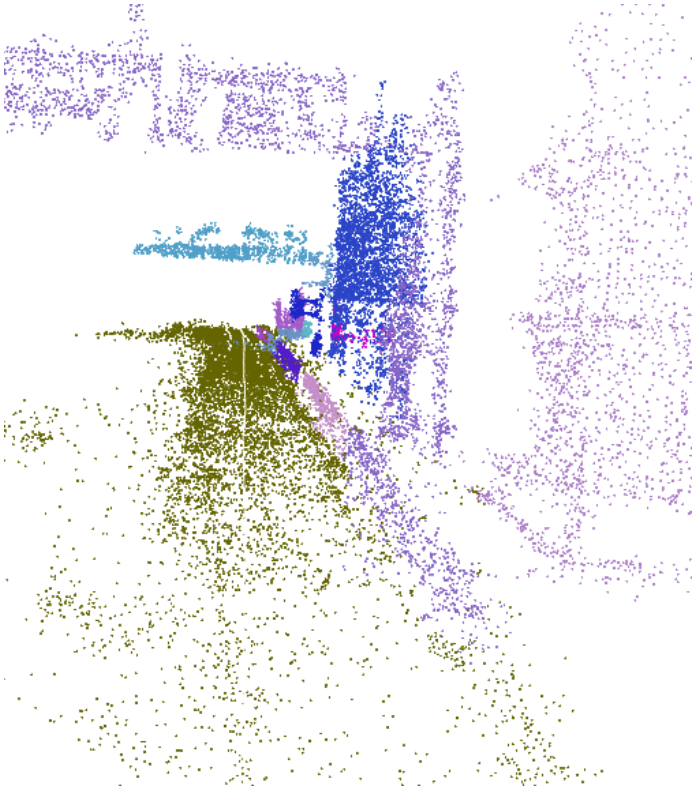
CHALMERS
UNIVERSITY OF TECHNOLOGY



Structure from motion and segmentation of road scenarios

A proposal for offline processing of video data

Master's Thesis in Biomedical engineering

Oskar Asphäll & Johan Ulfsson

Department of Signals & Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2015

Master's thesis EX039/2015

Structure from motion and segmentation of road scenarios

A proposal for offline processing of video data

Oskar Asphäll & Johan Ulfsson

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Signals & Systems
Division of Image Analysis & Computer Vision
Chalmers University of Technology
Gothenburg, Sweden 2015

Structure from motion and segmentation of road scenarios A proposal for offline processing of video data
Oskar Asphäll & Johan Ulfsson

Supervisor: Gustav Santesson, Volvo Car corporation
Examiner: Fredrik Kahl, Signals and Systems

Master's Thesis EX039/2015
Department of Signals & Systems
Division of Image Analysis & Computer Vision
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: A city environment rebuild from video data with SFM and segmented road and potential objects

Typeset in LaTeX
Gothenburg, Sweden 2015

Structure from motion and segmentation of road scenarios
A proposal for processing of offline video data
Oskar Asphäll & Johan Ulfsson
Department of Signals & Systems
Chalmers University of Technology

Abstract

The field of active safety is expanding rapidly within the automotive sector. Having safety as a sales argument among car-makers makes it a hot topic to improve. The time it takes for an active safety system to pass the stage of testing for commercial implementation is strictly depending on the amount of testing done. Supposing video data is available from previous test runs it could be reused to validate new systems. The arising problem is then to correctly identify relevant road scenarios in the video data.
An approach based on 3D segmentation using point cloud data obtained from structure from motion is proposed. The structure from motion technique is implemented using OpenCV and the segmentation methods using PCL. The experimental results of the structure from motion demonstrate that it is possible to recreate 3D point clouds from the video data collected by Volvo Cars. The geometric structures created are representing the environment. However the technology has some clear limitations such as field of view for the camera and inability to depict moving objects. Further experimental results show that the segmentation methods work well for segmenting the road and clear objects such as barriers, despite sparse point clouds. Additional work is needed to evaluate if it is dense enough for accurate scene classification.

Keywords: Structure from motion, SfM, keypoints, feature points, point cloud, segmentation, active safety, verification,

Acknowledgements

We would like to thank our examiner and supervisors at Chalmers, Fredrik Kahl for his support during the working of the project. A huge thanks also goes to our supervisor at Volvo Car Corporation, Gustav Santesson for his knowledge, support and ideas. We also want to thank Yury Tarakanov and Fredrik Persson, who supported the work with new ideas and knowledge. We would also like to send our gratitude to the people developing the Point Cloud Library, PCL, and OpenCV which helped a lot in the implementation of the tracker. We would also like to thank Samuel Scheidegger and Mathias Ernst for productive discussions and ideas.

Oskar Asphäll and Johan Ulfsson, Gothenburg, June 2015

Contents

List of Figures

List of Tables

1
Introduction

1.1 Active safety systems

Car manufacturers are currently competing to achieve the best safety possible for their costumers. Volvo cars have a clear vision that say "By 2020, nobody shall be seriously injured or killed in a new Volvo" [6]. To achieve this vision there is a need to create redundancy in the control of the car. For some years this has been partly achieved by systems aiding the driver in some situations, those situations are increasing in numbers. A further development to achieve safer driving can be autonomously driven cars. These systems rely on sensor data and machine learning algorithms working in real time in the car.
Volvo Cars is a fast-growing active safety developer with a clear aim to be branch leading in all safety aspects. For this to be possible both the development and verification of these systems need to be fast and accurate.

1.1.1 Verification

To verify the sensor a large amount of data has to be collected and verified. Collecting the data is mainly done by driving on roads and in specified test areas while logging the sensor data together with a vast span of data on the communication buss. When the data is collected problematic situations have to be found and evaluated in the massive amount of logs. This is to some extent done by hand due to lack of the possibility to have an automatic construction of ground truth. This process is time consuming and expensive, therefore there is a need to automate parts of the process.

1.2 The thesis approach

It would be a vast improvement if there were more information about the content of each log, information that could be used as a reference for online algorithms. This can be done by algorithms evaluating the information in the log after the log is collected. The advantage is that more information is accessible and we can use a computational cluster. There have been some developments where LIDAR is used to build up a 3D point cloud[7]. This gives a very dense point cloud with even rows of points. One problem however is that the LIDAR system is rather expensive. Another problem is the vast amount of logs that have only video and no LIDAR. To verify the logs with no LIDAR another approach is recommended. Therefore this

thesis project aims at building good 3D environments from just the video log. This can be done with Structure from Motion (SfM from here on). However there are some known limitations to the SfM method, for example objects not appearing in the video cannot appear in the point cloud and objects moving with respect to the ground, between frames will be noisy or filtered out.
A large inspiration for this project has been the object segmentation and classification from 3D LIDAR data, a large problem with LIDAR is the cost. Therefore it is interesting to evaluate the ability to create 3D structure from a simple single lens camera.

1.2.1 Offline processing

Functioning active safety systems are required to run in real time on processors in the car. This thesis has no aim of achieving that since the application is verification on logs already collected. The code generated and discussed will be offline code. Hence we can use more demanding algorithms and a computational cluster can be used to processes large amounts of logs.

1.2.2 Goal

The ultimate goal of this project is to construct an accurate 3D point cloud structure that could be segmented and finally used in a classifier to tag various scenarios. This came to be a massive undertaking and hence the thesis is limited to creating a point cloud and presenting segmentation methods. It will provide an understanding of the existing methods and review possibilities, advantages and disadvantages.
This is the first project creating rather dense SfM clouds for segmentation from Volvo Cars unique data. Methods will be applied for segmentation that likely only have bean used for LIDAR data before.

1.2.3 Data

This thesis aims to achieve algorithms that can be validated and implemented on Volvo Cars collected video. That data is collected with a resolution of 1280x960, each pixel is 12 bits and the frame rate is 18 frames per second. The lens that is collecting the data is of Read Clear Clear Clear (RCCC) type, further descriptions and information on that is found in the theory chapter. The image is also distorted when collected but the distortion is known so we can undistort the image. We also have access to information from the CAN bus such as speed of the car.

1.3 Report structure

The report is divided in to theory chapters, method and implementation, results, discussion, and conclusion. The theory part provides information concerning the general theories and methods for building SfM systems. The next part focuses on picking out segments from the SfM point cloud, the last theory part is intended to give some insight in to the classification possibilities.

An important focus point in this report is the method and implementation chapter. The reader will obtain insight regarding the method selection and implementation of algorithms that took part when applying the theory part to reach the results of this master thesis. There will also be more information on the base data used in the project. Also this chapter provides the information base for the problems and limitations discovered, later shown in results and handled in the discussion segment. Result and discussion explain, illustrate and compare the results of the different SfM and segmentation algorithms used in the project. The advantages and disadvantages of the different methods evaluated within the frame of the master thesis will be discussed.

1.3.1 Related work

This section will describe some related work that have been performed and can give interesting background information for this thesis.

1.3.1.1 Visual odometry

A lot of work has already been done in the area of visual odometry. One interesting approach using a planar road model to tackle the problem of scale ambiguity is described in [8], where no information other than the camera's height over ground is needed. Another implementation focusing on time efficient and geographically versatile implementations of simultaneous localization and mapping is presented by [9], with an impressive accuracy. Both those methods inspired the algorihtms that is implemented in the thesis. In order to evaluate SfM methods a benchmarking procedure is described in [10] where different errors describing accuracy in positioning is elaborated. This thesis use the reprojection error.

1.3.1.2 Segmentation

Normals have been used combined with region growing to segment parts of point clouds for numerous applications, often in LIDAR clouds. One such application is described in [11]. Here normals are used to segment smooth parts of industrial equipment with satisfying results. The thesis use similar methods for segmenting road.
Previously DoN segmentation have been implemented to segment unorganized 3D point clouds from LIDAR origin, one such implementation is described in [12]. Here the results are satisfying, and the article states that the algorithm "showed a clear segmentation of points belonging to various objects of interest at different scales, such as cars, road curbs, trees, and buildings" however our point cloud is rather different since it is from SfM origin. This algorithm is implemented for segmentation in this thesis.
Min-Cut Based Segmentation is a method for segmentation that does not rely on curvature estimation like DoN, but instead uses change in the horizontal plane to limit growth [13].
The Min-Cut algorithm limits the growth in the horizontal plane with aspect to the total growth including the vertical growth. The change in the horizontal plane as a

ratio of the total change including vertical change is limited. This means that the algorithm will stop growing when it reaches a flat surface. For example for a pole the algorithm should find the minimum cut where the pole ends and the ground begins. In [14] [15] an implementation of this algorithm is explained. It is possible to make an automatic implementation of the algorithm, however to have a better classification of all objects, seeds can be placed on the objects and background [14]. This algorithm is not implemented but refereed to both as comparison and further work.

2
Structure from Motion

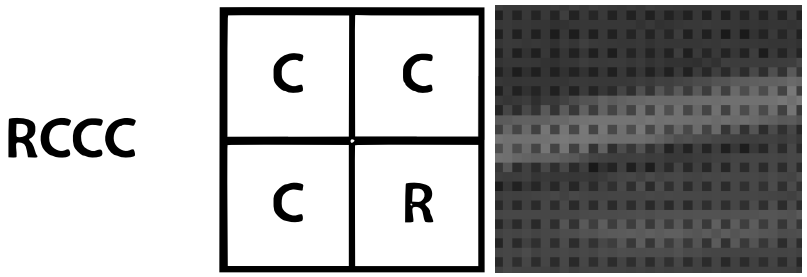
The process of extracting 3D structure in the form of point clouds from video data contains several steps, with theory from a wide variety of areas. This section explains the theory for the different steps to form a basis to understand the method used and results produced later in the report.

2.1 Preprocessing

To achieve good results from the data collected by Volvo some pre-processing had to be done, the pre-processing is mostly to get video in a format the algorithms developed can handle without damaging the result. For example pixels of drastically different intensity due to the design of the camera can make it hard to extract features.

2.1.1 Red Clear Clear Clear(RCCC) pixel model

The video feed is collected using a RCCC camera, a type of camera where the pixels are grouped as three pixels measuring the light intensity and the fourth the intensity of the red spectra 2.1a2.1.



(a) **(b)**

Figure 2.1: (a) The order of pixels in RCCC cameras. (b) A section of an image taken with RCCC camera.

For many of the later suggested algorithms this will cause issues and hence the effect of the red pixels is necessary to suppress. The filter suggested in [1] was applied to the red pixels and was chosen due to its advantage of considering information from both the red and clear pixels in its immediate surrounding. The filter can be

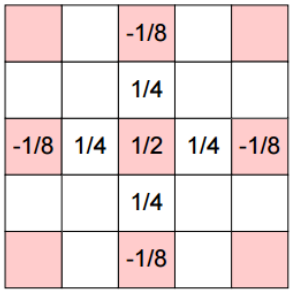


Figure 2.2: The averaging filter considers both the clear pixels and the red pixels [1].

depicted in figure 2.2 and will reduce the effect of the red pixels to a visually nearly perfect result.
Another complication that had to be addressed before any further image processing was correction of the radial distortion caused by the camera lens. Since the radial distortion coefficients were provided the images could easily be remapped to its undistorted form.

2.1.2 Object identification

The point cloud algorithm is depending on points to be stationary between frames to get an accurate estimation of the points. If an object is moving between frames it is considered as noise and should preferably be discarded. This will be partially handled by methods mentioned later in this report, but there are information regarding objects already detected in the videos provided such as cars and pedestrians. There are also information regarding speed for each object which is crucial for knowing if the objects are stationary or moving. This information will be used to reject information from moving objects to obtain better results.

