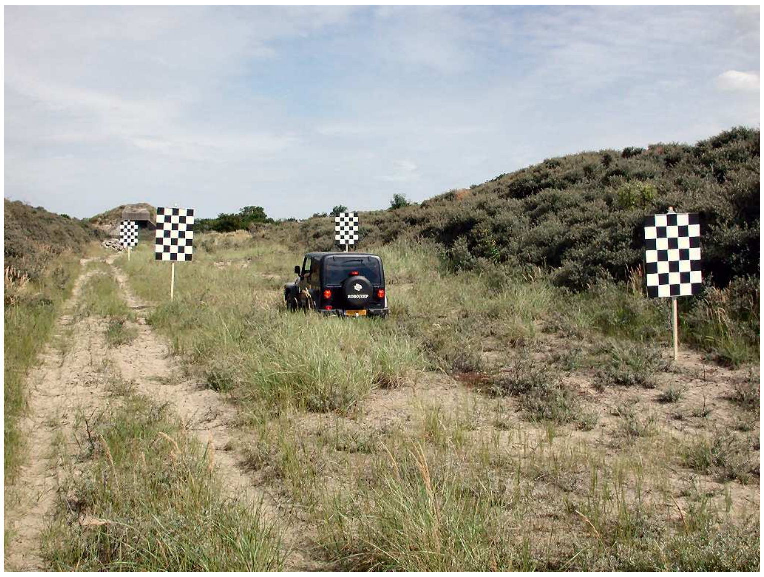


Figure 2.9: The test track in rough terrain used for our experiments. The test vehicle is driven over a straight path between the landmarks.

2.6.2 Landmark based ground truth estimation

Other approaches to ego-motion evaluation have mainly been based on GPS position measurements. Notable examples can be found in Olson et al. [66] and Mallet et al. [51]. Because GPS accuracy does not degrade over time it is suitable to monitor the error accumulation of ego-motion methods. Unfortunately, most GPS systems only provide longitude and latitude measurements that are sufficiently accurate for meaningful comparisons. With such data, only errors in 2-D map localisation can be investigated and vehicle orientation has to be ignored.

In order to recover the full 3-D motion we use a method based on landmarks. These were added at fixed positions in the vicinity of our test track. Fig. 2.9 shows our test vehicle driving on the test track. Landmarks have been added at either side of the track. Each landmark consists of a large plate, mounted to a pole approximately 1.5 m above the ground. A large sheet of paper with checkerboard like pattern is fixed to the plate. The pattern consist out of 5 by 7 black and white squares, each measuring 17 by 17 cm. Junctions between the black and white checkers serve as clear visual features and form a rectangular grid of 4 by 6 points.

The advantage of using such landmarks is that there are no bad or ambiguous point matches. The set of tracked image landmark features does not contain outliers, because landmarks are hand labelled and the grid dimensions dictate which points should correspond to each other. Furthermore, the junctions are almost perfect corner like features that can be easily extracted from the images with subpixel accuracy.

However, there will be inaccuracies in the extracted image positions. Any stereo reconstruction and ego-motion estimation based on this data will also contain inaccu-

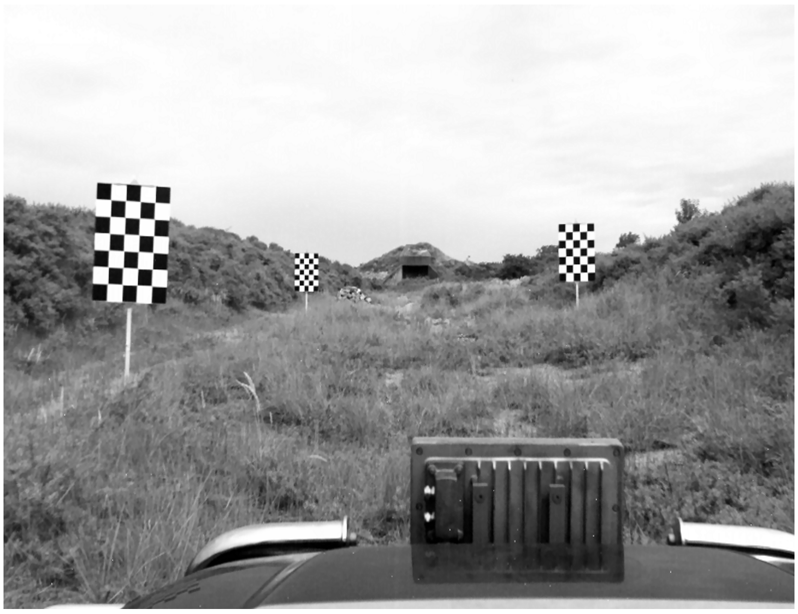


Figure 2.10: Rectified left stereo image of the first test sequence.

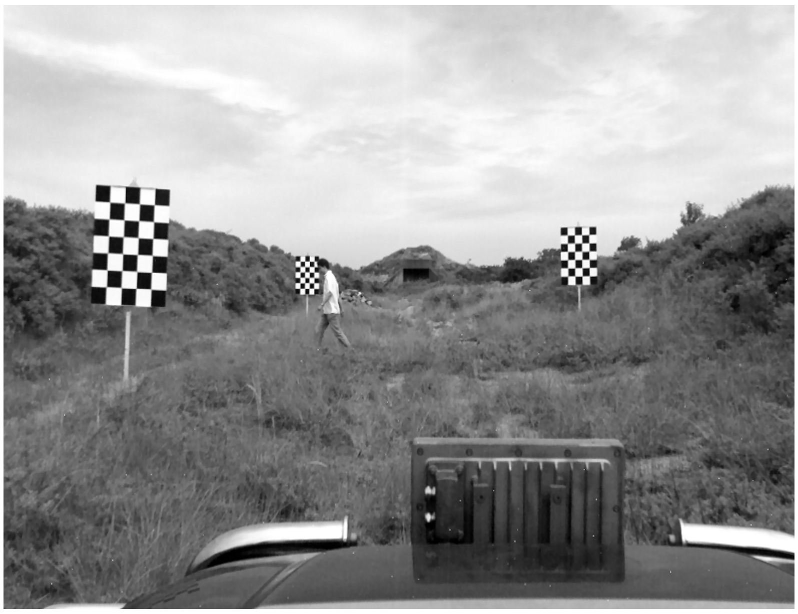


Figure 2.11: Rectified left stereo image of the second test sequence.

racies. Without knowing how this affects the reliability, it is difficult to use it as ground truth for evaluating the accuracy of other estimation methods. In the next section, the accuracy of the landmark based ground truth is assessed.

2.6.3 Obtaining the accuracy of stereo ego-motion estimation

Bootstrap analysis is applied to evaluate the accuracy of the landmark based ego-motion estimation. The original method was developed by Efron [21]. This is a statistical technique that can be used to investigate the accuracy of an estimator if no ground truth is available. We will first describe the general bootstrap method. An adaptation for ego-motion estimation is discussed next.

The bootstrap method

Suppose that that there are N data points $\{z_1, z_2, ..., z_N\}$. The result of applying an estimation process to these points is the estimate $\boldsymbol{g}$. If the data points are independent and identically distributed, the bootstrap method can be applied to evaluate the accuracy estimation process.

The method relies on a number of bootstrap sets that are sampled from the original set. This number is indicated by B and a particular set by index $*b = 1..B$. Each bootstrap set has the same number of entries $j = 1..N$ as the original set. They are assembled by repeatedly selecting a random z_i from the original set and putting it in the bootstrap set as z_j^{*b}. Note that this does not lead to B copies of the original set. Rather, it leads to B sets where a random fraction of the original points are replaced by duplicated original points.

Subsequently, the estimation process is applied to each of the the bootstrap sets. This lead to B estimates, each indicated by $\boldsymbol{g}^{*b}$. Now, the estimation process is said to be unbiased if:

$$\boldsymbol{g} = \bar{\boldsymbol{g}}^* \quad \text{with} \quad \bar{\boldsymbol{g}}^* = \frac{1}{B}\sum_{b=1}^{B} \boldsymbol{g}^{*b} \ . \tag{2.113}$$

However, in many cases there will be a bias indicated by vector $\breve{\boldsymbol{g}}$. In the bootstrap method it is approximated by calculating the difference between the mean $\bar{\boldsymbol{g}}^*$ and the estimate $\boldsymbol{g}$:

$$\breve{\boldsymbol{g}} \approx \bar{\boldsymbol{g}}^* - \boldsymbol{g} \ . \tag{2.114}$$

Another approximation is that estimator covariance $\boldsymbol{C}[\boldsymbol{g}]$ is equal to the covariance $\boldsymbol{C}[\boldsymbol{g}^*]$ of the bootstrap estimate vectors $\boldsymbol{g}^{*b}$:

$$\boldsymbol{C}[\boldsymbol{g}] \approx \boldsymbol{C}[\boldsymbol{g}^*] \quad \text{with} \quad \boldsymbol{C}[\boldsymbol{g}^*] = \frac{1}{B-1}\sum_{b=1}^{B} \left(\boldsymbol{g}^{*b} - \bar{\boldsymbol{g}}^*\right)\left(\boldsymbol{g}^{*b} - \bar{\boldsymbol{g}}^*\right)^\top \ . \tag{2.115}$$

Because the estimation process bias has been approximated, it can be removed from the covariance $\boldsymbol{C}[\boldsymbol{g}]$. This results in the bias corrected estimator covariance $\boldsymbol{C'}[\boldsymbol{g}]$:

$$\boldsymbol{C'}[\boldsymbol{g}] = \boldsymbol{C}[\boldsymbol{g}^*] + \breve{\boldsymbol{g}}\breve{\boldsymbol{g}}^\top \ . \tag{2.116}$$

It should be noted that in order to obtain consistent bootstrap estimates, B should be chosen to be sufficiently large. Also the number of points in the original set cannot be too small.

Applying bootstrap to stereo ego-motion estimation

Because of the anisotropic inhomogeneous noise in reconstructed stereo points, the bootstrap method cannot be applied directly. An adaptation of the bootstrap method for motion estimation with anisotropic inhomogeneous noisy data is described in the work of Matei [54]. It assumes that the vectors $\boldsymbol{r}_i$ and $\boldsymbol{s}_i$ are perturbed by noise from their true positions $\hat{\boldsymbol{r}}_i$ and $\hat{\boldsymbol{s}}_i$.

$$\boldsymbol{r}_i = \hat{\boldsymbol{r}}_i + \Delta\boldsymbol{r}_i \quad \text{and} \quad \boldsymbol{s}_i = \hat{\boldsymbol{s}}_i + \Delta\boldsymbol{s}_i \tag{2.117}$$

Bootstrap is based on the residuals $\Delta\boldsymbol{r}_i$ and $\Delta\boldsymbol{s}_i$. Because of the constraint $\boldsymbol{R}\hat{\boldsymbol{r}}_i + \boldsymbol{t} = \hat{\boldsymbol{s}}_i$, estimates for the true positions can be obtained by projecting $\boldsymbol{r}_i$ and $\boldsymbol{s}_i$ in such a way that they fit the motion. An estimate for $\boldsymbol{R}$ and $\boldsymbol{t}$ is obtained from $\boldsymbol{r}_i$ and $\boldsymbol{s}_i$ by applying the HEIV estimator. This estimate is then used to obtain $\hat{\boldsymbol{r}}_i$ and $\hat{\boldsymbol{s}}_i$:

$$\hat{\boldsymbol{r}}_i = \boldsymbol{R}^\top \boldsymbol{A}_i^- \boldsymbol{s}_i + \boldsymbol{B}_i^- \boldsymbol{r}_i - \boldsymbol{R}^\top \boldsymbol{A}_i^- \boldsymbol{t} \ , \tag{2.118}$$

$$\hat{\boldsymbol{s}}_i = \boldsymbol{A}_i^- \boldsymbol{s}_i + \boldsymbol{R}\boldsymbol{B}_i^- \boldsymbol{r}_i + \left(\boldsymbol{I} - \boldsymbol{R}^\top \boldsymbol{A}_i^-\right)\boldsymbol{t} \ . \tag{2.119}$$

Here, matrices $\boldsymbol{A}_i$ and $\boldsymbol{B}_i$ are based on the covariances $\boldsymbol{C}[\boldsymbol{r}_i]$ and $\boldsymbol{C}[\boldsymbol{s}_i]$ of the original vectors.

$$\boldsymbol{A}_i = \boldsymbol{I} + \boldsymbol{C}[\boldsymbol{s}_i]\boldsymbol{R}\boldsymbol{C}[\boldsymbol{r}_i]^- \boldsymbol{R}^\top \quad \text{and} \quad \boldsymbol{B}_i = \boldsymbol{I} + \boldsymbol{C}[\boldsymbol{r}_i]\boldsymbol{R}^\top \boldsymbol{C}[\boldsymbol{s}_i]^- \boldsymbol{R} \tag{2.120}$$

With the corrected vectors the residuals $\Delta\boldsymbol{r}_i$ and $\Delta\boldsymbol{s}_i$ can be calculated:

$$\Delta\boldsymbol{r}_i = \boldsymbol{r}_i - \hat{\boldsymbol{r}}_i \quad \text{and} \quad \Delta\boldsymbol{s}_i = \boldsymbol{s}_i - \hat{\boldsymbol{s}}_i \ . \tag{2.121}$$

As indicated earlier, the residuals are due to anisotropic inhomogeneous noise. Matei [54] uses a “whitening”-“colouring” cycle to enable bootstrap sampling. The basic idea behind this cycle is that the residuals are first transformed in such a way that their distributions become isotropic and homogeneous. Bootstrap sets are sampled from these transformed residuals. Afterwards, the distribution of residuals in the sets are transformed back to anisotropic inhomogeneous. The “whitening”-“colouring” can be performed because the residual covariances $\boldsymbol{C}[\boldsymbol{r}_i]$ and $\boldsymbol{C}[\boldsymbol{s}_i]$ are available from the stereo method discussed in section 2.3.4. The Cholesky decomposition of the covariances $\boldsymbol{C}[\boldsymbol{r}_i]$ and $\boldsymbol{C}[\boldsymbol{s}_i]$ is equal to:

$$\boldsymbol{C}[\boldsymbol{r}_i] = \boldsymbol{L}[\boldsymbol{r}]_i^\top \boldsymbol{L}[\boldsymbol{r}]_i \quad \text{and} \quad \boldsymbol{C}[\boldsymbol{s}_i] = \boldsymbol{L}[\boldsymbol{s}]_i^\top \boldsymbol{L}[\boldsymbol{s}]_i \ . \tag{2.122}$$

The residuals are whitened to isotropic homogeneous by applying the transposed inverse of Cholesky upper triangular matrix $\boldsymbol{L}$.

$$\Delta\boldsymbol{r}'_i = \boldsymbol{L}[\boldsymbol{r}]_i^{-\top} \Delta\boldsymbol{r}_i \quad \text{and} \quad \Delta\boldsymbol{s}'_i = \boldsymbol{L}[\boldsymbol{s}]_i^{-\top} \Delta\boldsymbol{s}_i \tag{2.123}$$

Now that the whitened residuals are available, bootstrap sets can be created. Only the residuals are sampled. The resulting random selections $\Delta\boldsymbol{r}'^{*b}_k$ and $\Delta\boldsymbol{r}'^{*b}_k$ are added to the estimated unperturbed positions $\hat{\boldsymbol{r}}_j$ and $\hat{\boldsymbol{s}}_j$. However, before they are added, the residuals are coloured with the anisotropic inhomogeneous distributions of the original points.

$$\boldsymbol{r}^{*b}_j = \hat{\boldsymbol{r}}_j + \sqrt{2}\boldsymbol{L}[\boldsymbol{r}]^\top_j \Delta\boldsymbol{r}'^{*b}_k \quad \text{and} \quad \boldsymbol{s}^{*b}_j = \hat{\boldsymbol{s}}_j + \sqrt{2}\boldsymbol{L}[\boldsymbol{s}]^\top_j \Delta\boldsymbol{s}'^{*b}_k \tag{2.124}$$

The scalar $\sqrt{2}$ is used by Matei [54] in order to insure a consistent bootstrapped covariance estimate.

Bootstrap rotation and translation estimates are obtained with the HEIV estimator from the coloured bootstrap sets. For translation, Eq. 2.113 until 2.116 can be used directly to calculate the mean bootstrap translation $\bar{\boldsymbol{t}}^*$, and its bias corrected covariance $\boldsymbol{C}'[\boldsymbol{t}]$. A vector representation of rotation has to be used in order to approximate its bootstrap covariance. Therefore, the estimated bootstrap rotation is converted to the rotation vector representation [20]. For the unit quaternion $\boldsymbol{q}$, the equivalent rotation vector $\boldsymbol{o}$ is computed by scaling the unit rotation axis vector $\boldsymbol{u}$ with rotation angle θ.

$$\boldsymbol{q} = \left(\cos(\tfrac{1}{2}\theta) \quad \left[\sin(\tfrac{1}{2}\theta)\boldsymbol{u}\right]\right)^\top \quad \leftrightarrow \quad \boldsymbol{o} = \theta\boldsymbol{u} \tag{2.125}$$

The same steps used for translation can now be used to compute the mean bootstrap rotation $\bar{\boldsymbol{o}}^*$ and its bias corrected covariance $\boldsymbol{C}'[\boldsymbol{o}]$. In order to get consistent estimates we use $B = 500$ bootstrap sets. From the resulting $\boldsymbol{C}'[\boldsymbol{o}]$ and $\boldsymbol{C}'[\boldsymbol{t}]$ we will obtain the desired accuracy estimates of the landmark based ground truth.

2.6.4 Evaluation of ego-motion methods for localisation tasks

In our evaluation the bootstrap means $\bar{\boldsymbol{t}}^*$ and $\bar{\boldsymbol{o}}^*$ are used as ground truth. They are not estimated in the conventional frame to frame fashion in order to avoid error accumulation. Instead, the first frame is regarded as a navigation starting point. Ego-motion is estimated between it and each other frame in the sequence. This ensures that there is no accumulation of estimation errors in the ground truth. However, the drawback is that all the stereo images must contain the same landmarks. This limits the number of usable frames in recorded test sequences.

A measure for the amount of error in the ground truth is extracted from the covariances $\boldsymbol{C}'[\boldsymbol{t}]$ and $\boldsymbol{C}'[\boldsymbol{o}]$. For translation we use error scalar ε_t that indicates the square root of the mean squared error for vector length $||\boldsymbol{t}||$:

$$\varepsilon_t = \sqrt{\operatorname{trace}(\boldsymbol{C}'[\mathbf{t}])} \ . \tag{2.126}$$

To investigate the error for each dimension separately, we also define the error vector $\mathbf{e}_t$. It is the square root of each diagonal element of the covariance matrix:

$$\boldsymbol{e}_t = \sqrt{\operatorname{diag}(\boldsymbol{C}'[\boldsymbol{t}])} \ . \tag{2.127}$$

It is more difficult to obtain an insight into the true rotation error from the rotation covariance. We therefore choose to calculate the most likely error rotation. This can be obtained by decomposing matrix $\boldsymbol{C'}[\boldsymbol{o}]$ into its eigenvectors and eigenvalues. The eigenvector $\boldsymbol{u}_e$ with the largest eigenvalue (principal component) represents the axis around which the error rotation is most likely to occur. The eigenvalue θ_e^2 is the mean squared error of this rotation's angle. With Eq. 2.125, a quaternion $\boldsymbol{q}_e$ can be defined that represents the error rotation.

Ego-motion estimation methods are evaluated with this ground truth. Each of the evaluated methods is provided with the same set of reconstructed points that have been tracked from frame to frame. The optimal stereo reconstruction method of section 2.3.4 is used to obtain these points from the naturally occurring terrain features. Rotation matrix $\boldsymbol{R}_i$ and translation vector $\boldsymbol{t}_i$ represent the resulting ego-motion estimates of one method at frame i. The following recursive formulas are used to transform it to a orientation $\boldsymbol{O}$ and position $\boldsymbol{p}$ estimate that can be compared to the ground truth:

$$\boldsymbol{O}_i = \begin{cases} \boldsymbol{I} & i \equiv 1 \\ \boldsymbol{R}_i \boldsymbol{O}_{i-1} & i > 1 \end{cases} \quad \text{and} \quad \boldsymbol{p}_i = \begin{cases} (0\,0\,0)^\top & i \equiv 1 \\ \boldsymbol{O}_i \boldsymbol{p}_{i-1} + \boldsymbol{t}_i & i > 1 \end{cases} \quad . \tag{2.128}$$

The estimates of each ego-motion method for $\boldsymbol{O}_i$ and $\boldsymbol{p}_i$ are compared to those of the ground truth. For translation, the error vector $\boldsymbol{e}_i$ is calculated.

$$\boldsymbol{e}_i = \bar{\boldsymbol{t}}_i^* - \boldsymbol{p}_i \tag{2.129}$$

For the rotation, both the rotation matrix $\boldsymbol{O}_i$ and the ground truth rotation quaternion $\bar{\boldsymbol{q}}_i^*$ are converted to the Euler angles; roll, pitch and yaw. The difference in these rotation angles is used as error measure.

2.7 Results

Two real stereo image sequences are used to evaluated the different ego-motion estimation methods. In the first sequence the vehicle follows a straight route. A left stereo image from this sequence is shown in fig. 2.10. The second sequence was recorded on the same terrain as the first sequence. However, in order to add outlier motion, a pedestrian now crosses the path of the vehicle. The pedestrian is visible in the example image of fig. 2.11.

In this section, we will first analyse the accuracy of the estimated ground truth. An examination of the results obtained with each individual test sequence will follow. Finally, we will discuss results achieved by the different ego-motion estimation methods.

2.7.1 Accuracy of the estimated ground truth

The discussed bootstrap method is used to obtain the ground truth ego-motion and its reliability for each of the two sequences. It is based on points tracked on landmarks to

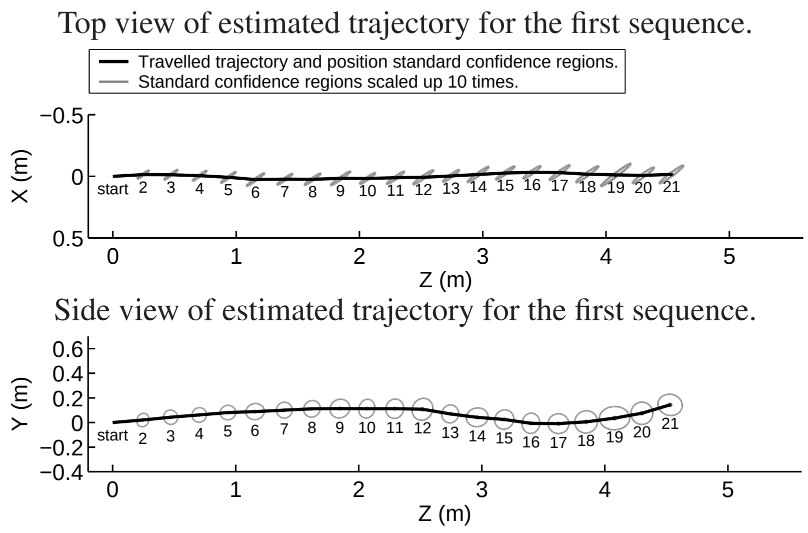


Figure 2.12: Top and side view of the estimated trajectory and errors for the first test sequence. The trajectory and the standard confidence ellipses of the bootstrapped covariances have been drawn in black. In order to illustrate the ellipse shapes more clearly, the gray variants have been added that are enlarged by a factor of ten.

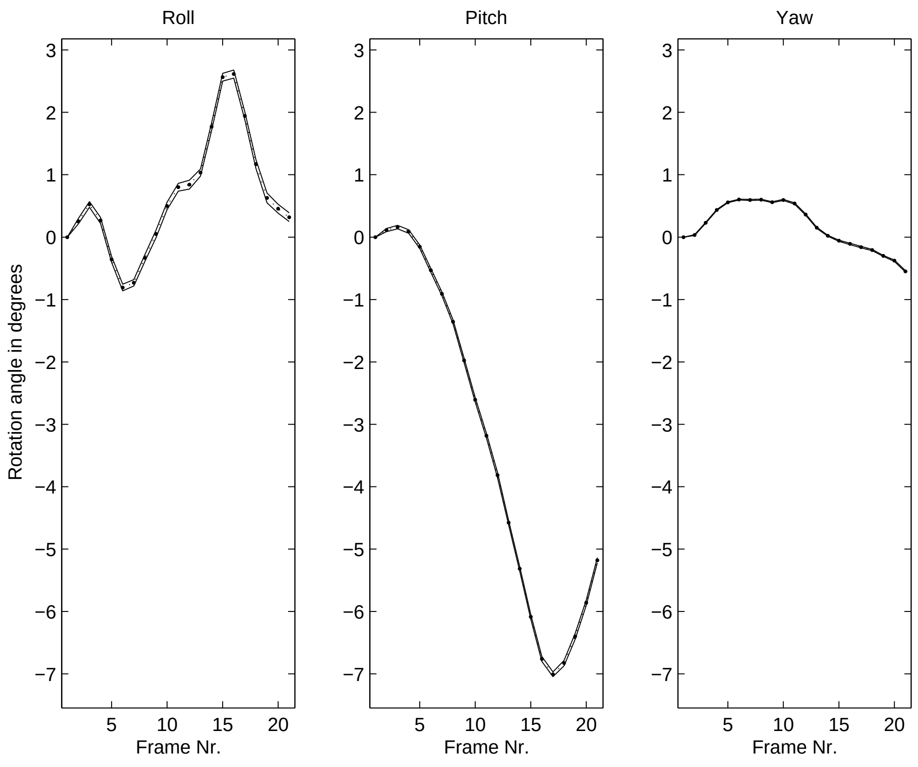


Figure 2.13: Ground truth rotation found for the first test sequence. The line thickness indicates the error in rotation.

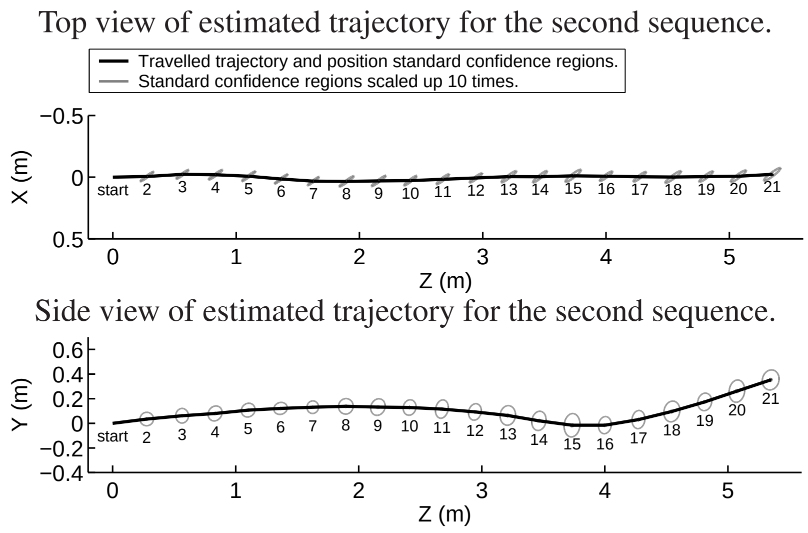


Figure 2.14: Top and side view of the estimated trajectory and errors for the second test sequence. The trajectory and the standard confidence ellipses of the bootstrapped covariances have been drawn in black. In order to illustrate the ellipse shapes more clearly, the gray variants have been added that are enlarged by a factor of ten.

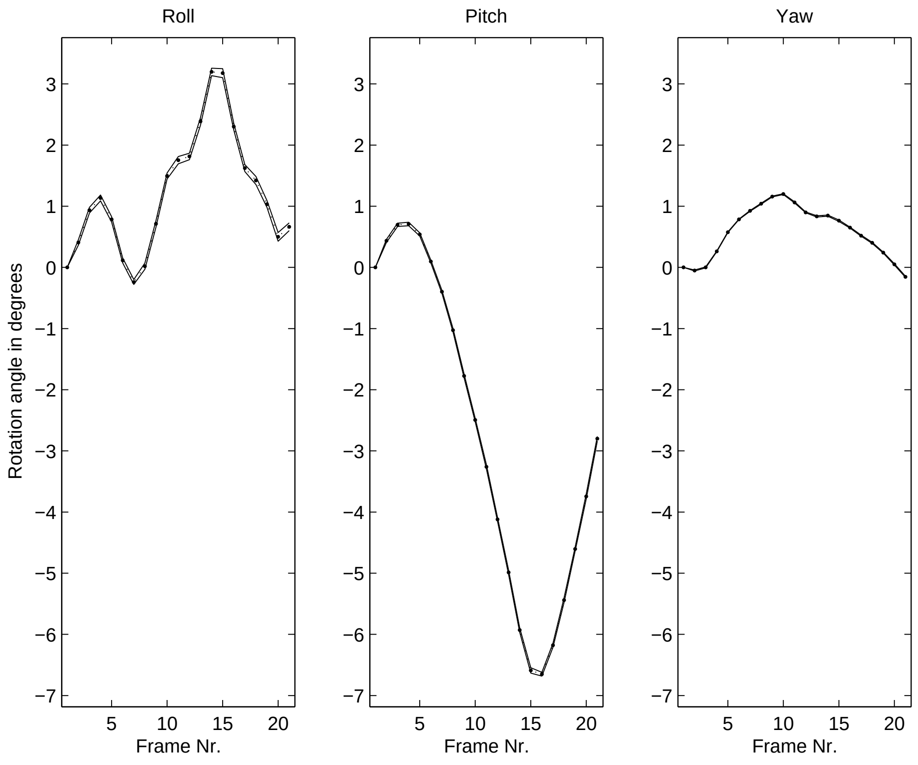


Figure 2.15: Ground truth rotation found for the second test sequence. The line thickness indicates the error in rotation.

Table 2.1: Estimated ground truth orientation and position at the end of each sequence by the bootstrap method.

	Rotation values			Position vector $\bar{t}^*_{i=21}$ in 10^{-3} m.			
Method	*roll*	*pitch*	*yaw*	$\bar{t}^*_{i=21} = (x\,y\,z)^\top$			$\lVert\bar{t}^*_{i=21}\rVert$
Sequence 1	0.3°	-5.2°	-0.5°	-15.7	143.3	4530.4	4532.7
Sequence 2	0.7°	-2.8°	-0.2°	-22.5	352.4	5347.7	5359.4

avoid outliers. Ground truth estimates for the stereo camera orientation and position at the end of each sequence are shown in table 2.1.

Figs. 2.12 and 2.14 shows the estimated ground truth trajectory for the first and second sequence. Each figure displays a top and a side view of the trajectory. Because they were recorded on the same test track there are similarities between the trajectories of both sequences. The top views confirm that the path followed by the vehicle was more or less straight in both sequences. The side views indicate that the terrain is uneven, because the track slopes up and down. The error in estimated position can be indicated by a standard confidence ellipse at each point on the trajectory. However, due to the high accuracy of the ground truth, the resulting ellipses are small. These are difficult to distinguish from the line that indicates the trajectory. For illustrative purposes only, the ellipses where scaled up ten times and added to the figs. 2.12 and 2.14 in the colour gray. The side view indicates that the error ellipses are more or less isotropic in the YZ-plane. However, the top-view shows that the error eclipses are anisotropic in the XZ-plane.

The rotation ground truth for the two sequences is plotted in the figs. 2.13 and 2.15. Each figure shows the rotation angles for roll, pitch and yaw. In both sequences there is a predominately pitch motion visible. Error in the rotation angles is indicated by the thickness of the plots. The limited thickness for all angles reveals that ground truth orientation estimates are also accurate.

2.7.2 Results of the ego-motion estimation methods

For the evaluation of the different ego-motion estimation methods, corner-like image features are extracted and tracked in each sequence. Features that are located on the landmarks are excluded to ensure that only naturally occurring features are used. The resulting test sets are used to evaluate the following four estimation methods:

1. Least-Squares (LS) estimator based on the rotation matrix from section 2.4.1.
2. Iterative (ITER) estimator from section 2.4.2.
3. The Optimal Rotation (OR) method from section 2.4.4.
4. The Heteroscedastic Error-In-Variables (HEIV) method from section 2.4.5.

During the evaluation, both “weak” and “robust” versions of each algorithm are tested. In the non-robust tests, the algorithms were provided with all the tracked data points, including outliers. For the robust tests the same data was used. However, each algorithm is extended with the LmedS technique for outlier detection.

Results with the first test sequence

In the first test sequence the vehicle follows a straight trajectory. According to table 2.1 it travels about 4.5 m. Among the feature tracks, there is a moderate number of outliers that are only caused by stereo mismatches and tracking errors.

Figs. 2.16 and 2.17 show the position errors made by the tested methods for each stereo frame in the first sequence. In fig. 2.16, errors are plotted that are the result of estimations with all data points. The top plot shows that each method’s error magnitude generally increases due to the accumulation of errors made at each frame. If the errors by the different methods are compared to each other, it becomes clear that the error increases for LS and OR are more severe than those for ITER and HEIV. The bottom plot provides a more detailed view of the latter two techniques. It shows that the error increases by the ITER approach are smaller than those by HEIV. The plot also displays the error in the ground truth. This demonstrates the reliability of this evaluation, because the ground truth errors are all smaller and do not increase over time.

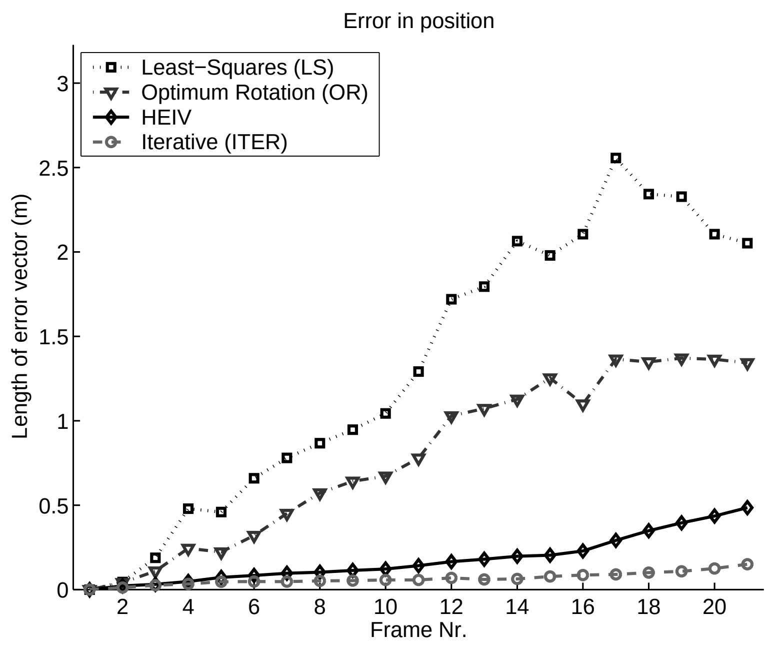
The same sequence was also used to evaluate robust variants of the estimators. In fig. 2.17, the error accumulation is shown when the LmedS technique is applied. As before, the top plot shows the errors of all estimators, while the bottom one shows only the two best performing methods and the ground truth errors. Estimators that do benefit from LmedS are OR and HEIV. Their errors are now significantly lower. This improvement is unclear for the LmedS ITER approach. Its errors are similar to those obtained without LmedS. As for the LmedS LS method, only the first part of the sequence indicates an improved result. In the second part, there are some large errors that actually worsen the end result.

Table 2.2 offers an insight in how much error has been accumulated by the different estimators at the end of the first test sequence. It contains orientation and position errors. The table shows that the errors of both LS variants are quite large. It is especially apparent when they are compared to the ground truth estimates in table 2.1. For example, the yaw angle error is larger than the ground truth rotation around this axis. Furthermore, the position error vector of the non-robust variant is about 45% of the length of the ground truth position vector. The best result is achieved by the non-robust ITER variant, with an error in position of about 3%.

Results with the second test sequence

Because they where recorded on the same test track, the results of the first and the second sequence can be compared to each other. The only major difference between them is that in the second sequence a pedestrian is visible, walking from right to left

Position errors in the first sequence by non-robust estimation



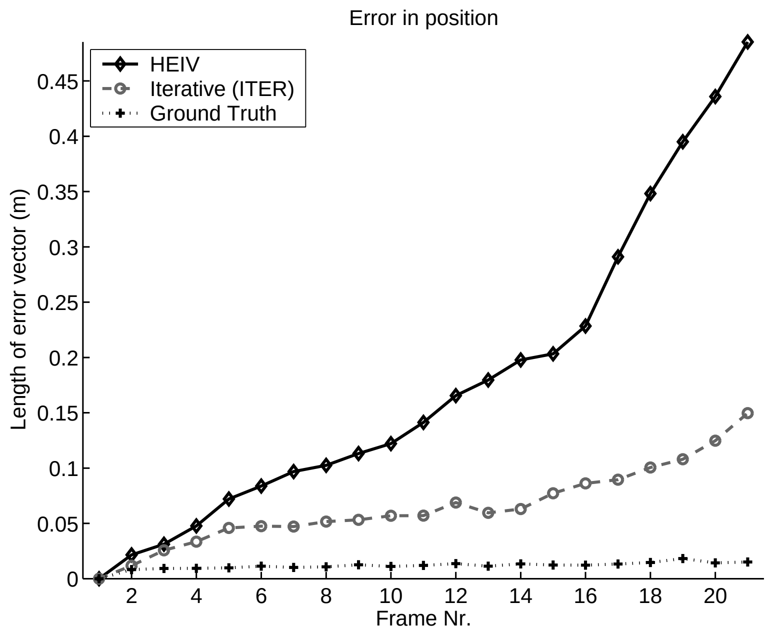
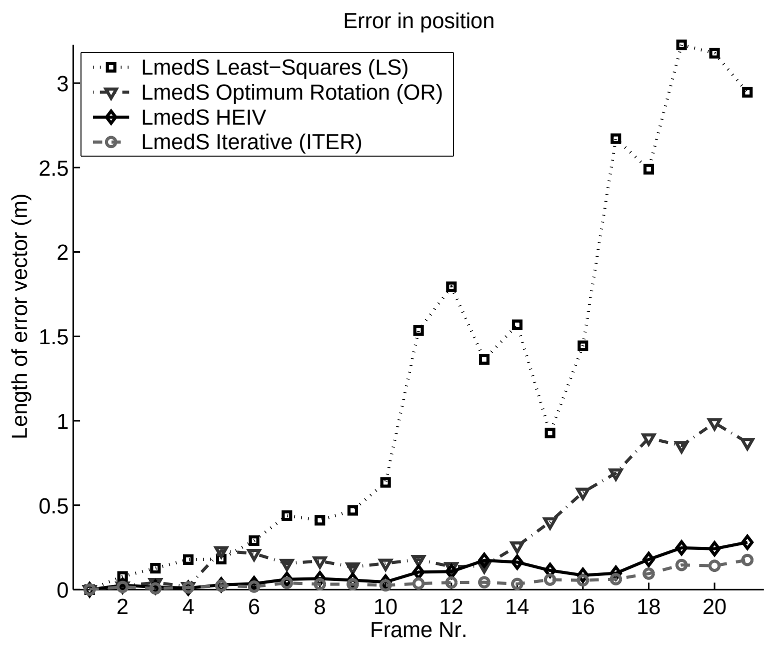


Figure 2.16: Plots of the error accumulation position for the various non-robust estimators for each frame in the first test sequence. A more detailed view of the best performing methods is provided by the bottom plot, errors in the ground truth position estimates are also included.

Position errors in the first sequence by robust estimation



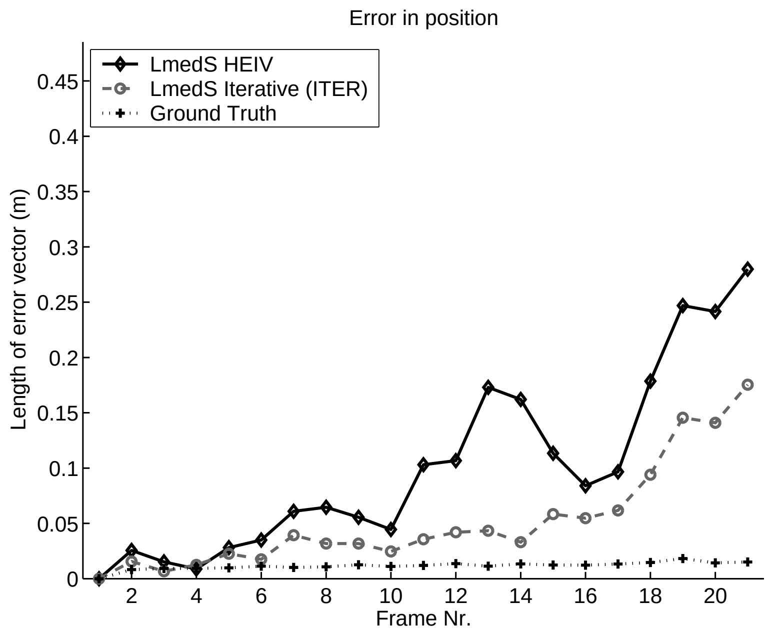
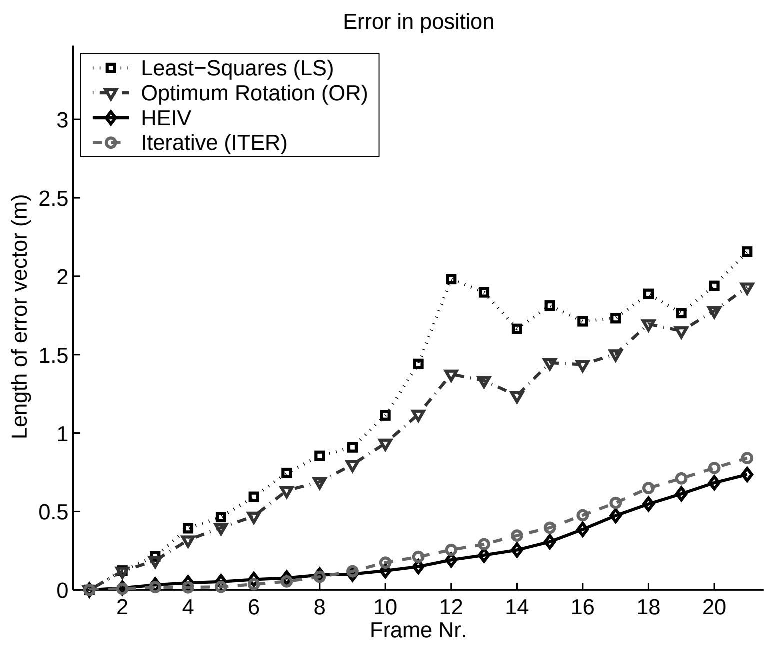


Figure 2.17: Plots of the error accumulation position for the various robust estimators for each frame in the first test sequence. A more detailed view of the best performing methods is provided by the bottom plot, errors in the ground truth position estimates are also included.

