Recognition of suddenly appearing obstacles using optical flow for autonomous vehicles

Master's thesis in Systems, Control and Mechatronics

JOHAN EDDELAND

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2016

Master's thesis EX030/2016

Recognition of suddenly appearing obstacles using optical flow for autonomous vehicles

JOHAN EDDELAND

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Signals and Systems
Image Analysis and Computer Vision
Chalmers University of Technology
Gothenburg, Sweden 2016

Recognition of suddenly appearing obstacles using optical flow for autonomous vehicles
JOHAN EDDELAND

Supervisor: Fredrik Kahl, Department of Signals and Systems
Examiner: Fredrik Kahl, Department of Signals and Systems

Master's Thesis EX030/2016
Department of Signals and Systems
Image Analysis and Computer Vision
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Top right and top left show subsequent frames from a video recorded and used for analysis in this thesis. Bottom left shows the calculated optical flow field, where the focus of expansion is marked with a red circle with black border. Bottom right shows the segmentation result when using an angle threshold of $\pi/2$, clearly indicating that the bicyclist seen coming in from the right in the two frames has been detected.

Typeset in LaTeX
Gothenburg, Sweden 2016

Recognition of suddenly appearing obstacles using optical flow for autonomous vehicles

JOHAN EDDELAND
Department of Signals and Systems
Chalmers University of Technology

Abstract

The problem of designing autonomous vehicles poses many interesting questions, among them the issue of active safety. To make sure an autonomous vehicle does not damage its environment or itself, many measures need to be taken with the help of sensors that help it acknowledge the world around it.

This thesis presents works in the area of safety for autonomous vehicles. More specifically, it regards the development of a small part of a safety system, which uses a single camera pointing forward to recognize when there are potential objects present that need to be avoided. The approach is to calculate optical flow in the video from the camera, then find the focus of expansion from the optical flow, and finally segment the given image for angle deviations based on a model where flow vectors are expected to diverge from the focus of expansion.

The results from evaluating the algorithm on recorded routes show that using the presented model leads to an over-segmentation of objects in the optical flow fields. This results in a system that when implemented would not allow the vehicle in question to drive autonomously as much as needed, as it would find too many potential objects where there are in fact none. In certain situations, however, the results show that some obstacles can be found and that the vehicle can make decisions that correspond well to intuitive manual analysis of the situations. The conclusion of this is that the work presented in this thesis can act as a framework for future development of an active safety system for autonomous vehicles, where improvements could be made by including more sensor information and optimizing parameter values for different parts of the system.

Keywords: optical flow, computer vision, obstacle avoidance, obstacle recognition, focus of expansion

Contents

List of Figures

List of Tables

1
Introduction

1.1 Background

The possibility of fully autonomous vehicles is coming closer and closer to reality, but together with exciting ideas about what can be done in a driver-less environment, the safety issues also make themselves apparent. To prevent casualties and damage, the autonomous vehicles must have advanced safety systems. This master thesis project is a small part of such an advanced safety system, where the main focus is to recognize when a suddenly appearing object appears in front of an autonomous vehicle with a single camera pointing forward. An example of an object that should be avoided can be found in Figure 1.1, where a bicyclist comes in from the right and creates a risk of collision with the camera.

Figure 1.1: An example of an object that should be avoided. The image is taken from a video which is recorded and used in this thesis work. A bicyclist, marked with a red rectangle, comes in from the right and creates a risk of collision with the camera.

Recognizing when there are objects in front of the vehicle is no simple task and it could be achieved in multiple different ways. The approach chosen here is to make use of *optical flow*, which is a technique that estimates how regions or pixels are moving in between two images. In other words, the work presented makes no attempt at distinguishing what *kind* of objects are in front of the vehicle, like trees or pedestrians, but rather whether there are any objects at all that should be avoided in some way. Since pedestrians standing still definitively should be avoided,

it is important that this work should be used in collaboration with other safety techniques for autonomous vehicles.

1.2 Purpose and Objectives

The purpose of the project is to create a part of a safety system for autonomous vehicles. The system created is supposed to be implementable on relatively low-cost hardware in real-time applications, which means that the calculation times for all parts of the program are very important. A desirable goal for such a system is that it should react quicker to dangerous situations than a human would, meaning that an investment into autonomous vehicles could pay off in the way of less accidents.

The objectives for the project can be divided into several smaller parts, as introduced in the following sections.

1.2.1 Recognizing when obstacles are present

To be able to evaluate if the algorithm can recognize when obstacles are present, it is important to define what an *object* refers to in this context. An object is in this case a cluster of pixels which moves in a direction different to what could be expected of it, based on a motion model with velocity vectors moving away from a certain point called the focus of expansion. This aim consists of the fact that the created program should be able to tell whether or not there are any objects present in between a given pair of images (as part of a video). This information should then be used to decide what control action will be used for the vehicle.

1.2.2 Creating decisions for different scenarios

Given that an obstacle is present, it is not trivial to decide how the autonomous vehicle should react. Depending on how fast the object is moving, and whether it is moving towards the edge of the images or towards the center, there might be different choices playing a part of how to influence the speed of the vehicle based on this analysis.

1.2.3 Evaluating performance of algorithm using pre-recorded data

Evaluating an algorithm that is supposed to recognize when obstacles are present is not straightforward, since there is no obvious ”ground truth” to compare to. This objective of the project consists of coming up with a consistent way of measuring how well the algorithm is performing based on some manual analysis of pre-recorded routes.

1.2.4 Evaluating the performance of certain optical flow algorithms

This thesis makes use of developed optical flow algorithms that have been presented by researchers in the area. This final objective of the report is to compare the performance of these optical flow algorithms against each other, to provide an outlook of a new application of optical flow.

1.3 Related Work

The work performed in this thesis is based on combining research from a few different sub-fields of computer vision, mainly optical flow and focus of expansion calculations. The idea to use optical flow to create systems that avoid obstacles has been presented before [4, 5], but in these cases the use of a motion model based on the calculated focus of expansion has not been tested. Optical flow in itself is chosen because it gives rise to a fairly simple motion model that, if it turns out to approximate real-world motion well enough, can be used for an obstacle avoidance scheme more easy to adjust than far more advanced techniques (e.g. SLAM [6], where instead of just considering the image plane, a more complex 3D-mapping of the environment is attempted).