2.2 Feature point localization and descriptor extraction

The transformation between two consecutive frames from a moving monochrome camera can be dechiffered into operations such as rotation, translation, scaling and various affine transformations. In addition to these transformations, other aspects such as variation in illumination and stochastic noise have to be considered in order to pick suitable features [16].
The three features selected that all are invariant, to a certain degree, to the transformations and interference mentioned above are the Scale-Invariant Feature Transform (SIFT), the Binary Robust Invariant Scalable Keypoints (BRISK) and the Speeded Up Robust Features (SURF). SIFT is expected to outperform both SURF and BRISK when it comes to accurately match features [17]. However, BRISK is up to a magnitude faster than the other to alternatives [4].
The construction of a each of these features can be divided into four sections:

1. Keypoint detection
2. Keypoint selection
3. Orientation assignment
4. Descriptor derivation.

2.2.1 SIFT

The first step in detecting interesting keypoints is to search for an image's extrema at several different spatial scales, which results in keypoints that are invariant to scale changes. The scaling is done by smoothing an image, $I(x, y)$, by convolving it with Gaussians with various sigmas, $G(x, y, \sigma)$ [18]:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y),$$

where the two dimensional Gaussian kernel is:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}.$$

Adjacent Gaussian convolved images are then subtracted to create a bundle of difference-of-Gaussian (DoG) images. Secondly the Gaussian convolved images are downsampled by a factor of two when the frequency information is low enough to be represented at a lower scale in order to be more computationally concise before the process starts over [16]. Each of these sessions are called octaves and are illustrated in figure 2.3.

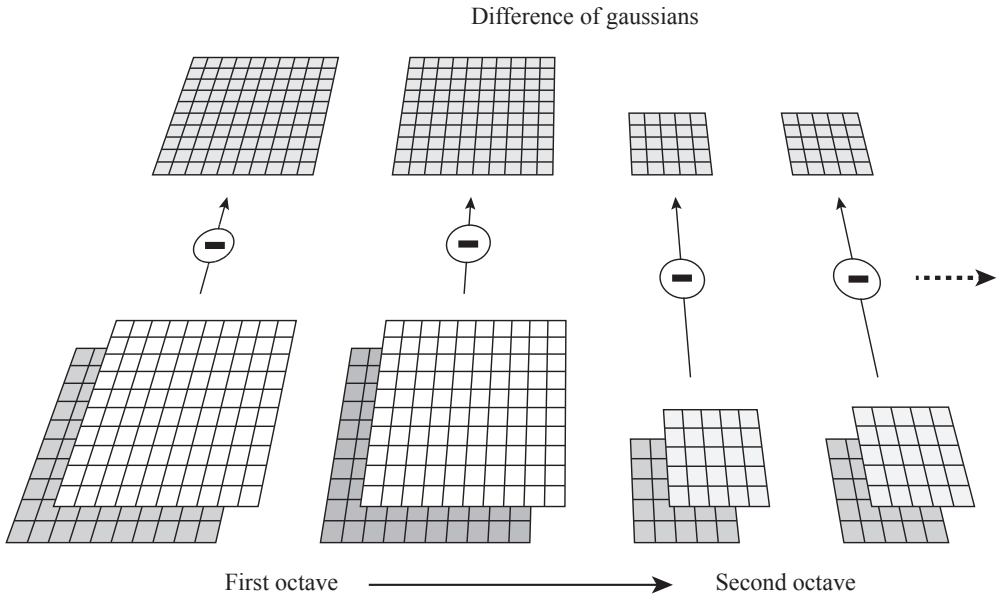


Figure 2.3: Adjacent Gaussian filtered images are subtracted to create the DoG seen in the top of the figure and the downsampling seen to the right.

The extraction of local minima or maxima from the DoG images is done by comparing each point to the surrounding eight neighbours in the same scale and the 18 neighbours from the adjacent scales. This procedure will detect extrema that are arbitrarily close to each other with no limit of the extrema amplitude. The points detected consisting of low amplitude are of no interest since they might be part of noise in the image or a spatially bad localization along edges. Hence there is of interest to discard these ambiguous keypoints before continuing to the descriptor extraction.
In order to remove keypoints with low contrast as well as refining the locations of the previously identified maxima a 3D quadratic function is fitted to the detected points of the DoG [19]. This approach takes advantage of the three initial terms in the Taylor expansion and yields the following expression:

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T\frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x}. \tag{2.1}$$

The refined extremum is then identified by calculating its derivative, setting it to zero and solving:

$$\hat{\mathbf{x}} = -\frac{\partial^2 D^{-1}}{\partial \mathbf{x}^2}\frac{\partial D}{\partial \mathbf{x}}, \tag{2.2}$$

The variable $\hat{\mathbf{x}}$ denotes the offset in relation to the original point. The maximum offset allowed before changing the origin of the initial point is set to 0.5 which implies that the point is diverging to a different extremum. Finally the offset is added to the initial keypoint in order to update the location of the keypoint. Furthermore, by substituting equation 2.2 into equation 2.1 the following expression is obtained which is used to discard low contrast keypoints:

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2}\frac{\partial D^T}{\partial \mathbf{x}}\hat{\mathbf{x}}. \tag{2.3}$$

The value of $|D(\hat{\mathbf{x}})|$ is a good indicator of the keypoint contrast and figure 2.4 illustrates the result with various thresholds.

(a) **(b)**

Figure 2.4: (a) Contrast threshold set to 0.04. (b) Contrast threshold set to 0.004.

It can clearly be depicted that by allowing keypoints with a lower contrast more points are detected in the road surface. However, this comes with the price of

increased number of detected points in the sky. This problem will be addressed under the section sky estimation as well as commented under the section estimate fundamental matrix.
In order to improve the quality of the keypoints it is of interest to remove those located on an edge. Points located along edges will be detected since the DoG has a strong edge response, which is unwanted since they are poorly spatially located along these edges [20]. The key to detecting keypoints located on an edge is to analyse the principal curvature of the DoG. The principal curvature will have a strong response perpendicular to the edge and a weak response otherwise. The principal curvature is described by the 2x2 Hessian matrix containing the local derivatives surrounding the keypoint:

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}. \tag{2.4}$$

By considering the relation of the eigenvalues of $\mathbf{H}$, according to [16], it is enough to calculate the sum of the horizontal and vertical derivatives, which yields the sum of eigenvalues $\alpha+\beta$, as well as the determinant, which yields the product of eigenvalues $\alpha\beta$. The relation between these two then has to be lower then a quantity expressed with a threshold r:

$$\frac{(\alpha+\beta)^2}{\alpha\beta} < \frac{(r+1)^2}{r}. \tag{2.5}$$

So far the keypoints have been derived using a scale space extrema method where the points with poor spatial localisation have been discarded. Next step is representing these keypoints relative to their rotation in order to achieve rotational invariant features. For each scale with detected keypoints an area around each keypoint in the corresponding Gaussian filtered image is selected in order to compute orientation as well as magnitude:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}, \tag{2.6}$$

$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y))). \tag{2.7}$$

The orientation is calculated in each pixle around a keypoint and weighted heavier the closer they are to the center. These orientations are expressed in a histogram consisting of 36 bins with 10 degrees each in order to cover all the angles. The peak in the histogram describes the keypoint orientation. It is possible for a keypoint to be assigned more than one orientation which is shown to increase stability when matching [16]. The feature will in some cases from now on be denoted with not only a scale, but also a color relative to its direction, figure 2.5.

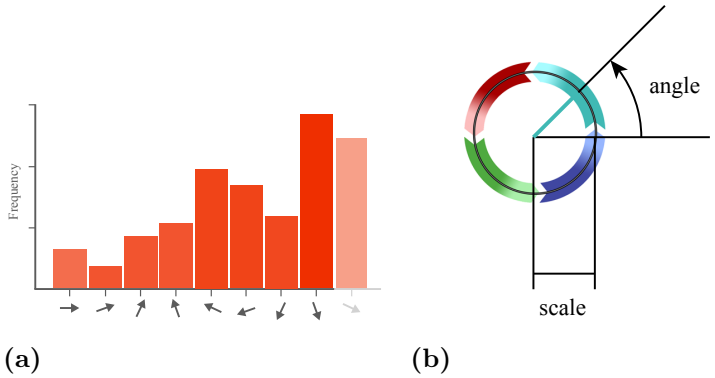


Figure 2.5: (a) Histogram of different direction representing each angles recurrence. (b) a visual representation of a keypoint.

Once the keypoints are localized and described invariant to both scale and orientation they have to be described uniquely in order to make it identifiable. The descriptor is created from 16 concatenated histograms each consisting of 8 bins all computed from an area of 16x16 pixels around each keypoint at its respective scale, resulting in a 128 valued descriptor. The procedure is depicted in figure 2.6.

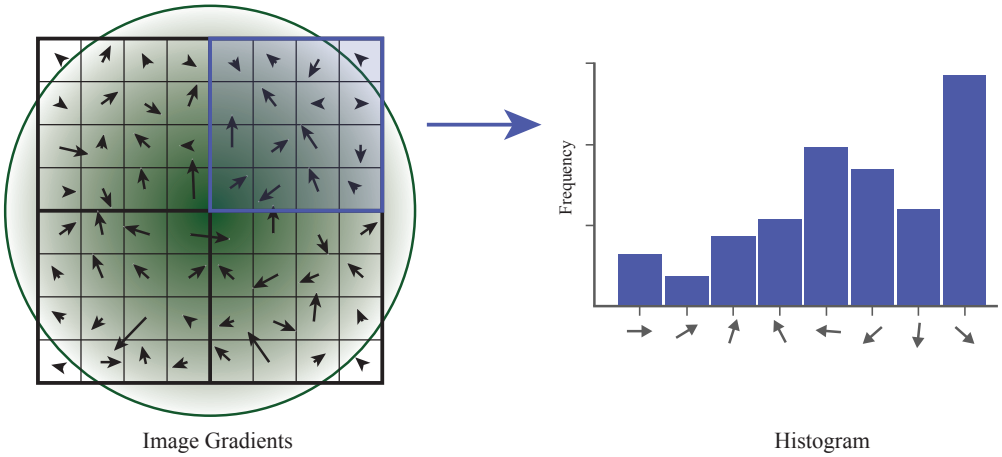


Figure 2.6: (a) The orientation of pixels surrounding a keypoint where the circle represent a Gaussian weight. (b) One of the in total 16 histograms describing the keypoint.

2.2.2 SURF

Both the derivation of detector and descriptor are in many ways similar to the extracting of SIFT features. The initial step is as in SIFT to detect a point of interest by analyzing extremum at different scales. SURF makes use of the determinant of the Hessian matrix. If the determinant is negative it means that the eigenvalues of the Hessian matrix have different signs simply because the determinant is the product of its eigenvalues. Based on the same conclusion a positive determinant is obtained when the both eigenvalues are of the same sign, hence a maximum. The Hessian is defined as following:

$$H(x, y, \sigma) = \begin{bmatrix} L_{xx}(x, y, \sigma) & L_{xy}(x, y, \sigma) \\ L_{xy}(x, y, \sigma) & L_{yy}(x, y, \sigma) \end{bmatrix}. \tag{2.8}$$

Where the four entries $L(x, y, \sigma)$ refers to the convolution of the image with a Gaussian kernel. The Gaussian kernel can be varied in order to smooth the image to different extent making it possible to calculate the determinant of the Hessian matrix at more than one scale. In order to increase the efficiency computing the entries to the Hessian the Gaussian kernels are approximated as box filters, see figure 2.7.

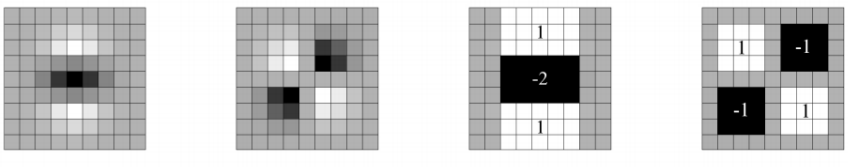
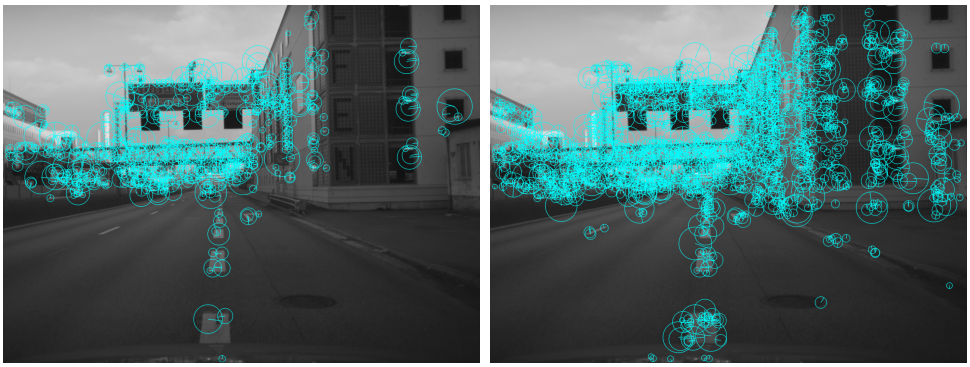


Figure 2.7: Gaussian kernels expressed as box filteres [2].

Further more, when the box filters are convolved with the integral image the computations are decreased significantly. An integral image makes it possible to reduce computations of the sum of a rectangular area in an image to four operations disregarding the size of the rectangle. The integral image consist of the sum of all pixels column wise and row wise between the point (x, y) and the origin of the image. In difference to the SIFT method when obtaining the scale-space SURF utilizes the fact that the same result can be obtained by increasing the scale of the Gaussian kernels which in turn has no effect on the number of computations [21].
In parity to the SIFT weak keypoints can be removed in order to fit the application. In SURF this is done by removing keypoints with a determinant lower than a specified threshold, see figure 2.8.



(a) **(b)**

Figure 2.8: SURF features, (a) Threshold set to 800. (b) Threshold set to 100.

When questionable keypoints have been discarded the procedure of finding the scale of the keypoint as well as refining its location is identical to SIFT described in the previous section. When the keypoint is successfully localized it has to be expressed

orientationally invariant. This is done in a similar manner to SIFT where local gradients around the keypoints are considered and weighted into general orientations. SURF however uses binary Haar wavelets with a size related to the scale to compute the orientations used for defining not only the general orientation but also for constructing the descriptor. An area around a keypoint is divided into subregions which can be expressed as [2]:

$$desc_{sub} = \left[\sum dx, \sum dy, \sum |dx|, \sum |dy|\right]. \tag{2.9}$$

Each descriptor consists of a set subdescriptors which in all results in either 64 or 128 elements.
To visualize the result of the extremum detector figure 2.9 illustrates the characteristics of the SURF detector where it clearly can be depicted that distinct maxima and minima has been detected and can be seen as a blob detector.