Position errors in the second sequence by non-robust estimation



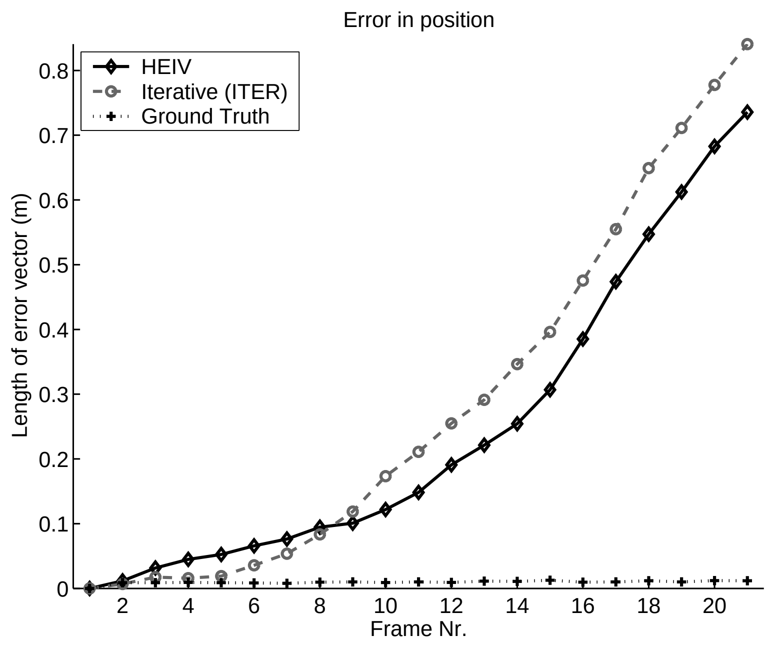
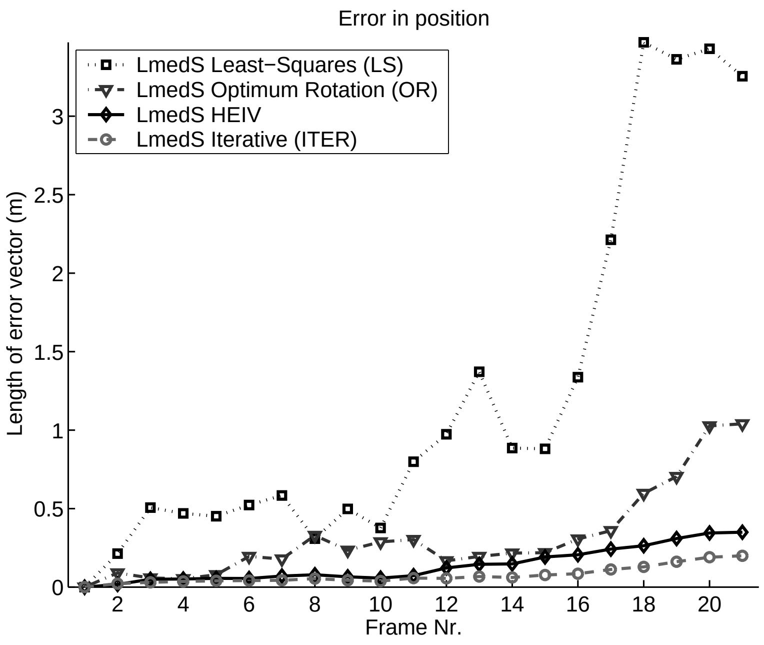


Figure 2.18: Plots of the error accumulation position for the various non-robust estimators for each frame in the first test sequence. A more detailed view of the best performing methods is provided by the bottom plot, errors in the ground truth position estimates are also included.

Position errors in the second sequence by robust estimation



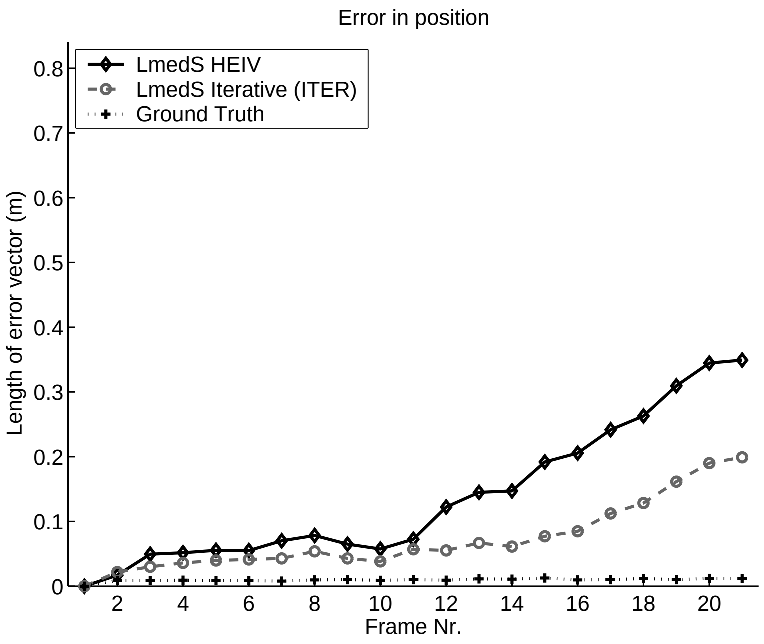


Figure 2.19: Plots of the error accumulation position for the various robust estimators for each frame in the first test sequence. A more detailed view of the best performing methods is provided by the bottom plot, errors in the ground truth position estimates are also included.

through the stereo images. This was done to test estimator robustness, because the pedestrian causes a large number of outliers among the tracked image features.

Fig. 2.18 shows the position errors of the different methods when no robust estimation technique is applied. If these plots are compared to those of fig. 2.16, it can be seen that the OR, ITER, HEIV and methods now show an increase in error accumulation. The larger errors are obviously caused by the outliers tracked on the independently moving pedestrian. The error accumulation by the OR approach is now similar to that of LS. Both HEIV and the ITER still show better performance over the other methods. However, the increased error accumulation indicates that these methods are sensitive for larger number of outliers.

Results of robust estimation with the LmedS technique are displayed in fig. 2.19. With the exception of LS, all estimation methods now show reduced accumulation errors. Their performance is similar to what was achieved with the first sequence. Therefore, for these estimators, the LmedS technique successfully identified and removed the outliers. This observation is supported by the numerical errors at the sequence that are shown in table 2.3. Except for LS, all methods show significant decrease in rotation and position errors when the LmedS technique is applied. For example, the non-robust ITER variant has a position percentage error of about 16%, while the robust LmedS variant only has a percentage of about 4%.

2.7.3 Discussion

After examining the individual test sequences, we can now discuss some of the overall results. The ground truth that is used in our evaluation, is itself also based on ego-motion estimation. Statistical bootstrap has been applied to analyse its accuracy. In the two test sequences both the accuracy of ground truth orientation and position is very high. For example, table 2.2 shows that the orientation errors are less than one tenth of a degree and that the position error is less than 1.5 cm. This is more than sufficient for reliable evaluation of the ego-motion estimation methods. However, it should be noted that this accuracy can only be achieved if the landmarks are relatively nearby ($\sim$20 m), because larger distances degrade the precision of image feature locations on the landmarks.

It is not surprisingly that LS is the least accurate ego-motion technique. This is due to the fact that the method does not consider the anisotropic inhomogeneous nature of noise in the reconstructed stereo features positions. Instead, it assumes the same error distribution for each data point while in reality, some of the points will have larger errors than the others. The data centroids, needed for translation and rotation separation, will therefore contain similarly large offsets from their true positions. This also occurs in the subsequent rotation estimation step.

The assumption of isotropic homogeneous noise by LS is also a problem for the application of the LmedS technique. In the first two test sequences, it is clear that LmedS accentually increases the error by LS. When LS uses a large amount of data points, errors might be reduced when a sufficient number of errors cancel each other

Table 2.2: Orientation and position errors at the end of the first test sequence.

	Orientation error			Error in position vector $\boldsymbol{e}$ in 10^{-3} m.			
Method	*roll*	*pitch*	*yaw*	$\boldsymbol{e} = (x\,y\,z)^\top$			$\|\boldsymbol{e}\|$
LS	-3.3°	1.0°	6.1°	1452.9	-456.2	-1374.7	2051.5
OR	0.6°	-0.8°	-3.1°	-517.6	47.0	-1238.4	1343.1
HEIV	0.5°	-0.3°	-0.6°	-266.8	47.5	-402.7	485.3
ITER	0.2°	-0.4°	-0.2°	-120.9	77.8	-41.8	149.7
LmedS LS	-3.7°	-4.8°	7.5°	2323.6	1236.5	-1322.4	2945.6
LmedS OR	1.1°	-0.7°	-2.5°	-662.9	205.5	-525.3	870.5
LmedS HEIV	0.1°	-0.2°	0°	-80.0	23.5	-267.2	279.9
LmedS ITER	0.1°	-0.1°	0°	-76.9	27.2	-155.3	175.4
Ground Truth	±0°	±0°	±0°	±7.4	±8.6	±9.9	15.1

Table 2.3: Orientation and position errors at the end of the second test sequence.

	Orientation error			Error in position vector $\boldsymbol{e}$ in 10^{-3} m.			
Method	*roll*	*pitch*	*yaw*	$\boldsymbol{e} = (x\,y\,z)^\top$			$\|\boldsymbol{v}\|$
LS	-2.9°	1.5°	3.5°	1008.5	-470.4	-1847.5	2156.7
OR	0.2°	-0.3°	-4.1°	-856.3	-101.4	-1726.6	1929.9
HEIV	0.6°	-0.1°	-1.9°	-568.7	-11.6	-466.4	735.6
ITER	0.3°	-0.3°	-2.7°	-704.5	66.1	454.2	840.8
LmedS LS	-1.4°	-7.6°	-5.2°	-1855.3	2404.4	-1171.8	3255.2
LmedS OR	0.9°	0.4°	-3.7°	-927.8	-264.7	388.0	1039.9
LmedS HEIV	0.1°	0°	0.8°	163.6	-25.9	-307.5	349.3
LmedS ITER	0°	-0.1°	0.6°	105.9	-9.2	-168.2	199.0
Ground Truth	±0°	±0°	±0°	±5.3	±8.1	±6.8	11.9

Table 2.4: Computation timings of the different ego-motion estimators with both test sequences. Timing is based on the average computation time needed for 100 points for estimating a single frame-to-frame ego-motion.

	Sequence 1		**Sequence 2**	
Method	Avg. time	Var.	Avg. time	Var.
LS	0.003 s	0.000	0.004 s	0.000
OR	0.705 s	0.007	0.767 s	0.008
HEIV	0.339 s	0.001	0.387 s	0.004
ITER	10.991 s	0.003	10.973 s	0.005

out. However, this is much more unlikely to occur in LmedS because the randomly selected subsets only contain a small number of points. Furthermore, erroneous estimates by LS, based on the points in the subsets, cause the wrong points to be selected as non-outliers.

The OR estimation technique can be regarded as an intermediate of Least-Squares and the HEIV technique because its accuracy lies in between both estimators. Its rotation estimation uses the appropriate Mahalanobis distance to assign weights to the individual data points. However, the method still relies on simple averages to compute the data centroids that isolate the rotation from translation. Similar to the problems encountered with LS, this can lead to large errors in the centroid positions. All results show that application of the LmedS technique improves OR estimation accuracy. Apparently, the OR method is very sensitive to outliers.

Overall, the two best performing estimators are the Iterative and the HEIV approach. Like the OR approach, weight computation in HEIV is based on the Mahalanobis distance. However, HEIV does correct for offsets in the data centroids. This leads to better performance for the rotation and translation estimates.

The results show that there are many instances where Iterative approach outperforms HEIV. This result might be unexpected, because the ITER method basically uses the same techniques as the LS estimator. Only the additional optimisation of the stereo reconstruction vector distances can be the cause of this performance advantage. We also noticed that in the first test sequence the estimator performance was not improved by LmedS. This can be attributed to robustness of the method to small numbers of outliers. An individual vector distance can be optimised to become zero, due to the repeated iteration steps. The distances of outlier points are often optimised to this value because it enables them to fit the dominant rotation and translation among the rest of the data points.

Unfortunately, the ITER approach comes with a cost. Table 2.4 shows the average computation time that each estimator needs to estimate the frame-to-frame ego-motion with 100 data points. This test was performed in Matlab 6.5 [1] on a Pentium 2.2 GHz computer with 512 MB of RAM. It is clear that the Iterative method is much more expensive than the HEIV method. This is due to the fact that the Iterative method requires multiple steps for the update of rotation, translation and stereo reconstruction. Because each step is only a sub optimal estimate, many iterations are needed to reach convergence. Each iteration in the HEIV estimator is an approximation of the optimal ego-motion estimation. The HEIV estimator can converge to a solution much faster and is therefore more suitable for real-time applications.

The accuracies achieved by the ITER and HEIV approach give an indication for which navigation tasks ego-motion estimation is suitable. If outliers are only due to matching and tracking errors, table 2.2 shows that the length of the LmedS HEIV error position vector covers about 6% of the actual length. The length of the LmedS ITER error position vector only covers about 4%. For a more difficult situation, where there are more outliers, table 2.3 shows that the LmedS HEIV error position vector covers about 7%, while the LmedS ITER error position vector only covers about 4%.

Stereo ego-motion estimation is not a viable alternative for replacement of absolute positioning sensors such as GPS, due to its error accumulation. Because the ego-motion position and orientation estimates are accurate for short distances it could serve as an alternative to both odometry and gyroscopic sensors. The advantage of ego-motion estimation over odometry is that it can measure displacement accurately in three dimensions. With ego-motion, it can be determined if the vehicle drives on or off a slope. Gyroscopes might still be able to measure orientation changes more accurately than ego-motion. However, orientation estimation with gyroscopes is based on the integration of measured rotational accelerations. This means that gyroscopic orientation accuracy also degrades over time when the vehicle does not move. With ego-motion estimation is it trivial to detect when the vehicle does not move.

An important application for ego-motion estimation is therefore precision navigation. This includes driving around newly detected obstacles, through narrow openings such as bridge heads and preventing that the vehicle tips over.

2.8 Conclusion

The goal of this research is to investigate the suitability of stereo vision based ego-motion estimation for mobile robot navigation in outdoor off-road terrain. The motion estimation itself is based on tracking detected corner-like image features that occur naturally the terrain. Stereo reconstruction is applied to recover the three-dimensional positions of these features, before and after an Euclidean inter-frame motion. An important property of these features is that the noise in their positions is anisotropic and inhomogeneous.

Our evaluation compares four different methods for ego-motion estimation. The first is a simple least-squares (LS) approach that incorrectly assumes isotropic and homogeneous noise for the reconstructed features. The second approach is an iterative technique (ITER) that is based on the least-squares approach. It does, however, include the estimation of the distances to the reconstructed stereo features as extra variables. In contrast to the first two estimators, the others do assume anisotropic and inhomogeneous noise. They use the more appropriate Mahalanobis distance as error measure. The third (OR) applies this distance to estimate only rotation optimally. The last technique, called Heteroscedastic Error-In-Variables (HEIV), applies it to both rotation and translation. Of each the four methods, a normal and a robust variant are evaluated. In the latter variant, the LmedS technique is applied to provide robustness against outliers.

Two stereo image sequences recorded in real off-road terrain are used for the evaluation. One of these sequences contains an independently moving pedestrian to also test estimation robustness. In contrast to other research on vehicle ego-motion, our evaluation considers all six-degrees of freedom of three-dimensional motion. A ground truth for this motion is estimated by using outlier free features located on special landmarks that where added to the terrain. Statistical bootstrap analysis shows that the ground truth is very accurate if the landmarks are sufficiently nearby. High precision landmark navigation, based on this method, could also be an interesting prospect for other applications.

Comparisons with the ground truth show that the LS and the OR method are ill-suited for stereo based ego-motion estimation. The errors made by the LS method are the largest because it assumes the wrong noise model for the data points. Our results also indicate that robust techniques, such as LmedS, actually degrade the LS estimation performance further because they rely on estimates based on small sets of points. However, the results for OR are somewhat better because it applies the correct noise model for rotation.

Better results are achieved with the ITER and HEIV approach. In spite of being based on LS, the ITER approach often outperforms the HEIV approach, while the later does use the correct noise model for both rotation and translation estimation. The ITER approach also demonstrates robustness against moderate numbers of outliers without relying the robust LmedS technique. Unfortunately, the large number of iteration required by ITER makes its less computationally attractive than the HEIV approach.

Overall, the results with HEIV and the Iterative approach show that accurate stereo based ego-motion estimation is possible for short distances. Errors accumulated by these approaches over longer distances could be corrected if an update for the absolute vehicle position is available. Such an update can be obtained from absolute positioning sensors such as GPS. It could also be obtained by closing-the-loop [62] in situations where it is known that the vehicle has driven over the same location twice.

Chapter 3

Real-Time Dense Stereo Disparity Estimation

3.1 Introduction

Reliable, robust and real-time obstacle detection methodologies are needed to ensure that autonomous vehicles can operate safely in complex and unstructured environments. Situational awareness is also an important topic for research into automotive safety systems. A goal here is to develop automatic systems that issue warnings or even actively intervene to avoid traffic accidents. This requires robust methods for detection of other traffic participants, such as cars and pedestrians. Vehicles that are fitted with a situational awareness system or that have autonomous navigation capabilities can be regarded as “intelligent vehicles”.

Stereo vision has the advantage that it is able to obtain an accurate and detailed 3D representation of the environment around a vehicle, by passive sensing and at a relatively low sensor cost. The work by Labayrade et al. [47] is an example of a real-time stereo system that is able to detect vehicles up to 80 m away. This and other previous applications (e.g. [25]) have mostly used sparse, feature-based approaches to stereo vision. Here only a subset of image pixels (e.g. vertical edge pixels) are matched.

However, by only using sparse depth data, it is more difficult to perform a subsequent object segmentation step. For example, the vertical edges of a single object are often separated. If edges of different objects are near to each other it is difficult to determine which of them belong to the same objects. This is especially difficult in unstructured off-road terrain. Here there are many edges due to the texture of, for example, bushes or foliage. Not being able to properly segment obstacles complicates the application of other processing steps such as classification and tracking. For this reason, it is attractive to use dense stereo vision, that tries to estimate disparity for all image points.

A large research community centres around dense stereo vision because it is attractive for a number of applications such as robot navigation, surveillance systems, 3D modelling, augmented reality and video conferences. Many systems for dense stereo vision or disparity estimation have been presented, as discussed in two large surveys of the field, one by Scharstein and Szeliski [75] and another by Brown et al. [10].

In contrast to previous surveys, this chapter does not aim to review the whole field of dense stereo. Our aim is to investigate if certain approaches to dense stereo vision are more suitable for intelligent vehicle applications than others. The criteria of this investigation are founded on practical considerations specifically related to the intelligent vehicle domain.

The first of these considerations is that application of dense stereo in intelligent vehicles is only possible if the disparity map can be calculated in real-time. Single Instruction Multiple Data (SIMD) offers an appealing and straightforward way for speeding up computation by carrying out one operation on multiple values simultaneously. Because the parallelism is only in terms of the data, difficult problems such as process synchronisation can be avoided.

Over the past few years, manufacturers have extended general purpose processors with SIMD capabilities in response to demanding multimedia applications (e.g. SSE2 instruction set for Intel processors, as used in this chapter). Yet SIMD also forms the basis architecture for special hardware as used in the automotive industry (e.g. DSPs), which faces particular demands with respect to power consumption, cost and compactness. Therefore, in order to evaluate dense stereo vision algorithms from an intelligent vehicles perspective, it is important to consider their suitability regarding SIMD parallelism.

In this chapter, we identify different methodological components and develop efficient underlying SIMD implementations. The latter are combined in a single framework enabling comparisons and analysis of the various approaches on an equal basis.

A second consideration is that the output of stereo algorithms on itself is not interesting for intelligent vehicle applications. Only subsequent steps, such as obstacle detection or segmentation, can provide useful information about the vehicle surroundings. Other work on dense stereo vision has often used error measures where only the quality of the disparity values was evaluated. We will present error measures and evaluation techniques which are more related to typical applications of stereo vision in the intelligent vehicles domain.

The outline of this chapter is as follows. Section 3.2 first discusses a number of concepts for dense disparity computation from the literature. In Section 3.3, we present the corresponding real-time SIMD implementations. Section 3.4 compares the resulting algorithms with additional, publicly available approaches [6, 75] on both simulated and real data depicting complex urban traffic scenes. Section 3.5 contains the conclusion.

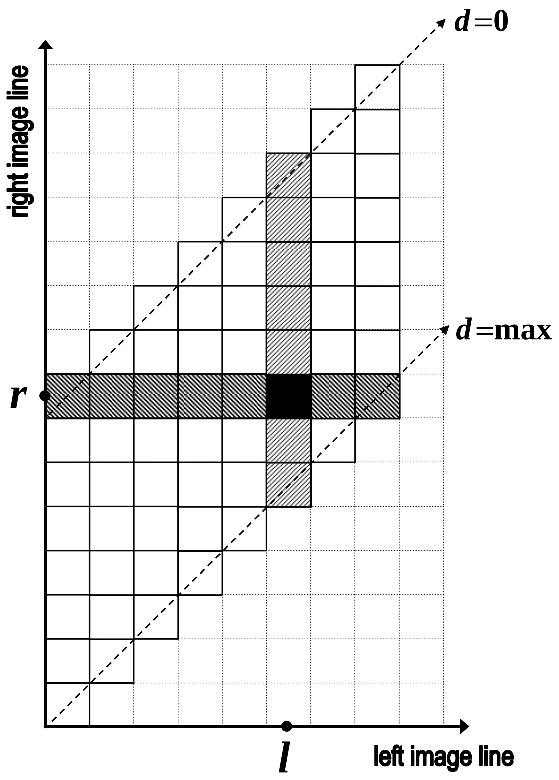


Figure 3.1: Disparity search space. The light gray area shows the left-to-right search area for a point match, while the dark gray area shows the right-to-left search area.

3.2 Approaches to stereo vision

The goal of stereo disparity estimation is finding the correct correspondences between image points from the left and right camera. For each point, the positions of possible matches in the other image is constrained to a single epipolar line, if the stereo camera geometry is known. Most approaches to disparity assume that the epipolar lines run parallel to the image lines, so that corresponding points lie on the same image lines. This situation can be achieved for stereo cameras by using a rectification technique [27, 32]. If images are rectified, the disparity d between a corresponding left point l and a right point r can be expressed as:

$$d = l - r\,. \tag{3.1}$$

All possible matches for points on a left and right image line are located in a two-dimensional space. This space is called the *disparity search space* and is shown in fig. 3.1. Left image pixel positions are located on the horizontal axis, while those of the right image reside on the vertical axis. Therefore, possible right point matches for a left point l can be found in the light gray coloured column of fig. 3.1. Similarly, the dark gray coloured row indicates the possible matches for a right point r. Often, a minimum d_{min} and maximum d_{max} disparity are used to bound the possible matches in the disparity search space.

Usually, one can distinguish two stages in a disparity estimator. In the first stage, cost values are calculated for comparing the different points in the disparity search space. These cost values are used in the second stage for searching the correct points (matches) in the disparity space. Some algorithms use additional pre- and post-processing steps. In order to simplify the matching step, pre-processing is applied to reduce the illumination differences between the stereo images. A typical post-processing step is the detection of occlusions, these are image regions only visible in one of the stereo images.

In the following sections we describe methods from literature that can be used for each step.

3.2.1 Pre-processing

A complicating factor for stereo matching is that the intensities of corresponding pixels from the stereo images can be different. This can be caused by unequal left and right camera sensor characteristics such as brightness and contrast. It is also due to differences in lighting conditions at each of the camera positions. If raw input images are used, it is necessary to use an illumination invariant similarity measure. Because invariant measures are computationally more expensive, pre-processing of input images is often applied to reduce the illumination differences beforehand. One approach subtracts median filtered versions from the original input images [81]. A more popular approach is the convolution of the input images with a Laplacian of Gaussian (LoG) kernel [13]. This reduces illumination influences because the response of the Laplacian is zero in areas with constant intensity while it is either positive or negative near edges with high intensity gradient.

Preprocessing can also be used to extract extra information in order to aid the subsequent disparity search. Hong and Chen [37] use colour based segmentation to find similarly coloured patches. They match patches instead of individual pixels because the assumption is that no large disparity discontinuities occur within the homogeneous coloured patches.

3.2.2 Similarity measures

The simplest similarity measures are based on the difference in pixel intensity, such as absolute difference (AD) or squared difference (SD). Algorithms that only use these single intensity measures in order to compare points are known as “pixel-to-pixel” algorithms. Unfortunately, discrete images have a quite limited number of different gray-level intensity values. It is possible to use colour instead [60]. However, colour is difficult to use when illumination conditions are less than ideal, for instance during nighttime.

Pixel-to-pixel measures are not very distinctive when intensities of different pixels are the same or corrupted by noise. Birchfield and Tomasi [6] also pointed out the sensitivity to image sampling. For example, the pixel intensities on corresponding stereo edges can be different due to aliasing. They have therefore designed a measure that is less sensitive to image sampling.

A more commonly used approach for improving distinction, is using a larger support region for aggregating the cost values. Typically, the sum of the pixel differences in a window around a pixel of interest is used, such as the sum of absolute differences (SAD) or the sum of squared differences (SSD).

The use of larger windows will lead to more robustness against noise. However, larger windows with fixed size and centre point will lead to less accuracy in disparity estimates. This is due to the fact that areas on slated surfaces will warp projectively between the stereo images. A square window on such a surface will therefore only correspond correctly to a projectively warped version in the other image.

Occlusions near object edges can also cause problems. When a large window is centred around a background point near a object edge it will almost certainly encapsulate a portion of the foreground object. In the case of an occlusion, a large amount of background pixels in the window will not be visible in the other image. The resulting similarity measure will wrongfully be biased towards disparities that belong to the foreground pixels.

In the literature, several ways can be found to improve window based matching. An adaptive size cost window was proposed by the Kanade and Okutomi [39]. Given an initial guess of disparity, they use a statistical technique in order to estimate the optimal size and box shape of each matching window. However, estimating the optimal window size at all points is computationally expensive.

Other ways of improving the cost computation include multi-scale techniques and the use of multiple windows. In some multi-scale approaches several matching windows of different sizes are used. The larger windows provide the robustness, while the smaller windows provide precision. On the downside, matching errors at the coarse scale might propagate to the finer scales.

A somewhat similar approach is not changing the size of the windows, but having different options for the location of the centre point. Both Bobick et al. [8] and Fusiello et al. [26] use nine different window centre points in their approaches. For real-time applications, Hirschmüller et al. [35] suggest the use of five-window configuration. Fig. 3.2 shows the regular matching window centred in the middle. On the edge points (A,B,C,D) of this window four additional windows have been indicated. Of these four windows the two with the lowest SAD values are searched. They are added to the value of the centre window. This step acts as a sort of deformable window, which means that the resulting window does not have to be centred on the point itself.

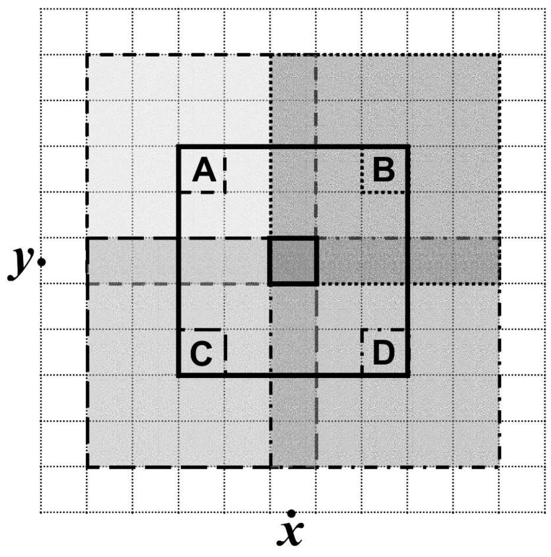


Figure 3.2: Multiple cost windows. Apart for the centre window indicated by the solid lines, four additional windows centred at A,B,C and D are used.

3.2.3 Disparity search

When similarity or cost values in the disparity search space have been calculated, the correct matches can be searched. A straightforward way of doing this is searching the disparity interval per point for a optimum value. This approach is known as the Winner-Takes-All (WTA) approach.

A drawback of the simple WTA approach is that it does not consider the presence of occlusions. This will result in wrong disparity estimates for those regions. Because WTA only searches for optimum cost values it is very sensitive to the results from the cost computation. The cost values computed for a textureless region often do not indicate a optimum match due to their similarity. Repetitive texture, such as bars, can lead to several ambiguous optima. In these areas, the WTA is prone to errors.

Assumptions about the scene geometry can be exploited to improve disparity estimates. Three often applied assumptions are the “smoothness constraint”, “uniqueness constraint” and “ordering constraint”. The smoothness constraint is based on the observation that changes in depth on surfaces are much smaller than those at the edges of objects.

The uniqueness constraint states that a 3D point only has exactly one projection in each of the stereo cameras. Therefore, only one correspondence has to be found for every stereo point pair. The constraint is only violated by image points on transparent material or occluded image parts. The ordering constraint states that the sequence in which points are ordered on a left image line, is the same for the right image line. If occlusions are present, points can be missing from one of the line sequences, but the ordering will remain. There are scene situations possible where the ordering constraint is violated [104]. However, it is assumed that they are not very common.

A cost function (CF) is one approach to enforcing smoothness. It can be used to locally add penalties to the values in the disparity search interval of each next point on the scanline. Mismatches can be suppressed by penalising large jumps in disparity between the scanline points. The difficulty of this approach is how to choose the penalty function. If penalties are too high, disparity jumps near edges are missed and if they are too low the smoothness in not enforced.

Given a correct stereo match in the disparity space, both the uniqueness and ordering constraint limit the number of possible matches for the following pixel on the reference image. Because of the limitations, finding the correct disparities is now akin to finding a valid path that takes the shortest route through the cost values. Special rules for how to transverse the search space can be added in order to handle occlusions and jumps in disparity. Dynamic-programming (DP) techniques are often used for this approach [8].

Another search optimisation technique called Scanline Optimisation (SO) was proposed by Scharstein and Szeliski [75]. Instead of enforcing special rules for occlusion and disparity jumps, they use a global variant of a smoothness cost function to add penalties to the matching costs. Then a disparity is searched for each scanline point that minimises the total cost.

In both the SO and DP the optimisation and search techniques are constrained to one single scanline. Just like CF, they are very dependent on the choice of rules that govern the path propagation or the cost functions that are used for searching the optimum solution. This can cause them to miss or wrongfully suppress large jumps in disparity, for example near object edges. These types of errors cause "streaks" of erroneous disparity estimates along scanlines. Interline consistency is difficult to maintain with these techniques. Recently, some work appeared on optimising in more than one direction. The two pass approach of Kim et al. [42], uses the results of the first optimisation pass along the scaliness to optimise the estimates across the scanlines in the second pass.

An alternative approach is to view disparity search as a directed graph labelling problem. The graph nodes can represent left and right pixel pairs with disparity as label. They can also represent whole pixel patches extracted with a pre-processing step. Each transition between nodes has a cost specified by a CF that partially depends on the assigned disparity label. In several approaches [45, 37, 18] graph cuts are used to the find disparity label assignments that minimise the transition costs. The graph optimisation approach carries a high computational cost. However, the algorithms based on it are among those that currently reside in the top positions of the performance ranking on the Middlebury stereo vision web site [2].

3.2.4 Post-processing

This section is devoted to some of the additional steps performed next to the disparity estimation. These include error detection, subpixel interpolation and occlusion removal.

It was mentioned earlier that image regions with little or repetitive texture are problematic for stereo algorithms. Some approaches try to detect possible errors in the disparity estimates. Fusiello et al. [26] use the variance of the cost values in the disparity range of a point as a measure of confidence. Other approaches [35] select the optimum C_1 and second best value C_2 from the disparity search range. The confidence in the selected optimum is expressed as:

$$C = \frac{C_2 - C_1}{C_1} \ . \tag{3.2}$$

The idea is based on the fact that the cost values of a textureless region will be very similar. Repetitive texture will lead to several optimums in the disparity interval. The measure of Eq. 3.2 will be low in both cases. However, Mühlmann et al. [60] argue that there are occurrences where the optimum correspondence is actually between two pixels. This also leads to two very similar optimum values in the disparity search space. To prevent discarding good estimates on these points, they suggest using the third best optimum instead of the second best.

Besides using integer values for disparity correspondence, floating-point values can be also used for subpixel accuracy. Many algorithms use an additional post-processing step for improving the disparity estimate to subpixel accuracy. If the SSD similarity measure is used, the cost values near an optimum can be approximated by a second degree polynomial. Given the cost values of the optimum and its two nearest neighbours, the subpixel estimate can be computed by:

$$d_{subpixel} = d + \frac{C_{d-1} - C_{d+1}}{2(C_{d-1} - 2C_d + C_{d+1})} \ . \tag{3.3}$$

Although this formula is intended for the SSD measure, many SAD based approaches [35, 60, 80] do also apply it for subpixel interpolation.

As indicated earlier, an extra step is necessary to detect and remove erroneous estimates on occlusions by WTA. The assumptions about scene geometry can also be exploited for this purpose.

One technique often used is the left-right consistency check. This check exploits the uniqueness constraint. It is assumed that the left to right and the right to left disparities are known. The algorithm checks that the disparity from left to right correspondence is the same value as when it is done conversely, searching the match from the right image to left image. Different value means inconsistency that could be caused by an occlusion and therefore, these matches are removed.

A drawback of this approach is that the minimum disparity has to be searched twice for every pixel. Stefano et al. [80] show that the uniqueness constraint can also be exploited for occlusion and error removal. During the search for left to right disparities, the cost value of a best match is stored for every newly matched right pixel. If a right pixel is encountered that has been matched before, the new and the old cost value of the match are compared. The old match is removed when the new cost value is smaller.

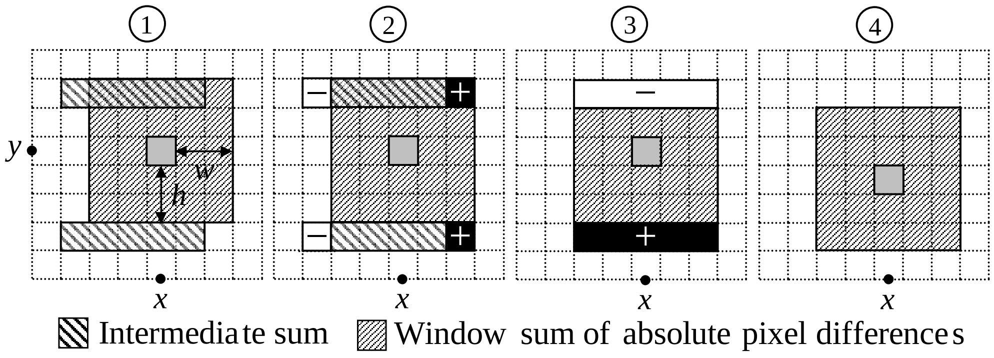


Figure 3.3: Computation of a new SAD value out of the value of an old one using intermediate row sums.

This allows the algorithms to "recover" from previously made bad matches.

3.3 Dense real-time stereo framework

In order to investigate the suitability of real-time dense algorithms for the intelligent vehicles domain on an equal basis, we have implemented our own framework of real-time dense stereo algorithms. These algorithms use the latest SIMD SSE2 instruction set available on the Intel Pentium 4 or AMD Athlon 64 processors. The SSE2 instruction set uses 128-bit registers. For integer operations these can contain packed 8, 16, 32 or 64-bit data buffers.

Implementation based on SIMD is not straightforward because it places restrictions on what kind of operations can be used. An algorithm can only be sped up if the necessary operations can be performed in parallel. Conditional constructs must be avoided because they often lead to stalls of the code path prediction unit on the processor [28]. Furthermore, processing can only achieve its maximum speed if the values reside in the (Level 1) cache of the processor. Because of the much lower speed of main memory, loads and stores to and from it should be kept to a minimum by doing all the operations on the data in one go.

These considerations and the real-time requirements for intelligent vehicles limit our choice of approaches from the previous section. For example, the current top ranking algorithms on the Middlebury stereo vision web site [2], based on the graph cuts technique, are not suitable due to their computational complexity and memory requirements. Our implementation therefore consists of several components, discussed in the previous section, that are more suitable for real-time operation. They can be combined to form different stereo algorithms.

The following sections explain how the components in our framework have been optimised.

3.3.1 SAD value computation

The SAD similarity measure can be computed efficiently by exploiting the fact that neighbouring windows overlap. For neighbouring windows with the same disparity, the overlapping pixels will contain equal absolute difference (AD) values. Therefore, a new SAD can be computed out of an old one by subtracting the values, which are only parts of the old window, and adding the values, which are only parts of the new window.

Fig. 3.3 shows how a new SAD value of a window of size width $2w+1$ and height $2h+1$ centred at $(x,y+1)$ can be computed from the previous sum of a window at (x,y). This process utilises the intermediate sums of AD values in rows of the same width as the matching window. Once such a sum is initialised at the beginning of a line, successive sums can be calculated recursively by first subtracting the most left AD value and adding the new AD value at the right-hand side. Calculated AD values for the additions are stored temporally in order to re-use them for the subtractions.

Two intermediate sums are applied in order to calculate the new SAD value of a window at $(x,y+1)$ from the old value of the window at (x,y). The first intermediate sum is used for the AD values that belong to the upper row of the old window. The second sum holds the AD values of the lower row of the window at the new position. By subtracting the upper row and adding the lower row, the new SAD value can be calculated. By repeating this process, a cascade function is created that only requires two pixel AD's, four subtractions and four additions in order to compute a new window sum.

The previously described steps can be executed for multiple disparity values simultaneously with SIMD type processing. For the computation of the AD values, SSE2 register $xmm0$ is filled with 16 copies of the left pixel value. The register $xmm1$ holds 16 pixels values from the right disparity interval. Only two saturated subtractions ($xmm0 - xmm1$ and $xmm1 - xmm0$) and the logical *or* of the results is needed to calculate the 16 AD's. The update of the intermediate sums and the SAD values themselves are carried out with regular SSE2 subtractions and additions.

3.3.2 Multiple windows

As explained earlier, the main deficiency of window based algorithms is their poor performance near object edges and on textureless regions. Multiple window approaches can improve the results.

Our implementation of the five window method by Hirschmüller et al. [35] uses a vertical and horizontal step. In the vertical compare step, newly computed SAD values are compared with the values of $2h$ processed image lines earlier. The results are temporally stored as the minimum and maximum values for the current centre image line of h image lines earlier. In the horizontal step, the vertical values on either side ($x-w$ and $x+w$) of a point are compared. The maximum of the left hand side is compared with the minimum of the right hand side and vice versa. The two resulting minimum values are also the two smallest values of the four windows. On average, only three compares are needed to find the smallest values with this method.

3.3.3 Left to right minimum search

An important step in a stereo algorithm is searching for the disparities with the lowest SAD values. Fig. 3.1 shows the search space for a disparity interval of $0 \leq d \leq max$. For a point l on the left image line, the possible matches with points on the right image line are indicated with light grey in the search space. Finding the correct match by searching the match with the lowest SAD value is computationally expensive because the whole disparity interval has to be searched.

In the work by Stefano et al. [81], a method was proposed for finding a minimum using multimedia extensions (MMX) instruction set that we have modified to work with SSE2. In this approach two types of SSE2 registers are used, one holding the range of disparity values and the other holding a copy of the associated SAD values. All values of two SAD registers can be compared in pairs using a single "compare smaller than" instruction. Both the smaller SAD values and accompanying disparity values can be selected afterwards, using the resulting binary mask. This algorithm can be used to process a large disparity range recursively, after which only eight values remain to be searched in order to find a single minimum.

3.3.4 Right to left minimum search

Disparity search can also be performed from right to left. Dark gray is used in fig. 3.1 to indicate possible matches on the left line for a right point r.

Right to left search does not require recomputation of the SAD values, the earlier computed left to right values can be reused. However, because of the arrangement of the values in memory, a different algorithm is needed in order to apply SIMD techniques. By evaluating the SAD value of multiple right points, instead of one point, this problem can be solved. Similar to the approach chosen for left to right search, we start out with an array holding the SAD values and another holding the disparity values. However, now the SAD values are compared with the SAD values in the next array. In order to align the next SAD values, the values are shifted down one position. By repeating these steps, all the SAD values in the disparity interval of a right pixel are compared sequentially. Eventually, the lowest SAD value and its disparity are the ones discarded from the arrays during the shift step.

3.3.5 Search optimising technique

Almost all real-time SIMD based algorithms known from literature use the simple WTA technique for disparity search. We have also implemented a search optimising method based on dynamic programming. It is based on a technique, proposed by Kraft

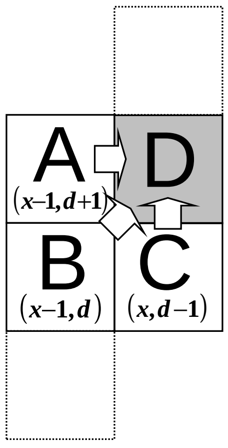


Figure 3.4: Possible predecessor points in cost propagation step. Predecessor point B has the same disparity as point D, while the points A and C are respectively an occlusion or part of a discontinuity.

and Jonker [46], that uses two stages; a cost value propagation stage and a collection stage for retrieving the best disparity estimates.

In the cost propagation stage, each disparity space point receives a cost value from a preceding point. The cost value for a left pixel x and disparity d is indicated by $LR(x,d)$, because left to right matching is used. The number of preceding points is limited to three possibilities, which are indicated for point D in fig. 3.4. Each preceding point has a different weight cost added to its accumulated cost value. Constant weights W_A and W_C are used for the points A and C which are occlusion and discontinuity points, respectively. The actual SAD value of point B is used as its weight because its disparity is equal to that of the current point. The predecessor with the lowest total cost value is selected as predecessor of the current point:

$$LR(x,d) = \min \begin{cases} LR(x-1,d+1) & +W_A \\ LR(x-1,d) & +\text{SAD}_B \\ LR(x,d-1) & +W_C \end{cases} . \tag{3.4}$$

Each point also stores the location of its predecessor. These references link up to form a path trough the disparity search space. At the end of the propagation stage, the best path is simply selected by searching for the lowest accumulated cost value. The best disparities are found in the collection stage by backtracking this path.

3.3.6 Subpixel interpolation

The algorithms of our framework estimate disparity with subpixel accuracy. Until now, we have described integer based operations with SSE2. The (older) SSE instruction set also contains SIMD operations for operations on 128 bit registers with packed 32 bit floats. We use SSE instructions for computing Eq. 3.3, which enables subpixel estimation for four points simultaneously.

3.3.7 Occlusion removal

We have implemented two types of algorithm components for removing pixels in occlusions and erroneous matches. The first type is the left-right check [35, 60]. The second type is the "recover" approach by Stefano et al. [81, 80]. Detected occluded pixels or erroneous matches are set to a predefined error code. This can be a value that is higher than the disparity maximum or simply zero. The rejected pixels are marked with an error code so that they can be identified for evaluation purposes.

Because of the conditional dependencies used in both approaches no straightforward application of SIMD is possible. Fortunately, each check only has to be executed once for every pixel. The required overhead is insignificant when compared to the other steps.

3.3.8 Algorithms

Using the described components we have created seven different stereo algorithms. All of them use the SAD computation method of section 3.3.1 and sub-pixel interpolation to enhance precision (section 3.3.6).

1. $\underline{SAD_L}$: Winner-Takes-All (WTA) approach that uses single window cost computation, left-to-right search (section 3.3.3) and no occlusion or error removal.

2. $\underline{SAD_{Rec}}$: Similar to 1, the recover approach (section 3.3.7) is used to remove occlusions or errors.

3. $\underline{SAD_{LR}}$: Similar to 1, also right-to-left search (section 3.3.3) is done in order to carry out the left-right consistency check (section 3.2.4).

4. $\underline{SAD_{MW5\,L}}$: Similar to 1, the cost computation is now based on multiple windows (section 3.3.2).

5. $\underline{SAD_{MW5\,Rec}}$: Similar to 2, the cost computation is now based on multiple windows (section 3.3.2).

6. $\underline{SAD_{MW5\,LR}}$: Similar to 3, the cost computation is now based on multiple windows (section 3.3.2).

7. $\underline{SAD_{DP}}$: Uses single window cost computation and the dynamic programming (section 3.3.5) disparity search optimisation technique.