1.3.1 Optical flow

The area of optical flow has been heavily researched since the beginning of the 1980s, when the work on global methods and local methods for flow calculations were presented by Horn & Schunk [7] and Lucas & Kanade [8], respectively. Since then, there have been many branches of research focusing on different aspects of the problems with approximating optical flow, like large displacement flow [9, 10] and real-time flow computations [11, 12, 13].

Almost all of the papers on optical flow do however focus on the flow calculations themselves, and not so much on the possible applications. In this thesis, the optical flow algorithms evaluated are fast, meaning that flow calculation times are in the order of one second on a standard CPU. The algorithms used are created by Farnebäck [14], Bao et al [1], and Wulff & Black [15]. These algorithms are explained more thoroughly in Sections 2.2.2 - 2.2.4.

1.3.2 Calculating the focus of expansion

When a camera moves while filming, the *Focus of Expansion* (FOE) is the projected point on the image which the camera is moving towards. Many different methods of calculating the FOE have been presented in literature, most of which can be divided into discrete and continuous methods [16, 17]. The continuous methods are the ones which rely on the use of optical flow, which make them the most relatable to this thesis.

One approach to approximating the FOE is to calculate it as the least-square solution to the pseudo-intersection of optical flow vectors [18]. Another method is to count the number of positive and negative horizontal flow components, take the difference and average over rows and columns to find the x- and y-components of the FOE separately [19]. The method used in this report is based on a matched filter technique [3], which accurately finds the FOE if it is in the image (which is not always the case e.g. if the vehicle with the camera is turning at a high rate).

1.4 Contributions

The main contribution of this thesis is the idea and evaluation of a new kind of method to avoid collisions between autonomous vehicles and suddenly appearing objects. As such, it is a base for further research and improvement with other existing techniques of active safety for vehicles, even though the created algorithm has not been implemented on a real prototype.

This report also provides an insight into whether or not it is suitable to use such a simple motion model to try and characterize dangerous situations for autonomous vehicles, in an attempt to be able to reduce accidents on roads. As it uses several previously presented optical flow algorithms, it provides information about how optical flow algorithms perform in such a kind of an application, as well as what properties of optical flow can be important in cases of detection of object presence for autonomous vehicles.

1.5 Limitations

Time constraints for the project lead to some major boundaries that could be worked on in future related projects. The algorithms created only consider making use of a forward-pointing camera as the only sensor on an autonomous vehicle. Any autonomous vehicle will likely be equipped with numerous auxiliary sensors, which could be used to enhance the performance of the proposed algorithm and make its part in a safety system even bigger.

It is also important to be clear about the fact that this thesis makes no attempt to try to *identify* any obstacles (e.g. pedestrians, other cars etc). This is of course a big limitation, as the created program would not be able to identify a child standing still in front of the moving self-driving vehicle – the algorithm considers only obstacles which are moving in a way that the motion of the camera can not easily predict.

2
Theory

This chapter presents the theory of the different parts needed to create a real-time algorithm for recognition of quickly-moving obstacles.

2.1 Image Processing

To be able to understand some of the methods used in this report, some basic concepts of image processing are needed. This section provides a quick insight in how to reduce the resolution of an image, something which is important when optimizing image or video calculations with respect to time.

2.1.1 Reducing resolution

Assume an image with resolution $M \times N$. This means that the image is a matrix with M rows and N columns, and the goal of performing one resolution reduction can be reached by the use of Gaussian pyramids [20]. Imagine a pyramid of images, where the original image is the base of the pyramid, and each upper layer is a version of the lower one with reduced size, as illustrated in Figure 2.1.

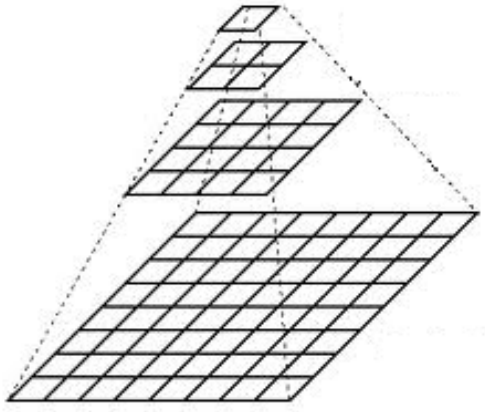

Figure 2.1: An illustration of what an image pyramid is. At the bottom of the pyramid is the original image, and each subsequent layer above is a version of the image below, but reduced in size.

Consider the original image to be the base G_i of the Gaussian pyramid. The next level G_{i+1} is created by convolving the image with the Gaussian kernel

$$\frac{1}{16}\begin{bmatrix}1 & 4 & 6 & 4 & 1\\ 4 & 16 & 24 & 16 & 4\\ 6 & 24 & 36 & 24 & 6\\ 4 & 16 & 24 & 16 & 4\\ 1 & 4 & 6 & 4 & 1\end{bmatrix}, \tag{2.1}$$

and afterwards removing every even-numbered row and column. This results in an image which has the resolution $\frac{M}{2} \times \frac{N}{2}$, which is exactly one quarter of the original resolution.

2.2 Optical Flow

Optical flow is the approximation of velocities in a pair of images, which can be calculated in many different ways. In this section, ways of calculation that are relevant to this thesis are presented.

An optical flow algorithm produces images where each point, corresponding to a pixel in the original pair of images, is represented by a two-dimensional vector. To visualize the flow vectors in a concise way, [21] propose a color coding scheme where direction of the vectors is coded by hue and length of the vectors is coded by saturation (see Figure 2.2).

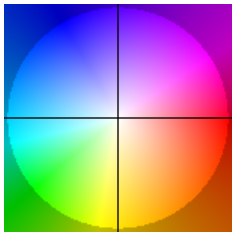

Figure 2.2: Color coding of flow vectors, where direction is colored by hue and length is coded by saturation.

An example of how the visualization of optical flow looks can be found in Figure 2.3.

Figure 2.3: An example of how optical flow can be visualized. The left image is the first frame, and the middle image is the second frame. The right image is a visualization of the resulting optical flow, as calculated by the EPPM algorithm [1].