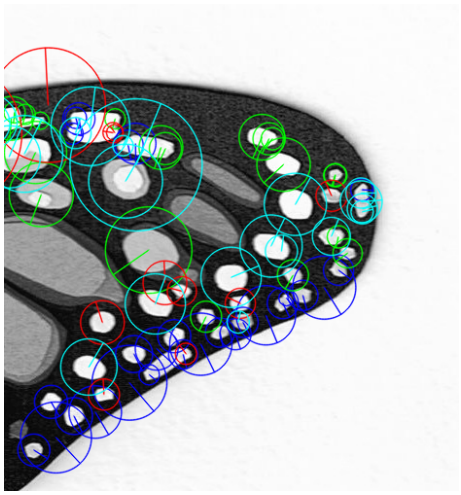


Figure 2.9: Using the SURF detector with a threshold of 400 one can see that it acts as a blob detector.

2.2.3 BRISK

A third method for detecting keypoints and describing it's neighbourhood is BRISK which is expected to yield similar results as SIFT and SURF but considerably faster. The keypoint detector used will be the corner detection method Features from accelerated segment test (FAST) as suggested by [4]. The fast detector is based on comparing the 16 neighbouring pixels to the central pixel, see figure 2.10. For a pixel to be considered interesting at least 9 consecutive pixels of the 16 pixels considered has to be remarkably different than the center pixel [3].

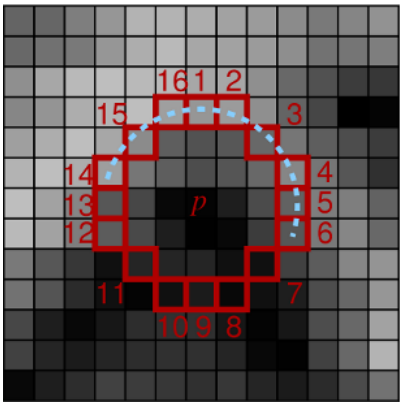


Figure 2.10: The 16 pixels considered to mark a pixel interesting or not [3].

As the SIFT and SURF detector the FAST detector fulfils the requirement of scale invariance by resampling the image and checking the FAST criterion at each scale. Using BRISK the scale space is created by downsampling the image with a factor of two called octaves, as familiar, but also intermediate octaves are created by downsampling of the image in between the octaves. Similar to SIFT and SURF the scale of the keypoint is determined by analyzing an interesting point in several scales. A detected keypoint is compared to its 8 neighbours in the same scale and if it is still considered an interesting point it is given a score. If this score is higher than the same points in the adjacent layers as illustrated in figure 2.11 the keypoint and corresponding scale are kept [4]. The location of the keypoint is then further refined with the same method described in SIFT.

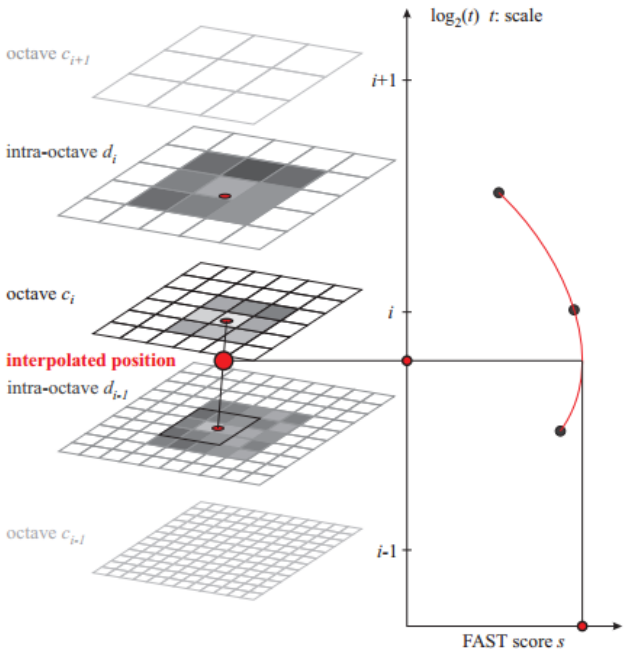


Figure 2.11: Interesting points are given a score depending on their 8 neighbours and compared to the adjacent scales [4].

A result using BRISK with varying thresholds can be depicted in figure 2.12.

(a) **(b)**

Figure 2.12: BRISK features, (a) Threshold set to 30. (b) Threshold set to 10.

BRISK is rotationally invariant by computing a general orientation for each keypoint expressing each descriptor relative to its orientation. To compute the descriptor 60 points are used which are placed in a concentric structure around the keypoint. These points are then used in brightness comparison tests with the results stored in a binary string. Additional reading of the derivation of orientation and descriptor can be found in [4].

2.3 Matcher

To match descriptors between frames a matcher has to be used. The matchers evaluated in this report are the Fast Library for Approximate Nearest Neighbors (FLANN) and a simple Brute Force matcher. The FLANN matching is expected to be computationally more efficient but less accurate than the BF matcher, which will be further described in the following sections in a simple manner. For a first step of filtering out the wrong and weak matches the concept of comparing the best match with the second best match will be used, known as the Lowe's criterion. When the score of the best match is close to the score of the second match the risk of not having selected the correct match increases. Hence, if the ratio between the score of the best and the second best match is higher than 0.8 they are discarded. By doing this up to 90% of the incorrect matches are removed with the cost of only 5% of the correct matches [16].

2.3.1 Brute force matching

The brute force matcher is a very straight forward procedure. It compare a descriptor from one set with all the descriptors from the second set and returns a score describing a distance. The distance types recommended for the features used in this project is euclidean distance for SIFT and SURF but hamming distance for BRISK [22]. As mentioned previously this procedure has to be repeated in order to also find the second best match needed to check Lowe's criterion.

2.3.2 FLANN

The second matcher evaluated was the FLANN matcher. As the brute force matcher it will only be described in general. The advantage of using a FLANN matcher compared to the brute force matcher is the speed. The FLANN matcher speed up matching but at the cost of some accuracy. The FLANN algorithm initially constructs Kd-trees similar to a classic Kd-tree. The difference is that while the classic Kd-tree splits data on the dimension with the highest variance, the FLANN kd-tree splits data randomly from the top 5 highest variance dimensions. Several such trees are then created and searched, for each search all feature vectors not marked as a neighbour are removed. This method sacrifices some accuracy for a large increase in speed [23].

2.3.3 Estimation of sky

In order to get a good representation of the surrounding it is important to successfully match features from all surfaces in an image except features from the sky. This is not possible just by trimming the contrast or intensity parameters of the keypoint detector. Hence the points detected in the sky are preferably discarded before any matching is done. This section describes a simple method that makes use of the irregularities in the image to isolate the sky. The method can be divide into the following steps:

1. Smoothing
2. Irregularity detection
3. Dilation and binarization
4. Erosion and removal of small unconnected areas.

The reason for smoothing is simply to remove any artifacts originated from the RCCC filtering. To identify irregularities both corners and edges are detected using harris feature detection respectively a sobel filter and are combined with equal weights.

2.4 Estimate motion

This section will cover the essentials of recovering the views of two adjacent images and acquisition of their geometric relation. Initially there will be a brief explanation of the camera model followed by describing the correspondence between points in two different views and finally deriving the camera motion.

2.4.1 Pinhole model

A pinhole camera model describes the relation between a 3D point and its projection onto the image plane. The origin of the model is set to the intersection of all rays due to the refraction of the camera lens. The distance of an arbitrary point in the 3D space will be described relative to this origin, see figure 2.13. Additional information

required to describe the projection of a 3D point is the distance between the origin and the image plane, the focal length, as well as the centre of the image plane, the principal point. These are all intrinsic parameters and are contained within the intrinsic matrix:

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.10}$$

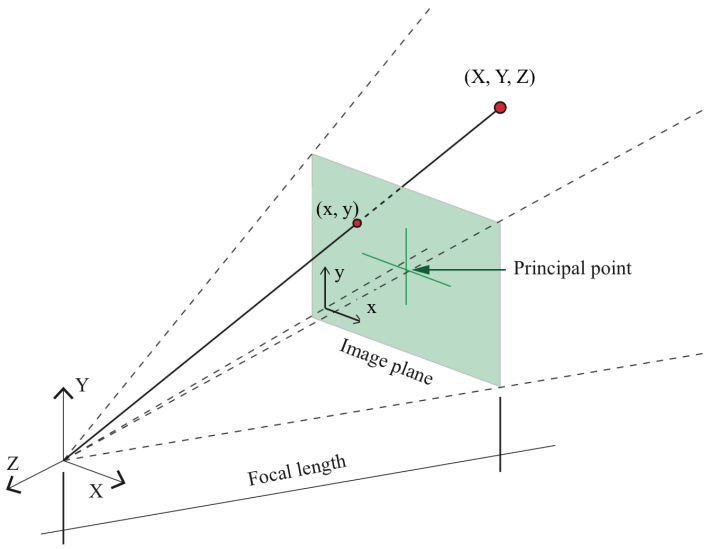


Figure 2.13: Illustration of the pinhole camera model.

The relation between a 3D point and a 2D point can be expressed as in equation 2.11 assuming no camera movement

$$s \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{2.11}$$

The parameter s in the previous equation is a scale factor converting the point from cartesian coordinates to homogeneous coordinates. It can be interpreted as a translation of the xy-plane in a perpendicular direction of length s from the axes *(x,y)* and is used in projective geometry [24]. The geometric relation between two images taken with stationary camera poses is simply unaltered. However, if the camera is under motion while capturing the images the relation between consecutive frames can be described as a translation and a rotation according to figure 2.14.

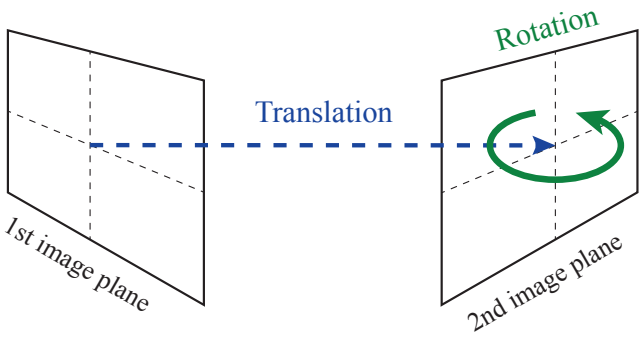


Figure 2.14: The relation between two frames is seen as a translation and a rotation.

If the camera is under influence of motion when capturing the frames the relation between a 3D point and a 2D point can then be expressed as

$$s \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{K}[\mathbf{R}|\mathbf{t}] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{2.12}$$

2.4.2 Point correspondence

In order to derive the geometric correspondence of translation and rotation relating two frames the constraint between point **x** in the first image plane and point **x'** in the second image plane has to be studied. Assume a 3D point is projected on two adjacent image planes, figure 2.15. These two points, combined with the origins of the respective camera, O and O', will all lay on a plane that is coplanar to both image planes. Hence, by only knowing the location of the point **x** in the first image, it is possible to know that point **x'** lies in the intersection between this plane and the second image plane, the epipolar line l'. It can also be depicted that the epipole, e, is the projection of the other image center. Several different 3D points are projected on to several different epilines all intersecting the epipole [25].

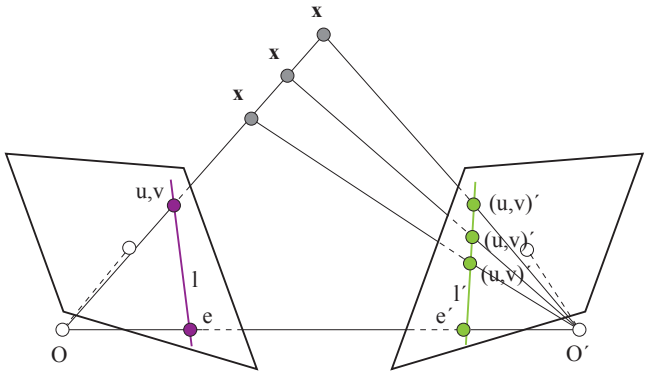


Figure 2.15: Projection of a 3D point on two image planes illustrating the epipolar line.