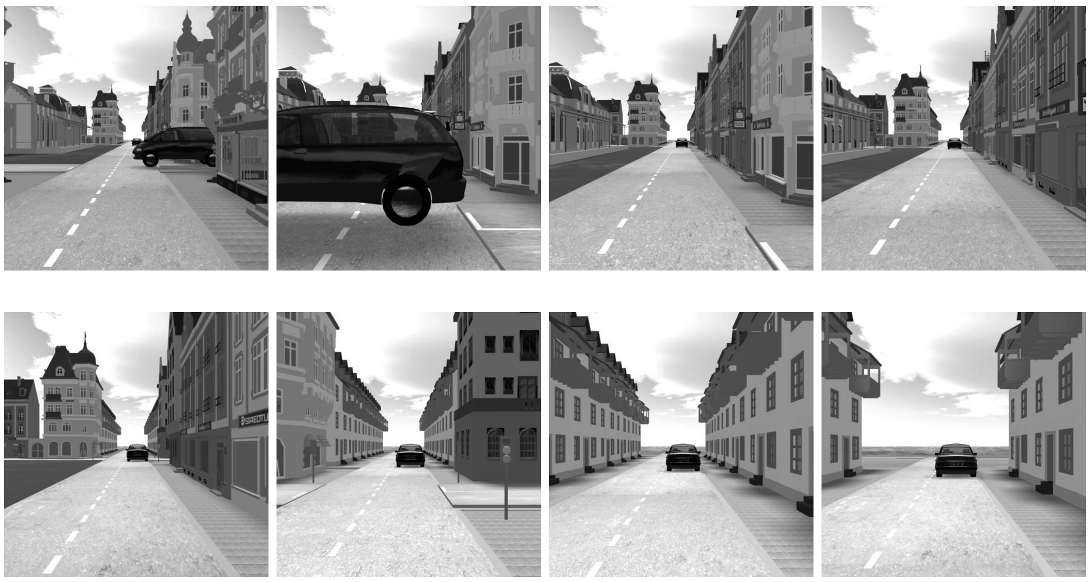


Figure 3.5: Virtual left stereo images from the sequence used for our experiments (frame nr. 20, 60, 100, 140, 160, 220, 260 and 300).

With the exception of SAD_L and $SAD_{MW5\,L}$, our algorithms will mark some pixels as rejected, because they are suspected to be occlusions or erroneous matches.

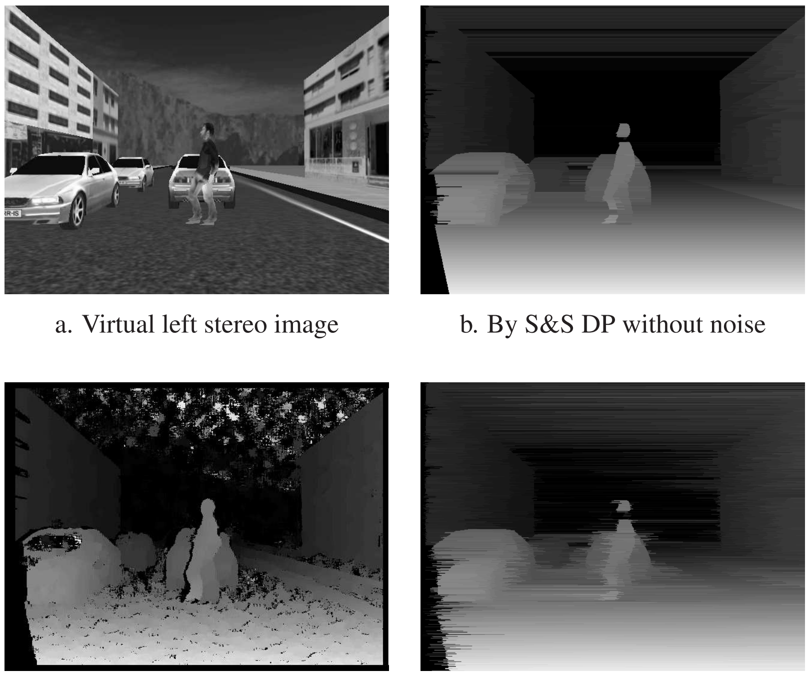
3.4 Experiments

The presented implementations have been tested and compared, together with four other publicly available algorithms. These are the SSD, DP and SO implementations, created by Scharstein and Szeliski (S&S) for their survey [75], that are available on the Internet [2]. Furthermore, an implementation of Birchfield and Tomasi's (B&T) [6] algorithm in the OpenCV library [3] is used.

The SSD algorithm is a WTA type algorithm like the majority of our implementations, however it uses the sum of squared differences for matching cost computation. Both DP and B&T use dynamic programming for searching the correct disparities. In contrast to the other algorithms, B&T does not use matching windows. Its measure is based on interpolating values between real pixels to achieve sampling invariance. The SO algorithm uses scanline optimisation for improving the disparity estimate.

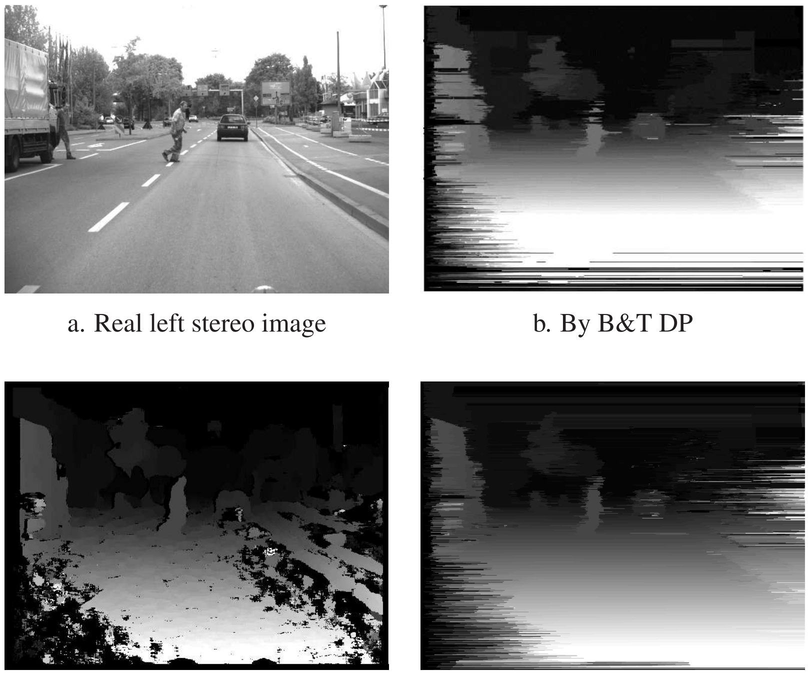
3.4.1 Stereo image test sequence

For evaluation purposes, several standard stereo image pairs with ground truth are available. A well know example are the stereo pairs of Tsukuba University. Unfortunately, the disparity range of these pairs is quite small (16 pixels) and the ground truth disparity is only given with 1 pixel accuracy.


a. Virtual left stereo image b. By S&S DP without noise

c. By WTA MW5 lr with noise d. By S&S DP with noise

Figure 3.6: Results with simulated stereo images. Except for the top left image, disparity images are shown where the values are indicated by a gray scale. Nearby disparity values are white and distant ones are black.


a. Real left stereo image b. By B&T DP

c. By WTA MW5 lr d. By S&S DP

Figure 3.7: Results with real stereo images. Except for the top left image, disparity images are shown where the values are indicated by a gray scale. Nearby disparity values are white and distant ones are black.

Another well known test set was introduced by Scharstein and Szeliski in their survey [75]. It does provide wider baseline stereo images with subpixel accurate ground truth disparity. However, in order to keep the acquisition of ground truth disparity simple, the images only contain planar surfaces.

Because our goal is to investigate the suitability of dense stereo vision algorithms for application in the intelligent vehicles domain, test images are needed that depict realistic situations in off-road or urban terrain. Unfortunately, none of the commonly used test sets resemble this type of data.

Instead, we use both real and synthetic data of urban traffic scenes to evaluate the algorithms. Compared to images of off-road terrain, the urban images do not contain large areas with natural textures such as grass, foliage or bushes. Evaluation with an urban data set is therefore a better test to see how the dense disparity estimators cope with insufficient image texture.

The real image sequences were recorded with a vehicle mounted stereo camera. These depict typical traffic scenes with obstacles such as pedestrians and other cars. Unfortunately, the real images do not come with a ground truth disparity. It is possible to obtain a ground truth disparity image with techniques such as active lighting [76]. However, this is not a practical approach for outdoor scenes and moving stereo camera rigs. Our analysis of the results with real stereo images is therefore limited to qualitative comparisons.

A stereo image pair and its ground truth disparity also can be generated synthetically from a 3D computer model. The MARS/PRESCAN software [48, 67] is a framework for simulation of different vehicle mounted sensors; such as radar, laser rangefinder or camera based systems such as stereo vision. With this simulator, a sequence was created of a virtual vehicle equipped with a stereo camera driving through traffic scenes. The used city-like scenery is complex and other moving vehicles are present.

As ground truth, the simulator provides the range images for each of the stereo images in the sequence. For our experiments, these are converted to disparity images.

The MARS/PRESCAN synthetic stereo images and ground truth data used in our experiments is publicly available for download from the Internet [4]. Some of the 326 images of this sequence, which have a resolution of 512 by 512 pixels and disparity range of 48 pixels, are shown in fig. 3.5.

3.4.2 Adding real image influences to synthetic images

We first looked at the qualitative similarities between the output disparity images generated with the simulated images and real images. The results of the simulated data were very good for all algorithms, see for example the DP result in the noiseless case (fig. 3.6b). This is caused by the fact that no image noise was added to the simulated images.

However, in the output for real stereo images we could more clearly see errors. An example of a real stereo image of a vehicle mounted camera is shown in fig. 3.7a. The output of dynamic programming approaches such as B&T and DP now shows a lot of "streaking" errors near object boundaries. WTA approaches such as in our framework and SSD, on the other hand, generate mainly errors on areas with insufficient texture. See for example the road surface in fig. 3.7c.

In order to approach the conditions of the real images more closely we added a number of stereo camera related perturbations. These perturbations are mainly due to the optics, sensor signal to noise ratio and the calibration of the stereo rig itself.

Light passing through the edges of a camera lens will hit the image sensor under a different angle than the light that passes trough the middle. The light rays at the image edges are scatted over a larger sensor area than those at the middle. This effect is known as "vignetting" and causes pixels far away from the image centre to be darkened. Fig. 3.8, shows the pixel weights that are multiplied with the original image pixels to add this effect. The weights where obtained with the $\cos^4$-law, which is the technique for calculating vignetting illumination fall-off [44].

In real cameras two forms of image noise are caused by the image sensor. The first type is called fixed pattern noise and is caused by physical differences between the light sensitive elements on the sensor. However, in almost all modern cameras the influence of this type of noise is negligible because non-uniformity correction is used. The other type is called temporal noise and is due to the sensors signal to the noise ratio. This type of noise was introduced to the synthetic images by adding white Gaussian noise with zero mean and a variance of 1 intensity level.

Because dense disparity estimation relies on steps for removing lens distortion and rectifying the stereo images, the stereo camera calibration itself is also a potential source of perturbations. In order to add this influence to the "perfectly" rectified synthetic images, we performed the undistortion and rectification steps on them with parameters based on typical residual errors of stereo calibration. Fig. 3.8 shows for each pixel in the distorted image the amount of displacement between its position and the location from which it was sampled in the original image. Pixels with a high displacement are indicated by light gray, while those with low displacement are indicated in black.

Outputs of the algorithms with these corrupted images now involve similar artifacts as observed in the real images, e.g. see $SAD_{MW5\,LR}$ and DP results in Fig. 3.6c & d and Fig. 3.7c & d.

3.4.3 Error measures

Several approaches to quantitative evaluation of stereo algorithms exists. One of the most simplest error measures is the averaged absolute mean error:

$$E_{abs} = \frac{1}{n}\sum_{i=1}^{n} |g_i - d_i| \ . \tag{3.5}$$

Where g is the estimate by the evaluated algorithm and d is the ground truth. The number of pixels in the image is defined by n. Larger errors can be accentuated by

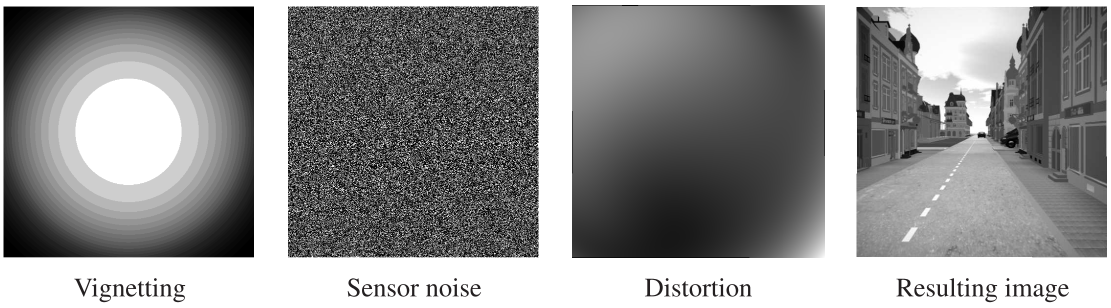


Figure 3.8: Influences on the quality of a stereo image pair.

using the squared error instead of the absolute error:

$$E_{sq} = \frac{1}{n}\sum_{i=1}^{n}(g_i - d_i)^2 \ . \tag{3.6}$$

The drawback of both error measures is that they do not distinguish well between disparity estimates with a lot of small errors and disparity estimates with only a few large errors.

Another error measure is the bad pixel percentage. It uses a threshold δ to set a maximum allowed absolute error. The absolute differences with the ground truth larger than this value are counted as bad pixels:

$$B = 100\% \frac{1}{n}\sum_{i=1}^{n}(|g_i - d_i| > \delta) \ . \tag{3.7}$$

In contrast to the previous two error measures, small errors are ignored while other errors are counted regardless of their magnitude.

It is difficult to relate the error measures presented so far to problems encountered when dense stereo vision is used as a sensor on an intelligent vehicle. Issues that are of importance here are the ability to detect objects such as obstacles and determine their range accurately.

Classifying a group of pixels as an obstacle requires that they are distinguishable from other background items such as the road surface, buildings or the sky. We therefore use the ground truth disparity from the MARS/PRESCAN simulator to divide pixels in each stereo image into four classes. These are foreground and background obstacles, road surface and sky. Pixels that have zero ground truth disparity are classified as being sky. Both pixels on the road and the curb belong to the road surface class. These are extracted from the ground truth with the V-disparity technique that will be discussed in section 3.4.7. Remaining pixels are regarded as obstacle pixels. A distinction is made by hand in the sequence between the two cars that are foreground obstacles and the buildings that are background obstacles. Examples of three pixel classes are shown in fig. 3.9.

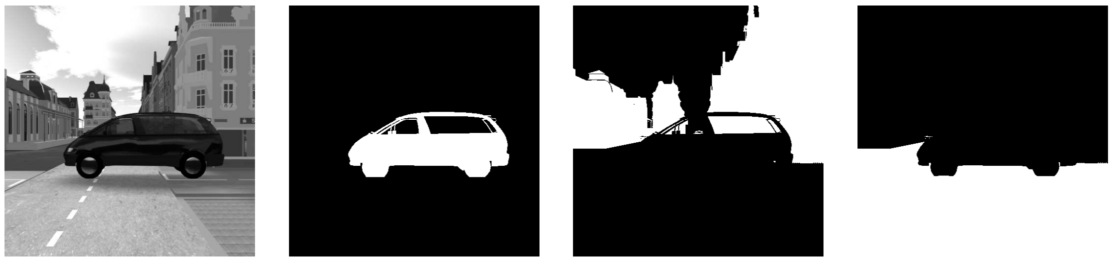


Figure 3.9: The different pixel classes used for evaluation.

A range can only be given for pixels that lie on surfaces, such as the foreground, background and road pixels. For these three pixel classes we define the estimation density D of a disparity image as:

$$D = 100\%\frac{m}{n} \quad \text{with} \quad m = \sum_{i=1}^{n} g_i > 0 \ . \tag{3.8}$$

This measures which percentage of the foreground, background or road surface pixels have been assigned a disparity estimate.

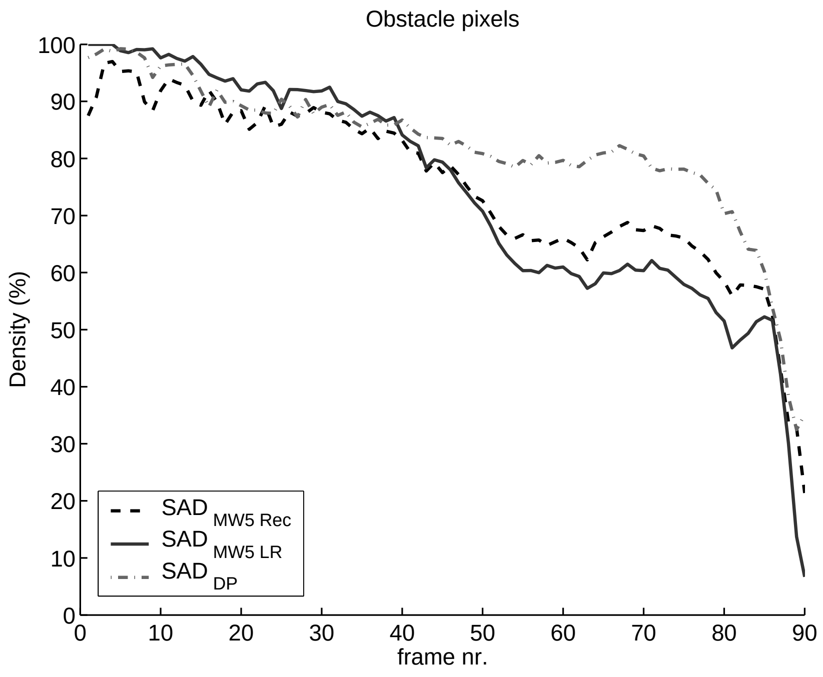
For stereo, range has an inverse relationship with disparity; small disparities correspond to large distances while large disparities correspond to small distances. Thus, an error in disparity for a far away point corresponds to a larger error in range than the same small error in disparity for a nearby point. The error measures of Eq. 3.5, 3.6 and 3.7 do not take this into account. In the Mean Relative Error measure the absolute error is divided by the ground truth disparity for each pixel. Therefore, this measure does actually relate to the expected error in range estimation.

$$E_{rel} = \frac{1}{m}\sum_{i=1}^{m}\left((g_i > 0)\frac{|g_i - d_i|}{d_i}\right) \ . \tag{3.9}$$

For intelligent vehicle applications it might be perceived that only the performance measures for the foreground obstacles are important. However, obstacle detection itself involves finding the correct foreground pixels among the other pixels classes. Since the road surface is more easily distinguishable than the (smaller) foreground objects it is actually searched first in some approaches. The performance measures for the other classes are therefore significant because bad estimates here can cause false positives.

3.4.4 Algorithms of the framework

The SIMD based algorithms of our framework and others from the public domain were tested with our sequence. The window based algorithms all used the same square window size of 9 by 9 pixels for cost computation. The weights for SAD_{DP} were set to



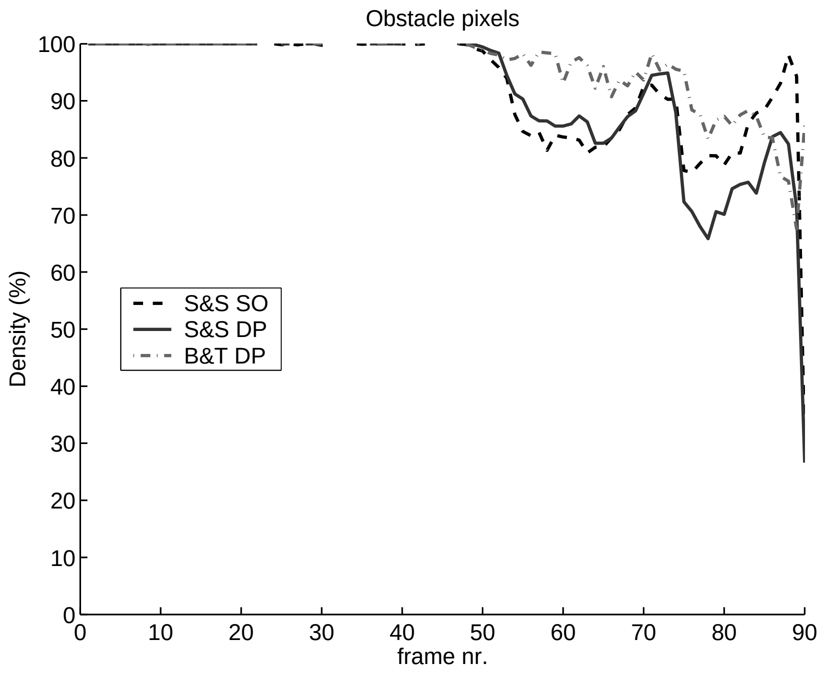
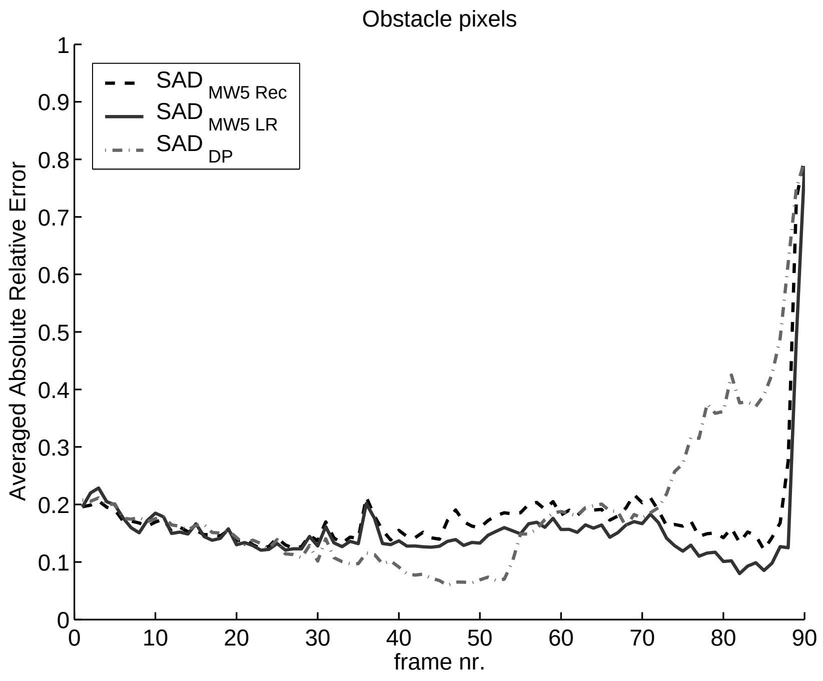


Figure 3.10: The estimation densities of six algorithms on foreground pixels.



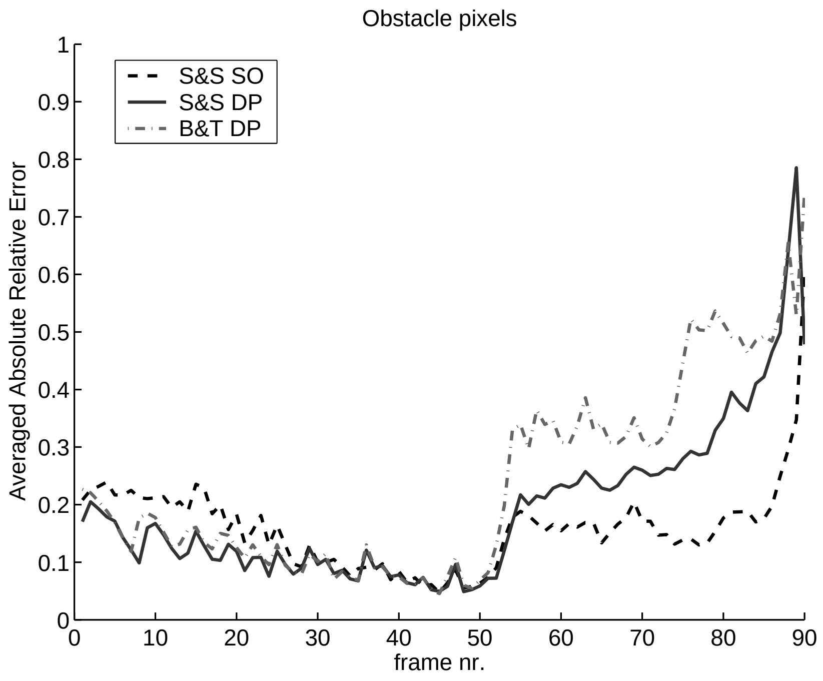


Figure 3.11: The mean relative error of six algorithms on foreground pixels.

Table 3.1: Results of the disparity estimators for different types of surface pixels.

	Foreground			Background			Road			All Surfaces		
	$R\%$	$D\%$	E_{rel}	$R\%$	$D\%$	E_{rel}	$R\%$	$D\%$	E_{rel}	$R\%$	$D\%$	E_{rel}
SAD_L	0	98.8	0.233	0	99.1	0.662	0	99.7	0.142	0	99.2	0.346
SAD_{Rec}	25.4	72.7	0.182	18.3	80.9	0.460	11.5	88.2	0.109	18.4	80.6	0.250
SAD_{LR}	26.8	73.0	0.156	17.7	82.0	0.386	9.0	90.9	0.100	17.8	82.0	0.214
SAD_{MW5L}	0	98.9	0.213	0	99.2	0.535	0	99.8	0.120	0	99.3	0.289
SAD_{MW5Rec}	20.9	75.8	0.172	14.4	84.3	0.397	11.0	88.1	0.098	15.5	82.7	0.222
SAD_{MW5LR}	24.5	75.3	0.149	12.4	87.4	0.340	7.6	92.4	0.088	14.8	85.0	0.192
SAD_{DP}	6.3	82.6	0.188	5.8	90.0	0.361	1.9	94.8	0.076	4.7	89.1	0.208
S&S SSD	0	98.2	0.239	0	97.2	0.574	0	98.8	0.159	0	98.1	0.324
S&S SO	0	93.9	0.153	0	97.5	0.457	0	95.4	0.126	0	95.6	0.246
S&S DP	0	92.8	0.188	0	99.2	0.335	0	95.6	0.087	0	95.9	0.203
B&T DP	0	96.3	0.233	0	99.6	0.333	0	99.2	0.110	0	98.4	0.226

$W_A = 34000$ and $W_C = 1000$. The tests were conducted on an Intel Pentium 4 3.2 GHz PC with 1.0 GB RAM.

The results for the full resolution frames 1 to 90 are shown in Table 3.1. For each of the surface pixel classes the estimate density $D\%$ and averaged relative mean error E_{rel} are shown. The rejection percentage $R\%$ indicates how many pixels are marked by our algorithms as an occlusion or error. Overall results averaged over all the surface pixels are shown in the fourth column titled “All Surfaces”.

We first studied the overall performance of the different algorithms from our framework. The algorithms that do not reject pixels show the highest error rates, see table 3.1. These are mainly caused by erroneous estimations in difficult areas such as occlusions and textureless regions. The algorithms with post-processing have lower error rates because they manage to reject many of these pixels. Comparing the percentages of pixels rejected by the recovery and the left-right approach, it is clear that the left-right approach rejects more pixels. Our DP approach rejects the least amount of pixels.

Considering the timing results for both half and full resolution frames in table 3.2, it is clear that the improvement by the post-processing steps comes at the price of a higher computational cost. The left-right consistency check is more expensive than the recovery approach. If the processing times of SAD_{Rec} and SAD_{LR} algorithms are compared to the time needed for SAD_L, the recovery approach shows a 10% increase, while the left-right check shows a 20% increase. The DP approach has the highest computational cost of our optimised algorithms. This is due to the fact that references have to be stored for back propagation phase, which increases the number of expensive memory operations.

It should be noted that the achieved run times only give an outlook on future performance because they have been achieved on general purpose computer hardware. For intelligent vehicles it is much more likely that more dedicated and low-power embedded SIMD hardware will be used.

As follows from table 3.1, the multiple window approach does improve results. It also decreases the number of rejected pixels. Of all our algorithms the $SAD_{MW5\,LR}$ algorithm has the lowest mean relative error for all surfaces pixels. From the timing

Table 3.2: Mean runtime for the tested algorithms. Performance is also shown in frames per second (fps). The algorithms are ordered by runtime.

	Image dimensions: (width, height, disparity)			
	(512,512,48)		(256,256,24)	
Algorithm	Runtime	Fps	Runtime	Fps
SAD_L	0.0879 s	11.3827	0.0189 s	52.9393
SAD_{Rec}	0.0976 s	10.2432	0.0202 s	49.5065
SAD_{LR}	0.1096 s	9.1240	0.0234 s	42.7709
$SAD_{MW5\,L}$	0.1665 s	6.0061	0.0322 s	31.0802
$SAD_{MW5\,Rec}$	0.1698 s	5.8905	0.0342 s	29.2219
$SAD_{MW5\,LR}$	0.1775 s	5.6350	0.0358 s	27.9397
SAD_{DP}	0.2456 s	4.0721	0.0466 s	21.4573
B&T DP	0.4594 s	2.1768	0.0742 s	13.4711
S&S SSD	5.3012 s	0.1886	0.7123 s	1.4039
S&S DP	6.5724 s	0.1522	0.8645 s	1.1567
S&S SO	10.5968 s	0.0944	1.0484 s	0.9538

results however, it is clear that the multiple window approach is expensive compared to the single window approach. The multiple window approach increases computation by about 50%.

3.4.5 Performance for different surface pixel types

The previous section provided an insight how well our various algorithms compare to each other and what computational cost they incur. In this section we investigate the performance figures for separate pixel classes. We have also included the results of other, publicly available algorithms in this analysis.

The relative mean error for all surface pixels in table 3.1 shows that the $SAD_{MW5\,LR}$ has the lowest overall error rate, followed by S&S DP as second best. However, this is not the case for all individual surface pixel classes.

For the background and road pixel classes it can be seen that approaches based on dynamic programming such as SAD_{DP} and S&S DP perform better than many of the WTA based algorithms. This is due to the fact that the road and background classes do not contain many disparity discontinuities, their disparity profiles adhere to the smoothness constraint. Because disparity discontinuities do occur at the edges of the foreground objects, the DP based approaches have more problems with these type of pixels. The WTA based algorithms that use the left-right check to remove errors are more successful on foreground pixels.

Regarding the processing time of various algorithms, table 3.2 reveals the benefit of the pursued SIMD SSE2 implementation. Our optimised algorithms are much faster than the SIMD non-optimised algorithms S&S DP, S&S SO, S&S SSD and B&T DP. The B&T DP is much faster than the S&S DP algorithm, because although both use dynamic programming, it is based on a pixel-based sampling invariant similarity measure, rather than the window-based measure used by S&S DP.

3.4.6 Analysis of performance variations

Until now, we have only considered averaged results for a part of the sequence. However, the performance of the tested algorithms clearly varies from frame to frame. This can seen in plots of the estimation density percentage and relative mean error per stereo image pair.

In fig. 3.10 two plots are shown of estimation density percentages ($D\%$) on foreground pixels for frame 0 until 90. The plot on left side shows the results of the $WTA_{MW5\,REC}$, $WTA_{MW5\,LR}$ and WTA_{DP} algorithms from our framework while the plot of the right side shows the results of S&S SO, S&S DP and B&T DP. Fig. 3.11 shows two plots for the same interval. This time, the relative mean error (E_{rel}) is shown for the six algorithms.

During this part of the sequence, a foreground object (a car) passes a road crossing. The vehicle drives into in the camera field of view from a side street. It is completely visible in frame 25. After frame 48 it starts to leave the cameras field of view. It is completely out of the field of view after frame 90.

If the different density percentage plots are compared it is clear that the algorithms from our frame work show a gradual decrease. This is due to the fact that the surface of the car contains little texture. In the beginning, when the car is far away, features such as edges, provide enough distinct points for the stereo matching. When the car is nearby more pixels in textureless areas are visible. This causes the decrease in estimation density.

Because WTA_{DP}, S&S SO, S&S DP and B&T DP use search optimisation techniques they are less effected by textureless regions. The plot of the latter three algorithms does show lower densities percentages after frame 48, which due to “streaking errors”. Because the disparity jump from background to foreground disparity is not visible anymore after frame 48, the dynamic programming technique wrongfully assigns zero disparities to some of the foreground pixels.

The effects of streaking errors by algorithms with search optimisation is more evident in the plots of fig. 3.11 where the relative mean error is shown. The algorithms $WTA_{MW5\,REC}$, $WTA_{MW5\,LR}$ that do not use search optimisation techniques show a fairly consistent error of about 0.2, while the other algorithms, that do use DP or SO show a sharp increase in error after frame 48.

3.4.7 Ground plane estimation experiment

In the previous sections we have studied the quantitative results of the different disparity estimators. We learned that the algorithms which use global search techniques are more effected by the scene complexity. In this section we will show some of the

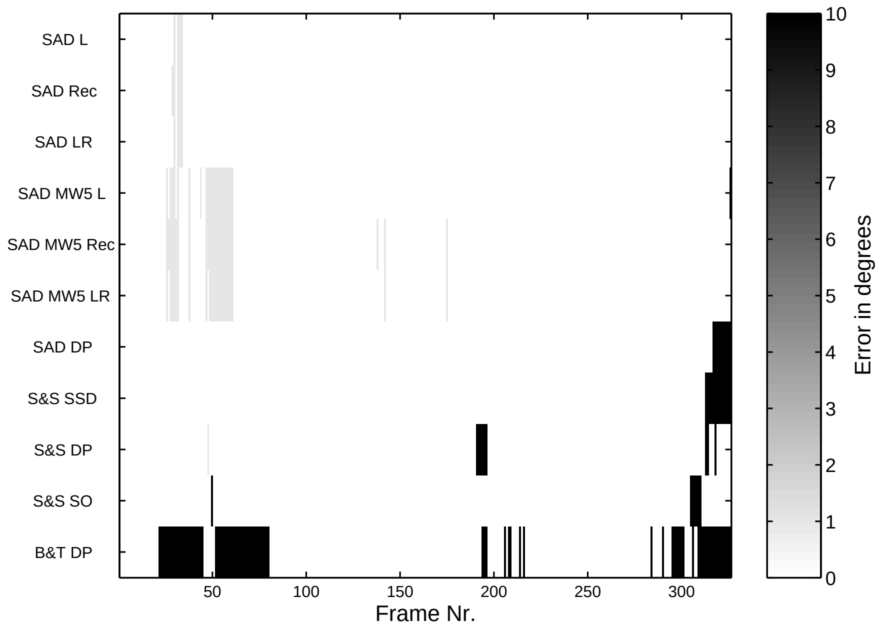


Figure 3.12: Error in ground plane angle estimation based on V-disparity.

consequences for a typical application of stereo vision in intelligent vehicles: ground plane estimation.

Ground plane estimation is required to distinguish obstacles such as other cars and pedestrians in the disparity image from the road surface. A robust and real-time method for doing this was developed by Labayrade et al. [47]. It is based on the assumption that for scanlines where the road surface is visible, the dominant disparity value is that of road surface pixels. In their method, each scanline is first converted to a disparity histogram. Dominant disparity values, belonging to the road surface, can now be found by looking for histogram bins with large pixel counts. The collection of all scanlines converted to histograms is called the “V-disparity” image. Its width is determined by the number of disparity bins and its height is equal to the original image. The road surface profile can be extracted from the V-disparity image by finding line features consisting out of concatenated bins with high pixel counts. The original method of Labayrade et al. [47] uses the Hough transform, to locate line features and approximates the road vertical curvature with a piecewise linear curve.

For our experiment, we use a simplified version of the Labayrade et al. [47] approach. To test an estimated disparity image it is first converted to a V-disparity image. Because the road surface of the synthetic images is flat, only a single dominant line feature is searched in the V-disparity image with the Hough transform. This line is then compared to the line found by the same method in the ground truth disparity image. The difference in angle between the two lines shows how ground plane estimation is affected by the quality of the disparity image.

In fig. 3.12 the differences in ground plane angle is shown for all images from test sequence. Each column shows the errors indicated by a gray-scale in degrees for one of the tested algorithms. These results show that error made by algorithms which use optimisation steps, affect the ground plane estimation more than the simpler approaches. The SAD_{DP}, S&S DP, S&S SO and B&T DP show severe errors in ground plane angle estimation in some the frames. Both the $SAD_{MW5\,L}$ and S&S SSD also show severe errors because they do not use rejection steps. It is also interesting to see that the multiple window approach, when compared to the single window approach, actually increases the error slightly for some parts of the sequence. This is due to the fact that V-disparity assumes that a majority of disparities belong to the road surface. It can therefore become biased in situations where this is not true. The multiple window approach gives more estimates for the textureless surface of the passing car than the single window approach. Surprisingly, this influences the ground plane estimation when the car is nearby and fills a large part of the image.

3.5 Conclusion

Application of dense stereo vision in intelligent vehicles requires accurate and robust disparity estimation algorithms that can run in real-time on small and power efficient computing hardware. Dense stereo vision algorithms which are based on Single Instruction Multiple Data processing are interesting because this type of instruction level parallelism is currently available on various normal, low power embedded and dedicated computing systems.

We implemented several real-time algorithms to investigate how well different approaches to dense stereo vision can benefit from SIMD optimisation and compare them on an equal basis. They are all based on a single framework that provides components for performing SAD cost computation, multiple window selection, disparity search based on WTA or dynamic programming and post-processing for occlusion or error rejection. We came up with fast SIMD optimised versions for most of these components using the SSE2 instruction set.

In order to test the algorithms under the varying conditions that can be encountered in city-like traffic, we used stereo images from a vehicle sensor simulator. In contrast to the commonly used single pair test images, our sequence contains hundreds of images with varying geometric and image properties. Effects that degrade image quality in real stereo cameras, such as vignetting, sensor noise and imperfect image undistortion and rectification were also added to enhance realism.

The first set of tests show that the post-processing steps can help reduce error percentages at the cost of a sparser disparity map. More advanced algorithms with multiple window technique or dynamic programming improve the disparity estimates on difficult image areas, albeit at higher computational cost. Nevertheless, the slowest of our algorithms still achieves real-time processing speeds.

We also studied the performance of our algorithms together with a set of four other publicly available stereo algorithms for different types of foreground disparity pixel classes. The results confirm that non-greedy matching algorithms (i.e. scan line optimisation, dynamic programming) perform better on surfaces with low geometrical complexity such as the road surface. However, the more simpler WTA techniques combined with error detection techniques can outperform these algorithms on more challenging surfaces such as nearby obstacles. This is caused by the fact that search optimisation algorithms can miss important disparity jumps at the edges of nearby objects and therefore assume the wrong disparities in later search or optimisation stages.

The consequences of these type of errors are demonstrated with an road surface inclination estimation experiment. The results show that ground plane estimation based on disparity estimates of the algorithms which do use optimisation techniques are less reliable.

Our research has shown that simple WTA techniques for dense stereo, combined with robust error rejection schemes such as the left-right check, are more suitable for intelligent vehicle applications. Because they do not use search optimisation techniques their processing speed is higher. Our investigation has also revealed that the basic search optimisation techniques such as dynamic programming can cause errors which interfere with subsequent steps needed for intelligent vehicle related sensing tasks.

The results suggest that future improvement of dense stereo algorithms can be achieved by applying search optimisation techniques only to the parts of the stereo images that can benefit from them, as discussed above. Preprocessing steps to detect textureless and edgeless regions could be exploited to detect which pixels are suitable.

Chapter 4
Obstacle Detection for Unstructured Environments

4.1 Introduction

Detection of obstacle hazards is an important issue for autonomous robot navigation in unknown terrain. Camera based techniques, such as stereo vision, can be applied to distinguish obstacles from drivable surfaces. However, the actual image processing involved, depends largely on the environment characteristics. If the environment is man-made, the surroundings are predominantly of an artificial nature. Generally, there is a flat ground surface to facilitate walking or driving. Paved road surfaces and office corridors are good examples of man-made environments. Visually, the road edge and corridor walls have clearly visible and repeating boundaries. Vision based approaches to autonomous vehicle guidance often exploit the presence of these visual features and the relatively simple geometry of artificial man-made environments. Already in 1987, Dickmanns et al. [19] demonstrated that visually detected lane markers and road edges of a highway could be used to guide an autonomously controlled van up to speeds of 96 km/h. Another advantage of only using feature points is that it reduces the computational loads of subsequent processing steps. In stereo vision approaches to vehicle guidance this is known as sparse stereo. Several sparse stereo and real-time approaches [25, 47, 61] to obstacle detection have been developed for on-road vehicle applications.

Outdoor terrains such as forests, dunes or fields are examples of natural environments. These environments can contain man-made objects such as poles and fences. However, they will mainly contain naturally occurring structures such as rocks, trees, bushes and sand dunes. These examples of natural structures do not have a constant or fixed shape and do not occur at regular intervals. They are much harder to distinguish visually. Furthermore, it is also not possible to assume that all drivable surfaces are flat. There might be steep inclines, such as sand dunes, the vehicle cannot traverse. The vehicle inclination can also change due to the non-flat ground surface. This influences the cameras relative orientation with the ground surface and view of obstacles.

Another problem is the presence of holes, ditches and trenches in unstructured environments. Because they are mainly located below the ground surface, these types of obstacles are often referred to as “negative obstacles”. The term “positive obstacles” is used to indicate obstacles above the ground surface.

In natural terrain it is much more difficult to rely on approaches that only use sparse image features. Objects such as steep sand dunes only have strong edge features near their object boundaries. However, the sand surface itself often lacks such features. In this situation, a feature based method cannot determine the steepness of a sand dune due to lack of information. Detection of negative obstacles, such as trenches is also difficult. Because only sparse range information is available on its edges, it is difficult to distinguish a trench edge from other normal object edges. The lack of flat ground surface geometry and reliable image features makes it necessary to use a dense approach to stereo vision in unstructured terrain.

In this chapter we will discuss our approach to stereo based obstacle detection in off-road terrain. The scope of the research is limited to the detection of positive obstacles. We will present an efficient approach that detects and clusters obstacle pixels into objects. It includes a novel method for dealing with the reduced range accuracy of stereo vision at larger distances. Our method also estimates the orientation difference between the cameras and the ground surface to compensate for uneven ground surface geometry. Unlike the majority of published research on the topic of obstacle detection in rough terrain, the evaluation of our method is not limited to qualitative images that demonstrate its capabilities. Several quantitative tests have been devised and conducted in order to evaluate the performance.

In the next section we will discuss several approaches to obstacle detection in off-road terrain known from literature. Section 4.3 describes our fast approach for searching obstacle and clustering points. It also explains a method for updating the ground plane orientation while driving over rough terrain. Results are presented in section 4.4. Conclusions can be found in section 4.5.

4.2 Approaches to obstacle detection

In this section, several approaches to obstacle detection from literature are reviewed that take a (dense) disparity image, obtained from stereo vision, as input.

A popular approach is to first estimate the ground plane. The idea behind this approach is that only obstacles remain present after removal of ground plane pixels. One of the techniques for ground plane estimation was introduced by Labayrade et al. [47]. This approach was later adopted by Broggi et al. [9] for obstacle detection on unpaved roads.

The main assumption behind this method is that the ground surface is mainly flat and is a dominant image feature. Therefore, there will be many image lines where a majority of the pixel disparities, stand out from the other disparities because they belong to the same flat surface. An example of a left stereo image recorded in a car

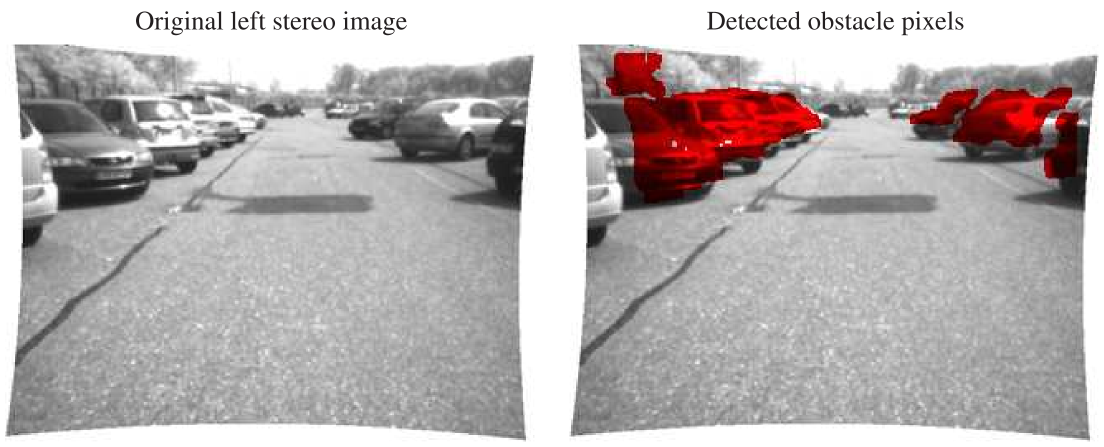


Figure 4.1: On the left, the original left stereo image is displayed. Detected obstacle pixels by the V-disparity technique are coloured red in the image on the right.