2.2.1 Gradient constraint equation

The standard constraint to assume when calculating optical flow is the *brightness constancy assumption*, which means that a certain area or object moving between two images I_1 and I_2 have the exact same intensity. This is mostly not true for real applications due to lightning, camera properties and shadows in the real world, but is a good starting point for optical flow estimation. Let $I(x, y, t)$ be the image intensity at the point (x, y) and time t, then the assumption is described by [22]

$$I(x, y, t) = I(x + u, y + v, t + 1). \tag{2.2}$$

In this case, u is the velocity in x-direction and v is the velocity in y-direction. The displaced image can be approximated by a Taylor-series, ignoring higher-order terms:

$$I(x + u, y + v, t + 1) \approx I(x, y, t) + u \cdot I_x(x, y, t) + v \cdot I_y(x, y, t) + I_t(x, y, t). \tag{2.3}$$

I_x and I_y denote spatial derivatives of the image I and I_t denotes the temporal derivative. Substituting this approximation into Equation (2.2) yields the *gradient constraint equation*:

$$I_x u + I_y v + I_t = 0. \tag{2.4}$$

The unknowns in this equation, u and v, are the components of the optical flow. As can be clearly seen, Equation (2.4) is underdetermined with one equation and two unknowns. This means that the flow velocity (u, v) cannot be computed without introducing more constraints [7]. The different methods that come up for calculating optical flow usually depend on which more constraints are included.

2.2.2 Optical flow based on polynomial expansion

In 2003, Farnebäck introduced the idea of approximating the neighborhood of each pixel in an image with a polynomial to then calculate optical flow [14]. Expressed in a local coordinate system, this can be written as

$$I(x) \sim x^T \mathbf{A} x + \mathbf{b}^T x + c, \tag{2.5}$$

where $\mathbf{A}$ is a symmetric matrix, $\mathbf{b}$ is a vector and c is a scalar. Let us now consider displacement of a neighborhood with a vector $\mathbf{d}$. Consider the first neighborhood to be approximated by

$$I_1(x) = x^T \mathbf{A}_1 x + \mathbf{b}_1^T x + c_1. \tag{2.6}$$

The new neighborhood, affected by displacement $\mathbf{d}$, is expressed by

$$\begin{aligned} I_2(x) &= I_1(x - \mathbf{d}) = (x - \mathbf{d})^T \mathbf{A}_1 (x - \mathbf{d}) + \mathbf{b}_1^T (x - \mathbf{d}) + c_1 \\ &= x^T \mathbf{A}_1 x + (\mathbf{b}_1 - 2\mathbf{A}_1 \mathbf{d})^T x + \mathbf{d}^T \mathbf{A}_1 \mathbf{d} - \mathbf{b}_1^T \mathbf{d} + c_1 \\ &= x^T \mathbf{A}_2 x + \mathbf{b}_2^T x + c_2. \end{aligned} \tag{2.7}$$

Equating the different coefficients from the two polynomials $I_2(x)$ and $I_1(x - \mathbf{d})$ yields

$$\mathbf{A}_2 = \mathbf{A}_1 \tag{2.8}$$
$$\mathbf{b}_2 = \mathbf{b}_1 - 2\mathbf{A}_1\mathbf{d} \tag{2.9}$$
$$c_2 = \mathbf{d}^T\mathbf{A}_1\mathbf{d} - \mathbf{b}_1^T\mathbf{d} + c_1. \tag{2.10}$$

Equation (2.9) can be solved for the translation $\mathbf{d}$ if $\mathbf{A_1}$ is non-singular. Since $\mathbf{d}$ is what we search for when trying to approximate optical flow, the solution in this case is

$$\mathbf{d} = -\frac{1}{2}\mathbf{A}_1^{-1}(\mathbf{b}_2 - \mathbf{b}_1). \tag{2.11}$$

2.2.3 Nearest neighbor field

One of the optical flow methods used in this thesis is an algorithm by Bao et al. which uses approximate *Nearest Neighbor Field* (NNF) [1]. The algorithm is local, which means that it does not optimize over the entire image, resulting in a shorter execution time. The main idea is to consider patches in the image, initializing random correspondence fields and iteratively propagating guesses among pixels that are nearby [2]. A good illustration of what this means is shown in Figure 2.4.

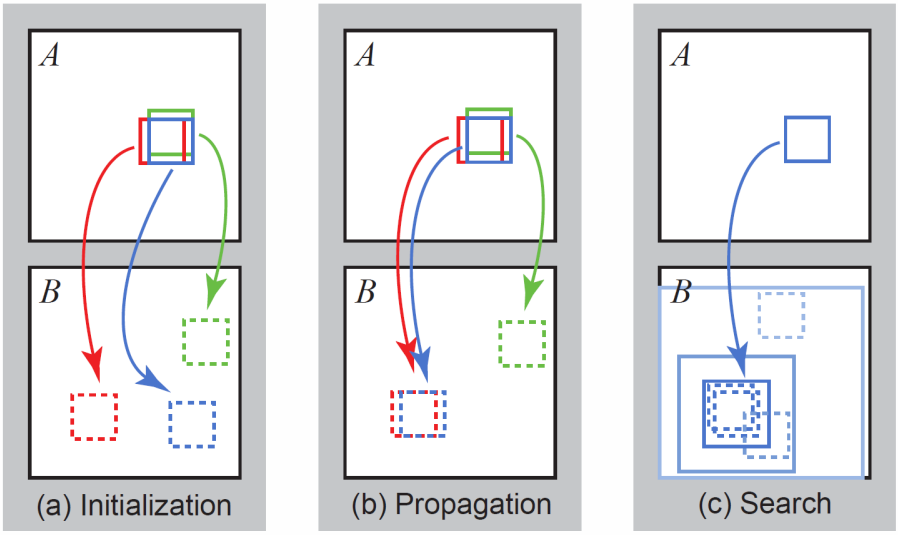

Figure 2.4: The different phases of a randomized nearest neighbor algorithm. (a) patches are initially randomly assigned. (b) the blue patch checks above/green and left/red neighbors to see whether they improve the blue mapping. (c) the patch searches for improvements randomly [2].

Assume that two patches with radius r, centered at location $\mathbf{a}(x_a, y_a)$ in image A and location $\mathbf{b}(x_b, y_b)$ in image B, are supposed to be matched. The cost function originally used between the two patches is

$$d(\mathbf{a}, \mathbf{b}) = \sum_{\mathbf{\Delta}(\Delta x, \Delta y): |\Delta x| \leq r, |\Delta y| \leq r} \|I^A(\mathbf{a} + \mathbf{\Delta}) - I^B(\mathbf{b} + \mathbf{\Delta})\|^2, \tag{2.12}$$

where I^A and I^B denote the CIELab color appearances of image A and B, respectively [23].
The algorithm produced by Bao et al. uses modifications of this patch cost, together with more adjustments and approximations to increase the speed of the optical flow calculations.