The 8-point algorithm will be used to calculate the fundamental matrix which describes the geometric relation between two consecutive frames. The fundamental matrix is based on the previously stated epipolar constraint which can be expressed as:

$$\mathbf{x}'^{T}\mathbf{F}\mathbf{x} = 0, \tag{2.13}$$

where $\mathbf{x}' = (x', y', 1)^T$ and $\mathbf{x} = (x, y, 1)^T$ will together give rise to one linear equation. By expanding equation 2.13 the following expression containing the known points from the left respectively the right image as well as the entries of the fundamental matrix is obtained

$$xx'F_{11} + xy'F_{21} + xF_{31} + yx'F_{12} + yy'F_{22} + yF_{32} + x'F_{13} + y'F_{23} + F_{33} = 0. \tag{2.14}$$

The equation can also be arranged as following where it is important to have at least eight corresponding points in order to get one unique solution:

$$\mathbf{A}\mathbf{f} = 0, \tag{2.15}$$

where $\mathbf{A}$ is the vector:

$$\begin{bmatrix} x_l x_r & y_l x_r & x_r & x_l y_r & y_l y_r & y_r & x_l & y_l & 1 \end{bmatrix}. \tag{2.16}$$

The next step is performing singular value decomposition (SVD) of $\mathbf{A}$ and selecting the last column of the right unitary matrix which correspond to the least singular value in the diagonal matrix and also the linear solution of the fundamental matrix [26]. In order to improve the result, an additional SVD is performed on the newly obtained fundamental matrix, setting the smallest singular value to zero. The improved result is obtained by multiplying the unitary matrix $\mathbf{V}$, the diagonal matrix $\mathbf{D}$ and the transpose of the unitary matrix $\mathbf{V}$ [27]:

$$\mathbf{F} = \mathbf{U}\mathbf{D}\mathbf{V}^{\mathbf{T}}. \tag{2.17}$$

It is suggested that in order to increase the robustness of the 8-point algorithm the points should first be normalized. In other words they should be transformed to a new coordinate system where the center is the mean of all points and the mean square distance of all points in one image to its center is equal to $\sqrt{2}$ [28].
To find the center of the new coordinate system, the centroid, the mean of all points is calculated:

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \tag{2.18}$$

$$\overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i. \tag{2.19}$$

Secondly the average distance from a point to the centroid is calculated and divided by $\sqrt{2}$ to ensure the new distance constraint.

$$\overline{d} = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \overline{x})^2 + (y_i - \overline{y})^2}{2n}}. \tag{2.20}$$

The normalized points can then be expressed accordingly:

$$\hat{\mathbf{p}}_i = \begin{bmatrix} \hat{x}_i \\ \hat{y}_i \\ 1 \end{bmatrix} = \begin{bmatrix} (x_i - \overline{x}_i)/\overline{d} \\ (y_i - \overline{y}_i)/\overline{d} \\ 1 \end{bmatrix}, \tag{2.21}$$

where the two images will have separate transformations:

$$\hat{\mathbf{p}}_l^i = \mathbf{T}_l \overline{\mathbf{P}}_i^l, \tag{2.22}$$

$$\hat{\mathbf{p}}_r^i = \mathbf{T}_r \overline{\mathbf{P}}_i^r. \tag{2.23}$$

The variable $\overline{\mathbf{P}}_i^l$ denotes the keypoints of the left frame and $\overline{\mathbf{P}}_i^r$ of the right. The transformation matrices can then be written as following, one for the left image and one for the right:

$$\mathbf{T}_{l,r} = \begin{bmatrix} 1/\overline{d} & 0 & -\overline{x}/\overline{d} \\ 0 & 1/\overline{d} & -\overline{y}/\overline{d} \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.24}$$

After calculating the fundamental matrix using the normalized points the actual fundamental matrix has to be recovered using following equation:

$$\mathbf{F} = \mathbf{T}_r^T \overline{\mathbf{F}} \mathbf{T}_l. \tag{2.25}$$

However, by just selecting eight of the matched points to derive the geometric relation makes the algorithm very susceptible to noise. One mismatch will lead to a faulty fundamental matrix. There are fortunately various methods applicable to improve the result when the observed data is subjected to outliers. For this purpose the Random sample consensus (RANSAC) method is chosen due to its proven efficiency regarding mismatches and possibility of identifying these outliers [29].

2.4.3 RANSAC

RANSAC is an iterative method based on randomly selecting samples from a collection in order to derive a model and see how well the total set of samples agree with this model [30]. One simple example is a line of points fused with scattered outliers, figure 2.16.

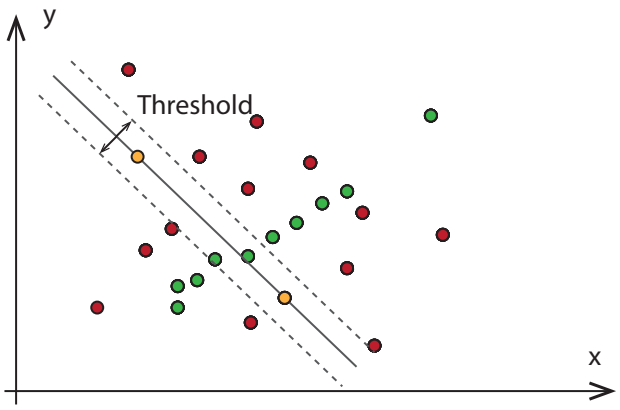


Figure 2.16: Red points are expected outliers, green points are expected inliers and the orange points connected by the line is a guess.

The first step is to define the model to which all the points will be fitted. It is however obvious by studying the line that they are not exactly located on a straight line, hence the accuracy has to be defined by a threshold in order to describe how much a point are allowed to deviate before it is classified as an outlier. When the model is defined and the threshold is specified a number of hypothetical inliers are selected and the model is adjusted to fit these samples. After the model is adjusted all points are tested against this model and all points that fall inside the predefined threshold are considered inliers and part of a consensus set whereas the rest are considered as outliers. The number of inliers are stored and the procedure is repeated a set number of times and the model that has the largest consensus set is assumed to be the correct model [31].
The procedure of obtaining the fundamental matrix and identifying outliers using RANSAC has proven to be very efficient in cases of big fractions of outliers [32]. The concept is fairly straight forward, eight random points are selected as hypothetical inliers from the matched points and the fundamental matrix is derived using the eight point algorithm. Secondly all the points are tested against the epipolar constraint described in equation 2.13. If the point lay on the epipolar line or within a chosen threshold it is considered an inlier. By using this strategy not only the fundamental matrix will be computed with as many inliers as possible, but the outliers can be removed accordingly.

2.4.4 Essential matrix

The fundamental matrix relates the geometry of two images regardless of the camera intrinsics mentioned in section 2.4.1 and has 7 degrees of freedom. But in order to relate this model to a 5 degree of freedom model, consisting of translation and rotation the camera calibration parameters are considered according to:

$$\mathbf{E} = \mathbf{K^T F K} \tag{2.26}$$

The essential matrix $\mathbf{E} = \mathbf{Rt}$ is scale ambiguous which will be addressed after the decomposition of $\mathbf{E}$ which is done using the two following matrices, the orthogonal matrix $\mathbf{W}$, and the skew-symmetric matrix $\mathbf{Z}$ [25]

$$\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.27}$$

$$\mathbf{Z} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{2.28}$$

The two non zero singular values of $\mathbf{E}$ have to be equal in order to reduce the degrees of freedoms to five, three for rotation and two for translation. Further more by assuming that the scale is arbitrarily the two singular values can both be set to 1:

$$\mathbf{E} = \mathbf{U} diag(\begin{bmatrix} 1 & 1 & 0 \end{bmatrix}) \mathbf{V^T}. \tag{2.29}$$

With the newly defined decomposed $\mathbf{E}$ two possible solutions for the translation respectively rotation can be computed [25]:

$$\mathbf{t_1} = -\mathbf{U Z U^T}, \tag{2.30}$$

$$\mathbf{t_2} = \mathbf{U Z U^T}, \tag{2.31}$$

$$\mathbf{R_1} = \mathbf{U W^T V^T}, \tag{2.32}$$

$$\mathbf{R_2} = \mathbf{U W V^T}. \tag{2.33}$$

The computed translation and rotation can combined freely describe four different camera motions, figure 2.17.

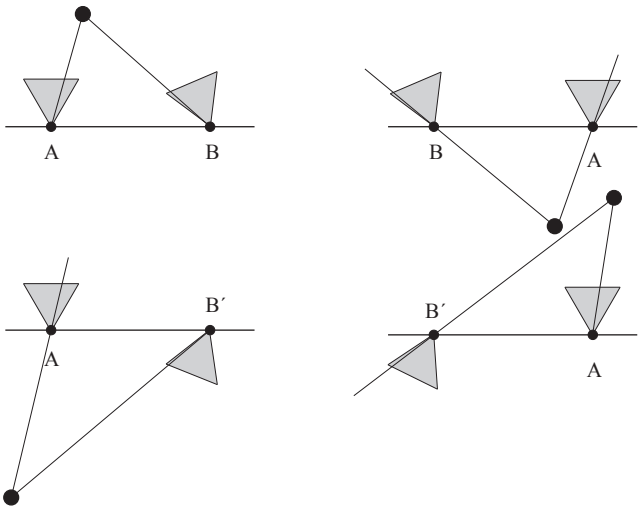


Figure 2.17: The four different combinations of camera rotation and translation.

To select the correct combination the assumptions are made that the translation correlates to the direction of the moving car and that the adjacent camera rotations are small. Additionally the obtained translation is arbitrarily in scale which means that the norm of the translation is 1. Hence the translation has to be multiplied with the total distance travelled in order to get the correct scale. The translation and rotation between the two cameras are horizontally concatenated in a matrix denoted extrinsic matrix:

$$[\mathbf{R}|\mathbf{t}] = \mathbf{P} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 v/\Delta t \\ r_{21} & r_{22} & r_{23} & t_2 v/\Delta t \\ r_{31} & r_{32} & r_{33} & t_3 v/\Delta t \end{bmatrix}. \tag{2.34}$$

2.5 Linear triangulation

Linear Triangulation is a linear algorithm suitable for mapping two data sets given a certain number of corresponding data points between the sets. It is based on the fact that an arbitrary point can be expressed as following where $\mathbf{P}$ denotes the extrinsic matrix of the camera and the coordinates are written as homogeneous coordinates, $\mathbf{x} = \lambda(x, y, 1)^T$, where λ is an unknown scale factor [33]

$$\mathbf{x} = \mathbf{PX}, \tag{2.35}$$

$$\mathbf{x}' = \mathbf{P}'\mathbf{X}. \tag{2.36}$$

The homogeneous scale factor is eliminated by a cross-product to give three equations for each image point visible in more than one of the cameras in the system. As an example the equation derived for a point in the first image would be given as $\mathbf{x}(\mathbf{PX}) = 0$. When expanded and the scale factor is substituted the following equations are obtatined [25]:

$$x(\mathbf{p}^{3T}\mathbf{X}) - (\mathbf{p}^{1T}\mathbf{X}) = 0, \tag{2.37}$$

$$y(\mathbf{p}^{3T}\mathbf{X}) - (\mathbf{p}^{2T}\mathbf{X}) = 0, \tag{2.38}$$

where each row in the extrinsic matrix is renamed for simplicity:

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} = \begin{bmatrix} \mathbf{p}^{1T} \\ \mathbf{p}^{2T} \\ \mathbf{p}^{2T} \end{bmatrix}. \tag{2.39}$$

The result is four linear equations:

$$\begin{bmatrix} x\mathbf{p}^{3T} - \mathbf{p}^{1T} \\ y\mathbf{p}^{3T} - \mathbf{p}^{2T} \end{bmatrix} \mathbf{X} = 0, \tag{2.40}$$

$$\begin{bmatrix} x'\mathbf{p}'^{3T} - \mathbf{p}'^{1T} \\ y'\mathbf{p}'^{3T} - \mathbf{p}'^{2T} \end{bmatrix} \mathbf{X} = 0, \tag{2.41}$$

which combined creates a 4x4 matrix with linear equations:

$$\mathbf{A}\mathbf{X} = 0, \tag{2.42}$$

$$\mathbf{A} = \begin{bmatrix} x\mathbf{p}^{3T} - \mathbf{p}^{1T} \\ y\mathbf{p}^{3T} - \mathbf{p}^{2T} \\ x'\mathbf{p}'^{3T} - \mathbf{p}'^{1T} \\ y'\mathbf{p}'^{3T} - \mathbf{p}'^{2T} \end{bmatrix}. \tag{2.43}$$

The four linear equations can be solved using several different approaches where one of them are the Linear Least Square approach. By neglecting the scale factor the number of unknown variables are reduced to three. The linear equation system can then be solved using singular value decomposition when the intrinsic and extrinsic matrices are known [33].

2.6 Reprojection error

The measured points $\mathbf{x}$ and $\mathbf{x}'$ are assumed to contain some inconsistencies in the sense of exact location between frames and scales. This lack of exact precision of the keypoints combined with some incorrect matches will result in a fundamental matrix drifting in accuracy. In other words equation 2.44 and 2.45 and will not be completely satisfied [25]

$$\mathbf{x} = \mathbf{P}\mathbf{X}, \tag{2.44}$$

$$\mathbf{x}' = \mathbf{P}'\mathbf{X}. \tag{2.45}$$

The accuracy can be numerically described by a so called reprojection error. The reprojection error is a geometric error corresponding to the euclidean distance between a keypoint and its reprojected 3D point.

2.7 Postprocessing

When the point clouds are created, there are often some points that are not correct due to inaccurate triangulation, wrong matches or trying to match points that do not belong to the stationary scenery. With the methods found in this project it is extremely difficult to get a high density with no erroneous points. Therefore some postprocessing can be done to reduce the impact of such errors which will briefly be described in this section.

2.7.1 Radial filtering

Radial filtering is a method for removing points that have to week connection to its nearest neighbours. To do this the method will take a specific radius surrounding the point in 3D space and check the number of points within that radius. If the number of points is within a threshold the point is kept, otherwise it will be discarded as an outlier.

2.7.2 Statistical outliers

Another method to remove outliers is to search for statistical outliers. This process assumes that the points and their distance to a given number of k nearest neighbours is Gaussian distributed. Therefore to remove the statistical outliers the distribution is calculated and all points that fall outside a given deviation threshold is removed.

3
Segmentation Theory

Segmentation is an important first step in automated understanding of point clouds. With a sufficient segmentation it can be determined which points belong to the same surface or region. This is necessary for any further classification or further processing.
There are several different methods for clustering large point clouds, this section will cover some methods, their advantage and disadvantages. This will form the basis for the decisions described under method.

3.1 Plane fitting and Polynomial fitting

By assuming that the road surface is piecewise flat to some extent it can be represented by patches of planes. Each of these 2D patches is contained within a 3D volume which in turn belongs to the complete point cloud. To identify a plane in a volume one can use the previously mentioned method RANSAC. By knowing the model, which is the equation of a plane, and a threshold, the points that belong to the best plane in a volume of points can be derived. In a similar manner other models, like a polynomial model, can be used that might capture the characteristics of the road more accurately which will be implemented with the same approach as for the plane fitting.

3.2 Normals

The normal of a point is calculated by taking a number of points in the proximity often selected by either taking all points within a radius or taking the closest N number of points. With both methods the radius or number of neighbours N is important design parameters. Witch limit to use and the parameter selection depends highly on the density of the cloud and the smoothness of density. When the neighbours is selected the best plane fit for those points is calculated and the normal to that plane is the normal to the point.