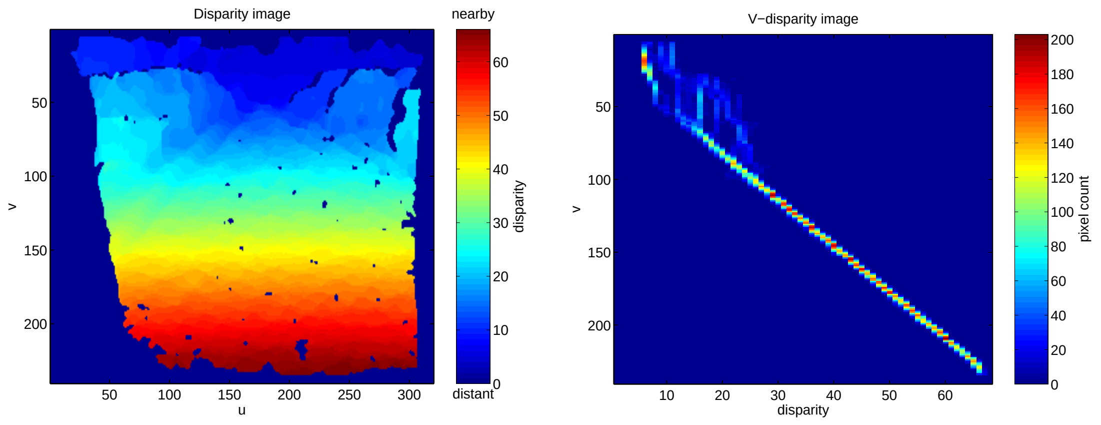


Figure 4.2: The stereo disparity image is shown on the left side. On the right side, the corresponding V-disparity image is shown.

park is shown on the left side of fig. 4.1. The estimated disparity image for this stereo pair is displayed on the left of fig. 4.2. It can be seen that the ground plane occupies a large area in both images.

A simple histogram technique is applied to find the ground plane. This involves converting each horizontal line of the input disparity image to a disparity histogram. Each histogram has enough bins to cover the integer disparity range of the original image. The result of converting all lines is called the "V-disparity" image. It has the same height as the original image while its width is equal to the number of histogram bins. The resulting V-disparity image of our example is shown on the right side of fig. 4.2.

If there are many pixels on a image line with the same disparity there will be a peak in the corresponding disparity histogram. In the V-disparity image, these peaks concatenate into a diagonal line that indicate disparities for a flat ground surface at the various images lines (v-pixel coordinate). This line is clearly visible in the example of fig. 4.2.

Obstacle pixels can now be easily identified because they correspond to points above the line in the V-disparity image. Uneven terrain, such as a slope, can be approximated by a piecewise linear function. Such an approximation can be found by applying the Hough-transform to the V-disparity image. Obstacle pixels identified in our example, by approximating the ground surface with a single line, are indicated in the image on the right side of fig. 4.1.

Approaches like V-disparity can estimate the inclination angle between the cameras and the ground plane. This is an advantage, because it reduces the sensitivity of the method to sudden changes in the vehicles pitch angle while driving over rough terrain.

The success of ground plane estimation depends on the ability to isolate points that belong to the ground surface. Unfortunately, there are situations where there is no clear ground plane. For instance, the vehicle camera system might be facing a vertical wall. A naive approach to ground plane detection might decide that the wall is the ground plane because a majority of the pixels belong to it. It is therefore necessary to always check the geometry of the estimated ground plane. Cases where there are not sufficient ground surface pixel disparities available, will also cause problems. This can occur in unstructured terrain when the vehicle is confronted with a negative obstacle such as an abyss. Or when the ground surface has little texture and the disparity estimation algorithm fails to find good stereo correspondences.

The use of a digital elevation map (DEM) [101, 97, 93] is an alternative for ground plane estimation. In order to create a DEM, the three-dimensional space in front of the stereo camera is carved up by a two-dimensional grid associated with the ground plane. Each grid cell corresponds with a certain area with fixed dimensions. The height of each cell can be obtained by determining which of the reconstructed stereo points fall within its bounds and computing their average elevation. Cells that are part of positive obstacles will be conspicuous, because their height value is larger than those that are part of the ground surface. Gradient analysis can be used to locate such obstacle cells in the DEM.

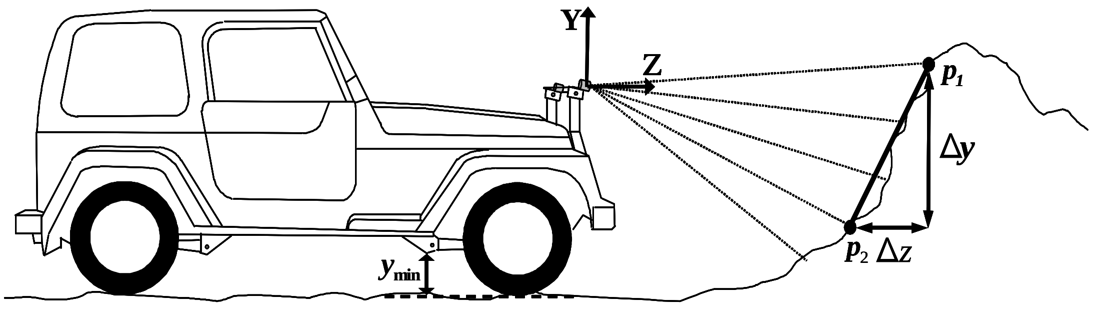


Figure 4.3: Example of geometry based positive obstacle detection.

Using DEM's has a number of advantages. It allows the computation of the terrain incline, which can help to determine whether or not an area is to steep for a vehicle to drive on. Furthermore, because DEMs are already in the correct format, they are convenient to use in map building applications. There are also a number of disadvantages associated with DEM's. A problem is the number of reconstructed stereo points that fall within grid cells at various distances. Because the range resolution of stereo is at its highest level nearby, the number of stereo points per grind cell will become progressively smaller at larger distances. This can lead to cells with unreliable or missing height values. Another problem is how to choose the dimensions of the grid cells. Small obstacles might not be detected if the cell size is set too large. On the other hand, the DEM can become susceptible to noise or missing values if the cell size is set too small. Finally, with a DEM comes the assumption that the orientation difference between cameras and ground plane remains the same. This can lead to problems if the vehicle undergoes roll and pitch motion while driving over rough terrain.

Geometry based approaches are an alternative to the ground plane estimation or DEM based obstacle detection approaches. An early example is the approach by Matthies et al. [57]. In their approach, each image column is regarded as a plane that vertically intersects the terrain in front of the vehicle. A range profile of the planes intersection with the terrain can be obtained from its image column disparity values. Obstacles are searched by analysing the one dimensional data in the range profile. This is illustrated for positive obstacles in fig. 4.3. The figure shows a side view of a vehicle mounted stereo camera that observes the points p_1 and p_2 on a positive obstacle. It can be seen that points have been selected that are separated by a fixed camera elevation angle. The distance Δz and height Δy differences between the points can be used to calculate the slope angle. If the slope angle is too high to be cleared by the vehicles ground clearance y_{min}, the pixels are indicated as obstacles.

Schäfer et al. [74] also process disparity images column-wise. However, they only use the disparity information to find positive obstacles. Their method uses a state machine that can switch between non-obstacle and obstacle states based on the differences between neighbouring disparities. While traversing the column from top to

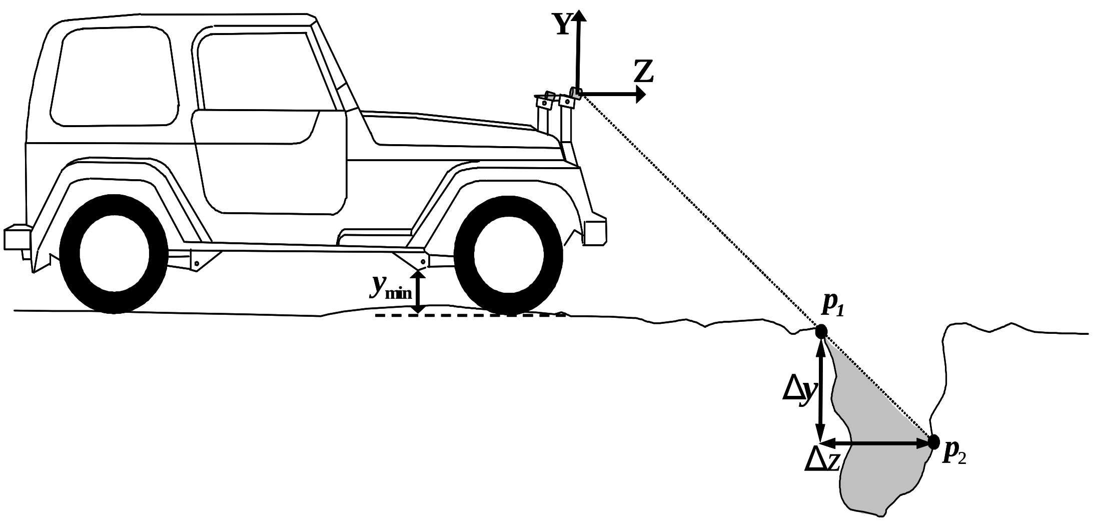


Figure 4.4: Example of geometry based negative obstacle (trench) detection.

bottom, large disparity changes signal the machine to go to an obstacle state. The machine remains in this state if the subsequent changes are small. If not, the state changes back to non-obstacle. The advantage of this approach is that overhanging obstacles, such as tree trunks, can be distinguished from normal positive obstacles. However, the approach requires an extra pre-processing step to fill in gaps in the disparity map. Missing disparity values or errors might disrupt the state machine.

Negative obstacle detection is also possible by column-wise processing. In the work by Manduchi et al. [52], gaps are searched in the reconstructed range profile. Because occlusions can also create gaps, the algorithms also searches for the downwards sloped edge visible at the opposite side of the negative obstacle. This is illustrated in fig. 4.4, where a vehicle mounted stereo camera observes a trench. The camera occluded area of the trench is indicated with gray. Therefore, the camera cannot see beyond point p_1 on the nearby edge and the point p_2 on the opposite side.

Manduchi et al. [52] have also proposed a more advanced method that both detect and cluster obstacle points. It uses two criteria to check if two reconstructed points belong to the same obstacle surface. The first criterion states that the vertical distance between two points of an obstacle must not exceed a pre-set maximum y_{max} and must be larger then a certain minimum y_{min}. The minimum is enforced in order to exclude points on objects that are too low to be an obstacle. The space between the minimum and the maximum enables clustering on surfaces that miss disparity values due to occlusions or errors.

The second criterion demands that the line connecting both points must have a sufficiently high angle θ with the ground plane, because positive obstacles have predominantly vertically oriented surfaces. Fig. 4.5 shows a number of reconstructed three-dimensional points. For one point, indicated by s_1, the space is indicated that

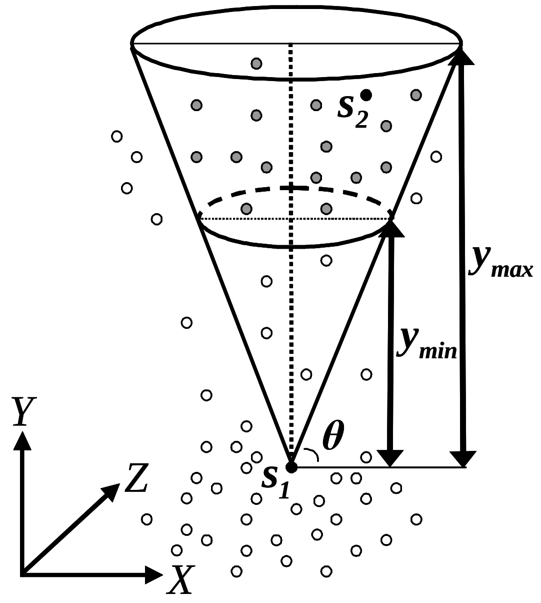


Figure 4.5: Inverted cone shaped detection area for obstacle points.

satisfies both criteria. Points in this space have been coloured gray instead of white. It can be seen that this space is an up-side down cone, limited by y_{min} and y_{max}.

In the original approach [52], the cone dimensions are projected onto the image to define a search window for each point. Every point in this search window is explicitly checked using the two criteria. Compatible points that are found are marked as being part of the same obstacle surface. For each obstacle, a search tree data structure is used to keep track of the points that belong to it. If two points from different obstacle trees are found to be compatible, their trees are merged into one that represents a single obstacle.

Clustering of point into obstacles opens up the possibility of geometric reasoning. Simple properties of the clusters such as number of points, volume of the enclosing box or height can be used to reject false detections. The ability to both detect and cluster obstacle points simultaneously is therefore a major advantage of the approach.

The computational complexity of this approach scales with the search windows dimensions. These in turn, depend on the choices for y_{min} and y_{max}. Due to the variation in terrain range, the sizes of windows also depend on the distance from the camera. Nearby, the windows are large and they become smaller for larger distances. Therefore, an image that contains more nearby points is computationally more expensive than an image with only far away points.

A more general problem with most of the geometry based methods is the assumption that the Y-axis is always perpendicular to the ground plane. Due to rough terrain, the vehicle can undergo roll and pitch motions. These motions change the stereo camera orientation relative to the ground surface. Detection might fail to find the correct

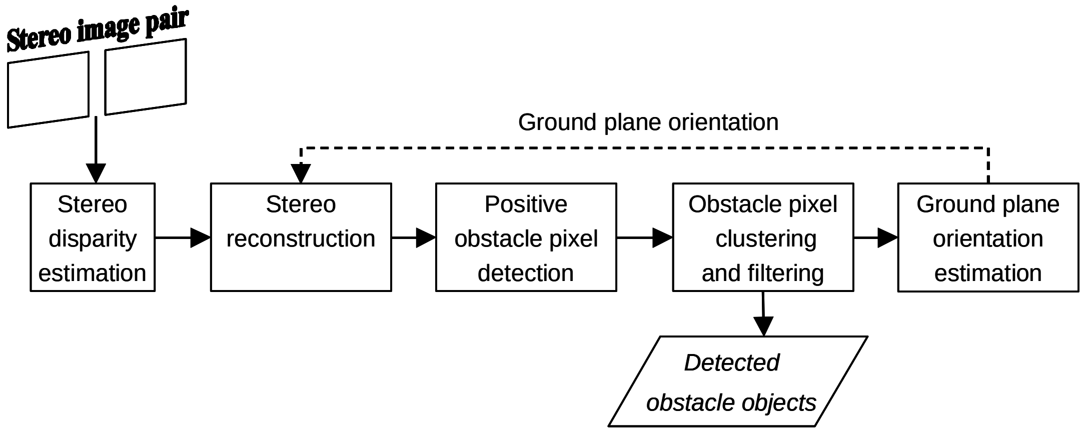

Stereo image pair
Ground plane orientation
Stereo disparity estimation
Stereo reconstruction
Positive obstacle pixel detection
Obstacle pixel clustering and filtering
Ground plane orientation estimation
Detected obstacle objects

Figure 4.6: Diagram of the main components of our approach. As input it takes a stereo image pair of unstructured terrain. The result is a list with the detected obstacle objects.

obstacle surfaces if the Y-axis orientation is not adjusted to these changes.

4.3 Obstacle detection with uncertainty

In the following sections we will present our approach to the detection of positive obstacles in rough outdoor terrain. A diagram of the main components in our approach is shown in fig. 4.6. It takes an image pair captured by a vehicle mounted stereo camera as input. First, disparity estimation is performed. The resulting disparity image is converted into a set of three dimensional points by applying stereo reconstruction. In the positive obstacle pixel detection step, points are searched in this set that belong to positive obstacle surfaces. The next step clusters the resulting points into a list of single object instances, which is the output of our method. As a final step, the current ground plane orientation is estimated. This new orientation is provided to the reconstruction step in order to improve the subsequent obstacle detection steps.

4.3.1 Stereo disparity estimation

The disparity estimator takes as input an image pair captured by the robot vehicle stereo camera. However, before disparity can be estimated, the stereo images require some preprocessing steps. Optical distortions such as radial and tangential distortion have to be removed because a pin-hole model for camera projection is assumed. More details on pin-hole camera projection and optical distortions can be found in chapter 2, sections 2.2.1 and 2.2.2.

Furthermore, the images have to be rectified in order to insure that the epipolar lines run in parallel with the image horizontal lines. Geometrically, this corresponds to a stereo camera where there is no orientation difference between the cameras. Its baseline is parallel to the epipolar lines. A more detailed explanation of the geometry involved in stereo imaging can be found in section 2.2.3 of chapter 2.

The parameters required for these operations can be obtained by performing a camera calibration procedure [33, 102]. The operations themselves are simple bilinear image interpolations that require pixel source coordinates and interpolation coefficients. These only have to be calculated once, for example during the necessary calibration procedure, and can be reused to enable fast preprocessing.

When the images have been undistorted and rectified, the disparity d between a corresponding left point l and a right point r horizontal coordinate, can be expressed as:

$$d = l - r\,. \tag{4.1}$$

In previous work, we developed several real-time dense disparity estimation algorithms [92] (chapter 3). For the obstacle detection in this study we use the single matching window variant with the left-right consistency check for error and occlusion detection (SAD_{LR}). Interpolation is used to obtain sub-pixel accurate estimates for the disparity values.

4.3.2 Parallel stereo reconstruction

The result of the disparity estimation is an image with values for pixels that do not lie on occlusions or are rejected as erroneous estimates. In the stereo reconstruction component, the three dimensional points that correspond to these pixels are recovered. The geometry involved uses the models for the normalised pin-hole projection and the parallel stereo camera that where explained in chapter 2, sections 2.2.1 and 2.2.4. Here, a coordinate system is assigned to each of the cameras in the stereo pair. It is indicated by XYZ for the left camera and by $X'Y'Z'$ for the right camera. Because the parallel cameras, both systems are separated by the baseline distance b that is limited to a translation over the X or X' axis.

Suppose that both cameras observe the same space point. In the left camera reference frame it is indicated by vector $\boldsymbol{r}$, while it is also indicated by vector $\boldsymbol{r'}$ in the right camera reference frame. Due to the camera projection, there is also a left $\boldsymbol{p}$ and a right $\boldsymbol{p'}$ image vector. Both vectors are related by the disparity d in the following way:

$$\boldsymbol{p'} = \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} x-d \\ y \\ 1 \end{pmatrix} \quad \text{and} \quad \boldsymbol{p} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}\,. \tag{4.2}$$

Here, the scalars x and y are the horizontal and vertical image positions.

In order to reconstruct $\boldsymbol{r}$ or $\boldsymbol{r'}$, the scalar difference λ has to be estimated from the disparity d and the camera baseline distance b:

$$\boldsymbol{r} = \lambda\boldsymbol{p} \quad \text{and} \quad \boldsymbol{r'} = \lambda\boldsymbol{p'} \quad \text{with} \quad \lambda = \frac{b}{d}\,. \tag{4.3}$$

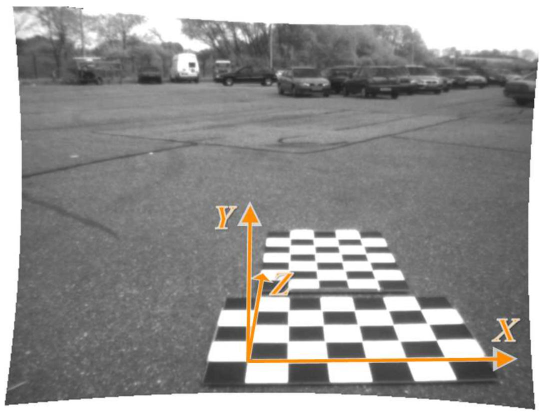


Figure 4.7: Image used for ground plane calibration. The XZ-plane is defined by the calibration plate surface.

A problem in stereo vision is that the disparity d cannot be estimated precisely. Errors in the estimate for d will have consequences for the precision of λ and consequently for $\boldsymbol{r}$. It is assumed that the distance errors in λ have a normal distribution with variance $V[\lambda]$. The propagation of the disparity error into the distance variance can be calculated with a method developed by Kanatani [40]:

$$V[\lambda] = \frac{2\varepsilon^2\lambda^4}{b^2} = 2\varepsilon^2\frac{b^2}{d^4} \quad . \tag{4.4}$$

Here, ε indicates the noise level in the vertical pixel coordinates y and y' relative to the epipolar geometry of the stereo camera. Because the images have been rectified and the disparity is estimated only along the scanline, y appears to be equal to y'. However, we assume that the pixel error with epipolar geometry is isotropic over the two-dimensional image plane. Therefore, ε is chosen to be equal to the error in the disparity estimate. In our disparity estimation algorithms [92], a quadratic method is used to obtain a sub-pixel disparity estimate. It is mentioned in the work by Hirschmüller et al. [35] that applying this method leads to an interpolation factor of approximately $1/8$ disparity. The value of ε is therefore set to $1/8$ of a pixel.

Due to the distance uncertainty, the position of stereo reconstructed points can only be determined up to a certain degree. Because we only consider errors in distance, the uncertainty is limited to a line emanating from camera, with the same direction as vector $\boldsymbol{p}$. The probability that the point is located on the line piece between the vectors $\boldsymbol{r}_{\min}$ and $\boldsymbol{r}_{\max}$, depends on the choice for parameter σ:

$$\boldsymbol{r}_{\min} = \left(\lambda - \sigma\sqrt{V[\lambda]}\right)\boldsymbol{p} \quad \text{and} \quad \boldsymbol{r}_{\max} = \left(\lambda + \sigma\sqrt{V[\lambda]}\right)\boldsymbol{p} \quad . \tag{4.5}$$

In our approach, we use $\sigma = 3.0$ so that it encompasses about 99% of the distance uncertainty.

Reconstructed vectors from stereo are defined in the left or right camera reference frame. For obstacle detection, it is more convenient to use a reference frame that is associated with the terrain. In our approach, we choose a reference frame where the the XZ-plane is parallel to the ground surface. The Z-axis points away from the vehicle, while the X-axis is orientated to the right side. Converting reconstructed vector $\boldsymbol{r}$ to vector $\boldsymbol{s}$, in this reference frame, involves a rotation defined by matrix $\boldsymbol{R}$:

$$\boldsymbol{s} = \boldsymbol{R}\boldsymbol{r} \, . \tag{4.6}$$

The same rotation can also be applied to convert both vectors $\boldsymbol{r}_{\text{min}}$ and $\boldsymbol{r}_{\text{max}}$ to $\boldsymbol{s}_{\text{min}}$ and $\boldsymbol{s}_{\text{max}}$.

For our experiments, we obtained an initial estimate for rotation matrix $\boldsymbol{R}$ while the vehicle is parked on a flat and paved surface. The procedure, that only has to be performed once, involves a large plate with a checkered pattern. The intersections between the checkers are corner features, arranged in a grid of known dimensions. For the measurement, the plate is placed on the ground in the field of view of the left camera. Fig. 4.7 shows the plate as viewed by the left stereo camera. It also shows the reference system associated with the plate. The XZ-plane is associated with the corner grid that can easily be extracted with image processing. A transformation from the cameras reference system to that of the plate involves a rotation and translation. We use the technique developed by Zhang [102], to estimate these extrinsic camera parameters. It bases its estimate on the corner image locations and their known location in the grid. The resulting rotation is the initial estimate for the matrix $\boldsymbol{R}$ that is used in the stereo reconstruction. While the vehicle is driving, it is updated automatically by a ground plane orientation estimation technique that will be described later on in this chapter.

4.3.3 Positive obstacle pixel detection

By applying stereo reconstruction, a disparity image can be converted into a large set of three-dimensional points. The problem is now to identify the points among this set that belong to obstacle surfaces. The method for detecting obstacle points is explained in this section.

Our approach is based on the method introduced by Manduchi et al. [52], because it enables the clustering of obstacle points into objects. This has the advantage that false positive detections can be rejected based on the geometric properties of the clustered object points. As discussed in the literature review, the original approach uses two criteria to check whether or not two points belong to the same obstacle surface. We have developed an approach where this check can be computed more efficiently. Furthermore, our approach also enables the inclusion of uncertainty in the stereo distance estimates.

Threshold based detection with uncertainty

The two criteria that decide whether two points belong to the same obstacle surface can be understood as a bounded space that limits the possible point locations. As is illustrated for a single point $\boldsymbol{s}_1$ in fig. 4.5, this space resembles a truncated up-side-down cone. Only points that are located within the space satisfy the two criteria. Fig. 4.8 shows the space on the right side and its image projection on the left side. Projected to the image domain, the space is reduced to a trapezoid shaped area. Its dimensions depend on where point $\boldsymbol{s}_1$ is located relative to the cameras in the ground plane reference frame.

Instead of determining the trapezoid dimensions for every point or possible distance, our approach only uses a fixed set of differently sized trapezoids. There are n number of trapezoids in the set and the corresponding truncated cones are spaced at regular distances of z_{step} apart. If z_{min} is the nearest and z_{max} is the furthest reconstructed distance, then the distance z_i of the i-th truncated cone is equal to:

$$z_i = z_{\text{min}} + i z_{\text{step}} \qquad \text{with} \qquad z_{\text{step}} = \frac{z_{\text{max}} - z_{\text{min}}}{n} \ . \tag{4.7}$$

Each pixel j within the trapezoid boundaries has its own distance threshold value Δz_j. As can be seen in fig. 4.8, this value corresponds to the distance between the cone surface and the gray coloured plane at distance z_i that splits the truncated cone in two halves. Therefore, a pixel with distance value z falls within the truncated cone space if the absolute difference with z_i is smaller than or equal to Δz_j:

$$|z - z_i| \leq \Delta z_j \ . \tag{4.8}$$

Because we use a fixed set of trapezoids, the Δz_j value of each trapezoid pixel only has to be calculated once. This can be done during initialisation so that the required computation during processing, is reduced to a threshold operation for each point pair.

A problem with the Δz_j threshold value is that it does not consider inaccuracies in stereo reconstructed distances. Because of the projective nature of imaging, these inaccuracies increase with distance. Therefore, pairs of points that lay far away might not be clustered to the same obstacle surface due to large distance estimation errors. In order to reduce this problem we propose to include a measure for this inaccuracy in the obstacle point check. As discussed in the stereo reconstruction section, Eq. 4.5 can provide a line piece that corresponds to the distance uncertainty of a reconstructed point. This line piece is between vectors $\boldsymbol{s}_{\text{min}}$ and $\boldsymbol{s}_{\text{max}}$ in the ground surface reference frame. Vector $\Delta\boldsymbol{s}$ is the difference of both vectors:

$$|\boldsymbol{s}_{\text{max}} - \boldsymbol{s}_{\text{min}}| = \Delta\boldsymbol{s} = \begin{pmatrix} \Delta s_x \\ \Delta s_y \\ \Delta s_z \end{pmatrix} \ . \tag{4.9}$$

The Δs_z value indicates the uncertainty in z_i of a trapezoid pixel. Note that the absolute difference with z_i is used in Eq. 4.8 to check whether or not a pixel is an obstacle point.

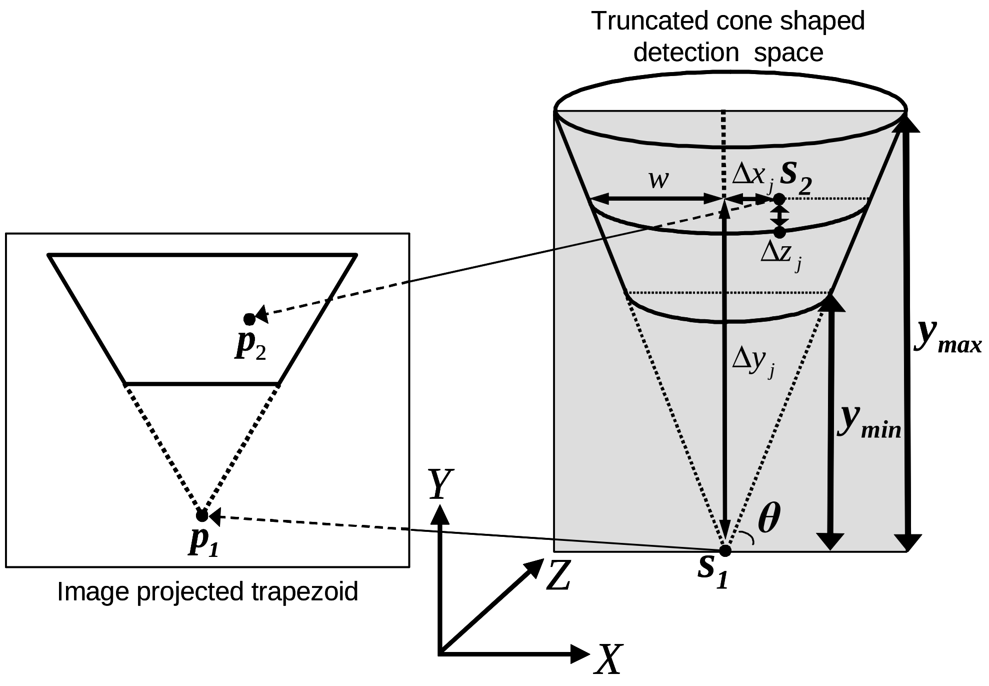


Figure 4.8: Geometry used for computing the Δz_j values.

Therefore, the Δs_z value can simply be added to Δz_j to include the distance uncertainty. This leads to the following criterion to identify obstacle points:

$$|z - z_i| \begin{cases} \leq \Delta z_j + \Delta s_z & \text{is an obstacle point.} \\ > \Delta z_j + \Delta s_z & \text{is not an obstacle point.} \end{cases} \tag{4.10}$$

Precalculation of the threshold trapezoids pixels

Our approach relies on performing an initialisation step where all the Δz_j values for all trapezoid pixels are calculated. The required geometry is explained in this section. Suppose that vector $\boldsymbol{s}_1$ indicates a point at distance z_i in the ground plane orientated reference frame. Fig. 4.8 shows this point and the plane, that lies at distance z_i. It also shows the front part of the cone shaped obstacle point search space with its origin at s_1.

The problem is now to calculate Δz for an arbitrary image point indicated by the vector $\boldsymbol{p}_2$. This is possible by first calculating the point s_2 where the rotated vector $\boldsymbol{p}_2$, extended by μ, intersects the plane at z_i. The plane can be defined by its support vector z_1 and a surface normal vector $\boldsymbol{n}$. Because the plane is perpendicular to Z-axis, vector $\boldsymbol{n}$ is parallel to it. Scalar μ is now equal to the fraction of the vector dot products of $\boldsymbol{n}$ with both $\boldsymbol{s}_1$ and $\boldsymbol{p}_2$:

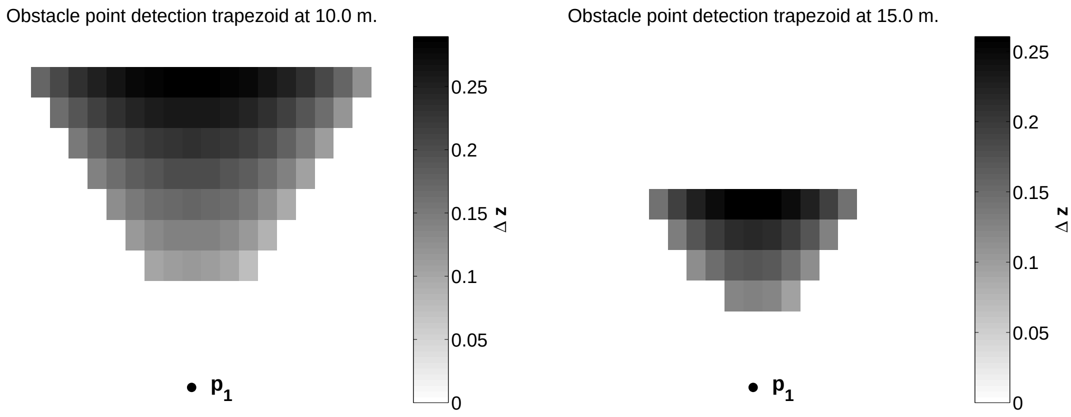


Figure 4.9: Two examples of trapezoid detection areas. The one on the left side is calculated for $z_i = 10.0$ m, while the other is calculated for $z_i = 15.0$ m.

$$s_2 = \mu(\boldsymbol{R}\boldsymbol{p}_2) \quad \text{with} \quad \mu = \frac{\boldsymbol{n}\cdot \boldsymbol{s}_1}{\boldsymbol{n}\cdot \boldsymbol{R}\boldsymbol{p}_2} \quad \text{and} \quad \boldsymbol{n} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} . \tag{4.11}$$

Fig. 4.8 also shows the intersection point $\boldsymbol{s}_2$. Because it lies in the same plane as the cone origin vector $\boldsymbol{s}_1$, there is only a horizontal Δx_j and vertical offset Δy_j. The radius w of the cone at vertical offset Δy_j can be computed with:

$$w = \Delta y_j \tan\left(\tfrac{1}{2}\pi - \theta\right) \quad \text{with} \quad |\boldsymbol{s}_2 - \boldsymbol{s}_1| = \begin{pmatrix} \Delta x_j \\ \Delta y_j \\ 0 \end{pmatrix} . \tag{4.12}$$

The desired depth difference Δz_j between the cone surface and vector $\boldsymbol{s}_2$ can be computed from the radius w and the horizontal Δx offset:

$$\Delta z_j = \tan(\varphi)\,\Delta x_j \quad \text{with} \quad \varphi = \arccos\left(\frac{\Delta x_j}{w}\right) . \tag{4.13}$$

Eqs. 4.11 until 4.13 can be used to compute all the Δz_j of pixels within the trapezoid boundaries. The calculations involved, only have to be performed during initialisation. The number of pixels per trapezoid is set to a limit of 50 in order to further reduce the computational complexity of applying the threshold. For trapezoid regions calculated at large distances this has no consequences because their pixel size is smaller than the limit. However, for nearby distances, the calculated threshold values are sorted on their Euclidian pixel distance with point $\boldsymbol{p}_1$. This insures that only the nearest pixels are inspected. Two examples of pre-calculated trapezoids are shown in fig. 4.9.

4.3.4 Obstacle point clustering and filtering

Points found by the positive obstacle pixel detection may belong to different obstacles. However, there can also be wrongly detected points on non-obstacle surfaces. Further geometric checks are needed to remove such false points. For this step, neighbouring obstacle pixels are first clustered. Each clustered set is regarded as an object surface. Several geometric properties can be checked to distinguish real obstacle surfaces from false ones.

We use a morphological approach to cluster the detected obstacle points into object clusters. Because each reconstructed point corresponds to a pixel, it has a maximum of eight neighbouring points. By computing the absolute difference of their z-distances it can be determined whether a point and its neighbour belong to the same obstacle surface. The threshold of this difference is equal to $z_{\text{step}} + \Delta s_z$ to include the distance uncertainty. Here, the distance uncertainly Δs_z is that of the centre (gray coloured) pixel.

Due to redundancy, not all eight neighbours need to be compared with the centre pixel. The left part of fig. 4.10 shows that, if the clustering procedure runs from left to right and from top to bottom, only three comparisons with other pixels (a, b and c) are required for each pixel. A two-pass algorithm is applied to exploit this fact. In the first pass only the horizontal connectivity (a) is checked among the pixels on each image line. Rows of connected pixels are stored efficiently by using run-length encoding. Each encoded row is assigned a reference to unique label. Vertical connections (b and c) between run-length encoded rows are checked in the second pass. If there is a connection between two rows, they are joined by assigning the same label reference. A problem here is that a set of connected rows might grow and become very large. Relabelling such a large set is computationally expensive. Instead, a list of references is kept that indicates for each label the row to which it is was assigned. Note that the reference between the label and the rows is now bidirectional. In case of a joining, the label belonging to one the rows is removed from the list. Its label reference is set to the row to which it is joined. For multiple joined rows, recursion through the label references is used to find the common label.

With the clustering algorithm, the obstacle points can be merged into one set per object. In most cases, this results into a list of several obstacle points sets. Our approach uses three properties to determine if a point set is a real obstacle. The number of points per set is the first property. If there are less than 10, the set is regarded as noise and is removed. Another property is the height of the bounding box that encloses a point set. A set will be removed if its height is less than y_{min}. Finally, the median slope of a point set is checked. This is used to find obstacles that are too “flat” to be a real danger to the vehicle. For its calculation, image columns are searched that intersect with the obstacle pixels of a set. The top and bottom obstacle pixel are determined in each column found. These pixels correspond to two space points on a line. The angle between this line and the Y-axis is regarded as an local approximation of the slope in the column. The median slope is obtained by collecting the slope for

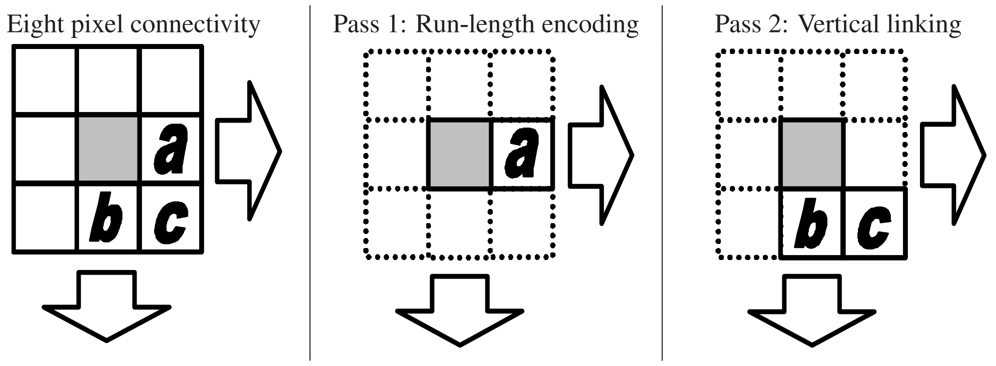


Figure 4.10: Pixels are clustered with a two pass algorithm. In the first pass, horizontally linked pixels are clustered by run-length encoding. Vertical links among the resulting rows are searched in the second pass.

all columns. When an obstacle is too flat its median slope will also be small. In our approach, obstacles with a median slope angle smaller than 5.0° are rejected as false obstacles.

4.3.5 Ground plane orientation update

Terrain elevation is an important feature for detecting obstacle pixels. An assumption in our approach is that it can be measured along the Y-axis of the reference system associated with the ground plane. A problem with driving over rough terrain is that the vehicle can undergo roll and pitch motions. It changes the ground plane orientation relative to the vehicle mounted stereo camera. If uncorrected, there will be a misalignment of terrain height with the coordinate system Y-axis. This can lead to problems with the obstacle detection. The approach of Manduchi et al. [52] briefly mentions the use of inclination sensors or a vision routine to correct for these orientation changes. Here, we propose our own vision based solution.

The change in ground plane orientation is modelled by a small rotation $\boldsymbol{R}_u$. It is applied to the initial orientation $\boldsymbol{R}$ obtained from the ground plane calibration:

$$\hat{\boldsymbol{R}} = \boldsymbol{R}_u \boldsymbol{R} \, . \tag{4.14}$$

The new orientation is defined by $\hat{\boldsymbol{R}}$. In order to measure $\boldsymbol{R}_u$, the ground surface pixels from the previous frames are used. An interesting aspect here is that the remaining set of points, after removal of obstacle points, must belong to the ground plane surfaces. Therefore, only the remaining pixels are used for orientation estimation.

Suppose that there is a set of N vectors $\boldsymbol{s}_{i=1...N}$ that belong to the ground surface. Rotation $\boldsymbol{R}_u$ can be estimated from this set using the following steps. First, compute the centroid of the set and subtract it from each vector. Then compose the matrix $\boldsymbol{M}$ by concatenating the resulting vectors $\bar{\boldsymbol{s}}_i$:

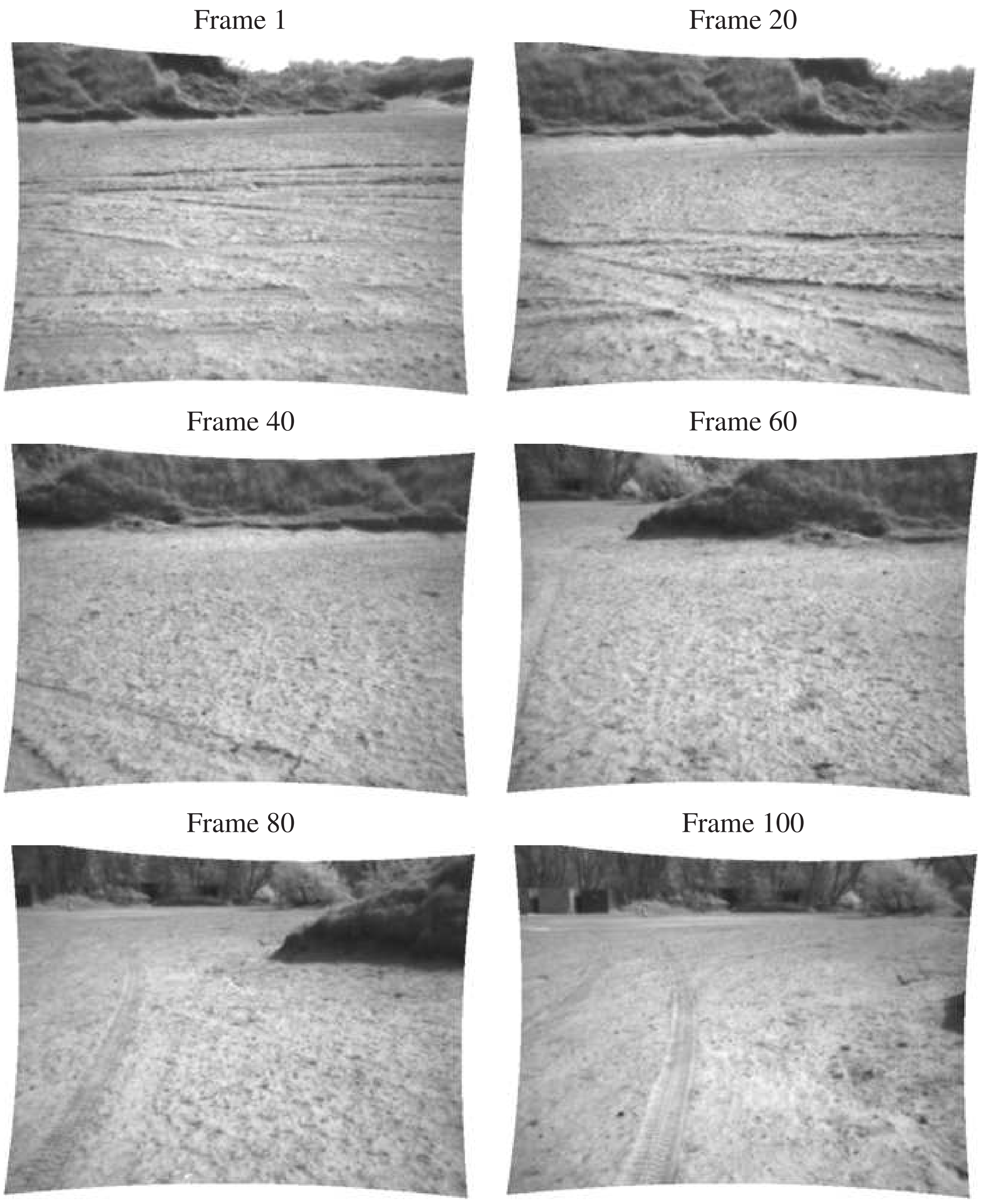


Figure 4.11: Example left stereo images from the test sequence.

$$\boldsymbol{M} = [\bar{\boldsymbol{s}}_1\, \bar{\boldsymbol{s}}_2 \cdots \bar{\boldsymbol{s}}_N]^\top \quad \text{with} \quad \bar{\boldsymbol{s}}_i = \boldsymbol{s}_i - \frac{1}{N}\sum_{j=1}^{N} \boldsymbol{s}_j \ . \tag{4.15}$$

With the singular value decomposition (SVD) [70] this matrix can be decomposed in:

$$\boldsymbol{M} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^\top \ . \tag{4.16}$$

The smallest singular value in the diagonal matrix $\boldsymbol{D}$ has a corresponding singular column vector $\boldsymbol{n}$ in matrix $\boldsymbol{V}$. This vector is the least squares fit for the normal of the plane.