2.2.4 Principal component analysis

Yet another idea of calculating optical flow is presented by Wulff & Black [15] in a report that shows a trained algorithm via *Principal Component Analysis* (PCA). PCA is a multivariate statistical technique that is used to extract only the most important information from a set of data [24], to compress the size of the set and, in this case specifically, simplifying the description of the set to in the end perform quicker optical flow calculations. This means that the algorithm in itself consists of two parts; the first step is an offline part where the principal components are calculated from a set of training data (in this case commercial movies), and the second step is the online part where the optical flow is calculated quickly using the principal components previously extracted.

The underlying assumption is that the optical flow fields in question can be approximated by a weighted sum of basis flow fields $\mathbf{b}_n, n = 1 \dots N$, with weights w_n

$$\mathbf{u} \approx \sum_{n=1}^{N} w_n \mathbf{b}_n. \tag{2.13}$$

In this particular case, $\mathbf{u}$ and $\mathbf{b}_n$ are vectorized optical flow fields, where the horizontal and vertical components are stacked as column vectors: $\mathbf{u} = (\mathbf{u}_x^T, \mathbf{u}_y^T)^T$.

2.3 Focus of Expansion and Expected Motion

The *Focus of Expansion* (FOE) is the pixel in the image corresponding to the intersection of the three-dimensional (3D) velocity vector describing the camera movement and the projection plane [3]. It is characterized (in the ideal case) by the fact that the optical flow vector at the FOE has zero magnitude and that all of the other optical flow vectors diverge radially from it, as can be seen in Figure 2.5.

2.3.1 Expected motion model

The FOE is calculated with respect to angles of the optical flow vectors, and the way to distinguish which pixels in the image belong to an object or to the background is also based on angles. More specifically, let $u(x, y)$ be the horizontal component

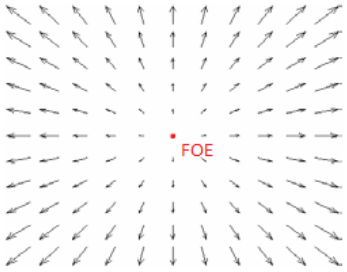

Figure 2.5: An example of how the optical flow vectors are expected to look around the FOE. The FOE is marked with a red dot. [3]

of the optical flow velocity vector and $v(x, y)$ be the vertical component. The angle $\alpha(x, y)$ for a given flow vector can be calculated according to

$$\alpha(x, y) = \arctan \frac{v(x, y)}{u(x, y)}. \tag{2.14}$$

A pixel at position (x, y) in the image is considered to be part of an object if $|\alpha(x, y) - \beta(x, y)|$ is larger than some predefined angle threshold value. Here $\beta(x, y)$ is the expected angle based on the FOE position in the image, which can be calculated as

$$\beta(x, y) = \arctan \frac{y - y_c}{x - x_c}, \tag{2.15}$$

where (x_c, y_c) is the position of the FOE.

2.3.2 Calculating the FOE with a matched filter

In [3], Sazbon et. al. present a method of finding the FOE of an image in a robust way, given that the FOE is inside the image. The approach is based on the assumption presented previously, namely that all optical flow vectors around the FOE are expected to diverge away from it. The algorithm presents a two-dimensional filter which optimizes a cost function that only attempts to match expected directions of flow vectors (hence not taking magnitude of the optical flow into account).

Assume a two-dimensional filter with filter size $(2w+1) \times (2w+1)$, where each pixel in the filter represents the angle between its corresponding grid point and the origin (i.e. the middle point of the filter). This gives the filter

$$F(m, n) = \arctan \frac{n}{m} \quad -w \leq m \leq w, \quad -w \leq n \leq w, \tag{2.16}$$

where m and n are coordinates in $y-$ and $x-$directions, respectively (cf. Equation (2.14)). The FOE is the pixel in the optical flow that minimizes

$$(\hat{x}_{FOE}, \hat{y}_{FOE}) = \arg\min_{(x,y)} S(x, y), \tag{2.17}$$

where $S(x, y)$ is the cost function defined as

$$S(x,y) = \Psi(u(x,y), v(x,y)) \sum_{m=-w}^{w} \sum_{n=-w}^{w} \Big((F(m,n) - \alpha(u(x+m, y+n), v(x+m, y+n)))^2 \Phi(u(x+m, y+n), v(x+m, y+n)) \Big). \quad (2.18)$$

Here $\alpha(u(x,y), v(x,y))$ is defined as in Equation (2.14) and $\Phi(u(x,y), v(x,y))$ is a weight function defined as

$$\Phi(u(x,y), v(x,y)) = \begin{cases} 1 & u(x,y)^2 + v(x,y)^2 \geq t \\ 0 & \text{otherwise,} \end{cases} \quad (2.19)$$

where t is a predefined threshold value. This threshold is used to prevent optical flow vectors with close to zero magnitude from affecting the sum in Equation (2.18), as their angle can be considered to unsure. In Equation (2.18), the term $\Psi(u(x,y), v(x,y))$ is then

$$\Psi(u(x,y), v(x,y)) = \Big(\sum_{m=-w}^{w} \sum_{n=-w}^{w} \Phi(u(x+m, y+n), v(x+m, y+n)) \Big)^{-1}, \quad (2.20)$$

meaning that $\Psi(u(x,y), v(x,y))$ is the number of neighbors that actually participate to form the sum of the weighted square differences for the (x,y) pixel.

To speed up the calculations somewhat, and also to not find false FOE in image pairs where the true FOE is not inside the flow image, the additional constraint proposed by Kumar et. al. in [25] is used. This means that only optical flow vectors that have a magnitude below some threshold are considered as possible candidates for being the FOE. More formally, this can be explained as setting the cost function value to infinity for flow vectors with magnitude below the threshold value, and otherwise calculating the cost function value according to Equation (2.18).

2.4 Receiver Operating Characteristic

A *receiver operating characteristic* (ROC) is a plot that illustrates the performance of a system where binary classifiers are used. Binary classifiers are used to separate test results into two groups, in the case of this report the groups ”object” and ”not object”. This ROC curve can be used to determine how well a given algorithm or test setting is working for different sets of parameters, and proves useful when having many cases of varying parameters.