3.3 Difference of normals

To find objects of a specific scale one is to calculate the Difference of Normal (DoN from now on). To calculate DoN two radiuses for normals calculation have to be given as input. The method creates a surface within the scales given by the two

radiuses, this results in two normals to the projected surfaces within both radiuses. If the point is a part of a larger smooth area then with the radius scales applied the DoN will be rather small. And if there is abrupt changes in area within the two scales applied the DoN will be small. An illustration can be viewed in 3.1, where r_1 and r_2 is the input radiuses that is given as a design parameter [12] [11]. This method is useful for point clouds where there are areas with small DoN and other areas with larger DoN, for specific scales

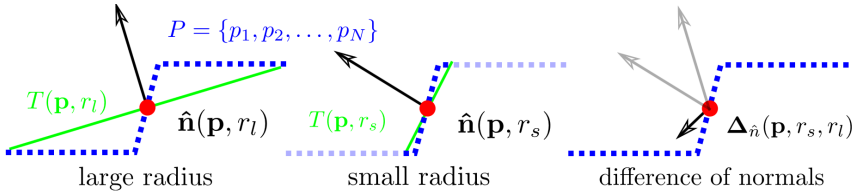


Figure 3.1: Illustration of the method used in calculating DoN [5].

$$\Delta_{\hat{n}}(p, r_1, r_2) = \frac{\hat{n}(p, r_1) - \hat{n}(p, r_2)}{2}. \tag{3.1}$$

Then the method can be used for detecting objects with a big DoN between the specified scales, if the surrounding points have a smaller DoN for the specified scales. This means that the input scales to the DoN algorithm need do differ depending on the objects you are trying to detect. According to [12] the optimal scales to use in the very dense LIDAR cloud evaluated on in the article was $r_s = 0.1$, $r_l = 0.4$ for pedestrians and $r_s = 0.4$, $r_l = 2.0$ for cars. This cloud is rather different from the SfM cloud, SfM gives more noise and more inconsistency in density. Therefore these parameters are not necessarily useful in the SfM case. The article also concludes that a ratio between the scales of approximately 10 times is often preferred. The complete procedure is given in Algorithm 1.

Algorithm 1 DoN

```
procedure DoN calculation
Remove all points classified as road
Extract a KDTree
    for All points do
        Calculate DoN for two scales
    end for
Filter point cloud with aspect to the DoN
Cluster the resluting points
end procedure
```

3.3.1 Euclidean clustering

After the point cloud has been processed and thresholded by DoN, some segments where the DoN was over the threshold needs to be formed.The points in those segments need to be connected into clusters that are assumed to be of the same object.

A simple way to do this (suggested in [12]) is Euclidean clustering, this method simply minimizes the Euclidean distance in the clusters according to equation 3.2 from [34]

$$dist(p,q) = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2 + (p_z - q_z)^2}. \tag{3.2}$$

The cluster method needs tree inputs, minimum cluster size, maximum cluster size and the radius condition that will stop cluster growth when no points are within the allowed radius.

3.4 Distance

For more specialised segmentation a distance can be used as a limit. Either from the expected centre of the cluster or, if using seeded segmentation the distance to the seed can be used. Furthermore the distance can be tweaked to fit certain needs, often euclidean distance is not the best parameter to limit or use as a cost parameter. Instead the change relative to an assumed surface or vector can be used to penalise some points, this is useful when the segment is 2D or 1D. The equation for euclidean distance can be viewed in equation 3.2.

3.5 Seeded(bottom up) and top down

Seeded segmentation is a strong bottom up method that needs an input where the segmentation is started. Then points are added to cluster, in many cases with region growing. The strength of this is that the algorithm can assume a point belongs to a type and then expand the points assumed to be of the same type. One of the disadvantages is that it might be hard to acquire such a point.
Top down segmentation usually has no seed as input but is starting with calculating one or several features for all the points. Then the points are connected to a segment with some type of filtering.

3.6 Region growing

Region growing is useful for expanding clusters, especially clusters started with a seed. The concept is best explained in algorithm 2 [35] [12]:
The interesting part in the algorithm used for region growing is the condition on which the decision to add a point is performed. There is a wide array of factors that can be used. Commonly distance is used in combination with one or several other feature relationships between the point evaluated in the segment and the point evaluated to be added to the segment. The problematic part with region growing is to find parameters that will perform a fast growth to the intended limit without going beyond the intended limit.

Algorithm 2 Region growing

```
procedure REGION GROWING
Add seed to Segment
    for all points in Segment do
        for all points in point cloud do
            if Point in segment and in point cloud within Limitation then
                add point in point cloud to Segment
            end if
        end for
    end for
end procedure
```

4
Method and implementation

To test and verify the algorithms for visual odometry and segmentation Matlab and C++ were used to construct the algorithms. The data used was mainly data collected by Volvo Cars active safety camera with additional information stored from the CAN bus. Two external libraries, Point Cloud Library (PCL) and Open Source Computer Vision (OpenCV) were used during the project. Both libraries are well documented and contain several already constructed methods related to image processing and point cloud processing.

4.1 Software implementation

Matlab seemed to be lacking the desired efficiency doing the computations, which tended to be rather heavy, and also the absence of a versatile visualizer of point clouds resulted in the change to C++. Another factor speaking for the advantage of using C++ is that the PCL library and other functions used are not limited by licences. PCL uses a BSD Licence [36][37] making it free to use for research and commercial purposes.

4.1.1 PCL Library

The PCL Library is composed of a vast amount of functions. This is a standalone open project library intended for the manipulation of point clouds. There are hundreds of contributors who contributed with code for a wide variety of applications [38] which makes it suitable for 3D altering.

4.1.2 OpenCV Library

Another library that has proven useful during the project is OpenCV. The principal for OpenCV is similar to that of PCL, the main difference is that OpenCV has focus on real-time computer vision mainly in two dimensions. This is useful for feature detection, feature matching, estimating fundamental matrix, triangulation and motion estimation [39]. As PCL, OpenCV is also realised under the BSD licence.

4.2 Method for visual odometry and structure from motion

To create a good 3D structure several steps had to be undertaken. This section will explain the process from video to a 3D structure.

4.2.1 Pre-processing

Since this thesis was executed at Volvo and intended for use on the existing video logs. The first step was to extract working video data that could be used for visual odometry. There are two main problems with the video frames in the condition they are stored, images is collected as RCCC and they have radial distortion.
To handle these problems a Matlab script was created in order to read the frames with the correct codex, filter them with the RCCC and remap the image information according to the radial distortion. Then all the frames where saved in their pre-processed format.

4.2.2 Feature point localization

To evaluate the use of feature points, functions from OpenCV were used. The features compared were:

- SIFT
- SURF
- BRISK.

The first step to evaluate the feature points is a visual inspection of the extracted points in a single frame. In this inspection factors such as the number of points and the density in different regions are assessed. In this project an important factor is to have features in all regions of the frame that is not located on a moving object or represented by sky. If all areas are not represented the result will be sections in the 3D representation with sparse or missing areas. Features located on the road are especially important in order to identify the road geometry which also is a tricky area since it is mainly homogeneous. To get different spread of feature points the thresholds are varied and the result before and after matching is evaluated.

4.2.3 Feature matching

When features are detected and the descriptors are derived they have to be matched between consecutive frames. The number of correct matches will be evaluated with respect to the time taken using both the FLANN matcher and the BF matcher for the different feature types.

4.2.4 Fundamental matrix and motion

As mentioned under the theory section the 8-point algorithm is used to calculate the fundamental matrix in combination with RANSAC utilizing epipolar constraint, where the entries to the 8-point algorithm are two sets of normalized keypoints. The fundamental matrix is used together with the intrinsic matrix to express the essential matrix used to derive the motion. As mentioned there are four possible combinations of the extrinsic matrix extracted from the essential matrix where the correct combination can be seen in figure 4.1.

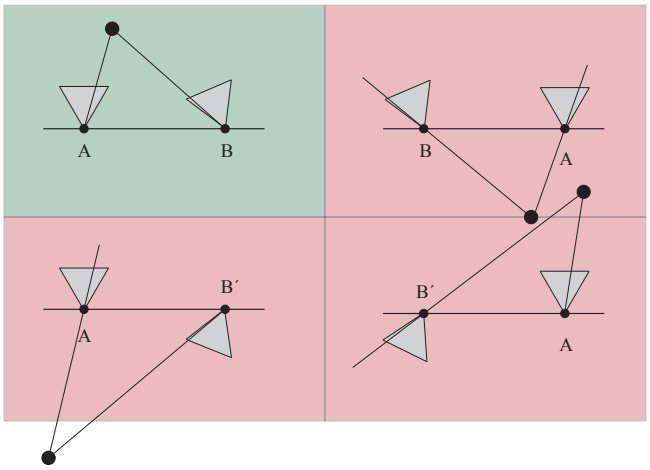


Figure 4.1: The four different solutions where the green is the correct one.

Below is a mock-up of the procedure selecting the correct combination and deriving the motion:

Algorithm 3 Motion algorithm

Input: Normalized points, RANSAC threshold
Output: Extrinsic matrix, index outliers
for 8 random points **do**
  Calculate fundamental matrix
  if Desired level of confidence achieved **then**
    Break
  end if
end for
Express essential matrix
Decompose essential matrix
if sign($t_{1,z}$) equals sign(*velocity*) **then**
  t_1 correct translation
else
  t_2 correct translation
end if
if trace(R) > trace($R2$) **then**
  R_1 correct rotation
else
  R_2 correct rotation
end if

The coordinate system is set up in a way that forward points in z-direction, yaw acts around z-axis and pitch around y-axis. When the transformation between two consecutive camera poses are determined it has to be related to the total transformation according to *(the matrices are padded with [0 0 0 1] to make them 4x4)*:

$$[\mathbf{R}|\mathbf{t}]_{tot} = [\mathbf{R}|\mathbf{t}]_{old}[\mathbf{R}|\mathbf{t}]_{new}. \tag{4.1}$$

4.2.5 Triangulation and post processing

When the transformation between camera poses are known the points can be back projected to the 3D point originally giving raise to the 2D point. This is done using the linear triangulation method described in section 2.5. The mock-up of this algorithm can be found below:

Algorithm 4 Linear triangulation

Input: Intrinsic matrix, extrinsic matrix, matches
Output: Back projected 3D points
for All matches **do**
  Convert matches to homogeneous coordinates
  Construct the four equations relating 2D to 3D
  Decompose and select the smallest eigenvector of the matrix **U**
  Divide 3D homogeneous coordinates with scale factor $\mathbf{U}(end, end)$
end for

In addition to the coordinates of the keypoints its intensity information is also stored. The intensity values are simply the value of the pixel containing the keypoint.
After the point cloud is created radial filtering and statistical outliers filtering described in 2.7 can be applied. For segmentation both those filtering algorithms where used for creating smother clouds and removing noise that wold cause a problem for the segmentation.

4.2.6 Reprojection error

The reprojection error is in this thesis used to validate the estimated motion and the precision of the derived 3D points. By using the transformation of a frame where a 3D point has been successfully derived the 3D point is transformed as if the image frame was placed in origin using $\mathbf{P_{back}}$, equation 4.2 *(the matrices are padded with [0 0 0 1] to make them 4x4)*. When done, the point can be projected onto a plane simply using normalized image coordinates. This point can then be related to the image plane when knowing the intrinsics of the camera

$$\mathbf{PP}_{back} = eye(4). \tag{4.2}$$

A mock-up of the reprojection procedure is described below:

Algorithm 5 Reprojection error

Input: Intrinsic matrix, extrinsic matrix, 3D points
Output: Reprojected points
Compute $\mathbf{P_{back}}$ for frame X
for All 3D points derived from frame X **do**
    Transform 3D point
    Project 3D point to plane
    Relate to image plane
    Compute euclidean distance between keypoint and projected point
end for

4.3 Method for road segmentation

To segment the road some assumptions had to be made. Two important assumptions made are that the road is a smooth surface with edges, and that the point closest to an imagined point is as far below the camera position as the camera is above ground is a part of the road. Since the camera can not see below itself it can not build a cloud under its current position. But after the first initial frames the car with camera will be driving over a segment that has points that should be classified as road.

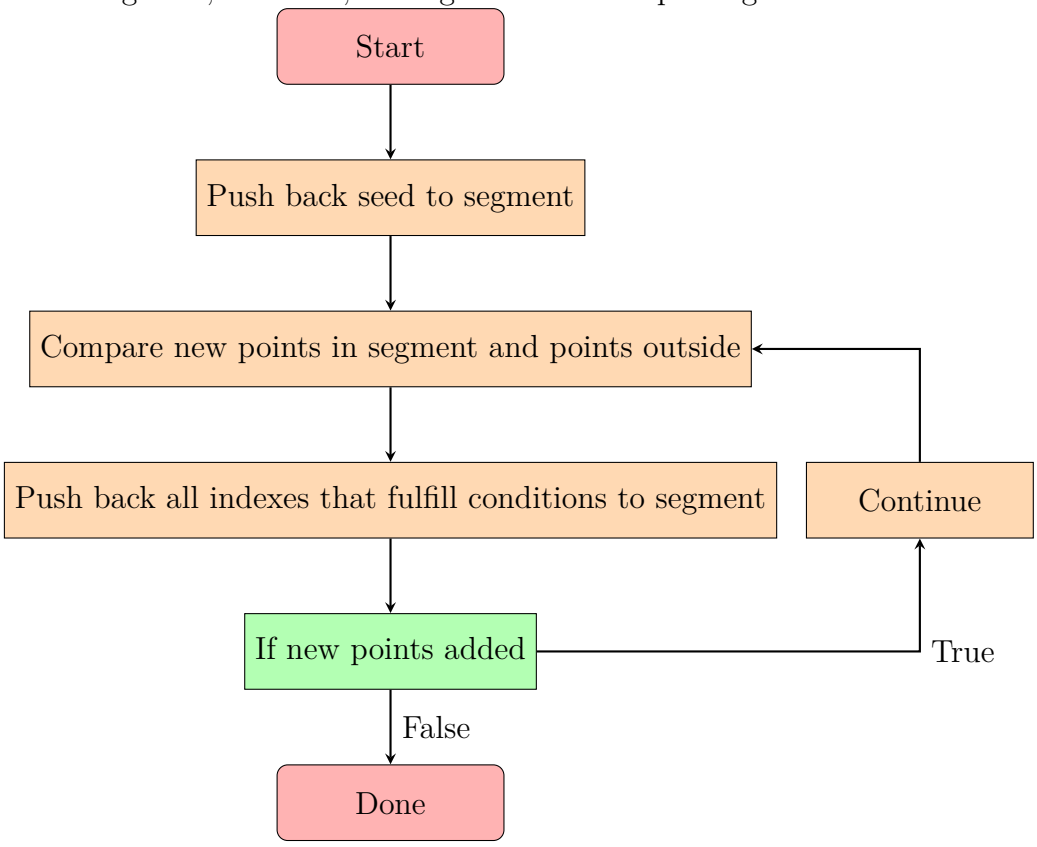
4.3.1 Seeded region growing

Since the road is expected to have a smooth surface with abrupt change when the road is ending a region growing algorithm might be able to manage the segmentation.