We seek the rotation that turns $\boldsymbol{n}$ into the vector $\boldsymbol{y}$ that is aligned with the Y-axis. The rotation axis, indicated by normalised vector $\boldsymbol{a}$, is obtained with the cross product:

$$\boldsymbol{a} = \frac{\boldsymbol{n} \times \boldsymbol{y}}{||\boldsymbol{n} \times \boldsymbol{y}||} \quad \text{with} \quad \boldsymbol{y} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \ . \tag{4.17}$$

The smallest rotation angle φ between both vectors can be computed with the dot product. Together, the rotation angle φ and axis $\boldsymbol{a}$ can be used to form a quaternion $\boldsymbol{q}$ (appendix B):

$$\boldsymbol{q} = \left(\cos(\tfrac{1}{2}\varphi) \ \left[\sin(\tfrac{1}{2}\varphi)\boldsymbol{a}^\top\right]\right)^\top \quad \text{with} \quad \varphi = \arccos(\boldsymbol{n} \cdot \boldsymbol{y}) \ . \tag{4.18}$$

By converting the quaternion $\boldsymbol{q}$, the rotation matrix $\boldsymbol{R}_u$ can be obtained.

It should be noted that at least three vectors are needed to estimate $\boldsymbol{R}_u$. These vectors should not indicate points that are collinear. The three-dimensional vectors $\boldsymbol{s}_1$, $\boldsymbol{s}_2$ and $\boldsymbol{s}_3$ are not collinear if:

$$||(\boldsymbol{s}_2 - \boldsymbol{s}_1) \times (\boldsymbol{s}_3 - \boldsymbol{s}_1)|| > 0 \ . \tag{4.19}$$

The estimation technique could be applied directly to the remaining points, because the points that do not belong to the ground surface have already been removed. However, there could still be some outliers left due to detection errors. We therefore seek a plane orientation that only fits a majority of the points and ignores possible outliers. The robust Least Median of Squares (LmedS) estimation technique [86] is used for this purpose. This technique is discussed in detail in chapter 2, section 2.5.

4.4 Results

One of the key differences between our approach and the original by Manduchi et al. [52] is the use of a fixed set of trapezoid shaped threshold windows. The other is that our approach considers the uncertainty in distance values obtained from stereo reconstruction. Therefore, we compare the results of our approach to those of a version that uses the original approach to check if pixels belong to the same obstacle surface.

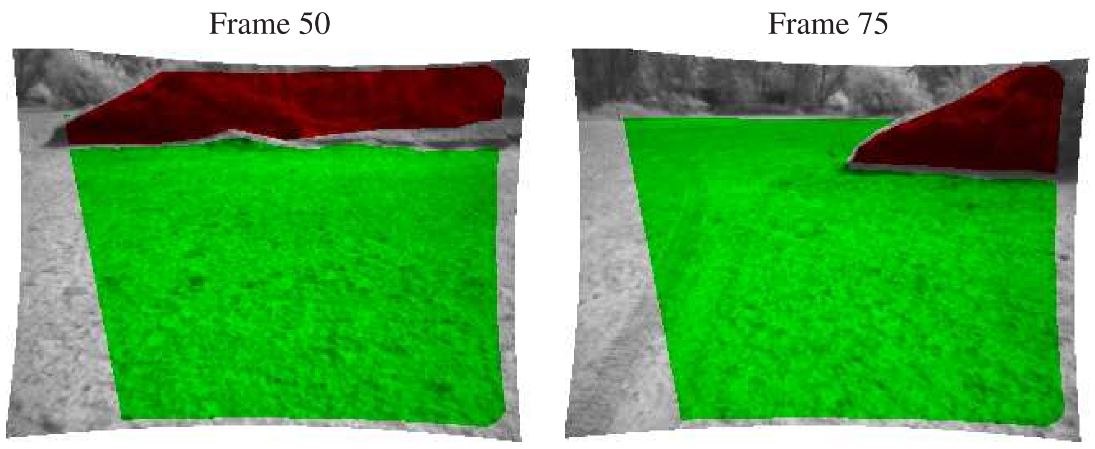


Figure 4.12: Example labelled images from the test sequence. The colour red indicates positive obstacles and the colour green indicates the ground surface.

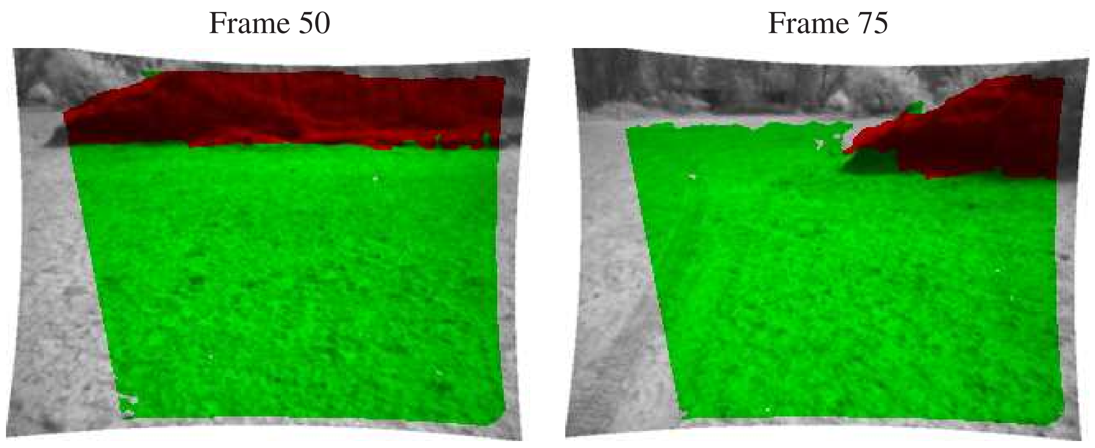


Figure 4.13: Obstacle detection output of our approach.

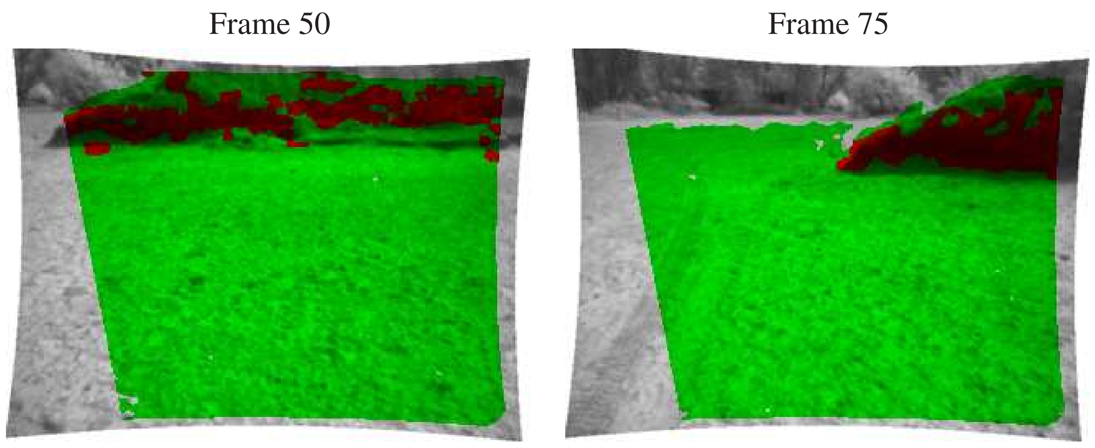


Figure 4.14: Obstacle detection output of the original approach.

Table 4.1: Pixel based detection probabilities.

			$P(C\|j)$	
Method	P_C	$\overline{P}_C$	Positive obstacle	Ground plane
Ours	0.988	0.966	0.942	0.991
Original	0.938	0.798	0.599	0.996

This involves computing the angle and height difference for each inspected point pair. It should be noted that this version does not use the tree-labelling algorithm, to cluster detected points into obstacle clusters, as described in the original paper [52]. In order to limit the differences between the two approaches, the pixel based method of section 4.3.4 is used instead. Both approaches also use the vision based orientation update routine described in section 4.3.5. During the evaluation, parameters that are used by both methods are set to; $y_{\min} = 0.1$ m, $y_{\max} = 0.3$ m and $\theta = 45°$. For our method, the trapezoid related parameters are set to; $z_{\min} = 2.0$ m, $z_{\max} = 30.0$ m and $z_{\text{step}} = 60$.

In the literature, the evaluation of new approaches to obstacle detection in off-road terrain is often limited to qualitative examples that demonstrate detection capabilities. Instead, we have devised several quantitative tests in order to investigate obstacle detection performance. The idea behind the tests is that positive obstacles pixels and ground surface pixels belong to two distinct classes. Misclassifying positive obstacle pixels as ground pixels leads to false negative detection, while misclassifying ground surface pixels as positive obstacle pixels can be regarded as false positive detection.

For this evaluation, a sequence of 102 stereo images was used where the vehicle drives towards a sand dune obstacle. Six images from this sequence are shown in fig. 4.11. At the beginning of the sequence, the obstacle is far away. During the sequence the vehicle moves closer to the obstacle.

Each of the sequence images was hand labelled with polygons to indicate pixels that belong to positive obstacles or the ground surface. Fig. 4.12 shows two images from the test sequence. Labelled pixels on a positive obstacle have been indicated with red, while ground surface pixels are coloured green. Detection results from our method are shown in fig. 4.13. It can be seen that a large majority of the obstacle pixels are detected correctly. Example output from the original method is shown in fig. 4.14. Here, it is clear that less obstacle pixels are detected as compared to our method. Many of the obstacle pixels are misclassified as being ground surface.

In the evaluation, the output of the obstacle detection algorithm is compared to the hand labelled ground truth. The number of times that a pixel with ground truth label j has been classified as k is counted. For each image, the result is collected in confusion matrices $CM_{j,k}$. Castano et al. [15] propose three quantitative performance measures based on confusion matrices. The first measure is the ratio of correctly classified points among the total number of points in the complete test set:

$$P_C = \frac{\sum_j CM_{j,j}}{\sum_j \sum_k CM_{j,k}} \ . \tag{4.20}$$

An estimate of the correct classification probability for each class j is given by:

$$P(C|\,j) = \frac{CM_{j,j}}{\sum_k CM_{j,k}} \ . \tag{4.21}$$

A disadvantage of the global measure P_C is that it is biased towards the class that occurs the most in the test set. The unbiased overall probability of correct classification $\overline{P}_C$ is equal to average of $P(C|\,j)$ for all classes:

$$\bar{P}_C = \frac{1}{2} \sum_j P(C|j) \ . \tag{4.22}$$

The plot in fig. 4.15 shows classification probability $P(C|\,j)$ for each sequence frame. Table 4.1 shows the average probabilities for P_C, $\overline{P}_C$ and $P(C|\,j)$ for the whole sequence. It can be seen that both our and the original approach score high classification probabilities for the ground surface pixels. The methods score an average probability of 0.991 and 0.996 respectively.

However, there is a significant difference in the classification probabilities for positive obstacle pixels. Our method scores an average classification probability of 0.942, while the original method only manages to score a probability of 0.599. The plot of fig. 4.15 shows that, at the start of the sequence, the classification score of the original method is lower as compared to our method. Its performance does increase during the sequence. At the end of the sequence the performance of both methods seems to drop dramatically. Because the object is leaving the stereo camera's field of view it can be seen that the number of labelled obstacle pixels also drops to zero in fig. 4.16.

The performance of both methods during the sequence can be explained when the obstacle distance is considered. A plot of the obstacle distance at each frame is shown in fig. 4.19. It is based on the median distance value of all labelled obstacle pixels. The plot shows that the obstacle is about 25 m away at the start of the sequence. During the sequence, the vehicle comes closer and passes it at 5 m distance.

The original method computes the angle and height differences between point pairs to check if they belong to the same obstacle surface. When the points lie far away there will be large distance errors in both the computed angle and height values. This degrades the ability of the original method to distinguish between obstacle and other points. However, when the vehicle drives towards the obstacle, the distance values become more accurate. Therefore, the distinction between obstacle and other points will also improve. This explains why the performance of the original method starts out low and improves during the sequence.

Our method does not compute the geometric check used by the original method explicitly. Instead, precalculated threshold values are applied together with a margin

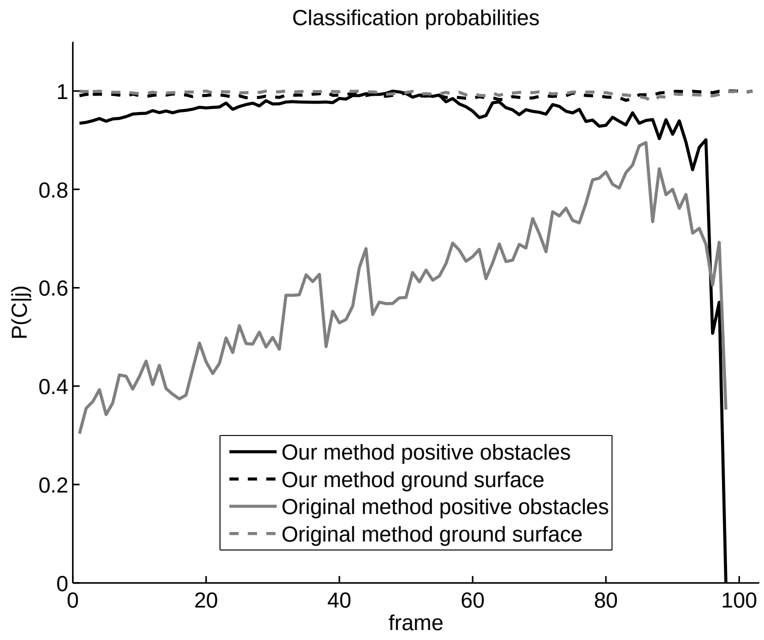


Figure 4.15: Detection probabilities for positive obstacle and ground surface pixel classes.

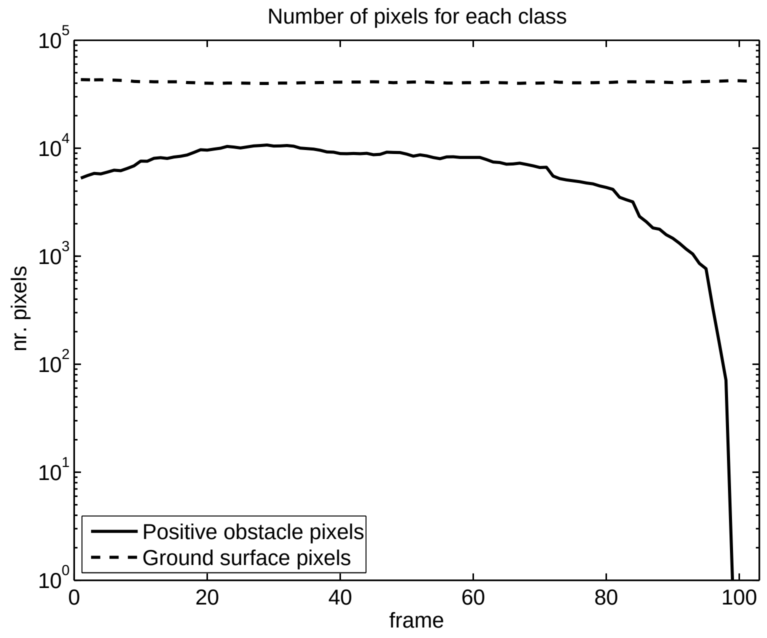


Figure 4.16: Pixel count for positive obstacle and ground surface pixel classes.

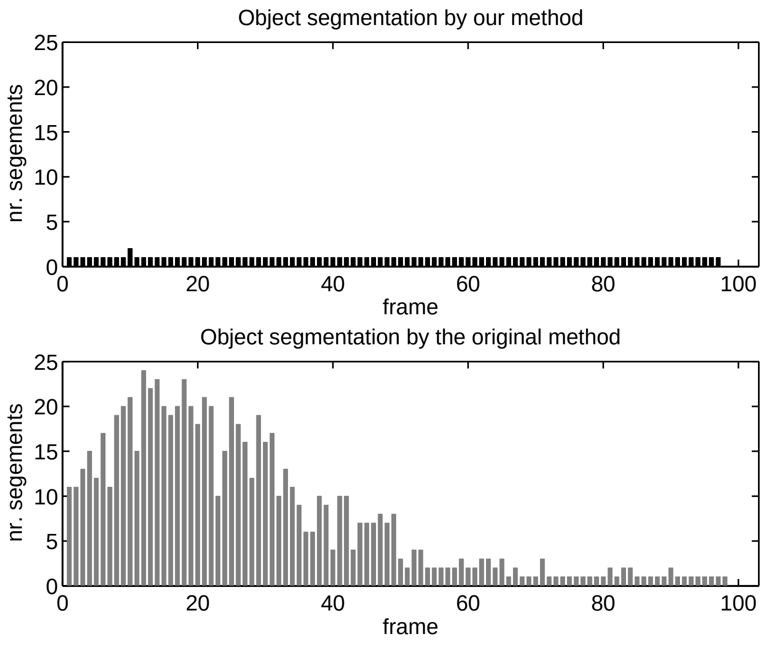


Figure 4.17: Number of detected segments in the labelled obstacle area.

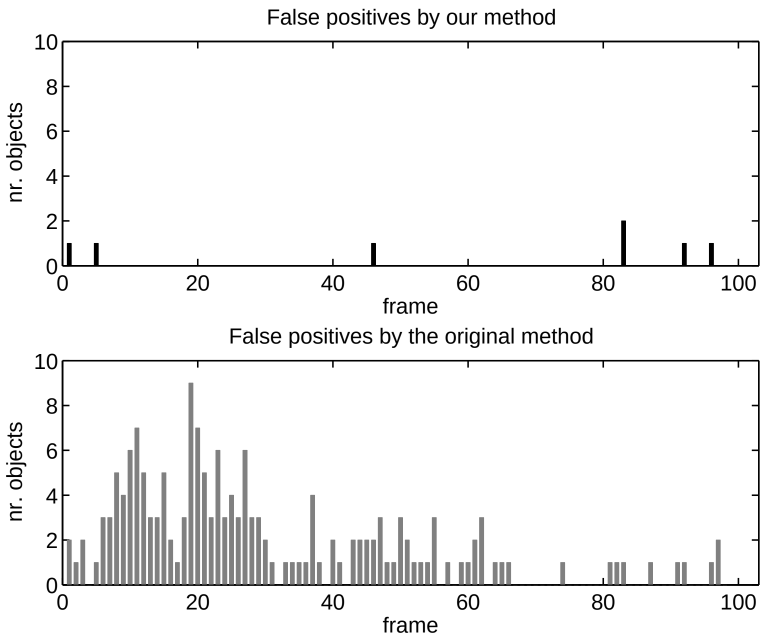


Figure 4.18: Number of falsely detected obstacles per frame.

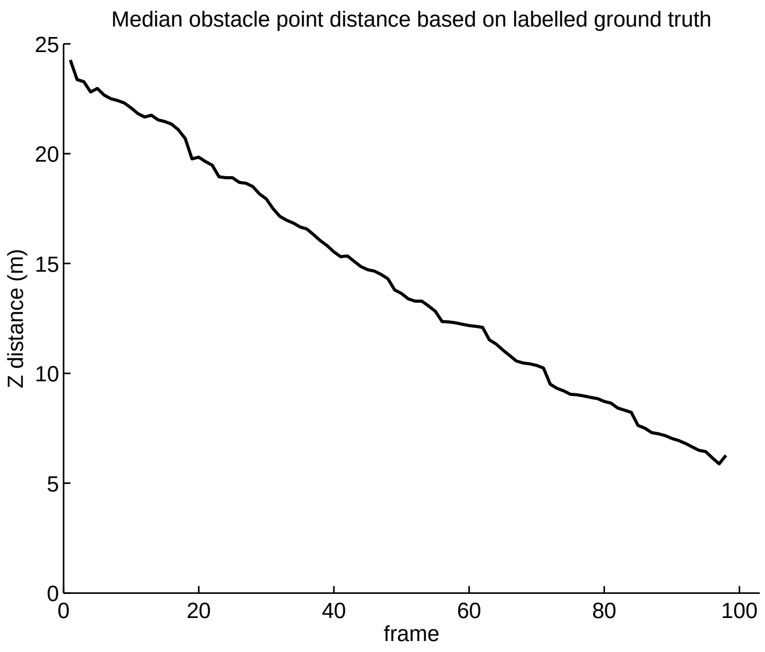


Figure 4.19: Obstacle distance for each frame based on the labelled pixels.

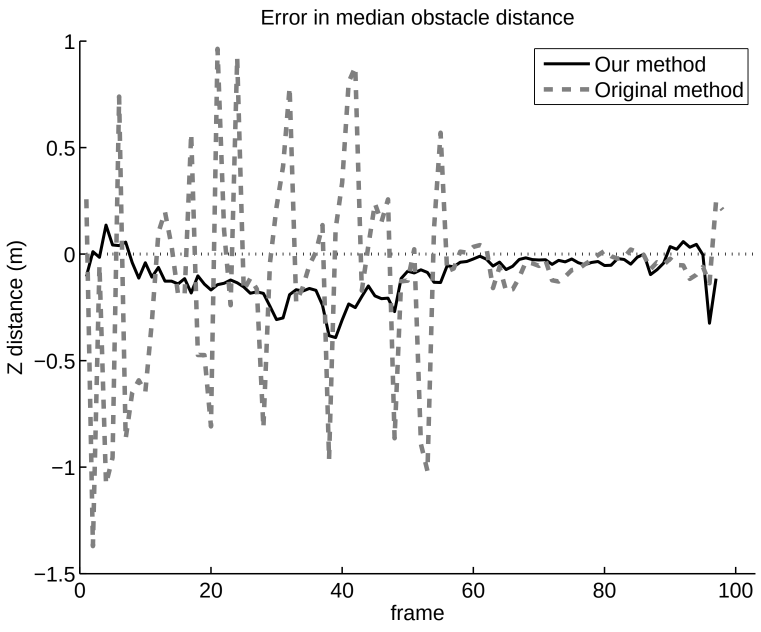


Figure 4.20: Error of the median estimated obstacle distance.

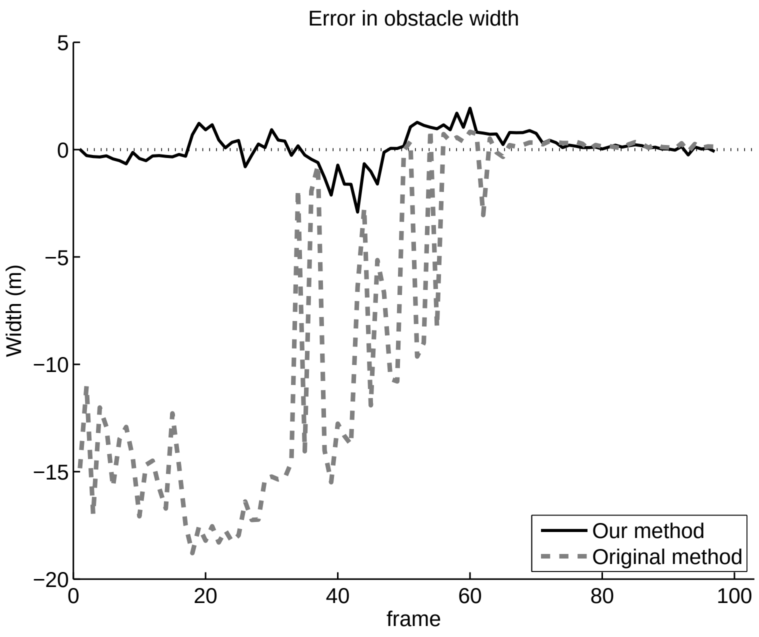


Figure 4.21: Error of the measured obstacle width.

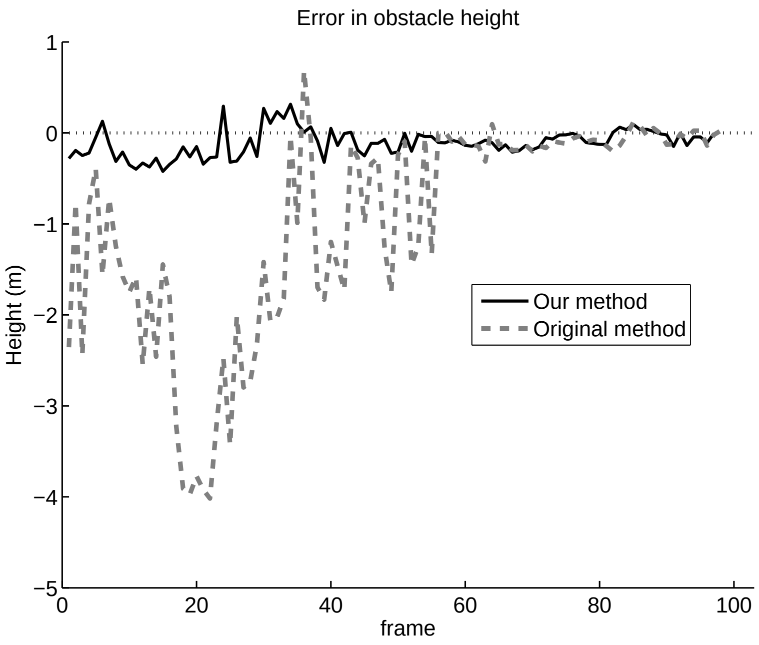


Figure 4.22: Error of the measured obstacle height.

to account for the distance uncertainty. The results indicate that this leads to a better distinction between obstacle and other points at further distances.

The classification probabilities only indicate the methods reliability to distinguish obstacle pixels from ground surface pixels. Another important issue is the ability of the methods to cluster obstacle pixels correctly. Monolithic objects should not be segmented in two or more separate objects. To test if over-segmentation occurs, the number of detected objects that occupy the image region of the labelled obstacles was counted. Fig. 4.17 shows the number of counted objects per labelled obstacle for each frame. For all but one frame, our method correctly clusters the labelled object pixels. The original method fails to achieve this for the majority of frames. Particulary when the obstacle is at a large distance, it detects multiple objects instead of one single object. This is a result of the fact that the original method detects fewer obstacle pixels at these distances. Apparently, the remaining pixels do not link up properly to form a single object.

Another property of detection is the amount of false alarms. The number of detected obstacles in the image region that belongs to the ground surface is shown in fig. 4.18. Of the 102 frames, there are only 6 frames where our method detects false obstacles. Substantially more false detections are found in the results of the original method. Most of these also occur when the real obstacle is far away.

Obstacle over-segmentation has consequences for the autonomous vehicle guidance. This can be demonstrated by inspecting the measured obstacle geometric properties. Remaining distance and width are properties that are important for obstacle avoidance. These should be measured accurately in order to avoid collisions. Another important property is height, because it is used in the subsequent filtering step to reject objects that are too small to be an obstacle.

In this evaluation, the geometric properties of the labelled pixels are used as ground truth. Obstacle distance is computed by taking the median of these pixel range values. Note that this was also used for plotting fig. 4.19. Width and height are obtained by finding a bounding box that fits the obstacle points. The geometric properties of detected obstacles are computed in the same way. Only one detected obstacle by each of the methods is compared with the ground truth. If there are multiple detections due to segmentation, the nearest obstacle to the ground truth is selected.

Fig. 4.20 shows the difference in median distance of the ground truth points and the detected obstacle points for both methods. It indicates the type of error in the distance estimate. Negative values indicate too small distance estimates and positive values are too large estimates. The plot shows that when the obstacle is nearby, both methods only display small errors. However, the errors of the original method are more severe than those of our method for larger distances. The results for obstacle width and height are shown in fig. 4.21 and fig. 4.22. These plots reveal that the original method underestimates the parameters when the obstacle is further away. Our method provides more consistent estimates.

The poor performance of the original method is clearly due to the over-segmentation. Comparisons with fig. 4.17 show that the large errors occur in the same frames where

Table 4.2: Processing times for each of the various algorithm steps.

Task	Time
Disparity estimation	0.038 s
Stereo reconstruction	0.002 s
Positive obstacle detection	0.060 s
Positive obstacle clustering	0.001 s
Ground orientation estimation	0.021 s
Total processing time	**0.122 s**

the object is divided in several segments. There are errors in the distance, width and height measurements because each segment only covers a part of the obstacle.

As indicated earlier, a problem for stereo is that the accuracy of three-dimensional reconstructed points decreases with distance. Because the original method does not consider this, it over-segments real obstacles and detects more false obstacles among points that lie far away. The results of our method show that inclusion of distance uncertainty measures, leads to more reliable detections.

An obstacle detection algorithm is not suitable for autonomous vehicle guidance if its computational demands do not allow for a sufficiently fast processing rate. We therefore measured the time needed to process stereo images by our method. Table 4.2 contains the measured processing times of the different components for one stereo pair. The timing tests were performed on an Intel Core 2 computer with a clock speed of 2.66 GHz and 2.0 GB of RAM. It can be seen that the most computational expensive step is the positive obstacle detection. The total time is 0.122 seconds, which allows for a processing rate of 8.197 stereo images per second.

4.5 Conclusion

We have presented our stereo vision based approach to obstacle detection in rough and off-road terrain. Dense disparity is used for obtaining three-dimensional terrain points from stereo, because obstacles in unstructured terrain have unpredictable shapes and often lack clear visual features. However, the application of stereo vision leads to the problem that the point reconstruction accuracy becomes progressively worse for larger distances. This is a problem for obstacle detection, where points on obstacle surfaces have to be found and clustered into single objects at both small and large distances.

The method for detecting points on the same obstacle surfaces is inspired by the method of Manduchi et al. [52] that uses two geometrical constraints. Instead of computing these constraints for every point pair, a fixed set of threshold values is computed in our approach. This has several advantages over the original method. Firstly, it reduces the computational complexity of the obstacle point check to a simple threshold operation. Secondly, the required threshold values are only computed during initialisation and are reused for each stereo pair. Furthermore, it allows for an easy way to deal with the uncertainty of the stereo reconstructed point distances. The uncertainty in reconstructed distance can simply be added to the threshold value.

To improve detection reliability, detected obstacle pixels are clustered into objects. Resulting objects with insufficient pixels, height and slope are discarded as false obstacles. The remaining ground surface pixels are used to estimate the orientation difference between the stereo camera and the ground surface. This is used to compensate changes in camera orientation while the vehicle is driving over rough terrain.

Quantitative tests were conducted to compare the performance of our method to that of the original approach. A sequence of stereo images with labelled obstacles was used for this purpose. The tests show that our method detects more obstacle pixels correctly. Due to the inclusion of distance uncertainty, our method is able to detect more pixels on obstacle surfaces, especially if they are located further away. This is advantageous for the subsequent clustering of obstacle points. In contrast to the original approach, our method does not segment obstacles into several smaller ones. Analysis of the measured obstacle dimensions has shown that over-segmentation by the original method can lead to inconsistent or underestimated values. In a map view of the obstacle, inconsistent values mean that the obstacle shape and size changes constantly. If a vehicle route is planned with such a map it will also have to be changed constantly. Underestimation is even a larger problem for vehicle guidance. A detected obstacle may be larger in reality than its measured dimensions. In such a case, the vehicle could still hit the obstacle, because steering decisions to avoid the obstacle will be based on incorrect information.

Because our method does not over-segment obstacles, it does not suffer from inconsistent or underestimated obstacle dimensions. It is therefore more suitable for application in autonomous vehicle guidance.

Chapter 5
Terrain Classification by Environment Resemblance

5.1 Introduction

Obstacle detection in off-road terrain traditionally has relied on range finding sensors such as laser or stereo vision. In natural terrain, not all objects that protrude out of the ground plane are potentially dangerous obstacles. For example; a rock is an obstacle and should be avoided, while tall grass can be safely driven over by the vehicle. If the obstacle detection is only based on height differences from range measurements, both types will be detected as obstacles. Consequently, the vehicle has to stop for, or drive around a lot of false obstacles. In the work by Castano and Matthies [14] a statistical method is used to distinguish between grass and solid objects, such as rocks, in laser-range finder data. However, based on only range information, tall grass is more difficult to distinguish from other non solid obstacles such as bushes. It is also more difficult to distinguish different soil types such as sand or gravel. Recognition of such terrain types is important in order to locate drivable areas such as dirt roads.

As an alternative there have been approaches that looked into the use of colour or texture information from computer vision. The methods of Buluswar et al. [12] and Manduchi et al. [52] use colour features to recognise terrain types. Methods based on colour have to deal with the fact that the apparent colour of an object depends on the colour of the light source, the reflectance of the object, illumination geometry, viewing geometry and camera properties. In outdoor images the colour of the illuminant varies with the time of day, cloud cover and atmospheric conditions. Buluswar et al. [12] have shown that because of the fact that there are many combinations possible, the variation in apparent colour can even be greater than the difference between two distinct colours.

In order to equalise the apparent colour under different illumination conditions, white point calibration is used by Manduchi et al. [52]. Unfortunately, this technique cannot compensate for all illumination conditions and requires that a white surface is fixed in the camera field of view.

Another problem for colour based approaches is that colour is not distinguishable in low-light conditions. Colour features are useless for terrain classification during night-time conditions without the additional illumination from non-monochromatic lamps.

Many terrain types can also be identified by their texture. The advantage of texture is that it is visible in both normal images and in the images of intensified or infrared cameras that are more suitable for use in nighttime conditions. Texture features extracted with Gabor filter banks are used by Castano et al. [15]. However, the difficulty of such texture features is that they vary in scale and orientation. Several filters of different orientation and scale have to be applied to an image in order to obtain usable features, resulting in time consuming operations.

A variety of machine learning techniques can be applied to the problem of terrain classification. Buluswar et al. [12] use multivariate decision trees to learn the regions in the *RGB* colour space occupied by different material types. Both Castano et al. [15] and Manduchi et al. [52] use a Maximum Likelihood (ML) approach and model the distribution of their terrain features with Gaussian Mixture Models (GMMs). These are obtained with the Expectation Maximisation (EM) algorithm.

Our approach to outdoor terrain classification will be presented in this chapter. Like Castano et al. [15] and Manduchi et al. [52], the basis of our approach is a ML classifier with GMM's. An advantage of this classifier is that it is not restricted to using one type of feature. Different types of features can be used or even combined. In contrast to other work on terrain recognition, combinations of colour with texture features will be investigated. We also investigate if classification of similar materials can be improved by inclusion of a simple geometric feature.

A new approach has been developed in order to deal with large variation of features, such as colour, in outdoor conditions. It is based on the idea that the pixels from different images, which show the same terrain types, are similar distributed in the feature space if environment conditions are the same. Given labelled input images, our method can find groups of images that share the same environment state. Classification of terrain types in separate states could be more successful because it is less influenced by the changes in environment conditions.

In the next section, the theory and influences involved in colour formation in outdoor natural terrain are introduced. Furthermore, the design of Gabor filter dictionaries for extracting texture features at different scales and orientations is explained. Section 5.3 describes our environment specific classification approach. It shows how a training set of images can be segmented into clusters of images that share the same environment state. In section 5.4 the approach is tested with various combinations of colour, texture and geometric features. Tests are conducted with images of natural outdoor terrain under different illumination conditions. Final conclusions are given in section 5.5.

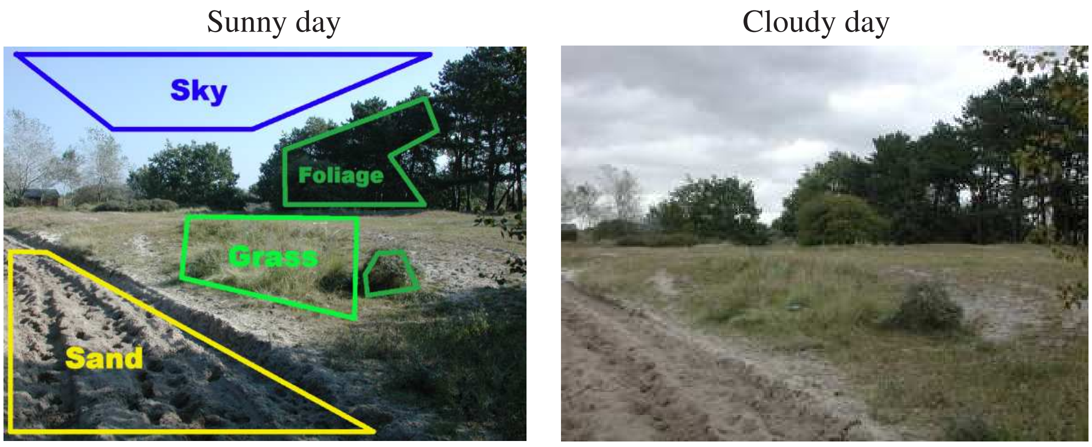


Figure 5.1: Both images show the same natural terrain scene. The left image was recorded on a sunny day. In this image, the different terrain types have been labelled. An image recorded on a cloudy day, is shown on the right.

5.2 Features for terrain recognition

An important guideline for choosing features in terrain recognition are the terrain types that should be distinguished. Our choice is applicable to the terrain in which we conduct experiments with autonomous robot navigation. This is a dune landscape nearby TNO's location in the Hague, the Netherlands. The terrain mainly consists of sand dunes, partially overgrown with grass, bushes and trees. There are also dirt roads of sand and gravel.

The terrain classes that we choose to distinguish are: sky, grass, foliage, sand and gravel. The importance of the latter two is straightforward; large areas of sand or gravel could help to identify a dirt road. Large patches of grass could indicate drivable areas, while the foliage of bushes and trees should be avoided. The sky class might be considered as an odd member of the terrain class family. Because it shows important clues about the environment state of an image it is included. For example; on sunny days the sky is often blue and on overcast days there are cloud patterns.

On the left of fig. 5.1 an example image from our data set is shown, where some of the terrain types are labelled. Note that not all pixels are labelled; only patches typical for each material are indicated roughly by a polygonal border.

In the following sections both colour and texture features are discussed. In order to understand why the colours of materials in images appear to be similar or different we first study the aspects of colour formation. Also the influence of camera parameters is reviewed. For extracting texture features, the use of dictionaries with Gabor filters of different orientations and scales is discussed.

5.2.1 Colour features

Fig. 5.1 shows two images of the same outdoor scene which contains a variety of terrain types. During a sunny day, the left image was recorded, while the image on the right was recorded during a cloudy day. It can be noticed that the same materials, such as sand and grass, have different colours due to the change in weather conditions. In the next sections we shall explain the factors that are responsible for colour formation in natural terrain.

Day-Light

Images such as the ones shown in fig. 5.1 are measurements of the light projected onto an imaging device by the camera lens. Light itself is a radiation energy which is part of the electro magnetic spectrum. Visible light occupies a part of the spectrum located between about λ=380 nm and 700 nm. In this part, each visible wavelength can be associated with a colour on the rainbow chart. A spectral power distribution $l(\lambda)$ can be used to represent the amount of energy present at each wavelength for a particular light source. If the distribution is uniform, e.g. an equal amount of energy for each wavelength, the light is considered to be white. However, it can have any colour if the distribution is non-uniform.

The colour of day-light varies due to the sun angle and weather conditions. On sunny days, surfaces will be illuminated by both direct sunlight and skylight. The skylight illumination is caused by sunlight that is reflected by the atmosphere. The atmosphere can also act as a filter. If the sun is low on the horizon (sunset), sunlight has to pass through more atmosphere. Due to the particles in the atmosphere direct sunlight appears to be red. On cloudy days no direct sunlight passes through the clouds and illumination is only provided by the ambient skylight. Intermediates are also possible; for example when the sun is partially obscured by a cloud while the rest of the sky is clear. The parametric model of the Commission Internationale de L' Eclairge (CIE) is a well known model for the variation in day-light and is based on actual measurements of sky colour. It indicates colour by means of a temperature. For example, a temperature of 6000 K indicates the colour of a bright midday sun while a colour of 10000 K indicates a heavily overcast sky. In his research on outdoor machine vision Buluswar [11] points out that the CIE parametric model of daylight is too limited for outdoor colour recognition. One problem is that the CIE data was collected using spectroscopes which only observed a small portion of the sky. Special precautions where taken to prevent light reflected from other surfaces to enter the spectroscope.

In order to improve this situation, Buluswar [11] proposes to use an extended model for day-light. He has compiled a table that is ordered on the sun angle. For each angle, a distinction is made between the percentage of cloud cover and the sun-visibility factor; which tells if the sun is blocked by a cloud or not. For each state, two types of colour measurements are available. The first type of measurements was used to obtain the colour of the incident light in the direction of the sun. The second type of measurements was used to obtain the colour of the light away from the sun. A drawback of this approach is that the sun angle and the weather type needs to be determined for each new image in order to retrieve the correct values for illumination from the table.

Reflection of light by surfaces

Surfaces are visible because they reflect light from illumination sources towards the camera. The reflected light by a surface can have different power distribution due to the material properties and scene geometry. We will first introduce the term surface "albedo" and then focus our attention to the different reflection types and their geometry.

The term "albedo" is used to refer to the surface material induced characteristic change in the power distribution spectrum of reflected light. It is defined as a function of wavelength $a_s(\lambda)$.

The geometry of a reflectance by a surface point can be described by the surface normal $\vec{n}$ and the normalised vectors $\vec{v}$ and $\vec{s}$ pointing in the direction of the illumination source and the reflection direction respectively.

If the reflection type is specular, the light will be reflected in a mirror-like fashion. The angle θ between $\vec{v}$ and $\vec{n}$ with be equal to the one between $\vec{v}$ and $\vec{s}$. The following equation can be used to model the light $l_s(\lambda)$ reflected by specular surface:

$$l_s(\lambda) = g_s(\vec{n},\vec{v},\vec{o})a_s(\lambda)l(\lambda) \ . \tag{5.1}$$

Here $g_s(\vec{n},\vec{v},\vec{o})$ is a geometric function that computes a scalar based on the vectors $\vec{n}$, $\vec{v}$ and $\vec{o}$ which points in the direction of the camera.

It has been observed [43] that many types of materials, such as certain plastics and ceramics, exhibit specular reflections which have almost the same colour as the light of the illuminator. This is why in some work [11, 29, 88] the albedo function for specular reflection is omitted.

Another type of reflection is called Lambertian or body reflectance. Like Eq. 5.1, the equation for light $l_b(\lambda)$ reflected by a Lambertian surface has its own geometry function g_b and albedo function a_b:

$$l_b(\lambda) = g_b(\vec{n},\vec{v})a_b(\lambda)l(\lambda) \ . \tag{5.2}$$

The difference is that it reflects incoming light towards many directions. Its geometry function is based on Lambert's law that states that the angle between $\vec{n}$ and $\vec{v}$ determines the brightness of an observed point:

$$g_s(\vec{n},\vec{v})\alpha\cos\theta \ . \tag{5.3}$$

There are materials that exhibit purely specular or Lambertian reflections, while other materials exhibit a mixture of both types. The dichromatic colour model was developed by Shafer [43] for such surfaces:

$$l_d(\lambda) = g_s(\vec{n},\vec{v},\vec{o})a_s(\lambda)l(\lambda) + g_b(\vec{n},\vec{v})a_b(\lambda)l(\lambda) \ . \tag{5.4}$$

How the apparent colour of a surface with complex geometry will appear in a colour space can be understood with this model. It is clear that the geometric term $\vec{o}$ will vary due to the camera projection while the term $\vec{n}$ will vary if the surface is non-planar. The term $\vec{v}$ can also vary if the light is emitted from a point source. In a static image, both the surface albedos a_s and a_b and the power distribution of the light source $l(\lambda)$ are constant. The apparent colour l_d will only vary over $a_s l(\lambda)$ and $a_b l(\lambda)$ in the colour space. In the *RGB* colour space, the vectors $a_s l(\lambda)$ and $a_b l(\lambda)$ span the dichromatic plane, where the direction of colour shades caused by Lambertian reflection is indicated by $a_b l(\lambda)$ and the direction caused by specular reflection is indicated by $a_s l(\lambda)$.