When evaluating the performance of the algorithm in this report, there will be a ground truth that describes where in the image there are objects that should be avoided. The algorithm will also generate a set of objects that should be avoided, and these two classifications can then be compared to see how well the algorithm

matches to the ground truth (which is, in this case, manual analysis).

Having two cases of binary classification means a total of four scenarios that can happen. The comparisons are made in a pixel-by-pixel fashion, which is summarized in the following way:

- When a pixel is marked as part of an object in the ground truth *and* in the automatic analysis, it is called a **true positive**
- When a pixel is marked are part of an object in the ground truth but not in the automatic analysis, it is called a **false negative**
- When a pixel is not marked as part of an object in the ground truth, but it is marked as part of an object in the automatic analysis, it is called a **false positive**
- When an object is not marked as part of an object in neither the ground truth nor the automatic analysis, it is called a **true negative**

Another type of result presented in Section 4 is ROC curves for binary decisions. A binary decision is in this case a simple control decision for the autonomous vehicle, which serves as an indicator of whether or not the algorithm proposed in this thesis is viable for real-life use or not. For both the ground truth and the automatic analysis, the binary decision will be "GO" if there are no objects in the image, and "STOP" if there is at least one object in the image. The different combinations of binary decisions give these classifications:

- When both the ground truth and the automatic analysis make the binary decision "STOP", it is called a **true positive**
- When the ground truth makes the binary decision "STOP" and the automatic analysis makes the binary decision "GO", it is called a **false negative**
- When the ground truth makes the binary decision "GO" and the automatic analysis makes the binary decision "STOP", it is called a **false positive**
- When both the ground truth and the automatic analysis make the binary decision "GO", it is called a **true negative**

The ROC curve is created by plotting the *True Positive Rate* (TPR) against the *False Positive Rate* (FPR). The TPR is calculated as

$$TPR = \frac{\text{number of true positives}}{\text{number of true positives} + \text{number of false negatives}}, \tag{2.21}$$

and the FPR is calculated as

$$FPR = 1 - \frac{\text{number of true negatives}}{\text{number of true negatives} + \text{number of false positives}}. \tag{2.22}$$

Note that TPR is also called *sensitivity*, and FPR is equal to 1 minus the *specificity*.

3
Methods

This chapter explains the steps that have been used to design the recognition of obstacles and evaluate the results of the algorithm.

3.1 Recording Routes for Evaluation

To be able to evaluate the algorithms created, data is needed. To best imitate a system that contains cheap, accessible hardware, the data was gathered by the use of ordinary cameras, such as smartphone cameras, mounted on bicycles and cars. The quality of the videos vary as the circumstances of each recording are not exactly the same, but the important theme of the videos are that they are recorded in an urban setting.

The videos are recorded in central Gothenburg, and each video is around 1 minute long. The number of objects which should be avoided vary, since there is no special setup for this and the objects in the videos are mostly cars, bicyclists and pedestrians that sometimes move in ways that are deemed unsafe when performing manual analysis of the recordings.

3.1.1 Reducing resolution

As the end goal of the main program created is to be implementable on a real-time system, computation times are of high importance. As such, the videos have their resolution reduced by a factor of 4 before undergoing any further computations. The resolution of each image in the video is reduced in a standard fashion by constructing Gaussian image pyramids as explained in section 2.1.1. Reducing the resolution of an image from $M \times N$ to $\frac{M}{2} \times \frac{N}{2}$ is done by convolving each image with a Gaussian kernel, which is performed one time to reach the desired resolution. The lower resolution results in a less detailed optical flow but faster calculations in general.

3.2 Generating a Ground Truth for Recognition of Obstacles

To be able to tell if the algorithm is performing well or not, it is important to have a ground truth to compare the calculations against. Since it is not entirely clear what an ”object” is at all times in a video, it is decided quite arbitrarily in a manual

pre-analysis. The objects are selected by bounding rectangles. This means that each time in a video that any object shows up that should be avoided due to possibility of collision with the vehicle, a rectangle overlay is created that fully surrounds the object in the image. These rectangle overlays are stored to be used at the evaluation process of the automatic recognition of objects in the program.

3.3 Optical Flow

Optical flow calculations are at the core of the proposed approach to recognize when obstacles are present. As such, four algorithms are implemented and compared in this thesis project:

- **Farnebäck's algorithm** based on polynomial expansion [14]
- **Fast Edge Preseving PatchMatch** (EPPM) by Bao et. al. [1]
- **PCA-Flow** by Wulff & Black [15]
- **PCA-Layers** by Wulff & Black [15]

For each pair of images in a given recorded video, the optical flow is calculated with each of the four algorithms presented above. This means that the number of calculated optical flow vector fields for each algorithm is one less than the number of frames in a video. It is the calculated flow fields that are considered for the rest of the work in the thesis, where the motion model used takes only the angles of the flow into account. A comparison between the four different optical flow algorithms is shown in Figure 3.1, where the flow is calculated for a pair of images from a video recorded and used in the thesis.

3.3.1 Approximate Runtimes

The optical flow algorithms chosen in this thesis are used because they are relatively fast algorithms when evaluated on the optical flow benchmark KITTI flow 2012 [26]. The official timings reported for each algorithm in the benchmark are shown in Table 3.1. The timings reflect approximately how long time it takes to perform one optical flow calculation for a pair of images.

Table 3.1: Approximate timings for each optical flow algorithm, as reported on the benchmark KITTI flow 2012.