The first key ingredient is a correct seed that will result in growth of the segment. Here it is possible to use the fact that the car will be passing over the road after the initial frames. There a seed can be placed under the camera at ground level with predetermined intervals with aspect to the distance travelled. One aspect that might help to determine an appropriate interval is the wideness of the road, the intervals if distance travelled when placing a new seed could be useful to have close to the wideness of the road. The parameters that will stop or allow a segment to grow with a specific point is:

- Normal change compared to all the points in the segment
- Normal change compared from the original seed, this is a higher limit to stop strange behaviour
- Distance from the horizontal plane compared to all the points in the cloud
- Euclidian distance from the seed this limit is also higher and is mainly to stop strange behaviour
- If all is within a specified limit the algorithm will add the point to the segment

If all is within a specified limit the algorithm will add the point to the segment. The algorithm will then go to the next point in the segment and check if there is a point that should be added with aspect to this point. After all the points in the segment have been looped trough, the algorithm will check if there was any new points added to the segment, and if so, the algorithm will loop trough them too.



4.3.2 Plane fitting

RANSAC can be used to iteratively fit any type of data to a predefined model which in this case will be the model of a plane in order to capture the points assumed to belong to the road. The method can however not be applied to the whole point cloud at once since it will contain several different planes and shapes. Hence, the point cloud will be divided into sections where a plane will be fitted to each of these section. The selection of sections is done by first performing a translation making the camera pose the origin of the point cloud followed by a rotation around the origin. Secondly a volume is selected around the origin with a shape of an ellipsoid followed by the inverse rotation and translation returning the point cloud to its initial state. The ellipsoid now has its origin at the camera position rotated to face the direction of the camera, see figure 4.2.

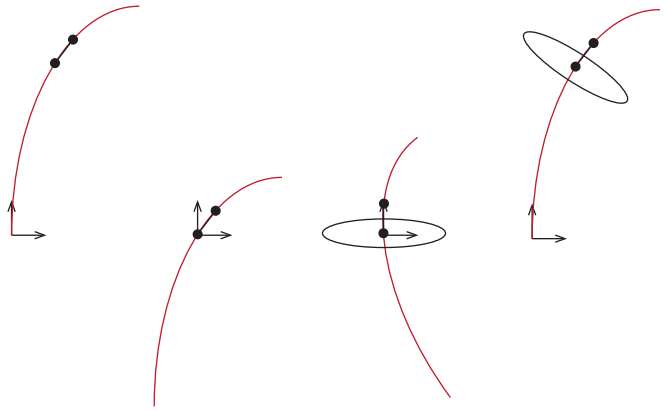


Figure 4.2: The illustration is seen from above. The black dots are two consecutive camera positions, the red line corresponds to the trajectory of the car and ellipses represent the selected ellipsoid.

When the section is selected and a threshold for RANSAC is set the inliers can be extracted and the coefficients in the planar equation can be obtained. A mock-up of the procedure is described below:

Algorithm 6 Plane fitting

Input: Point cloud, camera locations
Output: Plane
Translate point cloud -(camera position)
Calculate yaw angle between two frames relative the z-axis
Rotate point cloud yaw degrees around origin
Extract ellipsoid
Rotate points -(yaw angle)
Translate back to original location
for Select 3 points in ellipsoid **do**
    Derive coefficients and compute inliers
    if Desired level of confidence achieved **then**
        Break
    end if
end for
Repeat when reached X meters

An exact plane rarely is the actual shape of the road, concave curving, convex curving and slopes are expected to influence the shape. Since these geometries cant be captured by the equation of a plane a more accurate representation is hoped to be achieved by instead using a second order polynomial surface.
The extraction of the polynomial surface is based on the same principal as the plane fitting, however, instead of a planar model a polynomial of second order is chosen and by using RANSAC the parameters of the mathematical model can be estimated. The pros and cons will further be weighted under section 5.2.2.

4.3.3 Verification

To verify the segmentation of drivable area we manually created a ground truth according to 4.3. Then point clouds are created from the original area but a feature is added from the ground truth image to mark the points as road or not road in the ground truth cloud. One such cloud can be seen in 4.4. After the cload is segmented with the segmentation algorithm for road, true and false positive and negative is calculated. From that is it possible to calculate sensitivity and specificity. In this specific case a lot of the sidewalk is unfortunately segmented as part of the road. We created ground truth and calculated the sensitivity and specificity for two different scenarios.

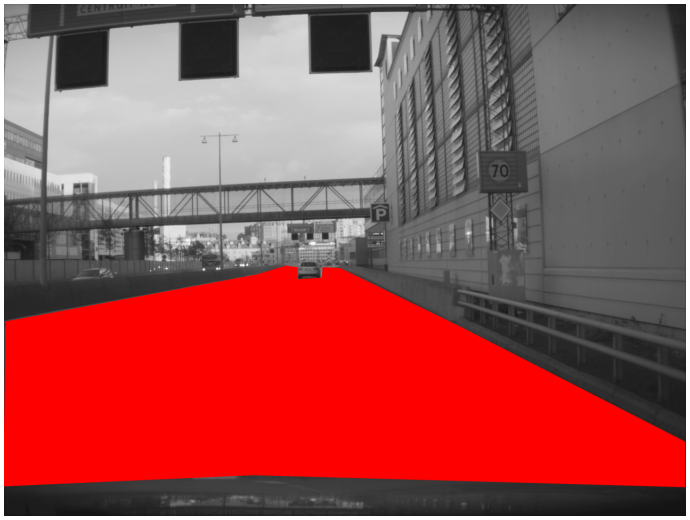


Figure 4.3: Image of case 1 with drivable area marked in red.

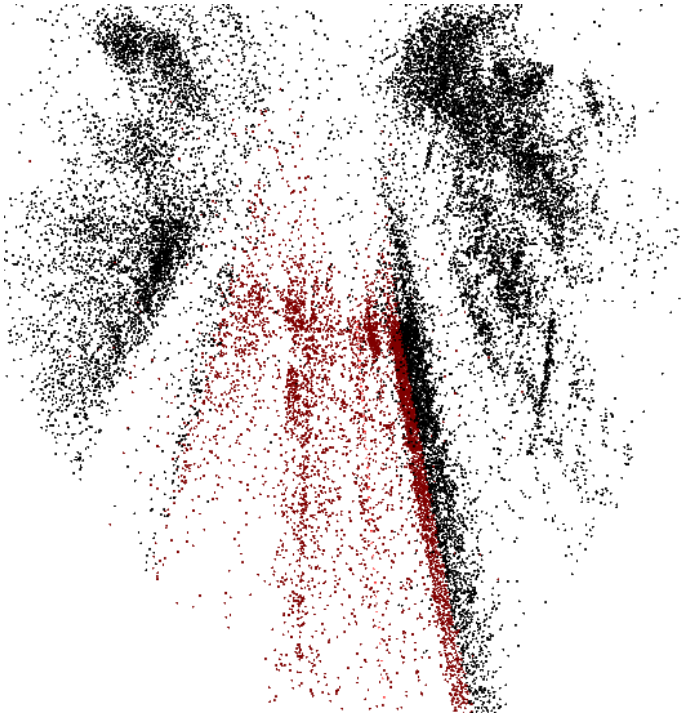


Figure 4.4: Point cloud with read points marked as drivable area and black points is not drivable area.

4.4 Potential objects

Segmenting possible objects needs another approach compared to segmenting the road, first of all it is more complicated to obtain a correct seed when there is no known position for each segment. The possible objects are also not smooth but rather fast changing. When segmenting possible objects the important thing is to

get a segment that can later be classified with another algorithm. Therefore the object segmentation will not know the type of object it is classifying.
The first step in the segmentation of possible objects is to calculate a feature which will be strong in the objects being searched for and weak otherwise. For this the DoN feature was selected. The DoN will be strongly depending on the characteristics of the different objects in relation to the radius for the different scales. This means that to cover all objects that should be classified there is a need to run the segmentation process with different DoN radius to segment different types of objects. When DoN parameters are tuned for things such as road barriers the road will get a low value since the change in road is not within the same scale. To remove points with weak DoN parameter in the interesting scales a threshholding algorithm is implemented. When the points have been filtered with a threshold after the DoN parameter, the remaining points need to be organised in clusters for any further processing to be done. This can be achieved with Euclidian clustering. The parameters needed in Euclidian clustering are strongly dependent on the density of the point cloud and therefore need to be tuned for different types of point clouds. A pseudo code illustration of this processes can be weaved in Algorithm 7

Algorithm 7 Objects segmentation

```
Remove all points classified as road
Extract a KDTree
for All points do
    calculate DoN for two scales
end for
Filter point cloud with aspect to the DoN
cluster the resluting points
```

Some of the parameters that are dependent on the object wanted to classify:

- Small radius scale
- Large radius scale
- Threshold for DoN
- Segmentation radius
- Min cluster size
- Max cluster size.

Most of these parameters are possible to guess or assume sufficient values on. But to fully test and reach the best potential some empirical studies with a trial and error method had to be applied. Often the assumed value performed correct but with slight changes the result could be improved. A correct start in the search for parameters in our application was [12] they suggest Small radius of 0.4m and large radius of 2m for cars. Since cars are not our focus and we have another type of cloud these values will not be optimal for us.

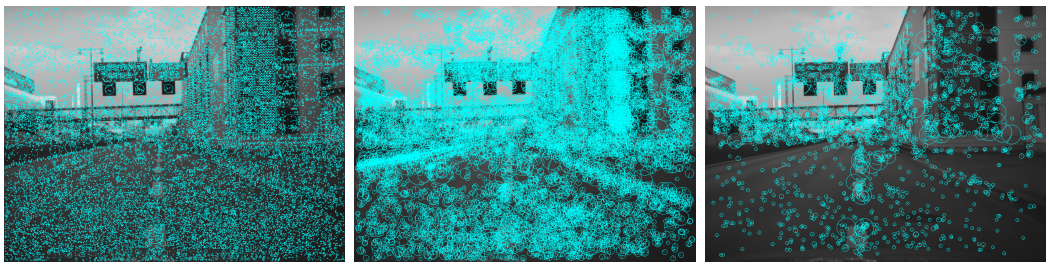
5
Results

This section will present the result after applying the implementing algorithms on the data provided by Volvo Cars.

5.1 Structure from motion

5.1.1 Feature and matcher selection

In order to compare the different feature types the parameters have been chosen in a way that gives a similar spread of points between the different methods with an additional requirement of having several features detected on the rather homogeneous road surface. The keypoints on the road are expected to be difficult to match but are required in order to later get a representation of the road plane in 3D. It can clearly be depicted by studying figure 5.1 and the corresponding table 5.2 that the number of features detected is significantly different. Even after lowering the thresholds of the SURF and BRISK they are unable to be competitors to SIFT with respect to the number of features detected.



(a) **(b)** **(c)**

Figure 5.1: (a) SIFT features with contrast threshold of 0.004 and edge threshold of 20. (b) SURF features with a threshold of 10. (c) BRISK features with a threshold of 5.

The two matchers used in this test are the BF matcher and the FLANN matcher and are compared regarding accuracy and computation time. The method to separate outliers from inliers is as mentioned earlier RANSAC which checks if the point lays on the epipolar line with an allowed deviation of 0.0005, bare in mind this is the distance from an epipolar line to a normalized point. The calculation is done not only for the best match but also for the second best in order to compare the scores between the best and the second best match. By comparing the number of features

kept after filtering and the corresponding computation time in table 5.2 it is clear that BRSIK is not a suitable choice due to the limited number of detected points and their poor matching. SURF is noticeably faster than SIFT with proportionally the same number of inliers. However, since this application is not time sensitive SIFT will be used since the number of inliers is more than tripled compared to the alternatives disregarding matching method. Comparing the different matching methods the number of kept matches are relatively unchanged whereas the time is significantly reduced using FLANN which qualifies it as the most adequate matcher for this purpose.

	SIFT	SURF	BRISK
Time detector (ms)	673	155	310
Time descriptor (ms)	1657	259	51
Nr. of features	22565	8304	2363
BF matcher			
Time matching (ms)	4402	518	52
Nr. of inliers	1631	520	47
FLANN matcher			
Time matching (ms)	777	259	5
Nr. of inliers	1867	426	31

Figure 5.2: Comparison between features considering time and accuracy. The result is obtained taking a mean over several frames.

5.1.2 Sky estimation

By applying the algorithm described in section 2.3.3 the sky can be successfully identified in the majority of the tried cases and the resulting binary mask can be applied in the process of feature detection in order to neglect the area assumed to be the sky. The accuracy of the algorithm can be seen in figure 5.3 where only visual verifications have been done. The estimation of sky has however proven to be unnecessary in excess to filtration of points using RANSAC where in almost all cases these keypoints are discarded.

(a) **(b)**

Figure 5.3: The darker parts in the images is classified as sky and can be used to reject potential feature points.

5.1.3 Fundamental matrix

After the SIFT descriptors are matched using FLANN and before any filtering is done it is simply impossible to visually separate the inliers from outliers which can be seen in figure 5.4b. After the matches with a distance ratio greater than 0.8, Lowe's criterion, are rejected as well as the matches shorter than 4 pixels several outliers have been successfully removed which can be depicted in figure 5.4b.