Outdoor experiments by Buluswar [11] with surfaces that exhibit both specular and Lambertian reflection have shown that $g_s(\vec{n},\vec{v},\vec{o})$ only has a value different from zero for a very limited limited range of angle θ. This is why specular reflections do not occur very often in scenes with irregular surfaces. Therefore, we assume that the majority of pixel colours in natural terrain images are the result of Lambertian reflectance.

Registration of light by cameras

Light can be filtered by reducing or blocking the energies at certain wavelengths. This principle is used in colour cameras to distinguish between different colours. Most colour cameras use either a beam splitter prism assembly or a Bayer filter to distinguish between three different colour channels; “Red”, “Green” and “Blue”. This is based on the human visual system which uses three different types of cone cells to distinguish the primary colours. Note that in both human and standard colour cameras, the different colour channels overlap each other at certain wavelengths. The sensitivity of a camera colour channel for each wavelength can be defined by the transition function $f_C(\lambda)$.

In electronic cameras, the light energy is converted to an electric signal by pixels on a CCD or a CMOS imager device. The intensity value I_C measured by such a pixel for light received with the power distribution $l(\lambda)$, can be calculated with the following well-known integral:

$$I_C = \int_{\lambda} f_C(\lambda) l(\lambda) d\lambda\,. \tag{5.5}$$

Pixels can become over-saturated if they receive too much light. Therefore, many cameras are fitted with several systems which regulate the amount of light received by the sensor or condition the output signal. A lens with a motorised diaphragm (iris) is an example of a system which controls the amount of light that is received. The diameter of a diaphragm aperture is equal to the division of the lens focal length trough the F-number. Modern cameras also alow that the amount of light is controlled “on-chip” by changing the exposure time. Alternatively, amplification or “gain” of the output sensor signal can be used to change the image brightness. We combine the influence of camera lens aperture, exposure and gain setting into a single term that is indicated by k_g.

In order to present an appealing image for human vision, cameras often use gamma correction. This allows for a non-linear scaling, for example by a logarithmic function, of the intensity values in order to enhance the contrast of an image. We denote the gamma setting of a camera by γ.

Most colour cameras are equipped with white balance. This is because of the colour of illumination sources, such as tungsten light bulbs, florescent lamps and day-light varies. In a raw camera image, different coloured materials may appear to have a similar colour due to the light source properties. This causes image areas, which normally appear white for humans, to appear differently coloured. The goal of white balancing is to transform the colours of these areas in such a way that they appear white. This can be achieved by applying a gain to each of the *RGB* colour channels. For the colour channel C we denote the white balance gain with k_w.

The relationship between output I'_C, its dependance on k_w, k_g and γ, and the actual measurement I_C can be expressed as:

$$I'_C = k_w(k_g I_C)^\gamma \,. \tag{5.6}$$

An approach that uses a calibrated light source to measure γ and other camera properties is presented in Withagen et al. [100]. For our research, we use a method developed by Novak et al. [63] that is based on the commercially available GretagMacbeth$^{\text{TM}}$ colour checker.

Fig. 5.2 shows three images of the colour checker illuminated by direct sunlight on a sunny day, skylight in a shadow on a sunny day and by the ambient skylight on a cloudy day. The images were recorded with a Nikon Coolpix 990 digital camera with a fixed white balance setting.

The bottom row of the colour checker show a range of gray squares of which the reflection properties are known. The percentage of light that is reflected by these Lambertian surfaces is (from left to right): 88.5%, 59.5%, 59.5%, 36.5%, 21.0%, 9.0% and 2.5% [63]. In fig. 5.3, the normalised percentage of light reflected is shown together with the average gray values measured in the shadow image of fig. 5.2. The non-linear relationship between the two is due to the fact that γ is not 1.0.

According to Novak et al. [63], the γ value can be estimated with a semi-log plot. In fig. 5.4, a log-log plotted of the normalised reflection percentages and the intensities is shown. The gradient of the line through the data points is equal to γ. The figure shows the line fitted through the points using a least squares estimator [71]. The estimate for γ is 0.61 and can be used to linearise the *RGB* colour values. Fig. 5.3 also shows the linearised intensities of the colour checker.

Colour constancy

Colour recognition is difficult to apply outdoors because the apparent colour of materials can change. Take, for example, the two images shown in fig. 5.1. Both images depict the same scene in different weather conditions. In fig. 5.5, several pixel distributions in the *RGB* colour space are shown. The left column of this figure shows the

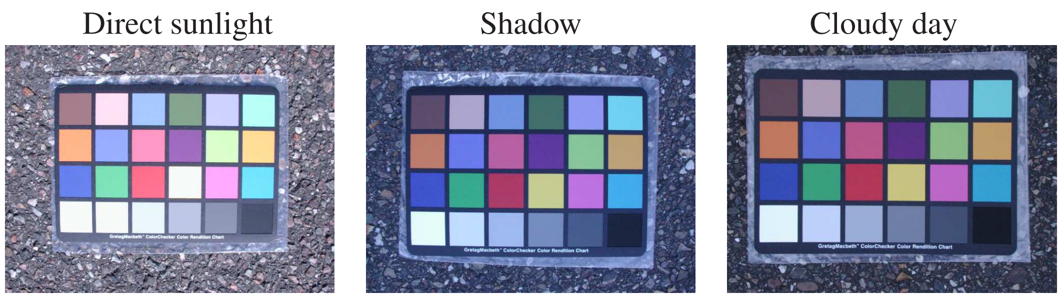


Figure 5.2: Three images of the colour chart under different illumination conditions.

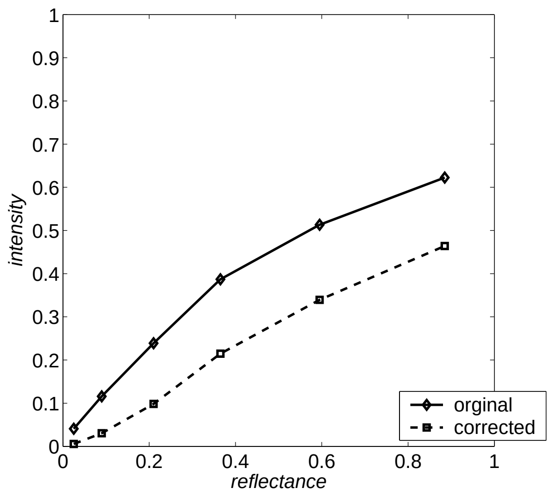


Figure 5.3: The solid curve shows the effect of gamma correction. The dashed line indicates the result after removing the gamma correction.

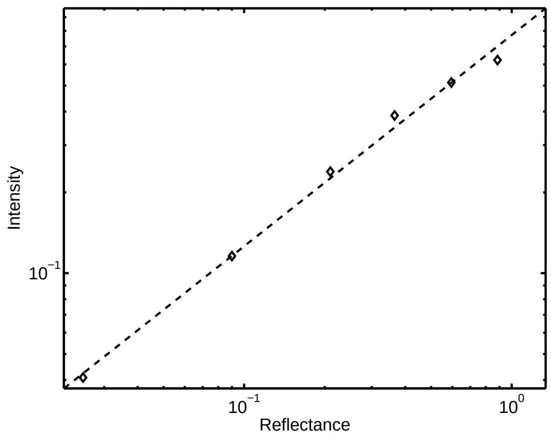


Figure 5.4: Log-Log plot of reflectance and normalised intensities. The dashed line indicates the Least-Squares fit.

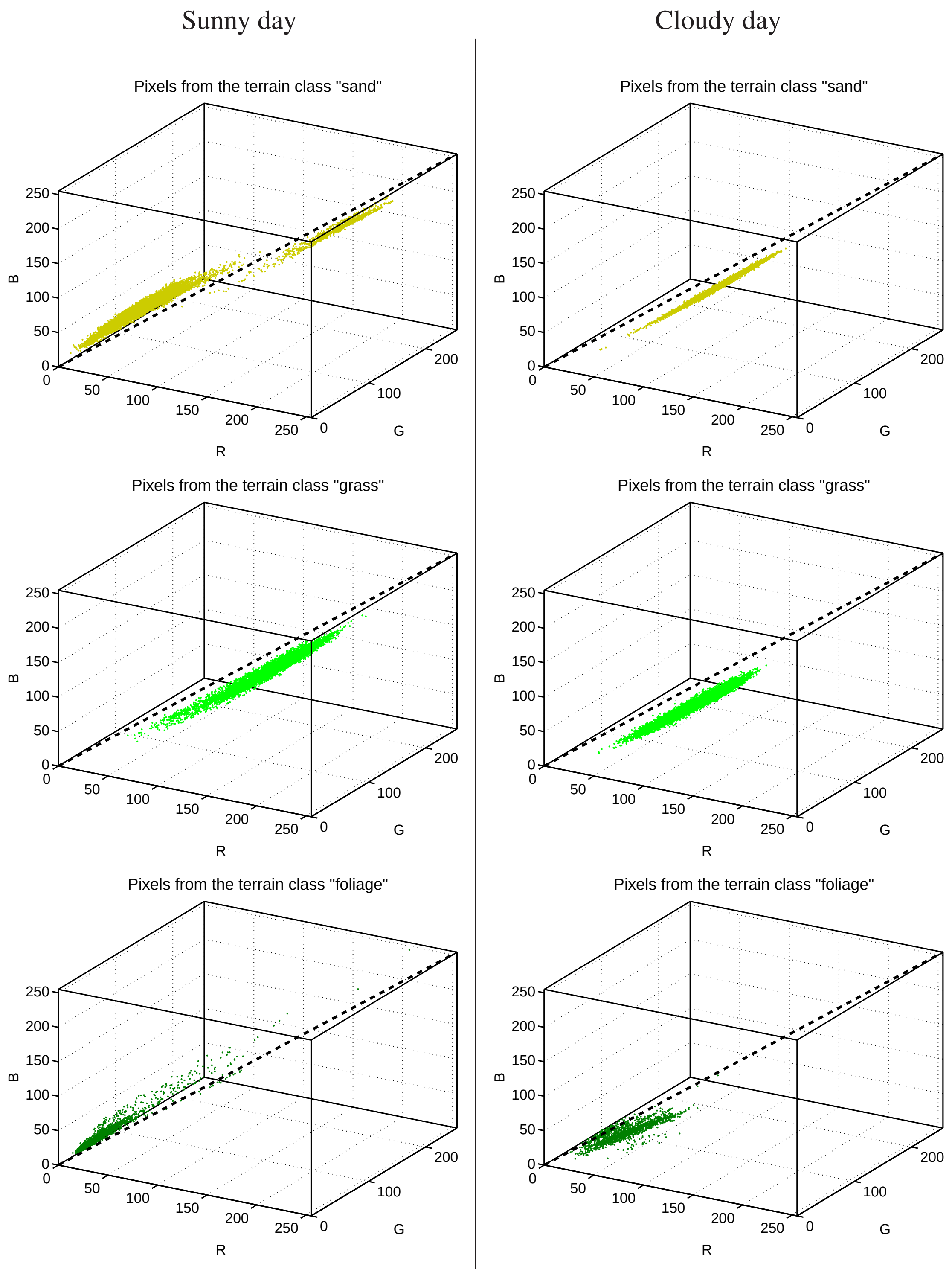


Figure 5.5: Colour distribution of selected sand, grass and foliage pixels in the *RGB* space. The left column shows the distributions of the sunny day image of fig. 5.1, while the right column shows the distributions for the cloudy day image.

pixels taken from materials in the sunny image, while the right side column shows the same, based on the cloudy image. A striped line has been added to each of the colour spaces to indicate the diagonal from black (0,0,0) to white (255,255,255).

Since the geometry of the material surfaces is irregular, most of the pixels apparent colours can be attributed to Lambertian reflectance. This is consistent with what can be seen in the colour spaces. Most pixels are located in single clusters due to the single illumination source and the Lambertian reflectance causes them to be elongated along the transition from dark to light colours.

The goal of colour constancy is to design measures or techniques that are invariant to changes in the apparent colour of a material. In research by Gevers et al. [29] it was shown that if the reflection properties of a material comply with the dichromatic model of Eq. 5.4, its apparent colour will be constant in colour spaces such as normalised *RGB*, even if the reflection geometry or the illumination intensity changes. Unfortunately, this is only true if the material is illuminated by white coloured light. On sunny days, there is differently coloured direct sunlight and skylight illumination. As can be seen in the left column of fig. 5.5, this leads to two pixels clusters in the colour space. The main axis of these clusters do not overlap because the illumination colour is different.

Methods for colour constancy outdoors need to be invariant for different types of illumination. There are measures [29, 88] that are invariant to the illumination colour. However, these types of invariants act like first order derivatives; they indicate changes in the albedo near surface edges and do not help to identify the material type.

An alternative method was developed by Tsin et al. [87]. A Bayesian likelihood model based on Lambertian reflection was developed. With this model, surface albedo and illumination are separately estimated for pixels in an image data set. Based on the estimates, a table is composed for each material class that contains samples of the averaged apparent colour under the different illumination types. Pixels from a new image are initially classified by searching the best matching apparent colours in the table. The retrieved surface albedo and illumination are used as initialisation values in a Maximum-A-Postiori (MAP) approach to optimise the estimation of each pixel's class, albedo and illumination.

The difficulty for this approach is that a class might contain materials with different albedos. In the cloudy example of fig. 5.1, different species of trees are visible. Due to the different albedos of the foliage, pixels are distributed over two elongated clusters. In the right column of fig. 5.5 these are located side-by-side.

5.2.2 Texture features

Colour is not the only available visual feature. Many terrain types also exhibit different types of textures. An example is the pattern caused by blades of grass. Due to different illumination angles and shadows, not every blade reflects the same amount of light. This causes changes in light intensity near their edges. When the blades have similar sizes, edges between them will occur at regular intervals. This corresponds to a signal

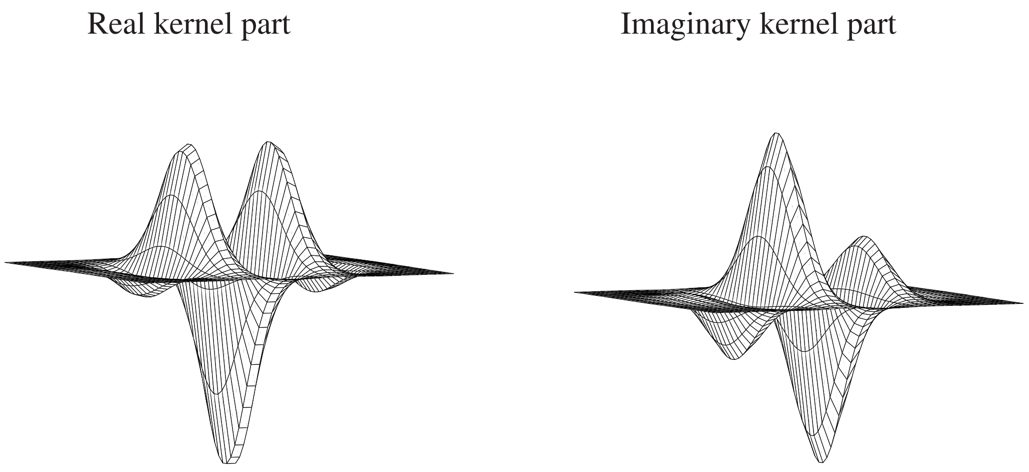


Figure 5.6: The Gabor filter kernel. Both the real part (left) and the imaginary part (right) of the kernel are shown.

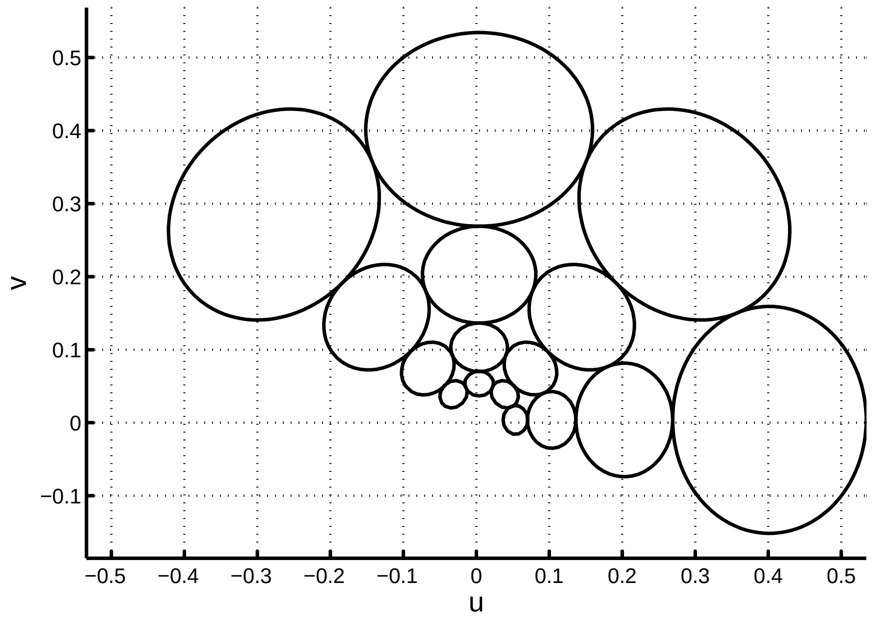


Figure 5.7: Half-way iso-lines of the Gabor filters magnitudes in the dictionary.

with a spatial frequency. If the grass is long, a majority of the edges between the blades will be orientated vertically. A texture can therefore have a specific spatial frequency and orientation.

A problem with applying texture measures in outdoor scenes is the large variation in distances. In our data set, intensity edges and lines on terrain types such as grass, gravel and sand are only distinguishable at close distances. At longer ranges, material patches often appear textureless, while edges and lines are only caused by transitions between different material types at object edges. Therefore, a material texture pattern can change at different ranges.

Gabor filters are scale and orientation variant detectors for edge and line features. These texture features can be extracted from an image by a convolution with a mask obtained from the two dimensional Gabor function $g(x,y)$:

$$g(x,y) = \left(\frac{1}{2\pi\sigma_x\sigma_y}\right)\exp\left(-\frac{x^2}{2\sigma_x^2}-\frac{y^2}{2\sigma_y^2}+2\pi\mathbf{i}\omega x\right) . \tag{5.7}$$

Here, σ_x and σ_y are the horizontal and vertical kernel scales and ω is the spatial frequency to which the filter is the most sensitive. Note that $g(x,y) \in \mathbb{C}$, the real part represents the symmetric part of the filter, while the asymmetric part is imaginary. Fig. 5.6 shows the filter kernel shape of both the symmetric and asymmetric part. The result r of a convolution with this filter also contains a real $r_\mathbb{R}$ and an imaginary $r_\mathbb{C}$ part. The magnitude of r is the amplitude A of the filter response:

$$A(x,y) = \sqrt{r_\mathbb{R}(x,y)^2 + r_\mathbb{C}(x,y)^2} . \tag{5.8}$$

The Gabor filter of Eq. 5.7 is the most sensitive to edges or lines that have spatial frequency ω and are vertically (y-axis) aligned in the images. In order to distinguish materials by their different textures, several filters with different orientations and frequencies are needed.

In Manjuntah and Ma [53], the use of a Gabor filter dictionary is proposed that covers a frequency interval bound by a lower ω_l and upper ω_h frequency, N different orientations and S scales. A filter in the dictionary is indicated with g_{sn}, where both s and n are integers with values $s = \{0,1,....,S-1\}$ and $n = \{0,1,....,N-1\}$. Each $g_{sn}(x,y)$ function can be computed by rotating and scaling the Gabor function $g(x',y')$ with $\omega = \omega_h$.

$$g_{sn}(x,y) = a^{-s}g(x',y') \tag{5.9}$$

Here, a^{-s} is a scale factor and the vector $(x'\,y')^\top$ is obtained by rotating and scaling the input coordinates vector $(x\,y)^\top$:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = a^{-s}\begin{pmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} \quad \text{with} \quad \phi = \frac{n\pi}{N} . \tag{5.10}$$

The redundancy in the dictionary is reduced by ensuring that the filters do not overlap too much in the frequency domain. The values of a, σ_x and σ_y are set in such a way that the half-way isolines of the filters magnitudes touch each other.

$$a = \left(\frac{\omega_h}{\omega_l}\right)^{\frac{1}{S-1}} \tag{5.11}$$

$$\sigma_x = \frac{2(a-1)\omega_h}{\pi(a+1)\sqrt{2\ln 2}} \tag{5.12}$$

$$\sigma_y = \frac{2}{\pi}\left(\tan\frac{\pi}{2N}\right)\left(\omega_h - 2\ln\left(\frac{2(\frac{\pi}{2}\sigma_x)^2}{\omega_h}\right)\right)\left(2\ln 2 - \left(\frac{(2\ln 2)^2(\frac{\pi}{2}\sigma_x)^2}{\omega_h}\right)\right)^{-\frac{1}{2}} \tag{5.13}$$

The Fourier transform of $g(x,y)$ is $G(u,v)$. Fig. 5.7 shows a dictionary with $G(u,v)$ functions on the interval $\omega_l = 0.05$, $\omega_h = 0.4$ with four orientations and scales. The ellipses indicate the half-way isolines of the filter magnitudes.

By filtering an intensity image with a Gabor dictionary, feature vectors of S times N amplitude values are obtained for each input pixel. It is also possible to use colour-texture features. In Hoang et al. [36] an approach to image segmentation is presented where colour channels are separately filtered with Gabor dictionaries. In order to prevent high computational costs due to large feature vectors, Principal Component Analysis (PCA) [71] is used to reduce the dimensionally of the feature space.

In our experiments we will not use texture extracted from colour images. Only textures obtained from intensity images are used. An important advantage of methods that do not rely on colour is that they are also suitable for night time conditions.

5.2.3 Geometric height feature

In the introduction it was explained that range information alone is not sufficient to distinguish between all the terrain types. This is why previous approaches have used either colour [12, 52] or texture [15] features. However, it could also be advantageous to combine different types of features into one feature space. For example, combining texture and colour could help to improve the distinction between very similar classes such as grass and foliage.

In spite of being an insufficient terrain type discriminator on its own, a combination of geometry features with other features could also lead to improved classification results. A simple geometric feature, available in every type of image, will be used to investigate if this is true. The feature is simply the vertical image position (y) of each pixel. The idea is that this acts as a "heuristic" that indicates the typical position of a terrain type in an image. For example, sky pixels are almost always located in the top part of the image. Generally, grass pixels occur more frequently on the bottom part of an image. Because the foliage pixels are more often located in the middle part of an image, the height feature could also help to improve the distinction between these pixels classes.

5.3 Environment state algorithm

The main problem for vision based approaches to terrain recognition is that features of a single terrain type can change from image to image due to influences such as illumination, scene geometry and camera properties. However, it is our assumption that if the features from corresponding materials types, taken from different images, are similarly distributed, that then there are also similarities between the environment influences of the individual images. Such a group of images is considered to share the same "environment state". Because there is less variation in the conditions in an environment state, terrain type features will also be less effected by them. Restricting terrain type features to a single environment state will therefore improve the distinction between them. This could help to increase the classification performance.

Fig. 5.8 shows an outline of the algorithm for finding environment states. It starts out with a set of labelled training images. Not all pixels are labelled; only patches typical for each material are indicated roughly by a polygon border. An example of an input image with labelled areas can be seen in fig. 5.1.

In the first step, the training set in segmented into clusters of similar images. Each cluster is regarded as an environment state. In the next step, environment specific models are trained for the different terrain types.

How new (unlabelled) images are classified with a set of environment specific terrain type models is shown in fig. 5.9. The new image is first compared to all the different environment states. It is subsequently classified with the terrain type models from the environment state to which it is most similar.

The next sections describe ML classification based on GMMs. Also the method that is used to segmented a training set of images into environment states is presented in detail.

5.3.1 Image pre-processing

The image pre-processing step is required to convert input image pixels to feature vectors. In order to reduce the influence of the camera gamma correction, colour values are linearised with the technique described in section 5.2.1.

For colour features, the input RGB colour space is transformed to the CIE *Lab* [16] colour space. This colour space is used because it has a separate intensity axis and has perceptually uniform colour distribution. The Gabor texture features are obtained from the intensity L channel of this space. Two types of texture vectors can be extracted; full length $(A_1, A_2, ..., A_{16})^\top$ or PCA reduced size $(A_{\text{PCA1}}, A_{\text{PCA2}}, A_{\text{PCA3}}, A_{\text{PCA4}})^\top$.

5.3.2 Maximum likelihood classifier

In our approach, a ML classifier with GMMs is used to model the distribution of features in the feature space. A single distribution is modelled by the multi-variate Gaus-

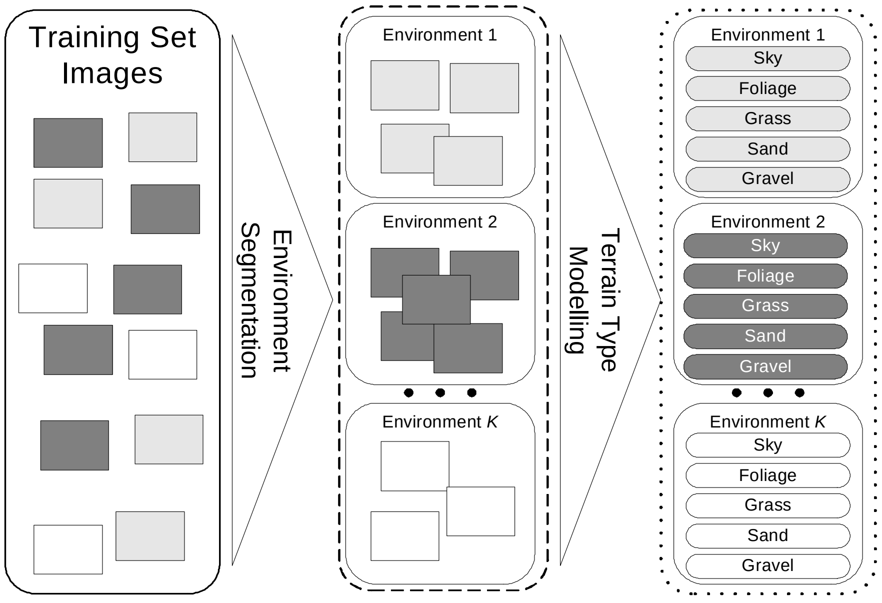


Figure 5.8: Outline of the algorithm for training environment specific terrain type models.

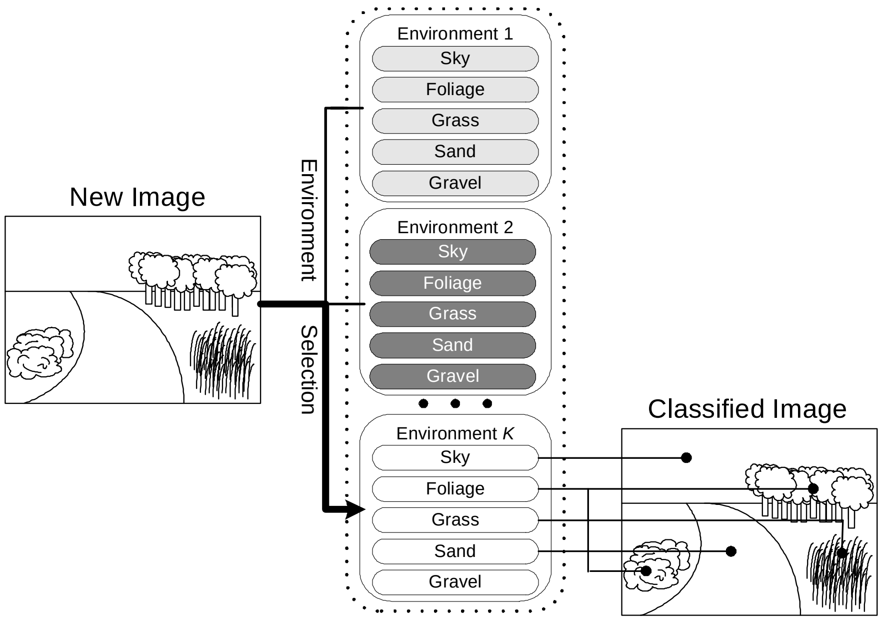


Figure 5.9: Outline of classification with the environment specific terrain type models.

sian density function:

$$\phi(\mathbf{x},\mathbf{m},\mathbf{C}) = \frac{\exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m})^\top \mathbf{C}^{-1}(\mathbf{x}-\mathbf{m})\right)}{(2\pi)^{\mathrm{n}/2}\sqrt{\det(\mathbf{C})}} \quad . \tag{5.14}$$

Where $\mathbf{x}$ is a vector in the feature space, $\mathbf{m}$ is the mean of the distribution and $\mathbf{C}$ defines its covariance matrix. The finite GMM composed of k Gaussian distributions, each with an individual positive weight w_j is defined as:

$$f_k(\mathbf{x}) = \sum_{j=1}^{k} w_j \phi(\mathbf{x}_j,\mathbf{m}_j,\mathbf{C}_j) \text{ with } \sum_{j=1}^{k} w_j = 1 \quad . \tag{5.15}$$

In order to obtain the GMM of a distribution with n pixels, the goal is to estimate the parameters for each of the k components in the mixture. The EM algorithm [71] is a widely used technique for estimating the values of these 'hidden' parameters. It uses iteration to maximise the log-likelihood of the mixture on all vectors:

$$L_k = \sum_{i=1}^{n} \log f_k(\mathbf{x}_\mathrm{i}) \quad . \tag{5.16}$$

Given a mixture containing k components each with initial guesses for their means and covariances, the algorithm optimises the estimation. The first step uses the current variables to estimate the probability that distribution j generated vector $\mathbf{x}_i$:

$$P(j|\mathbf{x}_i) = \frac{w_j \phi(\mathbf{x}_i,\mathbf{m}_j,\mathbf{C}_j)}{f_k(\mathbf{x}_i)} \quad . \tag{5.17}$$

Given the new probabilities the weights, means and covariances are updated:

$$w_j = \frac{1}{n}\sum_{i=1}^{n} P(j|\mathbf{x}_i) \quad , \tag{5.18}$$

$$\mathbf{m}_j = \frac{\sum_{i=1}^{n} P(j|\mathbf{x}_i)\mathbf{x}_i}{\sum_{i=1}^{n} P(j|\mathbf{x}_i)} \quad , \tag{5.19}$$

$$\mathbf{C}_j = \frac{\sum_{i=1}^{n} P(j|\mathbf{x}_i)\left(\mathbf{x}_i - \mathbf{m}_j\right)\left(\mathbf{x}_i - \mathbf{m}_j\right)^\top}{\sum_{i=1}^{n} P(j|\mathbf{x}_i)} \quad . \tag{5.20}$$

Because it has been shown that the log-likelihood cannot decrease after an EM step [98], the steps can be repeated until convergence.

In spite of the simple implementation and the guaranteed convergence, the basic EM algorithm has some limitations. Firstly, it assumes that the number of components of the Gaussian mixture is known a-priori. Secondly, it requires some way of initialisation or guess of the component parameters. Thirdly, the algorithm is sensitive to local maxima of the likelihood function.

Using a fixed number of components has some drawbacks. The data will be poorly modelled if the number of components is to small. It can also be over-fitted if too many components are used.

To overcome the drawbacks of the basic EM algorithm, Vlassis and Likas [98] developed the Greedy EM algorithm. We use this algorithm because it can also provide an estimate for the optimal number of components in a mixture. The algorithm works by adding components successively to the mixture. If $\phi(\mathbf{x},\mathbf{m},\mathbf{C})$ is a component it can be added to mixture $f_k(\mathbf{x})$, to form the new mixture:

$$f_{k+1}(\mathbf{x}) = (1-a)f_k(\mathbf{x}) + a\phi(\mathbf{x},\mathbf{m},\mathbf{C}) \ . \tag{5.21}$$

This algorithm starts out with a mixture which only contains one component with a mean and covariance based on the whole training set data. Regular EM steps are performed until the log-likelihood of the mixture converges. Then, the algorithm searches for a new component $\phi(\mathbf{x},\mathbf{m},\mathbf{C})$ and corresponding mixing weight a, which maximises the likelihood of L_{k+1}:

$$L_{k+1} = \sum_{i=1}^{n} \log\left((1-a)f_k(\mathbf{x}) + a\phi(\mathbf{x},\mathbf{m},\mathbf{C})\right) \ . \tag{5.22}$$

The parameters of the old mixture $f_k(\mathbf{x})$ are kept fixed during this search. For a detailed description for finding the optimal component we refer to the original paper of Vlassis and Likas [98]. More efficient and computationally faster methods can be found in Verbeek et al. [96] and Nunnink et al. [64].

After the component has been found, it is added to the mixture and regular EM steps can be used until convergence. The addition of a component to the current mixture can be repeated several times. Cross validation is used to determine the optimum number of mixture components. This requires the division of the original training set into a new training set and test set. The new training set is used to learn the components, while the likelihood on the whole data set, is used to determine the optimum number of mixture components.

Other approaches by Castano et al. [15] and Manduchi [52] used regular EM to model the feature distributions of different terrain type classes. In their approaches, each terrain type class c is modelled with a GMM $f\mathrm{c}_k$. The class of a pixel with feature vector $\mathbf{x}$ is obtained by finding the GMM with the maximum log-likelihood:

$$\mathrm{c} = \underset{\mathrm{c}\in Classes}{\arg\max}(\log f\mathrm{c}_k(\mathbf{x})) \ . \tag{5.23}$$

In our approach, the GMMs that model the terrain types are *environment specific*. Each terrain type is coupled to an environment state. GMMs are also used to model the environment states. They model the feature distribution for all terrain types in a set of similar images. New images are first classified to a particular environment using the ML approach. Then, the associated terrain type classes are used to classify individual image pixels. The next section explains our method for finding environment states in an image training set.

5.3.3 Finding environment states

Segmenting a training set of images into environment states requires that three types of feature distributions are modelled with GMMs. The first type is the environment specific feature distributions f of labelled terrain type pixels. These are modelled for each environment state after the training set has been segmented. The second type, which we will indicate with g, models the feature distribution of a single image. It is obtained by merging all labelled pixels in the image into one set. Environment states are basically sets of images with similar feature distributions. Therefore, the third type of GMM, which we will indicate with h, models environment states. These are obtained by merging labelled pixels from all the images within an environment state. The three types of GMMs are modelled on the different feature sets with the Greedy EM algorithm.

Grouping images into environment states requires a similarity measure that allows us to compare image GMMs to each other and to environment GMMs. In order to compare an image mixture model to an environment state, the Kullback-Leibler divergence (KL) [71], also known as relative entropy, is used. This is a measure for computing the distance between two GMMs g and h. The Kullback-Leibler divergence is not a true distance metric. For example, it is not symmetric: $KL(g||h) \neq KL(h||g)$. However, the metric is only zero if $g(\mathbf{x}_\mathrm{i}) = h(\mathbf{x}_\mathrm{i})$. It is also a convex function of $g(\mathbf{x})$ and its values are always positive.

It is well known [30] that Kullback-Leibler divergence between two GMMs cannot be analytically computed. If a set of feature $\mathbf{x}_\mathrm{i}$ vectors belonging to g is available, the following equation can be used to approximate KL:

$$KL(g||h) \cong \sum_{i}^{n} g(\mathbf{x}_i) \log \frac{g(\mathbf{x}_\mathrm{i})}{h(\mathbf{x}_\mathrm{i})} \quad . \tag{5.24}$$

Because this measure is asymmetric, it is only suitable for comparing multiple images g to a single environment state h. For comparing two images g_1 and g_2 an approximation of the symmetric KL distance is used:

$$D(g_1||g_2) \cong \frac{1}{n_1} \sum_{i=1}^{n_1} \log \frac{g_1(\mathbf{x}_{1\mathrm{i}})}{g_2(\mathbf{x}_{1\mathrm{i}})} + \frac{1}{n_2} \sum_{j=1}^{n_2} \log \frac{g_2(\mathbf{x}_{2\mathrm{j}})}{g_1(\mathbf{x}_{2\mathrm{j}})} \quad . \tag{5.25}$$

Both measures are used in an adaptation of the algorithm by Greenspan et al. [30] for an automatic image retrieval system.

At the start, the set of all training images is considered to be one environment state. Iteration is used to segment this state into more states.

In each segmentation step, the first task is to find the environment state that should be segmented into smaller ones. For the first iteration this is trivial because there is only one environment state. In subsequent steps, it is important to select the environment state that contains the most dissimilar images. This ensures a gradual segmentation from large differences between images in the first iterations to small differences in the latter iterations.

Eq. 5.25 is used to compute distances between the images in an environment state. For each environment state, all the unique pairwise distances $d(i,j)$ between the images mixtures are computed:

$$\forall ij \rightarrow i = 1..(m-1) \wedge j = (i+1)..m, \quad d(i,j) = D(g_i, g_j) \ . \tag{5.26}$$

The variance of all $d(i,j)$ indicates if the distances in a state are similar to each other or not. After this value is computed for all environment states, the state with the highest variation is selected for further segmentation.

The algorithm now tries to segment the selected environment state into two smaller states. This is an iterative process which involves determining two mixture models, each representing a new environment state, and finding out which of the images from the old environment state belongs to them.

In order to start this process with two GMMs that are the most unlike, our approach selects the pair of images with the largest distance D. In the iteration, Eq. 5.24 is used to compute the distance $KL(g_i,h_j)$ between each of the image GMM and the new environment GMM. Each image is assigned temporally to the environment to which it has the shortest distance. The two environment GMMs are then updated with Greedy EM using points collected from the images that were assigned to them. Hereafter, the likelihood is calculated of the new GMM with the data. The loop is repeated until there is no significant increase in likelihood. The resulting two new environments states replace the old one.

It is possible that during the iteration all the images are assigned to only one of the GMMs. In this case, the GMM with no images is removed and the old environment state is returned with a flag that it cannot be segmented any further.

Segmentation of all environment states is halted when a desired number K of states is reached or when they cannot be segmented any further.

5.4 Results with real outdoor images

A colour image collection of outdoor natural terrain was used to evaluate our methods. The images where recorded with a digital Nikon Coolpix 990 mega-pixel camera and reduced to a resolution of 320 by 240 pixels via linear interpolation. The images were recorded under three different weather types. The first type was sunny, the second overcast and the third rainy. Some images are shown in the left column of fig. 5.14.

The set used for our experiments consists out of 89 images. In each image, patches of the visible terrain types were labelled by hand. This set was divided over a training set (44 images) and a test set (45 images). In the training set, the labelled areas provide the terrain type information which is used in the cluster and learning steps. The labels of pixels in the test set are only used as ground truth in order to evaluate classification results.

As discussed earlier, colour, texture and height features are extracted from the image sets. In the experiments several feature combinations will be tested.

5.4.1 Performance measures

Confusion matrices can be used to evaluate the performance of classifiers. For our classifiers, each entry $CM_{j,k}$ in such a matrix describes the number of times a pixel with ground truth label j has been classified as k. Castano et al. [15] propose three quantitative performance measures based on confusion matrices. The first measure is the ratio of correctly classified points among the total number of points in the complete test set:

$$P_C = \frac{\sum_j CM_{j,j}}{\sum_j \sum_k CM_{j,k}} \ . \tag{5.27}$$

An estimate of the correct classification probability for each class j is given by:

$$P(C|\,j) = \frac{CM_{j,j}}{\sum_k CM_{j,k}} \ . \tag{5.28}$$

A disadvantage of the global measure P_C is that it is biased towards the class that occurs the most in the test set. The unbiased overall probability of correct classification $\overline{P}_C$ is equal to the average of $P(C|\,j)$ for all five classes:

$$\bar{P}_C = \frac{1}{5}\sum_j P(C|j) \ . \tag{5.29}$$

5.4.2 Segmentation experiments

The novelty of our approach is the environment specific terrain type classification. An important choice is the number of environment states K that should be used. Therefore, the influence of K on the classification performance should be investigated.

Based on only colour

For the first experiment only colour *Lab* features are used. The training set is segmented with our method until it cannot be segmented any further. Some of the segmentation results are shown in table 5.1. For $K = 1$ until 8, 27 and 28, the columns show the number of segmented environments. Each environment state is placed in the row which indicates the type day (sunny, cloudy or rainy) on which the majority of its images was recorded.

Table 5.1: The resulted environment clusters for varying threshold K using only colour features.

	Segmentation threshold K value									
	1	2	3	4	5	6	7	8	...	27	28
C			1x11/1	1x11/1	1x11/1	1x11/1	1x2/1	1x2/1	...	4x1/0	4x1/0
L							1x9/0	1x9/0	...	5x2/0	5x2/0
O											
U											
D											
R		1x13/0	1x13/0	1x1/0	1x1/0	1x1/0	1x1/0	2x1/0	...	6x1/0	6x1/0
A				1x12/0	1x5/0	1x5/0	1x5/0	1x4/0	...	1x2/0	1x2/0
I					1x7/0	1x7/0	1x7/0	1x7/0	...	2x3/0	2x3/0
N											
S	1x16/28	1x16/15	1x16/3	1x16/3	1x16/3	1x5/3	1x5/3	1x5/3	...	3x1/0	4x1/0
U						1x11/0	1x11/0	1x11/0	...	2x2/0	3x2/0
N									...	1x3/0	1x6/0
									...	1x6/0	

Where $n \times m/o$ means n clusters of m images of which o where not recorded on the same day as the majority.

Table 5.2: The resulted environment clusters for varying threshold K using both colour and height features.

	Segmentation threshold K value										
	1	2	3	4	5	6	7	8	...	26	27
C					1x13/2	1x13/2	1x1/0	1x1/0	...	5x1/0	5x1/0
L							1x13/2	1x2/0	...	4x2/0	4x2/0
O								1x11/2	...		
U											
D											
R		1x14/0	1x2/0	1x2/0	1x2/0	1x2/0	1x2/0	1x2/0	...	2x1/0	2x1/0
A			1x12/0	1x4/0	1x4/0	1x4/0	1x4/0	1x4/0	...	6x2/0	6x2/0
I				1x8/0	1x8/0	1x8/0	1x8/0	1x8/0			
N											
S	1x16/28	1x16/14	1x16/14	1x16/14	1x14/1	1x1/0	1x1/0	1x1/0	...	4x1/0	5x1/0
U						1x13/1	1x13/0	1x13/0	...	1x1/1	1x1/1
N									...	1x2/0	2x2/0
									...	1x3/0	1x6/0
									...	1x6/0	

Where $n \times m/o$ means n clusters of m images of which o where not recorded on the same day as the majority.

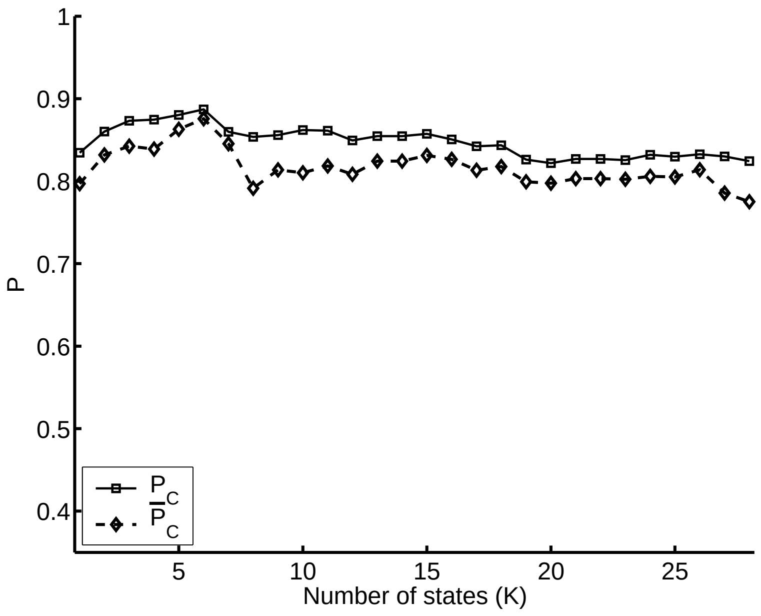


Figure 5.10: Overall classification probabilities with only colour features.