Algorithm	**Time**
Farnebäck's algorithm	1 s
EPPM	0.25 s
PCA-Flow	0.19 s
PCA-Layers	3.2 s

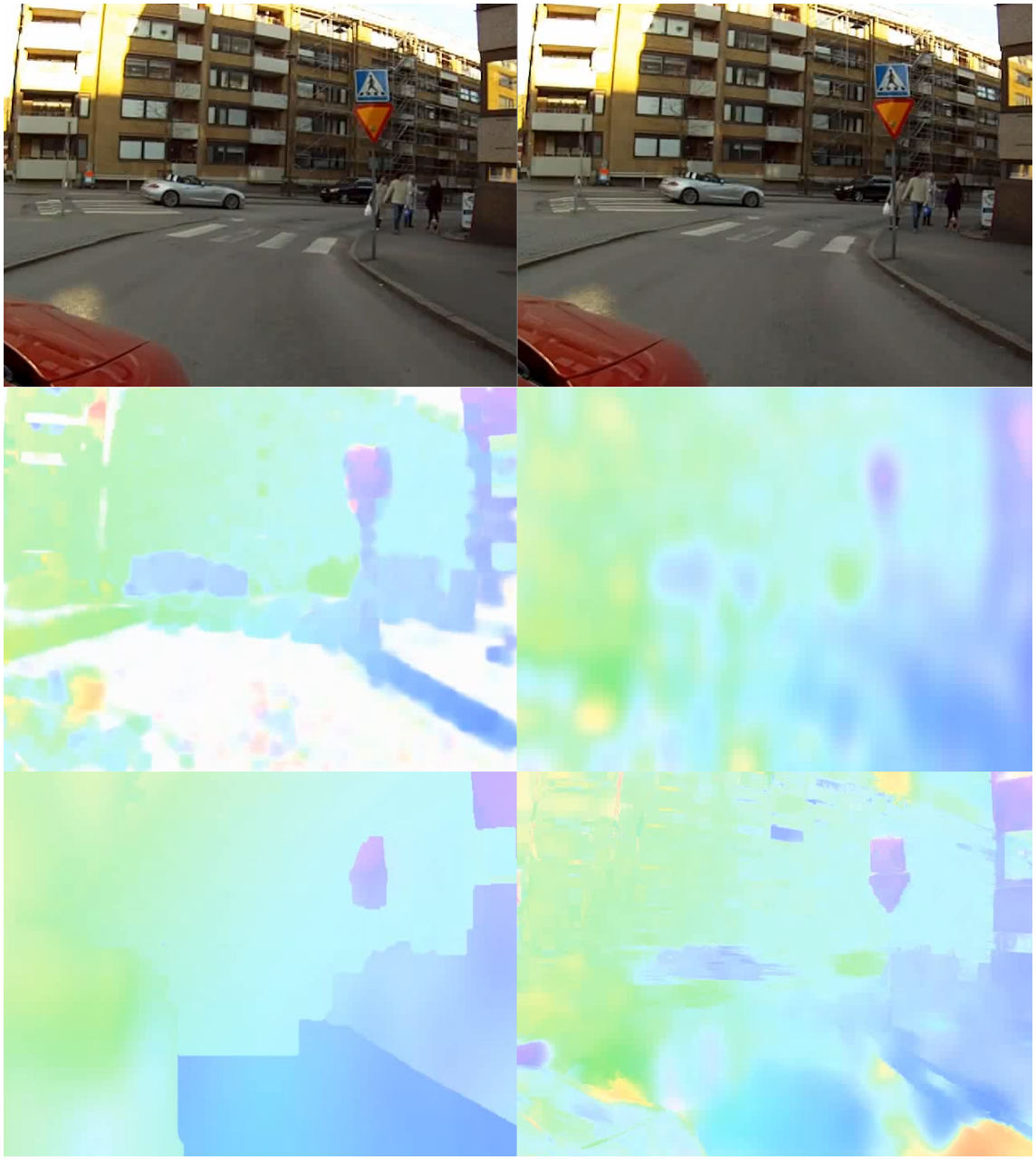

Figure 3.1: Illustration of the difference of the four different optical flow algorithms presented in this thesis. The flow fields are calculated for a pair of images from a video recorded and used in the thesis. Top left and top right shows frame one and two, respectively. Middle left shows the flow calculated by Farnebäck's algorithm, middle right shows the flow calculated by PCA-Flow, bottom left shows the flow calculated by PCA-Layers, and bottom right shows the flow calculated by EPPM. It is obvious that the flows calculated from different algorithms are far from identical, which also means that the final segmentation results will be different.

3.3.2 Using different frame jumps

As has been stated before, an optical flow calculation is naturally carried out on a pair of frames. In reality, most cameras capture the environment at around 30 frames per second, which means that the optical flow algorithms can not be expected to be able to calculate flow for each subsequent pair of frames in a video. A choice has to be made in regards to exactly which two frames are used for each calculation. The solution chosen to be able to test different approaches are so called frame jumps. A frame jump of 1 means that each algorithm performs optical flow calculations on each subsequent pair of images. A frame jump of 2 does however mean that every other frame is skipped, leading to less computations per second in a real-time environment, and also leading to larger flows in the image due to the camera motion being larger between the two chosen frames. In this report, the frame jumps evaluated are 1, 4 and 8, to get a good overview of how the frame jumps affect the algorithm as a whole. An illustration of what frame jumps mean can be seen in Figure 3.2.

Figure 3.2: Illustration of the meaning of frame jumps by the use of a 9-image sequence. The top row shows which are the first two frames used for optical flow calculations when frame jump is 1. The middle row shows frame jump 4, and the bottom row shows frame jump 8. The used frames are highlighted compared to the other ones.

3.4 Calculating the FOE

The calculation of the FOE is made with a matched filter according to Section 2.3.2. It is also important to store information about whether or not an FOE was actually found in the given optical flow field or not. To illustrate how this is done, Algorithm 1 contains pseudo-code of how the computations are made.

3.5 Performing Segmentation to Recognize Objects

When both the optical flow and the FOE are calculated, these parts are put together to examine whether there are any obstacles present that should be avoided

Algorithm 1 FOE calculation

```
minimal_cost ← inf
FOE_found ← False
for vector (u(x, y), v(x, y)) in optical flow field do
    if u² + v² > mag_threshold then
        Ignore this vector
    else
        Calculate S(x, y)
        if S(x, y) < minimal_cost then
            minimal_cost ← S(x, y)
if minimal_cost < inf then
    FOE_found ← True
```

or not. A vector in the flow field is considered to be part of an object if the absolute value of the angle difference between the vector and the expected angle is larger than a predefined threshold value, as explained in Section 2.3.1. Performing this segmentation for all vectors in the flow results in a binary image where white pixels correspond to objects and black pixels correspond to background. To avoid a lot of small clutterings of objects where the optical flow might be of low quality, objects smaller than a certain area threshold value are removed. The motivation for this object removal is to reduce the number of false positives when segmenting objects. As the area threshold is very small, it should not impact the safety of the program severely, but rather help to make the vehicle run smoother in cases where the optical flow has small regions of lower quality flow calculations.