(a) **(b)**

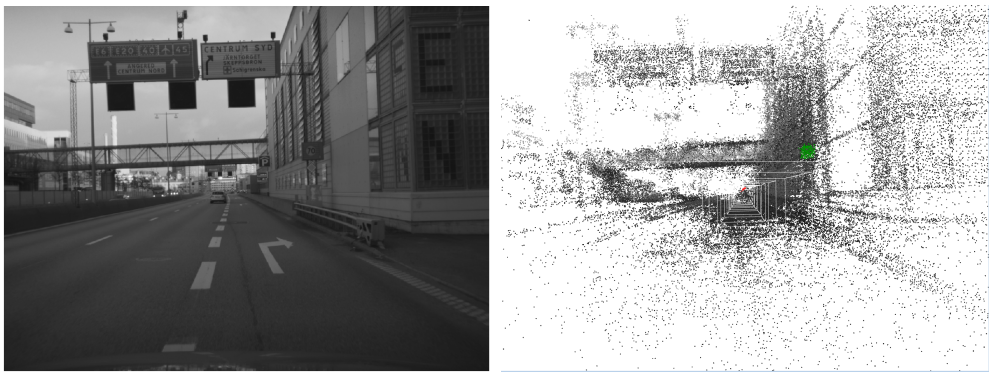
Figure 5.4: (a) Result after matching all points. (b) Result after removing matches that does not fulfille Lowe's criterion.

By estimating the fundamental matrix using RANSAC based on the epipolar constraint and discarding all the points that deviate more than a set threshold of 0.0005 the result can be visually verified by the fact that the majority of the outliers can be identified and removed. The remaining matches can be depicted in figure 5.7. The same figure illustrates how nearly all points located on the moving cars have been rejected since those points are moving in a direction that does not agree with the majority of points.

Figure 5.5: Matches that fulfille both Lowe's criterion and epipolar constraint.

5.1.4 Triangulation and reprojection error

The triangulated points with corresponding intensity values can be depicted in figure 5.6a. By visually comparing the point cloud to the video feed the accuracy of the SfM can be verified but only to some extent.



(a) **(b)**

Figure 5.6: (a) Rectified 2D image. (b) Triangulated points with intensity values based on the keypoints.

This is however a very basic validation of the methods used which is why the reprojection error was introduced to quantify the accuracy. The mean reprojection error for sequences of 200 frames where in the span of 1-3 pixels. The reprojection can be depicted in figure 5.7 were the white points correspond to the keypoints and the black points to the reprojected 3D points.

Figure 5.7: Reprojection error visualized for points that have been seen in four consecutive frames.

5.2 Segmentation

In this section the results of the implemented segmentation algorithms will be presented. Figures 5.8 and 5.9 show an overview of two segments that have been segmented, the road is dark yellow and all other colors are different segments segmented with the DoN algorithm described in theory and method.
There are some general limitations that will cause a problem for all segmentation, like the field of view of the camera. These limitations are more thoroughly described in discussion.
The parameters used for segmentation are explained here.

- **Road segmentation**
- Delta normal: The change in normal from the compared points
- Delta xyz: Euclidean distance
- Delta y: Change compared to the horizontal plane
- Delta norm difference: change in normal compared to the seed
- **DoN segmentation**
- Small Radius: Small radius in the DoN calculation
- Large radius: Large radius in the DoN calculation
- DoN treshhold: Threshold for the removal of low DoN
- Segmentation radius: Radius for the Euclidean clustering

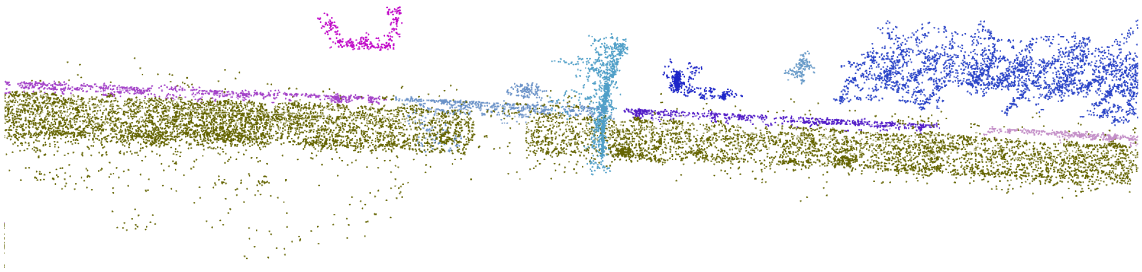


Figure 5.8: Overview of a segmentation from above, segmentation is done with the following road parameters: delta normal: 0.3 delta xyz: 4 delta y: 0.3 delta norm difference: 0.15
and the folowing DoN paramters: small Radius: 0.3 large radius: 2 DoN treshhold: 0.15 segmentation radius: 0.6

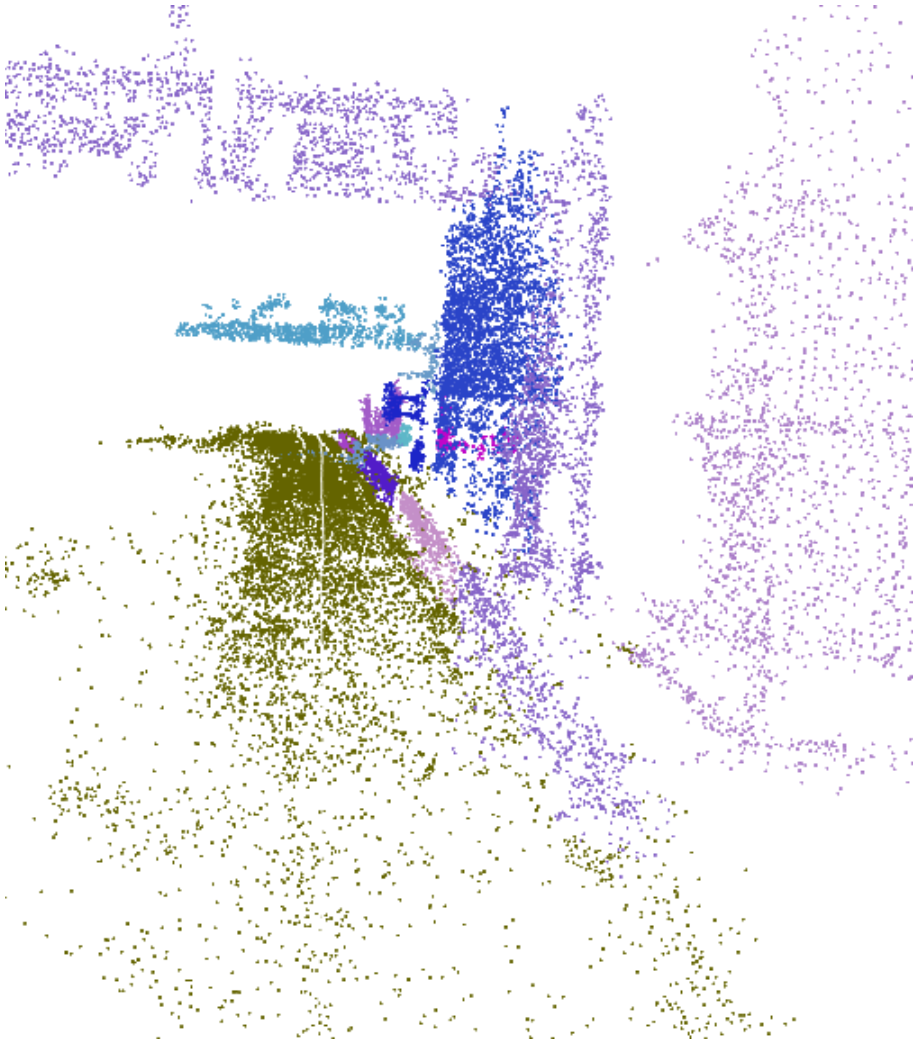


Figure 5.9: Overview of a segmentation from the driving direction, segmentation is done with the following road parameters: delta normal: 0.3 delta xyz: 4 delta y: 0.3 delta norm difference: 0.15
and the folowing DoN paramters: small Radius: 0.3 large radius: 2 DoN treshhold: 0.15 segmentation radius: 0.6

5.2.1 Region growing road segmentation

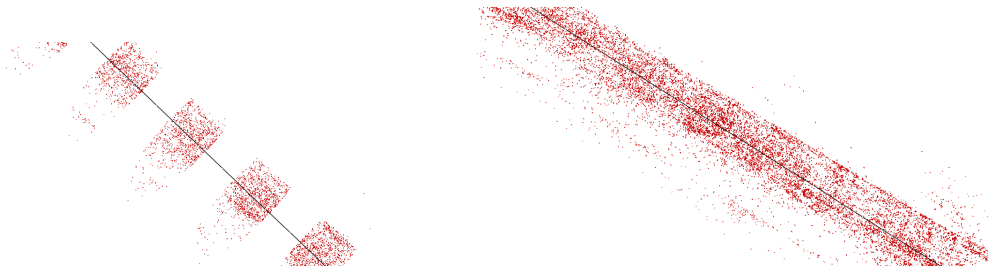
The seeded region growing segmentation with normals and distances limiting growth can be viewed in the overview images 5.8 and 5.9. The section of road where the data is collected fulfil the limitation, mainly that there is a strong geometric change at the edge of the road.
The sensitivity and specificity can be seen in 5.1, this is for case 1 where the edge is strong and case 2 where one of the edges is week.

	Sensitivity	specificity
Case 1	92%	97%
Case 2	81%	79%

Table 5.1: Table of sensitivity and specificity for two cases of road segmentation

5.2.2 Plane and polynomial fitting

The theory behind plane segmentation is as described in earlier chapters a straightforward method using RANSAC. By knowing the trajectory of the camera the approximated location of the road is simply the camera height below the trajectory. By extracting all points located in a specified volume relative to the cameras directions, ellipsoid used as described, the best planes are fitted. The distance between these volumes is set to 12 meters in order to visualize the concept, figure 5.10.



(a) **(b)**

Figure 5.10: (a) Ellipsoids extracted after a distance of 12 meters. (b) overlapping patches covering the whole road.

To select the correct patches following the camera trajectory the strategy described earlier is applied and the result can be depicted in figure 5.11.

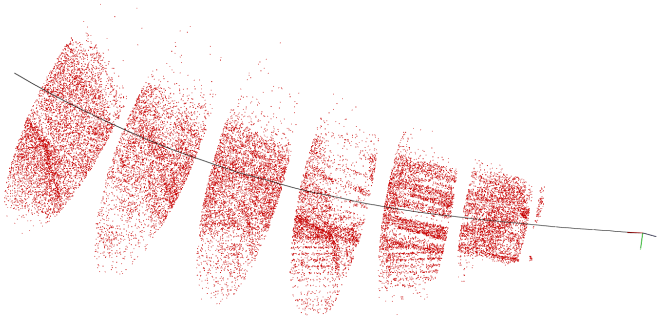


Figure 5.11: Illustration of the rotation of patches.

The estimated road using plane fitting can be seen in figure 5.12 where the estimated planes are highlighted in the point cloud.

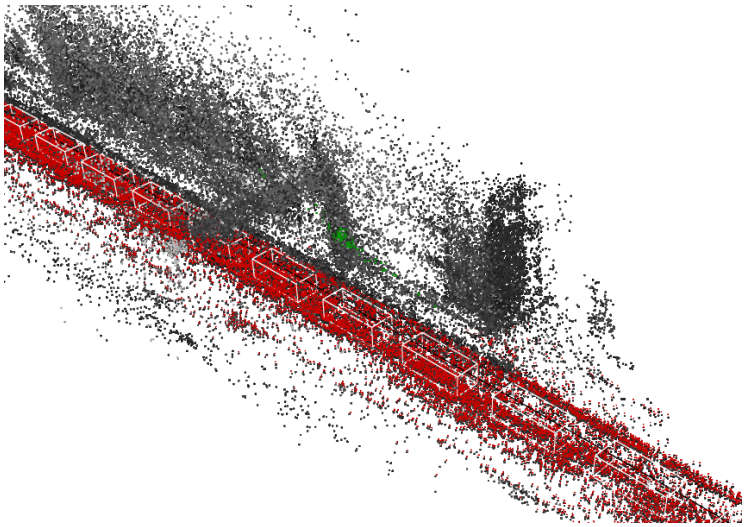


Figure 5.12: The result after using overlapping patches highlighted in the full point cloud.

The assumption of a piecewise almost completely flat road surface is of course a rough estimation which does not capture rapid slopes or bumps. One approach tested previously mentioned is increasing the threshold of the plane fitting and projecting these points to a second order polynomial. The result can be depicted in figure 5.13. It is not easy to visualize but it captures curvatures of the road fairly well but at the expense of points close to a sharp edge like a barrier will smooth the corner which can be depicted in the right side of the figure.

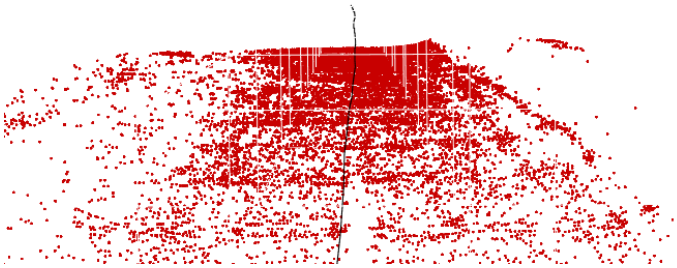


Figure 5.13: Overlapping patches resampled to a polynomial representation.

5.2.3 Potential objects

Since there is no classification implementation the algorithm for segmentation of potential objects has been visually inspected. This evaluation has mainly been done on road barriers since it is a possible end application that is common and rather easy to visually evaluate. It proved useful to remove the road before the DoN segmentation. One example of a segmented (but not classified) barrier is in figure 5.14.

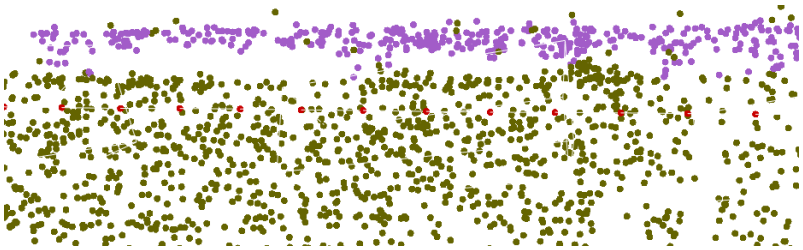


Figure 5.14: Close view of a barrier, segmentation is done with the following road parameters: delta normal: 0.3 delta xyz: 4.5 delta y: 0.3 delta norm difference: 0.15 and the folowing DoN paramters: small Radius: 0.3 large radius: 2 DoN treshhold: 0.15 segmentation radius: 0.6

It is important to note that the segmentation parameters as described in section 5.4 are selected to optimize functionality with aspect to road barriers.