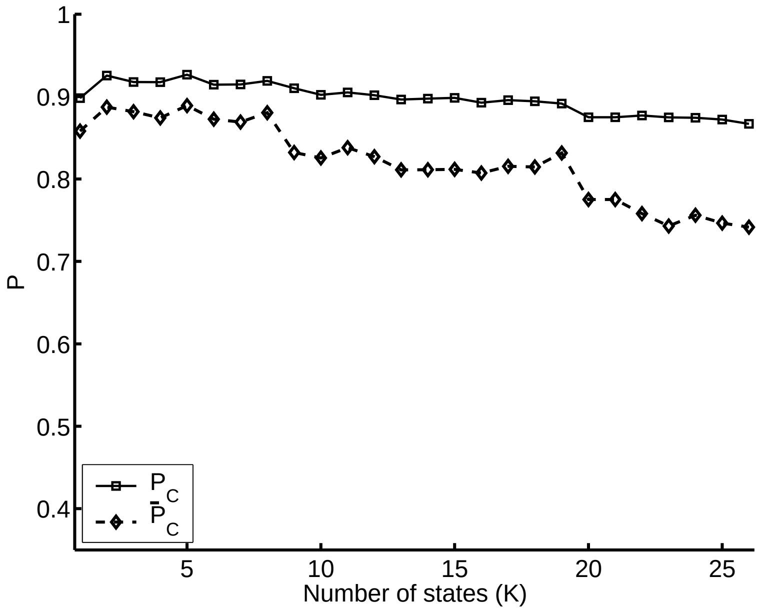


Figure 5.11: Overall classification probabilities with both colour and height.

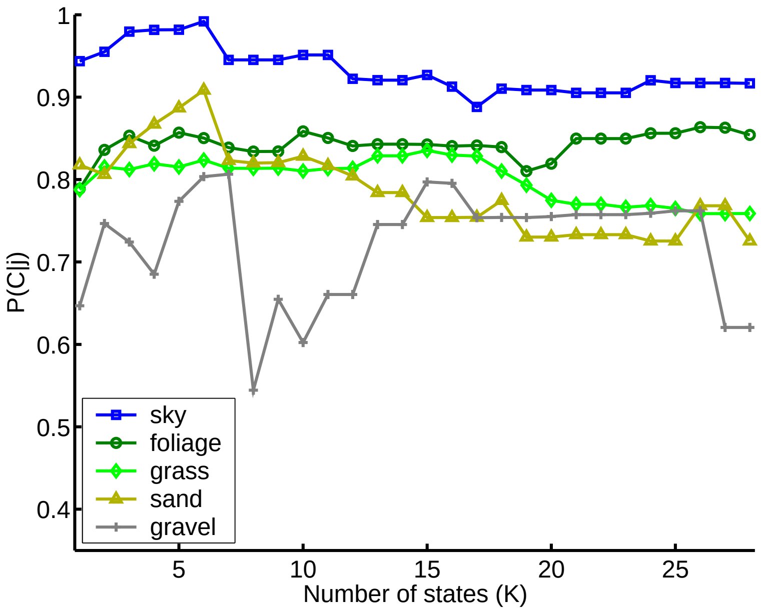


Figure 5.12: Individual terrain type classification probabilities with only colour features.

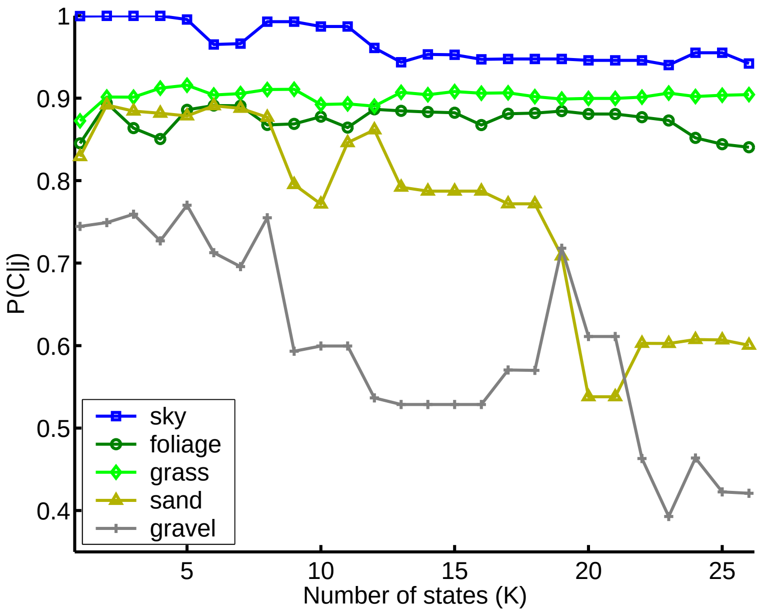


Figure 5.13: Individual terrain type classification probabilities with colour and height.

In case of the initial setting of $K = 1$, all images are grouped into one environment state. At the first segmentation of $K = 2$, an environment state with 13 of the 14 rainy images is found. The next step, ($K = 3$) creates an environment state with 11 of the 14 cloud images. After only two steps, each of the three environment states already contains a clear majority of images recorded under the same weather conditions. The steps of $K = 4$ until 5 are partitions of the "rain" environment state. At $K = 6$ the "sun" environment state is partitioned. Closer inspection revealed that the result was one state with sunny images showing a blue sky and another state with both sunny and cloudy images showing a white sky due to pixel saturation. In subsequent steps, the different environment states are segmented further. At $K = 28$ the iteration is halted because none of the environment state can be segmented anymore. Here, the majority of states contain less than three images.

All the environment states, found at each of the 28 segmentation steps, were used to classify the terrain type of pixels belonging to the 45 test set images. The resulting overall classification probabilities P_C and $\overline{P}_C$ based on only colour features are plotted in fig. 5.10. Both the global measure P_C and the unbiased measure $\overline{P}_C$ indicate that optimum performance is achieved for $K = 6$. In general, a lower or higher K leads to suboptimal classification performance.

In fig. 5.12 the individual classification probabilities for each terrain type are shown. The plot shows that the performance of terrain types such as sand and gravel is much more sensitive to the setting of K. Both these types show clear peaks for $K = 6$, while other show optimum performance for larger intervals of K. This is due to the fact that sand and gravel occur less in our data set than the other classes. An intermediate number for K improves performance and ensures that both gravel and sand occur in every environment state.

Based on colour and height

The same experiment was repeated with a combination of colour features and the geometric height feature that was explained in section 5.2.3. Table 5.2 shows which clusters have been found for $K = 1$ until 8, 26 and 27. Like the previous segmentation results (Table 5.1), the algorithm distinguishes rain images from the other images at $K = 2$. However, in $K = 3$ until 4 this environment state is further segmented before the cloud images are segmented from the sunny images in $K = 5$.

The overall classification probabilities are plotted in fig. 5.11. For $K = 1$ until 10 both measures indicate improved performance when compared to using only colour features. For higher K the measure P_C continues to show this improvement, while the unbiased measure $\overline{P}_C$ shows that from $K = 20$ the performance is actually less than that of the colour only classifier.

Fig. 5.13 shows the individual classification probabilities. Here there is also a differentiation between the most and the least occurring terrain types. The types sky, foliage and grass show an improvement when compared to using only colour features. On the other hand, sand only shows improvements for low K values, while the perfor-

Table 5.3: Performance of the classifiers when the optimal number of environment states is used (Best K). Also, the maximum number of environment states found is shown (Max. K). The different feature types are indicated by; "*Lab*"=Colour, "Gab"=Texture, "PCA"=Reduced texture and "Y"=Pixel height.

				$P(C\|j)$					Best	Max.
Type	Features	P_c	$\overline{P}_c$	sky	foliage	grass	sand	gravel	K	K
Manual	*Lab*	0.87	0.86	0.94	0.84	0.82	0.85	0.86	n.a.	n.a.
Semi	*Lab*	0.86	0.86	0.92	0.85	0.82	0.84	0.87	n.a.	n.a.
Ours	*Lab*	0.89	0.88	0.99	0.85	0.82	0.91	0.80	6	28
Ours	*Lab*, Y	0.93	0.89	1.00	0.89	0.92	0.88	0.77	5	27
Ours	Gab	0.53	0.48	0.93	0.34	0.33	0.52	0.29	2	3
Ours	Gab, Y	0.70	0.65	0.95	0.64	0.56	0.62	0.47	2	4
Ours	Gab, *Lab*,	0.83	0.81	0.98	0.74	0.81	0.70	0.80	2	6
Ours	Gab, *Lab*, Y.	0.90	0.88	1.00	0.88	0.85	0.80	0.88	2	7
Ours	PCA.	0.53	0.48	0.90	0.21	0.49	0.35	0.44	1	23
Ours	PCA, Y	0.71	0.66	0.97	0.71	0.54	0.50	0.60	1	25
Ours	PCA, *Lab*	0.88	0.84	0.97	0.86	0.83	0.80	0.75	4	28
Ours	PCA, *Lab*, Y	0.93	0.92	1.00	0.90	0.90	0.90	0.88	2	29

mance of gravel is lower than what is achieved with only colour features.

5.4.3 Classification performance for colour

Our method can find the number of desired environment states automatically. Alternatively, environment states could also be distinguished with the use of extra sensor information. Examples range from simple light intensity sensors to measuring the (global) illumination colour of a white Lambertian reflector [52].

For this experiment we assume that there is a sensor that measures the global illumination intensity. It can provide a reference that indicates whether an image was recorded in a sunny, cloudy or rainy environment. Our method is compared to two others that use this reference, which we provide by hand. The first, is called "Manual" and uses the reference both for training and classifying the terrain types. The second method is called "Semi" and only uses the reference for training.

The first three rows of table 5.3 show the quantitative results obtained with the classification performance measures. The global measures P_C and $\overline{P}_C$ show that our colour only method, which finds the environment states by itself, performs as well as the other methods which require an external reference.

All three methods show similar classification probabilities for the foliage and grass classes. Our method performs better on the sky and sand classes than the other approaches. It does show lower performance on the gravel class. Nevertheless, the lowest probability for correct classification of this terrain type is 0.80.

Fig. 5.14 shows some of the images from the test set and the classification results from the different classifiers. The colour blue is used to indicate sky, light green is grass, dark green is foliage, yellow is sand and gray is gravel.

The results show that each of the classifiers is capable of recognising the terrain types in images with large environment differences. Except for row six, there are no large differences between the results of the different classifiers. On row six, our approach classifies the sky correctly, while other approaches classify large parts as being gravel.

Some misclassifications do occur. In some of the images, edges between foliage and sky are misclassified as sand or gravel. Identifying shaded grass is troublesome for the classifiers that only use colour features. The example of the second row shows that this is misclassified as being foliage.

5.4.4 Texture and combinations with other features

In addition to colour, texture features were also extracted from the corresponding intensity images. A Gabor filters dictionary was used that covered a frequency interval of $\omega_l = 0.05$ to $\omega_h = 0.4$ with 4 scales and 4 orientations. The extracted 16 dimensional features can be used directly in our approach. However, using such high dimensional feature vectors carries a high computational cost. A PCA technique is used to create a second set of texture feature vectors with the dimensionality reduced to 4.

Several classification experiments were conducted in which both texture types were used. In addition, also various combinations of texture, colour and height features were tested. The results of these experiments have been included in table 5.3. For each experiment the combination of features is indicated; "*Lab*" for colour, "Gab" for full-length texture, "PCA" for reduced size texture and "Y" for the height features. The reported results for individual classification probabilities where achieved by using the optimum number of environment states (Best K). The table also shows the maximum number of environment states found (Max. K). Classification results for some of the images in our test set are shown in figs. 5.14, 5.15 and 5.16.

Classification with only texture features

The results of the experiments based on only texture features reveal low overall classification probabilities. Only the class "sky" is classified well with both types of texture features, while the other classes show much lower probabilities. Due to the large distance variation in outdoor images, textures of the same materials appear different nearby than those further away. The overall performance measures, P_C and $\overline{P}_C$, show the same results for the full and reduced size texture features. Apparently, there is sparseness in the full size texture feature space; the variation is only high for a limited set of sub-dimensions. The poor classification results with only full-length texture features are visible in the second column of fig. 5.15. The same is shown for the reduced size feature vectors in fig. 5.16.

Combinations of colour or texture with height

Compared to using only colour, the overall performance measures in table 5.3 indicate that the combination of colour with height leads to an improvement. For the individual terrain types, this is only true for foliage and grass. The sand and gravel terrain types show a decrease in performance. The fourth column in fig. 5.14 shows the results of this classifier. The examples of the second and the third row show that shaded grass is now correctly recognised. Examples on the sixth and seventh row also reveal a drawback; foliage in the bottom part of the image is misclassified as grass.

Table 5.3 shows that the performance achieved with texture-height combination is clearly better than that of texture only classification. This is understandable, because the height feature improves the distinction between textures that occur more frequently at a specific location in the image. The improved classification results are also apparent in fig. 5.15 and 5.16.

Combinations of colour, texture and height

The overall performance measures of table 5.3 do not indicate that combining colour with texture feature leads to an improvement over using only colour features.

Performance is improved if all the colour, texture and height features are combined. Table 5.3 indicates that the best overall performance with $P_C = 0.93$ and $\overline{P}_C = 0.92$ is achieved by the combination of colour, reduced texture and height.

In general, it can be observed that the combinations of reduced texture features perform slightly better than the combinations of full texture features. The combinations with reduced features can be modelled better with the GMMs. This seems to enforce the observation that the full size textures feature values are sparse.

Some of the classification results of these feature combinations can be seen in fig. 5.15 and 5.16.

5.5 Conclusion

Classification of terrain types is needed to improve the autonomy of robot vehicles in off-road terrain. Colour and texture can be used as features in computer vision based approaches to this problem.

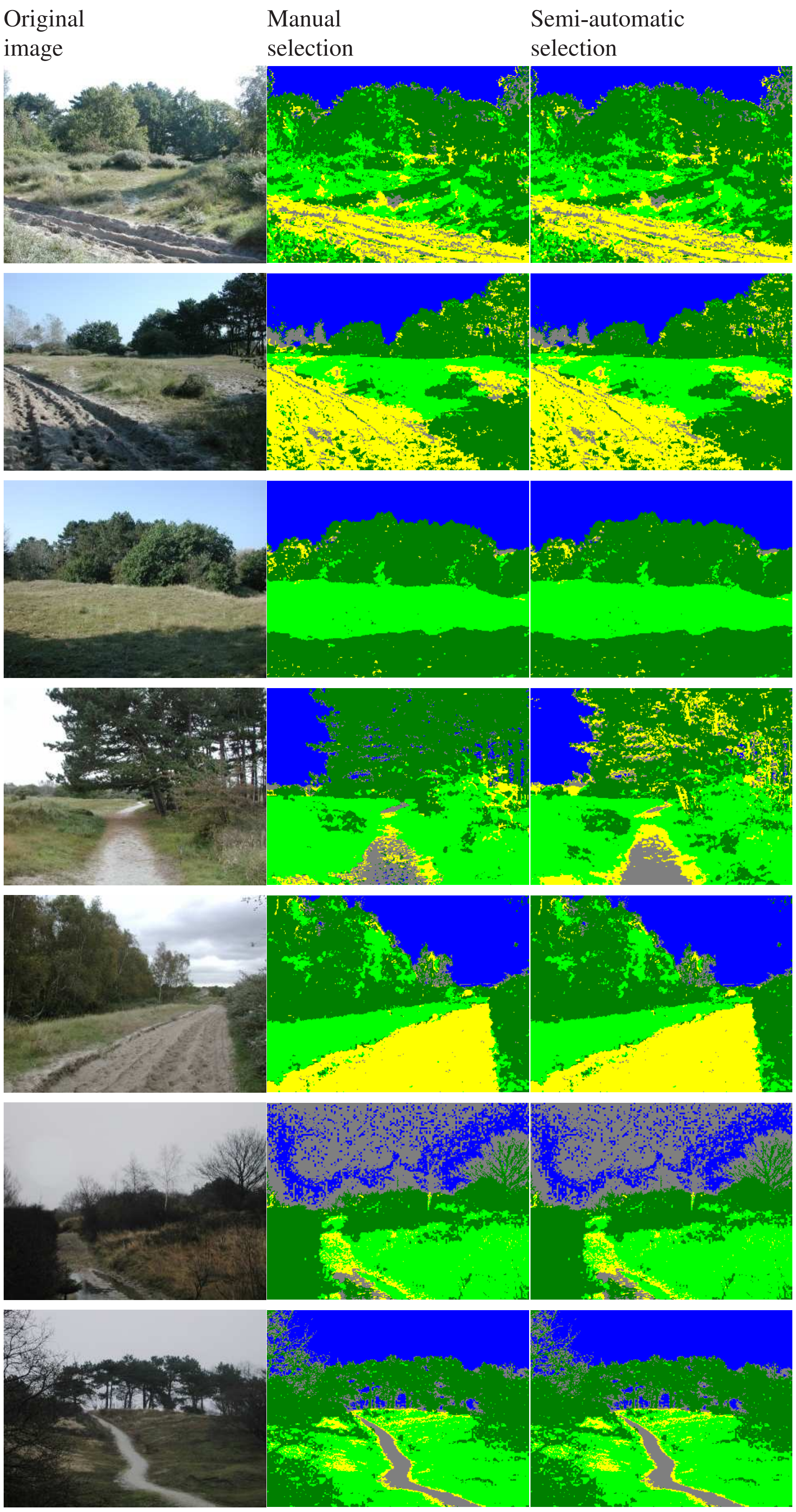
The problem for colour based approaches is that apparent material colour shows a large variation outdoors, due to influences such as illumination and scene geometry. The large variation of distance is also a challenge for texture based approaches, because this influences the appearance of an object texture.

In order to deal with this problem of large variation in the feature values outdoors, we have developed a method that find sets of images that share the same environment conditions. The main idea behind this approach is that feature values in such environment states are less effected by changes of influences such as illumination and geometry. Given a set of input images with labelled terrain types our method can automatically segment it into a desired number of environment states.

The method was evaluated with a set of outdoor images that were recorded under different weather and illumination conditions. Experiments with only colour features show that the environment states found by our method mostly contain similar images. They also show that the use of an intermediate number of states improves the terrain type classification performance. The performance of our colour based approach was compared to that of other methods that need an external reference in order to compensate for image illumination conditions during either training or classification of the terrain types. Despite the fact that our approach does not use such an external reference, the results show that its performance is competitive to that of the other methods. On the (unseen) test-set, our colour only method achieves an unbiased classification probability of 0.89.

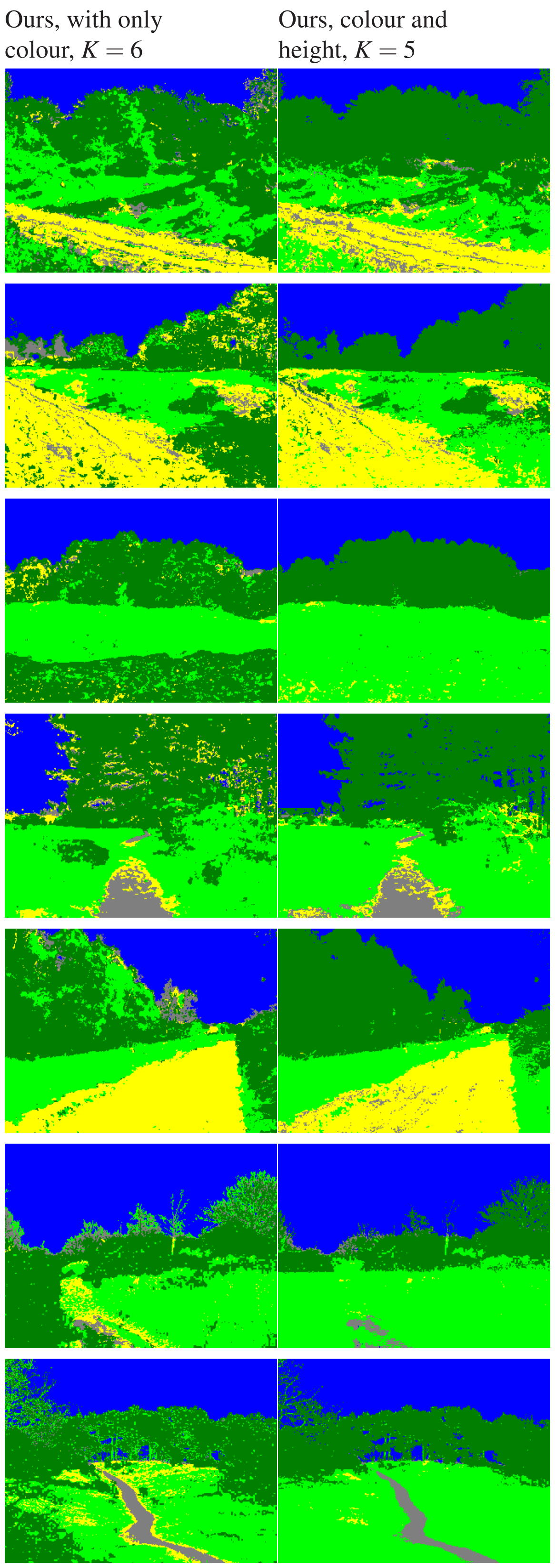
In addition to colour features, various combinations of texture features extracted with Gabor filter banks and geometric features were also tested with our approach. Experiments with only texture based features did not provide satisfactory results due to sparseness in the extracted features values. However, the results are significantly improved if texture is combined with a simple geometric height feature. Combinations of texture and colour turned out to be inferior to colour only classification. A slight performance increase was obtained by reducing the dimensionality of the texture data in combinations of texture with other features.

The best performance is achieved when our method is used with colour, reduced texture and height features. Its unbiased classification probability turned out to be 0.92.



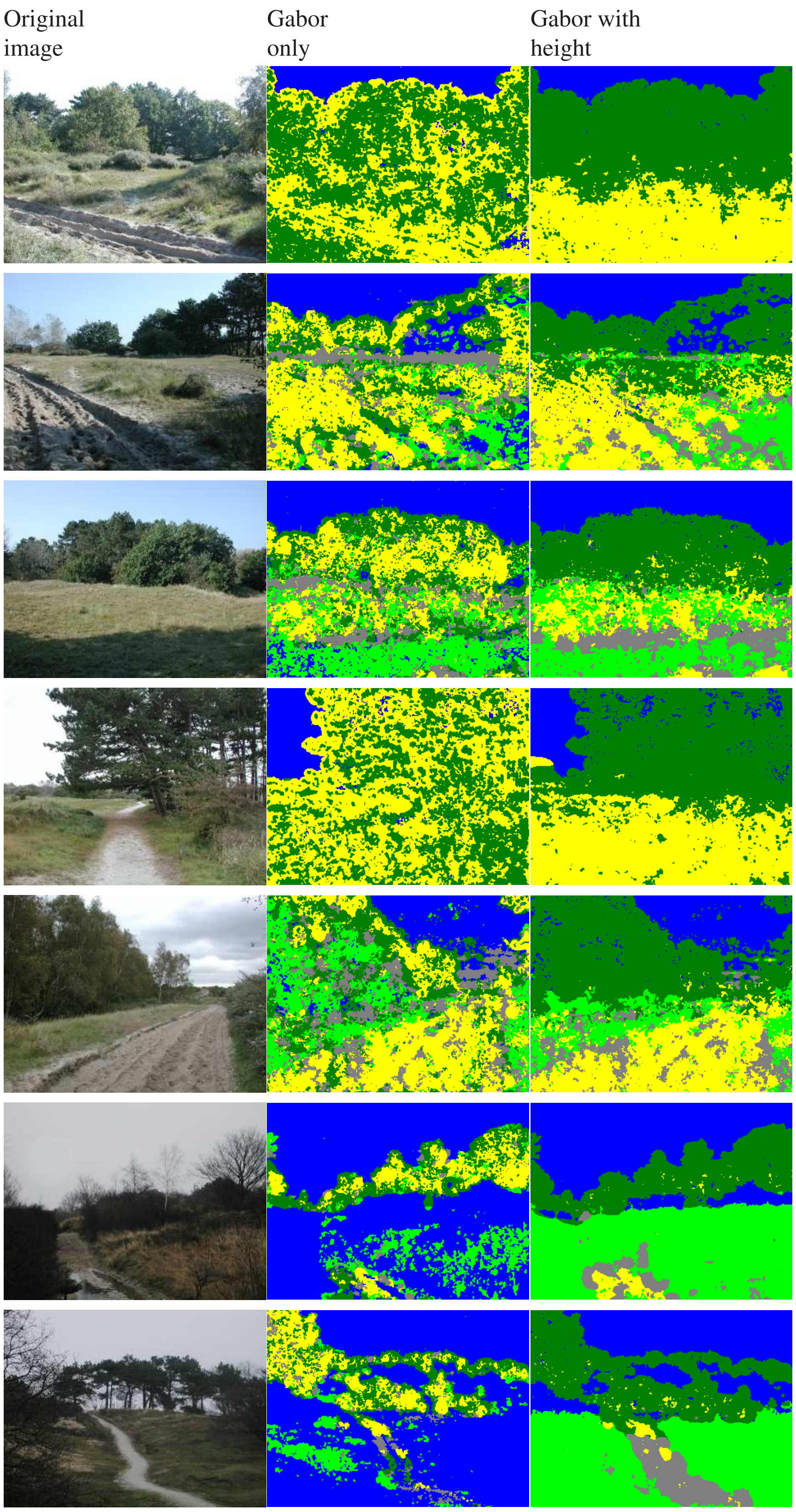
Terrain type classification: *blue*=sky, *dark green*=foliage, *green*=grass, *yellow*=sand and *gray*=gravel.

Figure 5.14: Results of the different colour classifiers on test set images with varying environment conditions.


Ours, with only colour, $K = 6$
Ours, colour and height, $K = 5$

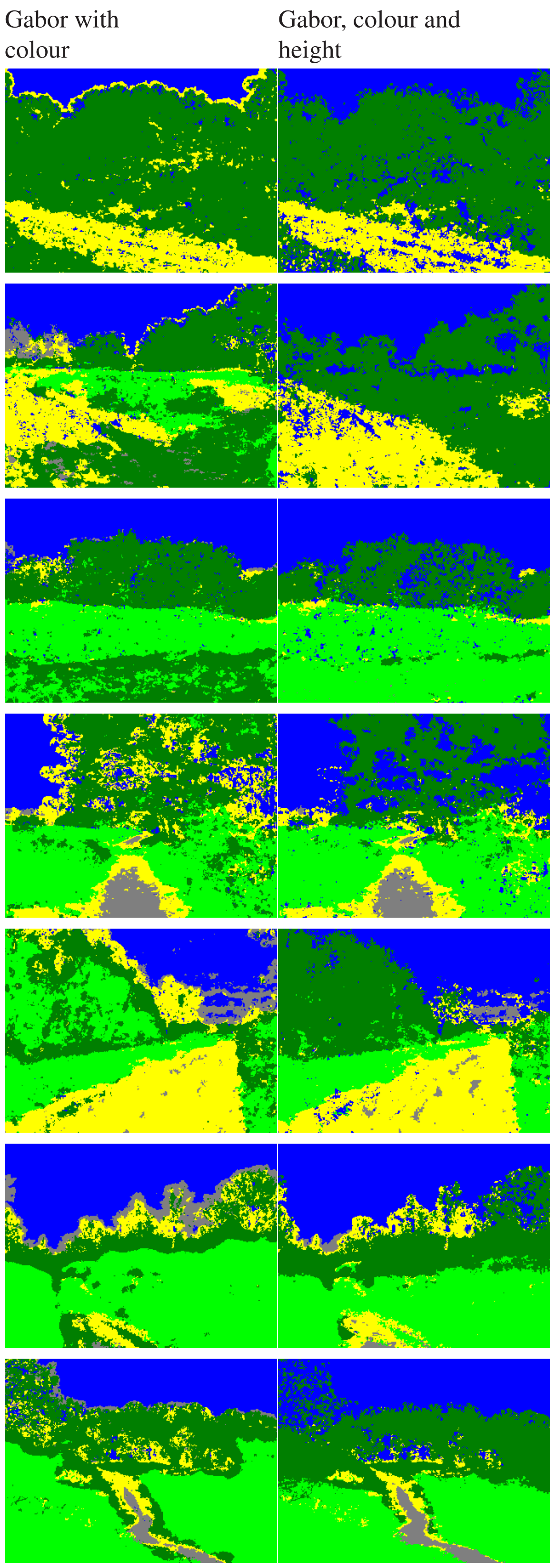
Terrain type classification: *blue*=sky, *dark green*=foliage, *green*=grass, *yellow*=sand and *gray*=gravel.

Figure 5.14 continued.



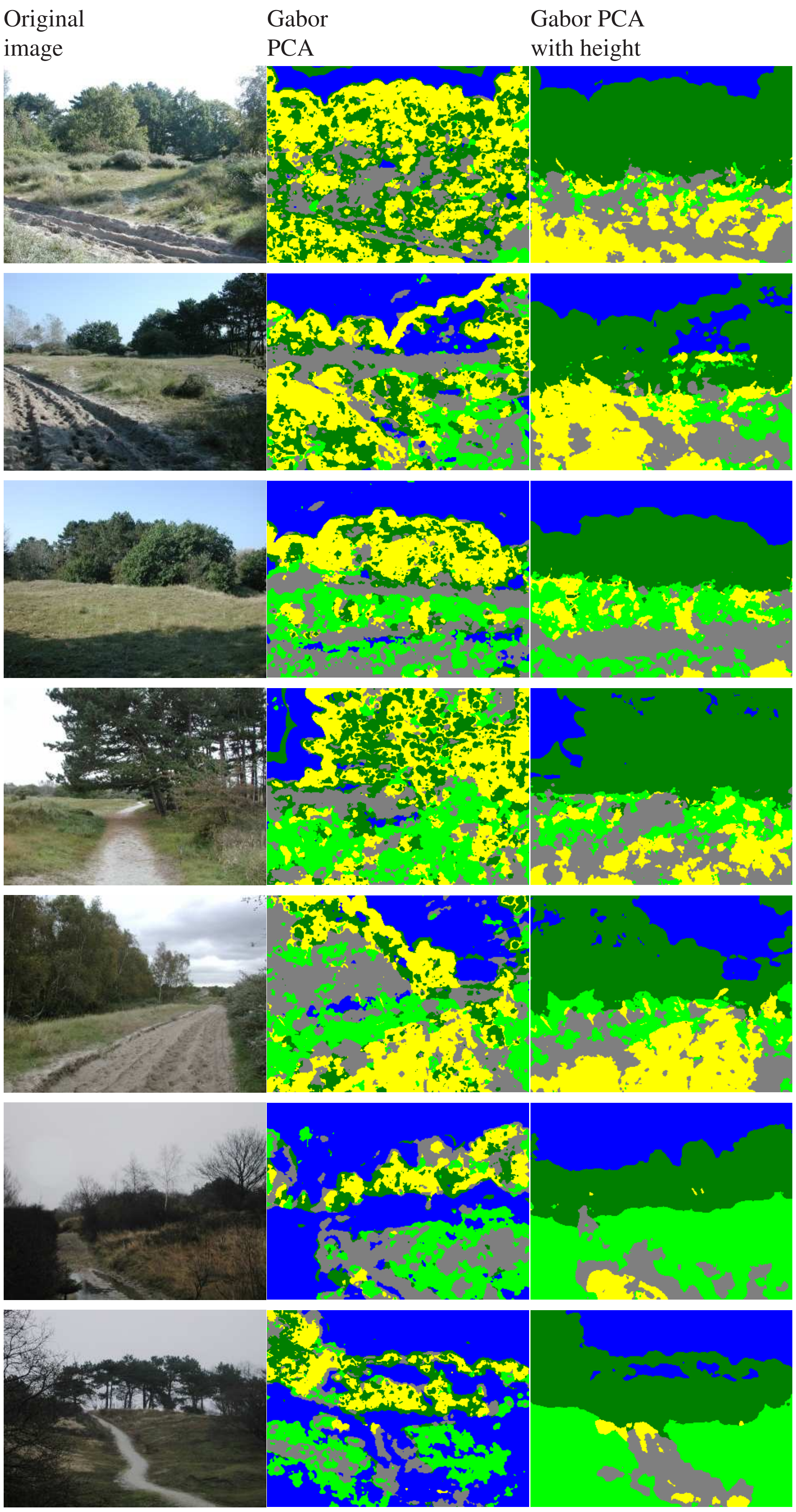
Terrain type classification: *blue*=sky, *dark green*=foliage, *green*=grass, *yellow*=sand and *gray*=gravel.

Figure 5.15: Results of the different Gabor texture classifiers on test set images with varying environment conditions.



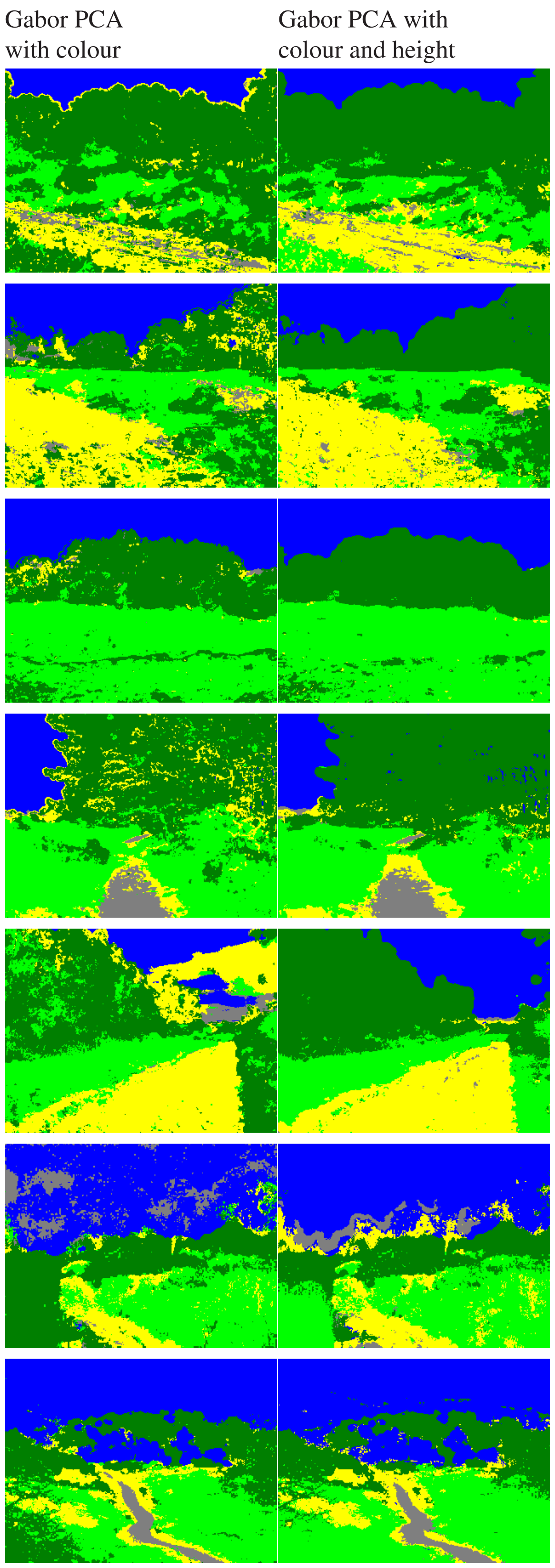
Terrain type classification: *blue*=sky, *dark green*=foliage, *green*=grass, *yellow*=sand and *gray*=gravel.

Figure 5.15 continued.



Terrain type classification: *blue*=sky, *dark green*=foliage, *green*=grass, *yellow*=sand and *gray*=gravel.

Figure 5.16: Results of the different reduced length texture classifiers on test set images with varying environment conditions.



Terrain type classification: *blue*=sky, *dark green*=foliage, *green*=grass, *yellow*=sand and *gray*=gravel.

Figure 5.16 continued.

Chapter 6

Summary, Conclusions and Future Work

In this thesis, we have examined the application of computer vision to problems related to off-road robot vehicle navigation, obstacle detection and terrain type recognition. A summary of the work including general conclusions and future research directions is given in this chapter.

6.1 Stereo vision and ego-motion estimation

In chapter 2, stereo vision based ego-motion is investigated as a method for accurate three-dimensional (3-D) vehicle navigation. The estimation is based on tracking corner-like image feature points that are present in the natural terrain that the vehicle is driving through. Due to the projective nature of imaging, the error distribution of stereo reconstructed 3-D points is anisotropic and inhomogeneous. Another problem is the presence of outliers that can be caused by mismatches or tracks on independently moving objects.

Several techniques for estimating motion from 3-D point sets are discussed. They range from simple least-squares techniques that assume isotropic and homogeneous noise to sophisticated techniques that use a more appropriate error model. However, none of the estimation techniques discussed consider the presence of outliers among the feature points. Therefore, the Least-Median-of-Squares method is included in the discussion as a robust estimation technique.

A technique was developed in order to evaluate the different approaches with real data captured by a vehicle that was driving through rough terrain. It uses artificial landmarks, placed near the vehicles test-tracks, to generate outlier-free image features. Statistical bootstrap is applied in order to obtain the accuracy of the landmark based motion estimates. The results show that our method is sufficiently accurate to serve as a “ground truth” in the evaluation of the estimation approaches.

The estimation methods are only tested with features found by edge detection in the terrain that do not belong to the landmarks. Not surprisingly, the least-square method performs much worse than the more elaborate methods. The results show that methods that try to correct errors introduced by the stereo reconstruction provide more accurate results. They are also more robust against modest quantities of outliers. If large quantities of outliers are present, such as an independently moving object, robust techniques are necessary. Overall, the results shows that stereo ego-motion estimation is a viable alternative to navigation sensors such as an Inertial Measurement Unit (IMU) or an odometer.

Future directions The work presented focusses on obtaining frame to frame ego-motion estimates. The results have shown that this is sufficiently accurate for navigation over short distances. Simultaneous localisation and mapping (SLAM) [84] approaches are commonly used for robot navigation over longer distances. Here, a map is constructed of the environment the robot is driving through, while its position in the map is also constantly updated. Most approaches to vehicle SLAM build a two dimensional map. However, because our method can provide accurate motion estimates for all six degrees of freedom, it could be extended to three dimensions. SLAM approaches also can incorporate absolute position readings in order to correct errors accumulated by the relative ego-motion estimates. These absolute readings can be retrieved from GPS or by closing-the-loop [62] in situations where it is known that the vehicle has driven over the same location twice.

6.2 Real-time dense stereo disparity estimation

Stereo vision is an attractive passive sensing technique for obtaining three-dimensional measurements. Recent hardware advances have given rise to a new class of real-time dense stereo disparity estimation algorithms. Chapter 3 examines their suitability for intelligent vehicle applications. In order to gain a better understanding of the performance and computational cost trade-off, we created a framework of real-time implementations. This consists of different methodical components based on Single Instruction Multiple Data (SIMD) techniques.

Furthermore, the resulting algorithmic variations are compared with other publicly available algorithms. We argue that existing, publicly available stereo data sets are not very suitable for the intelligent vehicle domain. Therefore, our evaluation of stereo algorithms is based on novel realistically looking simulated data, as well as real data from complex urban traffic scenes.

The results from this study reveal that there is a considerable influence of scene conditions on the performance of all tested algorithms. Approaches that aim for (global) search optimisation are more affected by this than other approaches. The best overall performance is achieved by the proposed multiple window algorithm which uses local matching and a left-right check for robust error rejection.

Timing results show that the simplest of the proposed SIMD variants are more than twice as fast as the most complex one. Nevertheless, the latter still achieve real-time processing speeds while their average accuracy is at least equal to that of publicly available non-SIMD algorithms.

Future directions The results suggest that future improvement of dense stereo algorithms can be achieved by applying search optimisation techniques only to the parts of the stereo images that can benefit from them. Preprocessing steps to detect textureless and edgeless regions could be exploited to select pixels are suitable.

6.3 Obstacle detection for unstructured environments

In chapter 4, a method is proposed that finds and clusters stereo reconstructed terrain points more efficiently into obstacles, by considering their position uncertainty. Basically, the obstacle detection method tries to find points that belong to the same obstacle surfaces. Two geometrical constraints; elevation difference and relative angle, have to be checked in order to decide if a point pair belongs to the same obstacle surface. This approach is problematic for real-time applications due to the number of points in dense image data. Furthermore, the points reconstruction accuracy becomes less reliable at larger distances. In our approach, the range of the reconstruction data is divided by regular intervals. The geometrical constraints are enforced by a set of pixel threshold values that are calculated for each interval. Because these values are all computed during an initialisation step, only simple pixel threshold operations remain to be performed during the real-time obstacle detection. An advantage of this novel approach is that the distance uncertainties can be incorporated into the thresholds.

Detected obstacle points are clustered into objects on the basis of their pixel connectivity. Objects with insufficient pixels, elevation and slope are rejected. Remaining non-obstacle pixels are regarded as ground surface points. They are used to update the orientation of the stereo camera relative to the ground surface. This prevents orientation errors during stereo reconstruction and the subsequent obstacle detection steps.

Our results show the drawbacks of ignoring the uncertainties in the stereo distance estimates for obstacle detection. It leads to over-segmentation and increases the number of falsely detected obstacles. Because our method incorporates these uncertainties, it can detect more of the obstacle surface pixels at larger distances. This leads to significantly fewer false obstacle detections.

Future directions The current method only uses single stereo frames to detect obstacles. Addition of obstacle tracking abilities would have a number of advantages. The position of stationary obstacles that are already detected, could be predicted in new frames with camera ego-motion estimates. This would improve the accuracy of measured obstacle distance and size and suppress detection misses or false detections. Another challenge would be to also track independently moving objects. This is necessary to avoid collisions in dynamic environments where, for example, people or other cars move freely around. Our ego-motion method could also be applied for this goal because it can identify outlier motion patterns.

6.4 Terrain classification by environment resemblance

A drawback of the approach presented in the previous chapter is that it cannot distinguish between solid obstacles such as rocks and soft obstacles such as tall patches of grass.

In this chapter we present a method to classify typical terrain coverings such as sand, grass or foliage. It is primarily based on colour recognition. However, it can also be used with other types of image features such as texture. Using colour recognition outdoors is difficult, because the observed colour of a material is heavily influenced by environment conditions such as the scene composition and illumination.

A new approach to terrain type classification is presented. It is based on the assumption that images with large similarities in environment related properties such as illumination, materials and geometry also have similar distributions in the feature space. The idea is to first partition the training image set into smaller sets of similar images, which are called environment states. Because the terrain type features are modelled separately for each environment state, there is less variation due to different environment conditions.

Our classification method is based on a Maximum Likelihood method with Gaussian Mixture Models (GMMs), to model to the environment states and their terrain types. It can automatically find a desired number of environment states and train GMMs for their terrain types. Terrain types in a new image are classified with the GMMs of the environment state that is the most similar to it.

Quantitative tests have been conducted with different image features types. The results with colour only features show that our approach is able to classify terrain types in images with large differences in illumination. Experiments with only texture based features did not provide satisfactory results, due to sparseness in the extracted feature values. Combinations of texture and colour turned out to be inferior to colour only classification. For combinations of texture with other features, a slight increase in performance was obtained by reducing the dimensionality of the texture data. The best performance is achieved when our method is used with colour, reduced texture and height features.

Future directions A large set of training images is required for the approach presented. The set should contain sufficient images that represent the various environment conditions. Creating such a set involves a lot of data collection and hand labelling effort. The work required could be reduced if images are used of a camera that records images of the same scene during different times of the day and weather conditions.

In our research we did not address the problem of detecting bodies of water such as lakes or rivers. These are hazardous to the robot vehicle and should also be detected. One advantage of our terrain classification approach is that for the sky a separate class is used. Pools of water often reflect the sky colours and can therefore be identified as regions of sky on the ground. In the work by Matthies et al. [56], sky colour reflection and other methods for detecting water hazards are investigated.

Appendix A

Mathematical Notation and Operator Definition

This appendix explains the notation used for vectors, matrices and operators in chapter 2. It also contains the definitions for the inner- and cross product that can be found in the book by Kanatani [40]. The matrix inner and exterior products definitions are taken from the paper by Otha and Kanatani [65].