Based on the binary image, analysis can determine how well the segmentation performs (see Chapter 4 for details). However, a binary decision is also made based on whether any objects are present in the segmented image or not. The binary decision is simply to tell the vehicle to "STOP" if there are any objects and to "GO" if there are no objects. If the FOE was not found for the image, the binary decision is "STOP" automatically, since it in this case is unsure how the motion model should look like at all. This binary classification gives a quicker understanding to how well a certain algorithm and parameter set performs with regards to the end goal of obstacle avoidance.

Algorithm 2 shows how the segmentation is performed in a short and concise way. This algorithm can be seen as a continuation of Algorithm 1.

3.6 Implementation Details

The optical flow algorithms, the FOE calculations and the rest of the program created in this thesis have been implemented partly on a Windows computer and partly on a Mac computer, using MATLAB and the programming language Python. The code presented for the EPPM algorithm in the paper by Bao et. al. runs on a GPU on Windows, and the code presented by Wulff & Black runs on Mac. The

Algorithm 2 Object segmentation

```
1: binary_image ← zero matrix dimensioned like flow field
2: if FOE_found =True then
3:     for vector (u(x, y), v(x, y)) in optical flow field do
4:         if |α(x, y) − β(x, y)| > angle_threshold then
5:             binary_image(x, y) ← 1
6:         else
7:             binary_image(x, y) ← 0
8:     remove objects smaller than area_threshold in binary_image
9:     if sum(binary_image) = 0 then
10:        binary_decision ←'GO'
11:    else
12:        binary_decision ←'STOP'
13: else if FOE_found =False then
14:    binary_decision ← 'STOP'
```

implementation of Farnebäck's optical flow algorithm is a built-in function in the Python library OpenCV [27]. To be clear, the code for each optical flow algorithm is unaltered by the author of this master thesis, and used as presented in respective article or report.

Each video is cut into parts of about 600 frames, equaling to around 20 seconds of video at 30 frames per second, due to memory limitations of the available computers. For each of these 600 frame videos, the program is run for all four optical flow algorithms, three different values for angle threshold, and three different values of frame jump. To perform computation of one 600 frame video of all these 36 scenarios takes around three days, meaning that there is a severe time constraint when evaluating the videos. It is important to note that one of the reasons that the calculations take relatively long time to perform is that one of the algorithms (PCA-Layers) has a significantly higher average computation time than the other algorithms, as seen in table 3.1. Also, the FOE calculations are chosen with respect to giving a good FOE rather than performing in the most time-optimal way – this could be changes in different ways (for example by varying the matched filter window size w).

3.6.1 Recording equipment

The majority of the videos have been recorded with a GoPro camera, which records videos in resolution 1280×960. These videos have the resolution reduced according to Section 3.1.1, resulting in a final resolution of 640×480 used for computations.

3.6.2 Parameters used

Even though the approach used throughout this report is based on a simple model, there are many possible variations of parameters. The different parameters chosen

are based on what seems usable in a scenario where the algorithm is implemented on a real-life prototype. However, because of the time constraints mentioned earlier, some parameters have not been changed (e.g. the resolution, which is kept at 640×480). An overview of all the parameters used are shown in Table 3.2.

Table 3.2: All the different parameter values used in the implemented algorithm.

Parameters	**Values**		
Angle threshold	$\pi/2$	$\pi/3$	$\pi/4$
Frame jump	1	4	8
Area threshold	1000		
Resolution	640×480		

4
Results

The results of the recognition of objects are presented with the aid of ROC curves and bar graphs. The metrics that are deemed interesting for evaluation of the algorithm are:

- **Individual pixel values** – Each pixel of each segmented image is compared to the ground truth
- **Binary decisions** – Based on whether or not there are any objects in a segmented image, the binary decision created is compared to the one for the ground truth

As the results shown are dependent on a lot of parameter combinations (based on differences in frame jump and angle threshold), Table 4.1 shows an overview of which parameter combinations are shown in which figure.

Table 4.1: Parameter combinations shown in results figures.

Figure	Subfigure	Curve type	Frame jump	Angle threshold
4.1	Left	Pixels TPR	$1, 4, 8$	$\pi/2$
	Middle	Pixels TPR	$1, 4, 8$	$\pi/3$
	Right	Pixels TPR	$1, 4, 8$	$\pi/4$
4.2	Left	Pixels FPR	$1, 4, 8$	$\pi/2$
	Middle	Pixels FPR	$1, 4, 8$	$\pi/3$
	Right	Pixels FPR	$1, 4, 8$	$\pi/4$
4.3	Top	Pixels ROC	1	$\pi/2, \pi/3, \pi/4$
	Middle	Pixels ROC	4	$\pi/2, \pi/3, \pi/4$
	Bottom	Pixels ROC	8	$\pi/2, \pi/3, \pi/4$
4.4	Top	Decisions ROC	1	$\pi/2, \pi/3, \pi/4$
	Middle	Decisions ROC	4	$\pi/2, \pi/3, \pi/4$
	Bottom	Decisions ROC	8	$\pi/2, \pi/3, \pi/4$

4.1 Varying the Frame Jump

In this section, bar graphs are presented that show the impact of varying the frame jump between 1, 4 and 8 frames, while the angle threshold stays constant. The true positive rate is shown in Figure 4.1, and the false positive rate is shown in Figure 4.2. Both the TPR and the FPR are shown for individual pixel values.

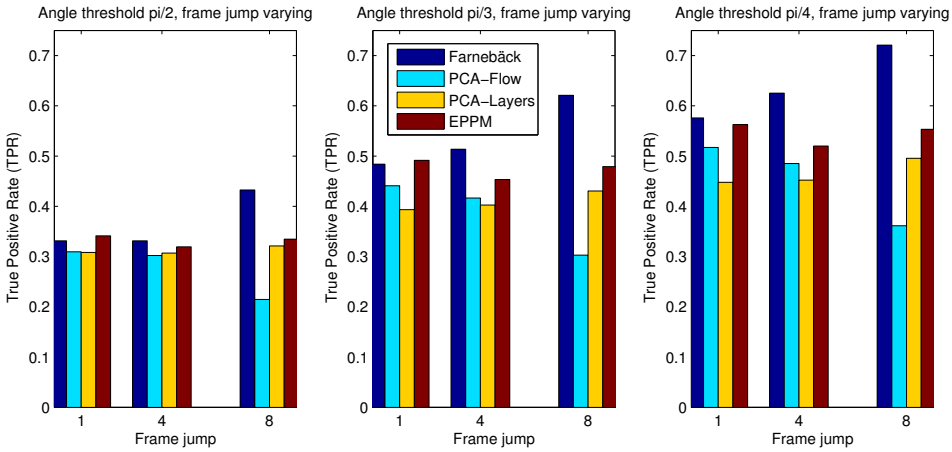

Figure 4.1: TPR for individual pixel values, showing the impact of varying the frame jump and the angle threshold. Left shows angle threshold $\pi/2$, middle shows angle threshold $\pi/3$, and right shows angle threshold $\pi/4$. For all three subplots, the frame jump is varied between 1, 4 and 8.