6
Discussion

During the project expected and unexpected results were reached and limitations with the chosen techniques discovered. This section will discuss those aspects.

6.1 Limitations

It is important to understand the limitation of SfM with a single front mounted camera. This section will describe some of the limitations discovered during the project. Some are absolute and will not be possible to work around with current sensor data. Other can be possible to solve, but not solely with the methods described in this thesis.
If an object's feature points are moving compared to the stationary feature points that are basis for triangulation, there is no way of doing a correct triangulation. This is because the ego motion estimation and triangulation are filtering away all outliers with RANSAC. Therefore a correct working algorithm will remove feature points not moving according to the estimated motion, hence if working correctly the algorithm will remove the moving objects completely from the cloud. An extreme case that would cause large problems is if the visual flow in the image is dominated by another object than the stationary ground, for example if more than half of the features are from the same large truck moving by and therefore the algorithm estimates motion compared to the large moving object and not the stationary reality.
If an area in the space that the algorithm is trying to reconstruct is not covered by the video for a few frames, suppose a sharp turn, it is not possible to reconstruct any 3D points from that area. Another scenario that will interfere with the density of the point cloud is if a big moving object is occluding the scenery. These are both events that can not be compensated for.
Currently this project has no good way of extracting colors in the image and applying it to the features extracted. An application where the color of the pixel in the centre of a feature is assumed to be the color of the feature has been constructed. The problem is illustrated in figure 6.1 where the features are located around changes in the image, but not necessarily on top of the object giving cause to the feature. In the image it is clear that the pole is giving cause to the feature, but the centre is located in the sky.

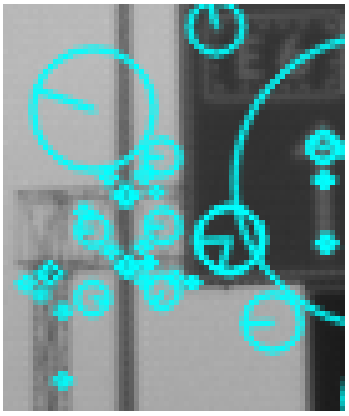


Figure 6.1: Example of feature centre not representing the object giving rise to the feature.

6.2 Structure from motion

The first step in continuing of the visual odometry should be verification. Since the data used lacked any positioning the accuracy of the computed motion is complicated to calculate and will not be studied more than visually within the limits of this thesis. The solution to this can be using external databases to get an idea of the accuracy. However, that will not be representative since the video resolution probably won't be the same. Additional focus on improvements should be on implementing bundle adjustment. Not only to increase the accuracy of the triangulated points but also to correct the motion. Other issues noticed are objects that are moving along with the estimated motion. In other words the direction of the matches is correct but not the length. This could be improved by implementing a clustering of matches to remove objects smaller than a threshold depending on the average from surrounding points, since the point is likely to be incorrect if the movement is likely to be faulty if movement is inconsistent. Using RANSAC to remove outliers based on the epipolar constraint is efficient but has one drawback. If two features are wrongly matched, just checking if the matched point lies on the epipolar line or not is not enough. It might in some rare cases lie on the epipolar line without being a correct match.

6.3 Segmentation

It is clear that we can reach a high sensitivity and specificity (above 90%) for some cases, one of the reasons for not classifying the last percentages right is that there are some error from the SfM Algorithms that places some points that have originated from the road in the image far from the expected road in the point cloud. In other cases the algorithm overgrows for example a small edge to the sidewalk, this will result in lower specificity.
Currently the segmentation algorithms need different parameters to work on different point clouds with various densities. Since there is no approach of automatically extracting correct parameters, the only way is to tune them manually.

The methods developed for segmentation using the appropriate parameters are promising, but since the segmentation algorithms are only tested on a very limited number of point clouds representing a very small part of possible scenarios it is difficult to know if they always are applicable.
The selection of seeded region growing segmentation is a strong method for segmenting the road using only the point cloud created from SfM. It uses both the strength in the knowledge that the camera is located above the road and that roads have a relatively constant geometry. One of the weaknesses is though that these assumptions can be violated by for example a high speed bump.
The region growing road segmentation is growing in a well defined manner and stopping in the right places. The current problem is that the parameters such as limits for distance and DoN is a real trade of between reaching out to all parts of the road and growing past the wanted areas. Hopefully this is also part of the problem with parameters not working equally in all cases, but it is possible that some kind of algorithm for dynamic limiting parameters depending on the properties of the point cloud have to be developed for a fully automatic implementation of the segmentation.
The DoN segmentation for objects shows some promising results, and the resulting segments might show to be a great base for classification. One of the main problems here similar to road segmentation is that the parameters needed for correct segmentation varies for different objects. It is also useful to remove the road before segmentation since it improves the segmentation.
The project also chose to go through with the DoN segmentation rather than the min cut segmentation since the DoN requires no seed to be placed. Placing a seed manually would be large amounts of work and not doable in the further applications. And the automatic seed algorithms for min cut did not appear to be either reliable or easy to implement.

6.3.1 LIDAR

A lot of the algorithms used in this project have been evaluated on LIDAR data within the frame of other projects. And one of the big questions when starting this project was if some of the information that is available in the LIDAR system can be extracted from video via the methods described in this project. It is important to note that some information that is present in the LIDAR setup is impossible to achieve due to the limitation described in this project. The more interesting question is if some useful and reliable information that LIDAR gives can be achieved without LIDAR. Therefore a comparison to the LIDAR system is somewhat relevant and here we can see that the work and complexity needed to achieve correct results is largely increased by removing the sensor data from the LIDAR. However it might be useful to use SfM to fuse LIDAR and video data, this is however outside the scope of this project.

6.4 Future work

To take this project further the first step would be to improve the SfM algorithms. A clear improvement would be to add bundle adjustment, the disadvantage would be that the algorithms would take considerable longer time to execute with bundle adjustment for back projection error. It is also clear that some smaller bugs is left in some parts of the PCL library used, if those bugs were eliminated by either the community behind that project or by further developing this project that would improve the performance.
If the SfM algorithms are improved with aspect to even density and accuracy it would improve the segmentation too, since slight noise or inconsistency is a problem for the segmentation. If this will not solve the problem with tuning parameters, algorithms for automatic finding the best parameters would be useful. During this thesis we encountered no literature on such dynamic parameter tuning and therefore have no good suggestion for such implementation. It would also be interesting to implement some kind of min cut algorithm described in theory, possible in combination with DoN segmentation for different types of objects. The combination of different methods for segmentation might prove useful, but there was no time in in this project to fully evaluate this possibility.
Preferable after the improvements mentioned above the next step would be to implement some kind of classification described in theory. The real test for the segmentation and SfM is to determent weather an accurate classification is possible with the segments created. But currently there is several areas that needs to improve before reaching that stage.

7
Conclusion

This project can conclude that based on experimental results it is possible to create 3D point clouds from the video logs collected by Volvo cars. Further it is possible to segment roads and some type of objects in this 3D environment. The SfM algorithms give satisfying low reprojection error. And the road segmentation has a high performance in some cases but further work is needed to handle all cases. Potential objects can be segmented but to evaluate accuracy there is a need for further evaluation. Therefore further work is needed to reach a higher level of accuracy in the segmentation and SfM.
There is important data in the LIDAR sensor that this projects experimental algorithms cannot replace by SfM methods, partly because of the limitation of moving objects and the field of view for the camera. From other areas the point cloud is of high density but with a non negligible degree of noise.

Bibliography

[1] G. Karanam, Interfacing red/clear sensors to adsp-bf609® blackfin processors (2013).
URL http://www.analog.com/static/imported-files/application_notes/EE358.pdf

[2] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, Speeded-up robust features (surf), Computer vision and image understanding 110 (3) (2008) 346–359.

[3] E. Rosten, T. Drummond, Machine learning for high-speed corner detection, in: Computer Vision–ECCV 2006, Springer, 2006, pp. 430–443.

[4] S. Leutenegger, M. Chli, R. Y. Siegwart, Brisk: Binary robust invariant scalable keypoints, in: Computer Vision (ICCV), 2011 IEEE International Conference on, IEEE, 2011, pp. 2548–2555.

[5] Difference of normals based segmentation (2015).
URL http://pointclouds.org/documentation/tutorials/don_segmentation.php

[6] V. Cars, Vision 2020 (2015).
URL http://web.origin.volvocars.com/intl/top/corporate-old/volvo-sustainability/safety/pages/vision-2020.aspx

[7] J. VILLYSSON, Robust tracking of dynamic objects in lidar point clouds, Master's thesis, Chalmers University of Technology, Sweden (2014).

[8] B. M. Kitt, J. Rehder, A. D. Chambers, M. Schonbein, H. Lategahn, S. Singh, Monocular visual odometry using a planar road model to solve scale ambiguity.

[9] R. Mur-Artal, J. Montiel, J. D. Tardos, Orb-slam: a versatile and accurate monocular slam system, arXiv preprint arXiv:1502.00956.

[10] J. Sturm, W. Burgard, D. Cremers, Evaluating egomotion and structure-from-motion approaches using the tum rgb-d benchmark, in: Proc. of the Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RJS IROS, 2012.

[11] T. Rabbani, F. van den Heuvel, G. Vosselmann, Segmentation of point clouds using smoothness constraint, International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 36 (5) (2006) 248–253.

[12] Y. Ioannou, B. Taati, R. Harrap, M. Greenspan, Difference of normals as a multi-scale operator in unorganized point clouds, in: 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on, IEEE, 2012, pp. 501–508.

[13] G. Karanam, Interfacing red/clear sensors to adsp-bf609® blackfin processors (2015).
URL http://pointclouds.org/documentation/tutorials/min_cut_segmentation.php

[14] A. Golovinskiy, T. Funkhouser, Min-cut based segmentation of point clouds, in: Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on, IEEE, 2009, pp. 39–46.

[15] B. Douillard, J. Underwood, N. Kuntz, V. Vlaskine, A. Quadros, P. Morton, A. Frenkel, On the segmentation of 3d lidar point clouds, in: Robotics and Automation (ICRA), 2011 IEEE International Conference on, IEEE, 2011, pp. 2798–2805.

[16] D. G. Lowe, Distinctive image features from scale-invariant keypoints, International journal of computer vision 60 (2) (2004) 91–110.

[17] S. Sundaram, Y.-H. Lee, Y. Kim, J.-H. Park, H.-S. Cho, Performance evaluation of point-based image descriptors, in: Information Science and Applications (ICISA), 2014 International Conference on, IEEE, 2014, pp. 1–2.

[18] F. Fraundorfer, D. Scaramuzza, Visual odometry: Part ii: Matching, robustness, optimization, and applications, Robotics & Automation Magazine, IEEE 19 (2) (2012) 78–90.

[19] M. Brown, D. G. Lowe, Invariant features from interest point groups., in: BMVC, no. s 1, 2002.

[20] R. J. Alitappeh, F. Mahmoudi, Mgs-sift: A new illumination invariant feature based on sift descriptor, International Journal of Computer Theory & Engineering 5 (1).

[21] T. Shukla, N. Mishra, S. Sharma, Automatic image annotation using surf features, International Journal of Computer Applications 68 (4) (2013) 17–24.

[22] opencv dev team, Feature matching (2014).
URL http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_matcher/py_matcher.html

[23] M. Muja, D. G. Lowe, Fast approximate nearest neighbors with automatic algorithm configuration., VISAPP (1) 2.

[24] S. Ghali, Homogeneous coordinates for projective geometry, Introduction to Geometric Computing (2008) 119–142.

[25] R. Hartley, A. Zisserman, Multiple view geometry in computer vision, Cambridge university press, 2003.

[26] R. I. Hartley, In defense of the eight-point algorithm, Pattern Analysis and Machine Intelligence, IEEE Transactions on 19 (6) (1997) 580–593.

[27] R. Y. Tsai, T. S. Huang, Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces, Pattern Analysis and Machine Intelligence, IEEE Transactions on (1) (1984) 13–27.

[28] F. Wu, Z. Hu, F. Duan, 8-point algorithm revisited: Factorized 8-point algorithm, in: Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on, Vol. 1, IEEE, 2005, pp. 488–494.

[29] C. Feng, Y. Hung, A robust method for estimating the fundamental matrix., in: DICTA, Citeseer, 2003, pp. 633–642.

[30] B. Ruzgiene, W. Förstner, Ransac for outlier detection, Geodezija ir kartografija 31 (3) (2005) 83–87.

[31] Z. Tian, B. C. Li, Optimize preview model parameters evaluation of ransac, in: Applied Mechanics and Materials, Vol. 687, Trans Tech Publ, 2014, pp. 3984–3987.

[32] O. Chum, J. Matas, Optimal randomized ransac, Pattern Analysis and Machine Intelligence, IEEE Transactions on 30 (8) (2008) 1472–1482.

[33] R. I. Hartley, P. Sturm, Triangulation, Computer vision and image understanding 68 (2) (1997) 146–157.

[34] C. Darken, J. Moody, Fast adaptive k-means clustering: some empirical results, in: Neural Networks, 1990., 1990 IJCNN International Joint Conference on, IEEE, 1990, pp. 233–238.

[35] A. Sampath, J. Shan, Segmentation and reconstruction of polyhedral building roofs from aerial lidar point clouds, Geoscience and Remote Sensing, IEEE Transactions on 48 (3) (2010) 1554–1567.

[36] pointclouds.org/ (2015).
URL `http://pointclouds.org/`

[37] Bsd-licens (2015).
URL `http://www.linfo.org/bsdlicense.html`

[38] R. B. Rusu, S. Cousins, 3d is here: Point cloud library (pcl), in: Robotics and Automation (ICRA), 2011 IEEE International Conference on, IEEE, 2011, pp. 1–4.

[39] Bsd-licens (2015).
URL `http://opencv.org/`