A.1 Vector and matrix notation

A column vector $\boldsymbol{a}$ consists of n values a_i:

$$\boldsymbol{a} = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} . \tag{A.1}$$

Vector transpose can be used to change from column vector, to row vector and visa versa:

$$\boldsymbol{a}^\top = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}^\top = \begin{pmatrix} a_1 & a_2 & \cdots & a_n \end{pmatrix} . \tag{A.2}$$

A matrix $\boldsymbol{C}$ is an array of m column vectors, each with n entries:

$$\boldsymbol{C} = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1m} \\ c_{21} & c_{22} & & c_{2m} \\ \vdots & & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nm} \end{pmatrix} . \tag{A.3}$$

Some operations require the selection of specific entries from the vectors or the matrices. One such operation is t_3, as defined in [65], that creates the following vector out of the 3 by 3 matrix $\boldsymbol{C}$:

$$t_3[\boldsymbol{C}] = \begin{pmatrix} c_{32} \\ c_{13} \\ c_{21} \end{pmatrix} \quad \text{with} \quad \boldsymbol{C} = \begin{pmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{pmatrix} . \tag{A.4}$$

A.2 Inner product

Given two vectors of equal length, their inner product $(\cdot)$ is equal to the product of their norms and the cosine of the angle θ between them:

$$\boldsymbol{a} \cdot \boldsymbol{b} = \|\boldsymbol{a}\| \, \|\boldsymbol{b}\| \cos\theta \ . \tag{A.5}$$

Evidently, if $\boldsymbol{a}$ is perpendicular to $\boldsymbol{b}$, then $\boldsymbol{a} \cdot \boldsymbol{b} = 0$.
The inner product of two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ is equal to:

$$\boldsymbol{a} \cdot \boldsymbol{b} = \boldsymbol{a}^\top \boldsymbol{b} = \sum_{i=1}^{n} a_i b_i \ . \tag{A.6}$$

A.3 Cross product

The result of the cross product $(\times)$ with two 3-D vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ is equal to:

$$\boldsymbol{a} \times \boldsymbol{b} = \left(\sum_{j=1}^{3} \sum_{k=1}^{3} \varepsilon_{ijk} a_i b_j \right) = \begin{pmatrix} a_2 a_3 - a_3 b_2 \\ a_3 a_1 - a_1 b_3 \\ a_1 a_2 - a_2 b_1 \end{pmatrix} . \tag{A.7}$$

Where ε_{ijk} is the Levi-Civita symbol for 3-D:

$$\varepsilon_{ijk} = \begin{cases} 1 \ \text{ if } (i,j,k) \text{ is } (1,2,3),\ (2,3,1) \text{ or } (3,1,2) \ . \\ -1 \ \text{ if } (i,j,k) \text{ is } (3,2,1),\ (1,3,2) \text{ or } (2,1,3) \ . \\ 0 \ \text{ in case of } i = j,\ j = k \text{ or } k = i \ . \end{cases} \tag{A.8}$$

If the angle between the two vectors is 0 or π, the norm of their cross product will be equal to zero. Otherwise, the vectors form a 2-D plane in the 3-D space. Their cross product gives a vector that is perpendicular to this plane.
The cross product is also defined for a 3-D vector $\boldsymbol{a}$ and 3 by 3 matrix $\boldsymbol{B}$:

$$\boldsymbol{a} \times \boldsymbol{B} = \left(\boldsymbol{a} \times \begin{pmatrix} b_{11} \\ b_{21} \\ b_{31} \end{pmatrix} \boldsymbol{a} \times \begin{pmatrix} b_{12} \\ b_{22} \\ b_{32} \end{pmatrix} \boldsymbol{a} \times \begin{pmatrix} b_{13} \\ b_{23} \\ b_{33} \end{pmatrix} \right) . \tag{A.9}$$

The cross product of a 3-D vector $\boldsymbol{a}$ and the 3 by 3 identify matrix $\boldsymbol{I}$ is therefore equal to:

$$\boldsymbol{a} \times \boldsymbol{I} = \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix} \ . \tag{A.10}$$

The cross product of a 3 by 3 matrix $\boldsymbol{B}$ and the 3-D vector $\boldsymbol{a}$ is defined as:

$$\boldsymbol{B} \times \boldsymbol{a} = \boldsymbol{B}(\boldsymbol{a} \times \boldsymbol{I})^\top \ . \tag{A.11}$$

A.4 Matrix inner and exterior product

In Otha and Kanatani [65], the inner product of two 3 by 3 matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ is defined as:

$$\boldsymbol{A} \vee \boldsymbol{B} = \sum_{i=1}^{3} \sum_{j=1}^{3} \boldsymbol{A}_{ij} \boldsymbol{B}_{ij} \ . \tag{A.12}$$

The exterior product $\boldsymbol{A} \wedge \boldsymbol{B}$ is a matrix whose i,j element is equal to:

$$\sum_{k}^{3} \sum_{l}^{3} \sum_{m}^{3} \sum_{n}^{3} \varepsilon_{ikl} \varepsilon_{jmn} \boldsymbol{A}_{km} \boldsymbol{B}_{ln} \ . \tag{A.13}$$

Appendix B

Quaternions

Complex numbers can be interpreted as points in a two-dimensional space. Quaternions are extensions of complex numbers which can be interpreted as points in a four-dimensional space. In addition to **i**, there are also the imaginary numbers **j** and **k**. A quaternion q is defined as [68]:

$$q = w + x\mathbf{i} + y\mathbf{j} + z\mathbf{k} \ . \tag{B.1}$$

Here, u, v, w and z are real valued scalars. Akin to imaginary numbers, the square product of the **i**, **j** and **k** is equal to -1:

$$\mathbf{i}^2 = -1, \quad \mathbf{j}^2 = -1, \quad \mathbf{k}^2 = -1 \ . \tag{B.2}$$

Multiplication of the imaginary numbers with each other is governed by the following rules:

$$\begin{aligned} \mathbf{i} * \mathbf{j} = -\mathbf{j} * \mathbf{i} = \mathbf{k} \ , \\ \mathbf{j} * \mathbf{k} = -\mathbf{k} * \mathbf{j} = \mathbf{i} \ , \\ \mathbf{k} * \mathbf{i} = -\mathbf{i} * \mathbf{k} = \mathbf{j} \ . \end{aligned} \tag{B.3}$$

Using these rules, the basic operations of quaternion addition and multiplication can be defined as:

$$q_1 + q_2 = (w_1 + w_2) + (x_1 + x_2)\mathbf{i} + (y_1 + y_2)\mathbf{j} + (z_1 + z_2)\mathbf{k} \ , \tag{B.4}$$

$$\begin{array}{rlllllllll} q_1 * q_2 = & (w_1w_2 & - & x_1x_2 & - & y_1y_2 & - & z_1z_2) & & + \\ & (w_1x_2 & + & x_1w_2 & + & y_1z_2 & - & z_1y_2) & \mathbf{i} & + \\ & (w_1y_2 & - & x_1z_2 & + & y_1w_2 & + & z_1x_2) & \mathbf{j} & + \\ & (w_1z_2 & + & x_1y_2 & - & y_1x_2 & + & z_1w_2) & \mathbf{k} & . \end{array} \tag{B.5}$$

Due to the imaginary numbers, quaternion multiplication is associative but not commutative:

$$\begin{aligned} (q_1 * q_2) * q_3 = q_1 * (q_2 * q_3) \ , \\ q_1 * q_2 \neq q_2 * q_1 \ . \end{aligned} \tag{B.6}$$

Quaternions also have a conjugate $\bar{q}$ and a magnitude $||q||$ that are defined as:

$$\bar{q} = w - x\mathbf{i} - y\mathbf{j} - z\mathbf{k} \ , \tag{B.7}$$

$$||q|| = \sqrt{q * \bar{q}} = \sqrt{w^2 + x^2 + y^2 + z^2} \ . \tag{B.8}$$

Quaternions whose magnitude equals 1, are called unit quaternions. The inverse of an unit quaternion is equal to its conjugate; $q^{-1} = \bar{q}$.

There are several representations possible of quaternions in linear algebra. They can be defined as a scalar vector pair, where the scalar is w and the vector is $\boldsymbol{v} = (x\, y\, z)^\top$. The four scalar values or the scalar vector representation can also be combined into one four-dimensional quaternion vector $\boldsymbol{q}$:

$$\boldsymbol{q} = (w\, \boldsymbol{v})^\top = (w\ [x\, y\, z])^\top \ . \tag{B.9}$$

It is clear that the norm of vector $\boldsymbol{q}$ is equal to the quaternion magnitude. The quaternion vector conjugate or the inverse of an unit quaternion vector can be found by changing the sign of $\boldsymbol{v}$. Addition of quaternion vectors is simply their sum and their multiplication is defined as:

$$\boldsymbol{q}_1 * \boldsymbol{q}_2 = (w_1 w_2 - \boldsymbol{v}_1 \cdot \boldsymbol{v}_2 \quad [w_1 \boldsymbol{v}_2 + w_2 \boldsymbol{v}_1 + \boldsymbol{v}_1 \times \boldsymbol{v}_2])^\top \ . \tag{B.10}$$

An important application of unit quaternions is three-dimensional rotation [20]. Suppose that $\boldsymbol{u}$ is a three-dimensional unit vector that indicates a rotation axis. An arbitrary vector $\boldsymbol{p}$ can be rotated around this axis with an angle θ to the vector ***p'***. The quaternion involved in this rotation is given by:

$$\boldsymbol{q} = \left(\cos(\tfrac{1}{2}\theta) \ \ \left[\sin(\tfrac{1}{2}\theta)\boldsymbol{u}\right]\right)^\top \ . \tag{B.11}$$

Note that the half angle is used in the quaternion. If $\tilde{\boldsymbol{p}} = (0\ [\boldsymbol{p}])^\top$ is the four-dimensional representation of $\boldsymbol{p}$, then the actual rotation operation is defined as:

$$\tilde{\boldsymbol{p}}' = \boldsymbol{q} * \tilde{\boldsymbol{p}} * \boldsymbol{q}^{-1} \ . \tag{B.12}$$

The inverted rotation can simply be obtained by:

$$\tilde{\boldsymbol{p}} = \boldsymbol{q}^{-1} * \tilde{\boldsymbol{p}}' * \boldsymbol{q} \ . \tag{B.13}$$

Two consecutive rotations, the first defined by $\boldsymbol{q}_1$ and the second by $\boldsymbol{q}_2$, can be concatenated to form a new unit quaternion $\boldsymbol{q}_3$:

$$\boldsymbol{q}_2 * (\boldsymbol{q}_1 * \tilde{\boldsymbol{p}} * \boldsymbol{q}_1^{-1}) * \boldsymbol{q}_2^{-1} = (\boldsymbol{q}_2 * \boldsymbol{q}_1) * \tilde{\boldsymbol{p}} * (\boldsymbol{q}_2 * \boldsymbol{q}_1)^{-1} = \boldsymbol{q}_3 * \tilde{\boldsymbol{p}} * \boldsymbol{q}_3{}^{-1} \ . \tag{B.14}$$

Bibliography

[1] "Matlab product description (The MathWorks)." available online: http://www.mathworks.com/products/matlab/description1.html

[2] "Middlebury college stereo vision research page." available online: http://www.middlebury.edu/stereo

[3] "The open source computer vision library." available online: http://www.intel.com/technology/computing/opencv

[4] "Stereo image data for algorithm evaluation." available online: http://stereodatasets.wvandermark.com

[5] P. Bellutta, R. Manduchi, L. Matthies, K. Owens, and A. Rankin, "Terrain perception for DEMO III," in *Proceedings of the IEEE Intelligent Vehicles Conference*, pp. 326–331, 3-5 October 2000.

[6] S. Birchfield and C. Tomasi, "Depth discontinuities by pixel-to-pixel stereo," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1073–1080, Bombay, India, January 1998.

[7] M. Björkman and J.-O. Eklundh, "A real-time system for epipolar geometry and ego-motion estimation," in *Proceedings of IEEE Conference Computer Vision and Pattern Recognition*, vol. 2, pp. 506–513, Hilton Head, SC, 2000.

[8] A. F. Bobick and S. S. Intille, "Large occlusion stereo," *International Journal of Computer Vision*, vol. 33, no. 3, pp. 181–200, September 1999.

[9] A. Broggi, C. Caraffi, R. Fedriga, and P. Grisleri, "Obstacle detection with stereo vision for off-road vehicle navigation," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 3, pp. 65–73, San Diego, 20-26 June 2005.

[10] M. Z. Brown and D. Burschka, "Advances in computational stereo," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 8, August 2003.

[11] S. D. Buluswar, "Color-based models for outdoor machine vision," Ph.D. dissertation, The University of Massachusetts Amherst, February 2002.

[12] S. D. Buluswar and B. A. Draper, "Color machine vision for autonomous vehicles," *International Journal for Engineering Applications of Artificial Intelligence*, vol. 11, no. 2, pp. 245–256, April 1998.

[13] R. Bunschoten and B. Kröse, "Range estimation from a pair of omnidirectional images," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Seoul, Korea, 21-26 May 2000.

[14] A. Castano and L. Matthies, "Foliage discrimination using a rotating ladar," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1–6, Taipei, Taiwan, 14-19 September 2003.

[15] R. Castano, R. Manduchi, and J. Fox, "Classification experiments on real-world textures," in *Workshop on Empirical Evaluation in Computer Vision*, Kauai, HI, December 2001.

[16] C. I. de L' Eclairge (CIE), "Official recommendations on uniform color spaces, color-difference equations, and metric color terms," Pub. No. 15, Supp. Num 2 (E-1.3.1), 1979.

[17] D. Demirdjian and R. Horaud, "Motion-egomotion discrimination and motion segmentation from image-pair streams," *Computer Vision and Image Understanding*, vol. 78, no. 1, pp. 53–68, April 2000.

[18] H. Y. Deng, Q. Yang, X. Lin, and X. Tang, "A symmetric patch-based correspondence model for occlusion handling," in *Tenth IEEE International Conference on Computer Vision*, vol. 2, pp. 1316–1322, Beijing, China, 15-21 October 2005.

[19] E. Dickmanns and B. Mysliwetz, "Recursive 3d road and relative ego-state recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence, Special Issue on Interpretation of 3D Scenes*, pp. 199–213, February 1992.

[20] J. Diebel, "Representing attitude: Euler angles, unit quaternions, and rotation vectors," Stanford University, Stanford, California 94301-9010, Tech. Rep., 20 October 2006. available online:
http://ai.stanford.edu/∼diebel/attitude/attitude.pdf

[21] B. Efron and R. J. Tisbshirani, *An Introduction to the Bootstrap*, ser. Monograps on Statistics and Applied Probability 57. Chapman & Hall/CRC, 1993.

[22] O. Faugeras and Q.-T. Luong, *The Geometry of Mutiple Images: the laws that govern the formation of multiple images of a scene and some of their applications*. MIT Press, 2001.

[23] M. Fischler and R. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Commun. ACM*, vol. 24, no. 6, pp. 381–395, June 1981.

[24] U. Franke, D. M. Gavrila, S. Görzig, F. Lindner, F. Paetzold, and C. Wöhler, “Autonomous driving goes downtown,” *IEEE Intelligent Systems*, vol. 13, no. 6, pp. 40–48, 1998.

[25] U. Franke and S. Heinrich, “Fast obstacle detection for urban traffic situations,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 3, no. 3, September 2002.

[26] A. Fusiello, V. Roberto, and E. Trucco, “Symmetric stereo with multiple windowing,” *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 14, no. 8, pp. 1053–1066, December 2000.

[27] A. Fusiello, E. Trucco, and A. Verri, “Rectification with unconstrained stereo geometry,” in *Proceedings of the Eighth British Machine Vision Conference*, 1997.

[28] R. Gerber, *The Software Optimization Cookbook.* Intel Cooperation, Hillsboro, OR: Intel Press, 2002.

[29] T. Gevers and A. Smeulders, “Color based object recognition,” *Pattern Recognition*, vol. 32, pp. 453–464, March 1999.

[30] H. Greenspan, S. Gordon, and J. Goldberger, “Probabilistic models for generating, modelling and matching image categories,” in *Proceedings of the 16th International Conference on Pattern Recognition*, vol. 3, Quebec, Canada, 11-15 August 2002.

[31] C. Harris and M. Stephens, “A combined corner and edge detector,” in *Proceedings of the fourth Alvey Vision Conference*, pp. 147–151, 1988.

[32] R. Hartley, “Theory and practice of projective rectification,” *International Journal of Computer Vision*, vol. 35, no. 2, pp. 115–127, November 1999.

[33] J. Heikkilä and O. Sivén, “A four-step camera calibration procedure with implicit image correction,” in *Proceedings of the IEEE Computer Vision and Pattern Recognition conference*, pp. 1106–1112, 1997.

[34] H. C. L. Higgins, “A computer algorithm for reconstructing a scene from two projections,” *Nature*, pp. 133–135, September 1981.

[35] H. Hirschmüller, P. R. Innocent, and J. M. Garibaldi, “Real-time correlation-based stereo vision with reduced border errors,” *International Journal of Computer Vision*, vol. 47, no. 1-3, pp. 229–246, 2002.

[36] M. A. Hoang, J. M. Geusebroek, and A. W. M. Smeulders, “Color texture measurement and segmentation,” *Signal Processing*, vol. 85, no. 2, pp. 265–275, February 2005.

[37] L. Hong and G. Chen, “Segment-based stereo matching using graph cuts,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. I, pp. 74–81, Washington DC, 27 June - 2 July 2004.

[38] B. K. P. Horn, “Closed-form solution of absolute orientation using unit quaternions,” *Journal of the Optical Society of America*, vol. 4, no. 2, pp. 1127–1134, July 1988.

[39] T. Kanade and M. Okutomi, “A stereo matching algorithm with an adative window: theory and experiment,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 9, pp. 920–932, September 1994.

[40] K. Kanatani, *Statistical Optimization for Geometric Computation: Theory and Practice*, ser. Machine Intelligence and Pattern Recognition 18, L. Kanal and A. Rosenfeld, Eds. Elsevier Science/Dover Publications, 1996.

[41] Y. Kanazawa and K. Kanatani, “Do we really have to consider covariance matrices for image feature points?” *Electronics and Communications in Japan, Part III: Fundamental Electronic Science*, vol. 86, no. 1, pp. 1–10, 2003.

[42] J. C. Kim, K. M. Lee, B. T. Choi, and S. U. Lee, “A dense stereo matching using two-pass dynamic programming with generalized ground control points,” in *Proceedings IEEE International Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 1075–1082, San Diego, CA, 20-25 June 2005.

[43] G. J. Klinker, S. A. Shafer, and T. Kanade, “The measurement of highlights in color images,” *International Journal of Computer Vision*, vol. 2, no. 1, pp. 7–32, June 1988.

[44] C. Kolb, D. Mitchell, and P. Hanrahan, “A realistic camera model for computer graphics,” in *Computer Graphics (Proceedings of SIGGRAPH ’95)*, pp. 317–324. ACM SIGGRAPH, 1995.

[45] V. Kolmogorov and R. Zabih, “Computing visual correspondence with occlusions using graph cuts,” in *Proceedings of the IEEE International Conference on Computer Vision*, Vancouver, Canada, 9-12 July 2001.

[46] G. Kraft and P. P. Jonker, “Real-time stereo with dense output by a SIMD-computed dynamic programming algorithm,” in *International Conference on Parallel and Distributed Processing Techniques and Applications*, vol. III, pp. 1031–1036, Las Vegas, Nevada, 24-27 June 2002.

[47] R. D. Labayrade and J. P. Tarel, “Real time obstacle detection on non flat road geometry through ‘v-disparity’ representation,” in *Proceedings of the IEEE Intelligent Vehicle Symposium*, vol. 2, pp. 646–651, Versailles, France, 17-21 June 2002.

[48] K. Labibes, Z. Papp, A. C. H. Thean, P. P. M. Lemmen, M. Dorrepaal, and F. J. W. Leneman, “An integrated design and validation environment for intelligent vehicle safety systems (IVSS),” in *10th World Congress and exhibition on ITS*, Madrid, Spain, 16-20 November 2003, proceedings on CD-ROM.

[49] J. Lasenby and A. Stevenson, “Using geometric algebra for optical motion capture,” in *Geometric Algebra with Applications in Science and Engineering*, E. Corrochano and G. Sobczyk, Eds. Birkhäuser Boston, 2001, pp. 147–169.

[50] J. N. Maki, J. F. Bell, K. E. Herkenhoffand, S. W. Squyres, A. Kiely, M. Klimesh, M. Schwochert, T. Litwin, A. J. R. Willson, M. Maimone, E. Baumgartner, A. Collins, M. Wadsworth, S. T. Elliot, A. Dingizian, D. Brown, E. C. Hagerott, R. D. L. Scherr, D. Alexander, and J. Lorre, “The mars exploration rover engineering cameras,” *Journal of Geophysical Research – Planets, Special Issue*, vol. 108, no. E12, pp. 12–1 to 12–24, 11 December 2003.

[51] A. Mallet, S. Lacroix, and L. Gallo, “Position estimation in outdoor environments using pixel tracking and stereovision,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 4, pp. 3519–3524, 24-28 April 2000.

[52] R. Manduchi, A. Castano, A. Talukder, and L. Matthies, “Detection and terrain classification for autonomous off-road navigation,” *Autonomous Robots*, vol. 18, pp. 81–102, 2005.

[53] B. S. Manjunath and W. Y. Ma, “Texture features for browsing and retrieval of image data,” *IEEE Transactions on Pattern Analysis and Machine Intelligence - Special issue on Digital Libraries*, vol. 18, no. 8, pp. 837–842, Augustus 1996.

[54] B. C. Matei, “Heteroscedastic errors-in-variables models in computer vision,” Ph.D. dissertation, The State University of New Jersey, New Brunswick, New Jersey, May 2001.

[55] L. Matthies, “Stereo vision for planetary rovers: Stochastic modelling to near real-time implementation,” *International Journal of Computer Vision*, vol. 8, no. 1, pp. 71–91, 1992.

[56] L. Matthies, P. Bellutta, and M. McHenry, “Detecting water hazards for autonomous off-road navigation,” in *Unmanned ground vehicle technology, Procedings of SPIE*, vol. 5083, pp. 231–242, Orlando FL, 2003.

[57] L. Matthies and P. Grandjean, "Stochastic performance modeling and evaluation of obstacle detectability with imaging range sensors," *Transactions on Robotics and Automation, Special Issue on Perception-based Real World Navigation*, vol. 10, no. 6, December 1994.

[58] L. Matthies, T. Litwin, K. Owens, A. Rankin, K. Murphy, D. Coorobs, J. Gilsinn, T. Hong, S. Legowik, M. Nashman, and B. Yoshimi, "Performance evaluation of UGV obstacle detection with CCD/FLIR stereo vision and LADAR," in *IEEE Workshop on Perception for Mobile Agents*, Santa Barbara, CA, June 1998.

[59] N. Molton and M. Brady, "Practical structure and motion from stereo when motion is unconstrained," *International Journal of Computer Vision*, vol. 39, no. 1, pp. 5–23, 2000.

[60] K. Mühlmann, D. Maier, J. Hesser, and R. Männer, "Dense disparity maps from color stereo images, an efficient implementation," *International Journal of Computer Vision*, vol. 47, no. 1-3, pp. 79–88, April - June 2002.

[61] S. Nedevschi, R. Danescu, D. Frentiu, T. Marita, F. Oniga, C. Pocol, R. Schmidt, and T. Graf, "High accuracy stereo vision system for far distance obstacle detection," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, pp. 292–297, Parma, Italy, 14-17 June 2004.

[62] P. Newman and K. L. Ho, "Slam- loop closing with visually salient features," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 644–651, Barcelona, Spain, 18-22 April 2005.

[63] C. L. Novak, S. A. Shafer, and R. G. Willson, "Obtaining accurate color images for machine vision research," in *Proceedings of SPIE*, vol. 1250, pp. 54–68, 1990.

[64] J. R. J. Nunnink, J. J. Verbeek, and N. Vlassis, "Accelerated greedy mixture learning," in *Proceedings of the Annual Machine Learning Conference of Belgium and The Netherlands*, pp. 80–86, Brussels, Belgium, January 2004.

[65] N. Ohta and K. Kanatani, "Optimal estimation of threedimensional rotation and reliability evaluation," in *Computer Vision - ECCV '98: 5th European Conference on Computer Vision*, B. Neumann and H. Burkhardt, Eds., vol. I (Lecture Notes in Computer Science), pp. 175–187. Freiburg, Germany: Springer, 2-6 June 1998.

[66] C. F. Olson, L. H. Matthies, M. Schoppers, and M. W. Maimone, "Rover navigation using stereo ego-motion," *Robotics and Autonomous Systems*, vol. 43, no. 4, pp. 215–229, 30 June 2003.

[67] Z. Papp, K. Labibes, A. H. C. Thean, and M. G. van Elk, "Multi-agent based HIL simulator with high fidelity virtual sensors," in *Proceedings of IEEE Intelligent Vehicle Symposium*, pp. 213–218, Columbus, OH, 9-11 June 2003.

[68] E. Pervin and J. Webb, "Quaternions for computer vision and robotics," *Computer Vision and Pattern Recognition*, pp. 382–383, 1983, also avalible as CMU-CS-82-150 tech. report.

[69] B. Porr, A. Cozzo, and F. Wörgötter, "How to "hear" visual disparities: real-time stereoscopic spatial depth analysis using temporal resonance," *Biological Cybernetics*, vol. 78, no. 5, pp. 329–336, May 1998.

[70] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed. Cambridge University Press, 1992.

[71] B. D. Ripley, *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

[72] Robosoft, "The powerful outdoor mobile robot for civil and military applications," 2006. available online:
http://www.robosoft.fr

[73] P. Saeedi, P. Lawrence, and D. Lowe, "3D motion tracking of a mobile robot in a natural environment," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1682–1687, 24-28 April 2000.

[74] B. H. Schäfer, M. Proetzsch, and K. Berns, "Obstacle avoidance in rough outdoor terrain," in *International Symposium on Motor Control and Robotics*, October 2005.

[75] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International Journal of Computer Vision*, vol. 47, pp. 7–42, April 2002.

[76] D. Scharstein and R. Szeliski, "High-accuracy stereo depth maps using structured light," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 195–202, Madison, WI, 18-20 June 2003.

[77] J. Schavemaker, E. den Breejen, K. Benoist, K. Schutte, P. Tettelaar, M. de Bijl, P. Fritz, L. Cohen, W. van der Mark, and R. Chignell, "Lotus field demonstration of integrated multisensor mine detection system in bosnia," in *Detection and Remediation Technologies for Mines and Minelike Targets VIII, Proceedings of SPIE*, R. Harmon, J. Holloway, and J. Broach, Eds., vol. 5089, pp. 1324–1335, September 2003.

[78] I. Schwartz, “PRIMUS: autonomous driving robot for military applications,” in *Unmanned Ground Vehicle Technology II, Proceedings of SPIE*, vol. 4024, July 2000.

[79] C. M. Shoemaker and J. A. Bornstein, “Overview and update of the Demo III experimental unmanned vehicle program,” in *Unmanned Ground Vehicle Technology II, Proceedings of SPIE*, vol. 4024, July 2000.

[80] L. D. Stefano, M. Marchionni, S. Mattoccia, and G. Neri, “A fast area-based stereo matching algorithm,” in *15th IAPR/CIPRS International Conference on Vision Interface*, Calgary, Canada, 27-29 May 2002.

[81] L. D. Stefano and S. Mattoccia, “Fast stereo matching for the videt system using a general purpose processor with multimedia extensions,” in *Fifth IEEE International Workshop on Computer Architectures for Machine Perception*, pp. 356–362, Padova, Italy, 11-13 September 2000.

[82] H. W. Stone, “Mars pathfinder microrover: A low-cost, low-power spacecraft,” in *Proceedings of the 1996 AIAA Forum on Advanced Developments in Space Robotics*, Madison, WI, August 1996.

[83] S. Thrun, “Robotic mapping: A survey,” in *Exploring Artificial Intelligence in the New Millenium*, G. Lakemeyer and B. Nebel, Eds. Morgan Kaufmann, July 2002, also available as CMU Tech report CMU-CS-02-111.

[84] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. The MIT Press, September 2005.

[85] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, “Stanley, the robot that won the DARPA grand challenge,” *Journal of Field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[86] P. H. S. Torr and D. W. Murray, “The development and comparison of robust methods for estimating the fundamental matrix,” *International Journal of Computer Vision*, vol. 24, no. 3, pp. 271–300, 1997.

[87] Y. Tsin, R. Collins, V. Ramesh, and T. Kanade, “Bayesian color constancy for outdoor object recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 1132–1139, December 2001.

[88] J. van de Weijer, “Color features and local structure in images,” Ph.D. dissertation, Informatics Institute, University of Amsterdam, March 2005.

[89] T. W. van den Berg, W. Huiskamp, and J. C. van den Heuvel, “Vehicle control using simulation and virtual reality techniques,” in *Proceedings of the IEEE Intelligent Transport Systems Conference*, pp. 895–900, Oakland (CA), 25-29 August 2001.

[90] J. C. van den Heuvel, J. Kleijweg, W. van der Mark, M. Lievers, and L. Kester, “Obstacle detection for people movers using vision and radar,” in *10th World Conference on Intelligent Transport Systems and Services*, Madrid, Spain, 16-20 November 2003, proceedings on CD-ROM.

[91] W. van der Mark, D. Fontijne, L. Dorst, and F. C. A. Groen, “Vehicle ego-motion estimation with geometric algebra,” in *Proceedings of the IEEE Intelligent Vehicle Symposium*, vol. 1, pp. 58–63, Versailles, France, 17-21 June 2002.

[92] W. van der Mark and D. M. Gavrila, “Real-time dense stereo for intelligent vehicles,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 1, pp. 38–50, March 2006.

[93] W. van der Mark, F. C. A. Groen, and J. C. van den Heuvel, “Stereo based navigation in unstructured environments,” in *Proceedings of IEEE Instrumentation and Measurement Technology Conference*, vol. 3, pp. 2038–2043, Budapest, Hungary, 21-23 May 2001.

[94] W. van der Mark, J. C. van den Heuvel, E. den Breejen, and F. C. A. Groen, “Camera based motion tracking for data fusion in a landmine detection system,” in *Proceedings of the IEEE Instrumentation and Measurement Technology Conference*, vol. 1, pp. 817–822, Vail, CO, 20-22 May 2003.

[95] W. van der Mark, J. C. van den Heuvel, E. den Breejen, and F. C. A. Groen, “Camera-based platform and sensor motion tracking for data fusion in a landmine detection system,” in *Unmanned Ground Vehicle Technology V, Proceedings of SPIE*, vol. 5083, pp. 380–389, September 2003.

[96] J. J. Verbeek, N. Vlassis, and B. J. A. Kröse, “Efficient greedy learning of gaussian mixture models,” *Neural Computation*, vol. 15, no. 2, pp. 469–485, 2003.

[97] M. Vergauwen, M. Pollefeys, and L. V. Gool, “A stereo-vision system for support of planetary surface exploration,” *Journal Machine Vision and Applications*, vol. 14, no. 1, pp. 5–14, April 2003.

[98] N. Vlassis and A. Likas, “A greedy EM algorithm for gaussian mixture learning,” *Neural Processing Letters*, vol. 15, no. 1, pp. 77–87, February 2002.

[99] D. Wettergreen, D. Bapna, M. Maimone, and G. Thomas, “Developing Nomad for robotic exploration of the Atacama desert,” *Robotics and Autonomous Systems Journal*, vol. 26, no. 2-3, pp. 127–148, 2-3 February 1999.

[100] P. J. Withagen, F. C. A. Groen, and K. Schutte, "CCD characterization for a range of color cameras," in *Proceedings of the IEEE Instrumentation and Measurement Technology Conference*, pp. 2232–2235, Ottawa, Ontario, Canada, 17-19 May 2005.

[101] Z. Zhang, "A stereovision system for a planetary rover: Calibration, correlation, registration, and fusion," in *Proceedings of the IEEE Workshop on Planetary Rover Technology and Systems*, Minneapolis, Minnesota, 23 April 1996.

[102] Z. Zhang, "Flexible camera calibration by viewing a plane from unknown orientations," in *Seventh IEEE International Conference on Computer Vision*, pp. 666–673, Corfu, Greece, 20-27 September 1999.

[103] Z. Zhang, "Camera calibration with one-dimensional objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 7, pp. 892–899, June 2004.

[104] M. Ziegler, "Region-based analysis and coding of stereoscopic video," Ph.D. dissertation, Technische Universiteit Delft, Delft, The Netherlands, 1997.

List of Publications

The research conducted for this thesis has led to the following publications:

- W. van der Mark, F. C. A. Groen and J. C. van den Heuvel, “Stereo based navigation in unstructured environments,” in *Proceedings of IEEE Instrumentation and Measurement Technology Conference*, vol. 3, pp. 2038–2043, Budapest, Hungary, 21-23 May 2001.
- W. van der Mark, D. Fontijne, L. Dorst and F. C. A. Groen, “Vehicle ego-motion estimation with geometric algebra,” in *Proceedings of the IEEE Intelligent Vehicle Symposium*, vol. 1, pp. 58–63, Versailles, France, 17-21 June 2002.
- W. van der Mark, J. C. van den Heuvel, E. den Breejen and F. C. A. Groen, “Camera based motion tracking for data fusion in a landmine detection system,” in *Proceedings of the IEEE Instrumentation and Measurement Technology Conference*, vol. 1, pp. 817–822, Vail, CO, 20-22 May 2003.
- W. van der Mark, J. C. van den Heuvel, E. den Breejen and F. C. A. Groen, “Camera-based platform and sensor motion tracking for data fusion in a landmine detection system,” in, C. M. Shoemaker, D. W. Gage, editors. *Unmanned Ground Vehicle Technology V, Proceedings of SPIE*, vol. 5083, pp. 380–389, September 2003.
- H. Sunyoto, W. van der Mark and D. M. Gavrila, “A comparative study of fast dense stereo vision algorithms,” in *Proceedings of the IEEE Intelligent Vehicle Symposium*, pp. 319–324, Parma, Italy, 14-17 June 2004.
- P. Jansen, W. van der Mark, J. C. van den Heuvel and F. C. A. Groen, “Colour based off-road environment and terrain type classification,” in *Proceedings of the 8th International IEEE Conference on Intelligent Transportation Systems*, pp. 61–66, Vienna, Austria, 13-16 September 2005.
- W. van der Mark and D. M. Gavrila, “Real-time dense stereo for intelligent vehicles,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 1, pp. 38–50, March 2006.
- W. van der Mark, J. C. van den Heuvel and F. C. A. Groen, “Stereo based obstacle detection with uncertainty in rough terrain,” to appear in: *Proceedings of the IEEE Intelligent Vehicle Symposium*, Istanbul, Turkey, 13-15 June 2007.

Samenvatting[*]

In dit proefschrift zijn verschillende beeldverwerkingstechnieken onderzocht om robotvoertuigen beter in staat te stellen zelfstandig hun weg te vinden over ruw terrein. De bestudeerde onderwerpen betreffen: het bepalen van de voertuigverplaatsing, snelle obstakeldetectie en het herkennen van verschillende terreintypen. Het eerste hoofdstuk bevat een inventarisatie van de literatuur over de recente ontwikkelingen op het gebied van autonome robotvoertuigen die door ongestructureerd terrein rijden.

Het tweede hoofdstuk gaat over de onderzoeksvraag hoe met stereovisie de driedimensionale (3-D) beweging van het voertuig nauwkeurig bepaald kan worden. Omdat cameraprojectie van punten in de 3-D ruimte naar 2-D beeldpunten hierbij een belangrijke rol speelt, wordt eerst de geometrie hiervan besproken. Ook het omgekeerde proces komt aan bod: het reconstrueren van de 3-D positie van beeldpunten met behulp van stereovisie. Vervolgens richten we ons op het schatten van veranderingen in de positie en de oriëntatie van een bewegende stereocamera. Er zijn verschillende methoden die hiervoor 3-D punten, verkregen door middel van stereovisie, gebruiken. Deze variëren van eenvoudige kleinste-kwadratenmethoden tot meer geavanceerde methoden waarbij rekening gehouden wordt met verschillen in de omvang en distributie van fouten in de 3-D posities van de punten. Om de nauwkeurigheid van de verschillende aanpakken te kunnen beoordelen wordt de werkelijke camerabeweging bepaald met een methode die gebruik maakt van kunstmatige referentiepunten die zijn aangebracht in het terrein. Hiermee kan zowel de nauwkeurigheid en de robuustheid van de verschillende schattingsmethoden worden bepaald. De meest nauwkeurige resultaten worden behaald met de methoden die rekening houden met afstandsfouten in de 3-D punten. Door het toepassen van een robuuste techniek kunnen de methoden ook omgaan met grotere aantallen punten die de camerabeweging niet volgen. Over de gehele linie laten de resultaten zien dat het schatten van voertuigbeweging door middel van stereovisie geschikt is voor navigatietaken over een korte afstand die veel precisie vereisen.

In het derde hoofdstuk presenteren we een raamwerk van snelle algoritmen waarmee de dispariteit van grote aantallen pixels in stereobeelden geschat kan worden. Deze maken gebruik van instructies van de computerprocessor die tegelijkertijd op meerdere data-eenheden een bewerking uitvoeren. We maken eerst een onderscheid tussen de verschillende onderdelen van de algoritmen en onderzoeken hoe hun nauwkeurigheid zich verhoudt ten opzichte van de benodigde rekentijd. Verder worden de prestaties vergeleken met die van andere algoritmen uit het publieke domein. De evaluatie is

[*]Summary in Dutch.

gebaseerd op synthetische beelden van een verkeerssituatie in stedelijk gebied. Verstoringen zoals ruis, foutieve kalibratie en lensvervormingen, die ook voorkomen in echte stereo beelden, zijn aan de data toegevoegd. Voor de evaluatie wordt niet alleen de nauwkeurigheid van de schatting van de dispariteit geanalyseerd. Er wordt ook onderzocht wat de gevolgen zijn van schattingsfouten bij de detectie van obstakels. De resultaten laten zien dat er een aanzienlijke invloed is van de omgevingsgerelateerde condities op de prestaties van de algoritmen. Algoritmen die gebruik maken van (globale) zoekoptimalisatie ondervinden hier grotere negatieve gevolgen van dan de meer eenvoudige aanpakken. De benodigde rekenduur van elk algoritme uit het raamwerk laat zien dat de simpele varianten meer dan twee keer zo snel zijn dan de complexere varianten. Echter, de verwerkingssnelheid die deze laatsten kunnen behalen is acceptabel, terwijl hun gemiddelde nauwkeurigheid gelijk is aan dat van de publiek verkrijgbare algoritmen.

De ontwikkelde techniek voor obstakeldetectie wordt besproken in het vierde hoofdstuk. Er is een efficiënte methode ontwikkeld die in staat is om pixels op positieve obstakels te vinden en te clusteren. Deze nieuwe methode houdt rekening met de hogere onnauwkeurigheid in stereoschattingen op grotere afstanden. De methode houdt ook bij of de oriëntatie van de camera's ten opzichte van het grondvlak is veranderend. Dit is noodzakelijk wanneer het voertuig door ruw terrein rijdt. Met verschillende kwantitatieve experimenten hebben we de presentaties van onze aanpak bestudeerd. De resultaten laten zien dat het nadelig is wanneer er geen rekening wordt gehouden met de onzekerheid in de afstand. Dit leidt tot over-segmentatie en verhoogt het aantal vals gedetecteerde obstakels. Omdat onze methode wel rekening houdt met deze onzekerheid is deze beter in staat om obstakelpunten te onderscheiden op grotere afstanden. Dit leidt tot significant minder gedetecteerde valse obstakels.

Het classificeren van verschillende terreintypen in beelden wordt onderzocht in het vijfde hoofdstuk. Toepassen van beeldkenmerken, zoals kleur, voor classificatie doeleinden is problematisch in de openlucht. De waargenomen kleur van terreintypen kan enorm variëren omdat de invloeden uit de omgeving, zoals de belichting, veel kunnen veranderen. We presenteren een nieuwe aanpak die in staat is om een set van voorbeeldbeelden van terreintypen op te delen in kleinere groepen. Deze groepen bevatten beelden die sterk op elkaar lijken omdat de invloeden van hun omgevingscondities overeenkomen. De classificatie van terrein typen wordt verbeterd doordat deze voor iedere set apart getraind wordt. We hebben geëxperimenteerd met verschillende beeldkenmerken zoals kleur, textuur en geometrische informatie. De resultaten met alleen kleurkenmerken laten zien dat onze aanpak in staat is om terreintypen te herkennen in beelden met grote onderlinge verschillen in belichting. Doordat er weinig textuur in de dataset aanwezig was, leverden experimenten die alleen gebruik maakten van textuur geen goede resultaten op. Combinaties van kleur en textuur presteerden niet beter dan de aanpakken die alleen gebruik maakten van kleurkenmerken. Echter, de resultaten konden wel licht verbeterd worden wanneer de dimensionaliteit van de textuurkenmerken verminderd werd. De beste prestaties werden geleverd door het combineren van dimensiegereduceerde textuur-, kleur- en geometrie kenmerken.

Dankwoord[*]

Mijn promotie onderzoek was een lange zit. Aan het eind van deze periode vol spanningen, onmogelijk lijkende uitdagingen, stressmomenten, tegenslagen en vertragingen heb ik echter een gevoel van voldoening en is het alle moeite waard geweest. Gelukkig heb ik tijdens deze periode ook veel professionele en morele steun ontvangen. De personen die hiervoor verantwoordelijk zijn wil ik daarom van harte bedanken.

Ten eerste wil ik graag Johan van den Heuvel bedanken voor de dagelijkse begeleiding. Johan, je was altijd bereid tot een goede discussie en had steeds behulpzaam advies paraat. Natuurlijk wil ik ook mijn promotor Frans Groen bedanken. Frans, er zijn maar weinig mensen met zoveel wetenschappelijk inzicht. Tijdens onze besprekingen had ik meer tijd nodig om de problemen voor te leggen, dan jij nodig had om ze te begrijpen en oplossingsrichtingen aan te geven.

Het heeft veel moeite gekost om goede stereodatasets voor het onderzoek te verkrijgen. Met name de hulp van Han van Bezooijen was hierbij onmisbaar. Han, ondanks dat mijn wensen qua camera's en computers voor op de RoboJeep soms absurde vormen leken aan te nemen, lukte het je toch om alles goed te monteren. Ook wil ik graag Leo Cohen bedanken voor het oplossen van menig elektrisch probleem. Verder ben ik Marinus Maris erg dankbaar voor het onderhouden en aanpassen van de RoboJeep software gedurende de jaren van het onderzoek. Daarnaast ben ik Michel van Elk zeer erkentelijk voor het maken van de synthetische stereobeelden die in hoofdstuk 3 gebruikt worden.

Het is buitengewoon motiverend als je onderzoek enthousiast gevolgd wordt door je collega's. In het bijzonder ben ik Eric den Breejen dankbaar, die al vroeg mogelijkheden zag om mijn onderzoek toe te passen in projecten zoals LOTUS. Ook wil ik graag Koen Benoist, Bas van den Broek, Ton ten Kate, Henk Lensen, Jan Olijslager, Robin de Rooy, John Schavemaker, Harm van Seijen, Peter Tettelaar, Paul Withagen en alle anderen bedanken voor de goede werksfeer en de vele aanmoedigen.

Ik heb veel dinsdagen doorgebracht bij de ISLA-groep van de Universiteit van Amsterdam. Het was leuk en nuttig om met leden van deze groep te overleggen. Ik wil daarom Roland Bunschoten, Leo Dorst, Dariu Gavrila, Jelle Kok, Rien van Leeuwen, Matthijs Spaan, Bas Terwijn, Sjaak Verbeek, Nikos Vlassis en Wojciech Zajdel voor de verleende gastvrijheid.

Tenslotte wil ik uiteraard mijn ouders bedanken voor de steun en begrip die ik heb ontvangen. Beste Alice en Frits, zonder jullie was dit nooit gelukt.

[*]Acknowledgments in Dutch.