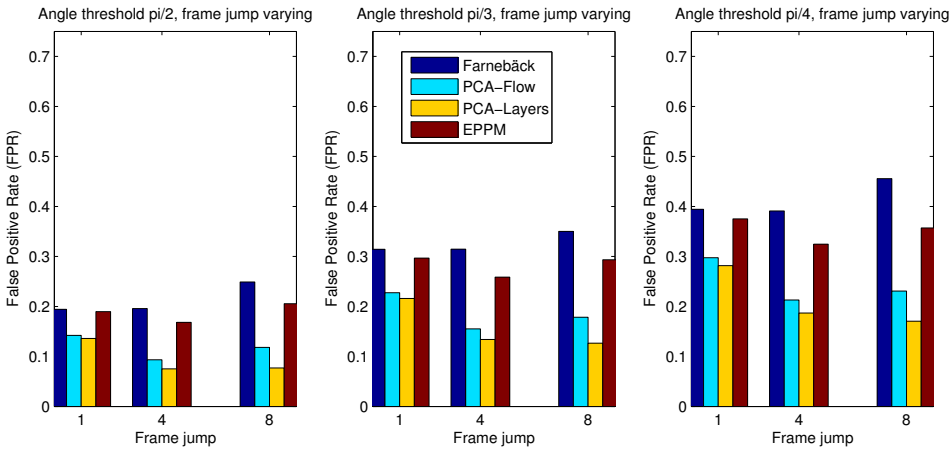

Figure 4.2: FPR for individual pixel values, showing the impact of varying the frame jump and the angle threshold. Left shows angle threshold $\pi/2$, middle shows angle threshold $\pi/3$, and right shows angle threshold $\pi/4$. For all three subplots, the frame jump is varied between 1, 4 and 8.

4.2 Varying the Angle Threshold

In this section, the ROC curves shown indicate what happens with the algorithm when the frame jump is kept constant and the angle threshold varies between the values $\pi/2$, $\pi/3$ and $\pi/4$. In these curves, angle threshold $\pi/2$ is connected to angle threshold $\pi/3$, and angle threshold $\pi/3$ is connected to angle threshold $\pi/4$. In the case of individual pixels, the endpoints of the angle threshold ranges are once again connected to coordinates $(0, 0)$ and $(1, 1)$ to make it easier to see the trend of the curves.
Figure 4.3 shows the ROC curves for individual pixels, and Figure 4.4 shows the ROC curves for binary decisions.

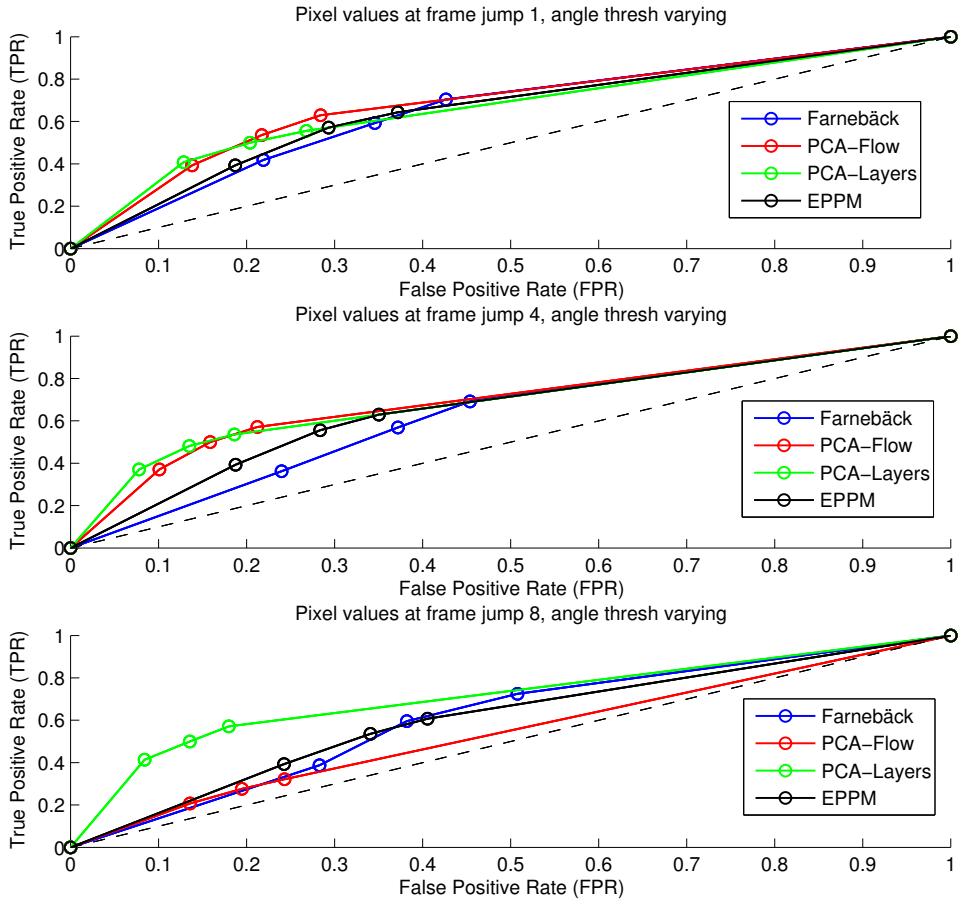

Figure 4.3: ROC curves of individual pixel values, illustrating what happens when the angle threshold varies and the frame jump is kept constant. The top figure shows frame jump 1, the middle figure shows frame jump 4, and the bottom figure shows frame jump 8. In all three figures, angle threshold are varying through the values $\pi/2, \pi/3, \pi/4$.

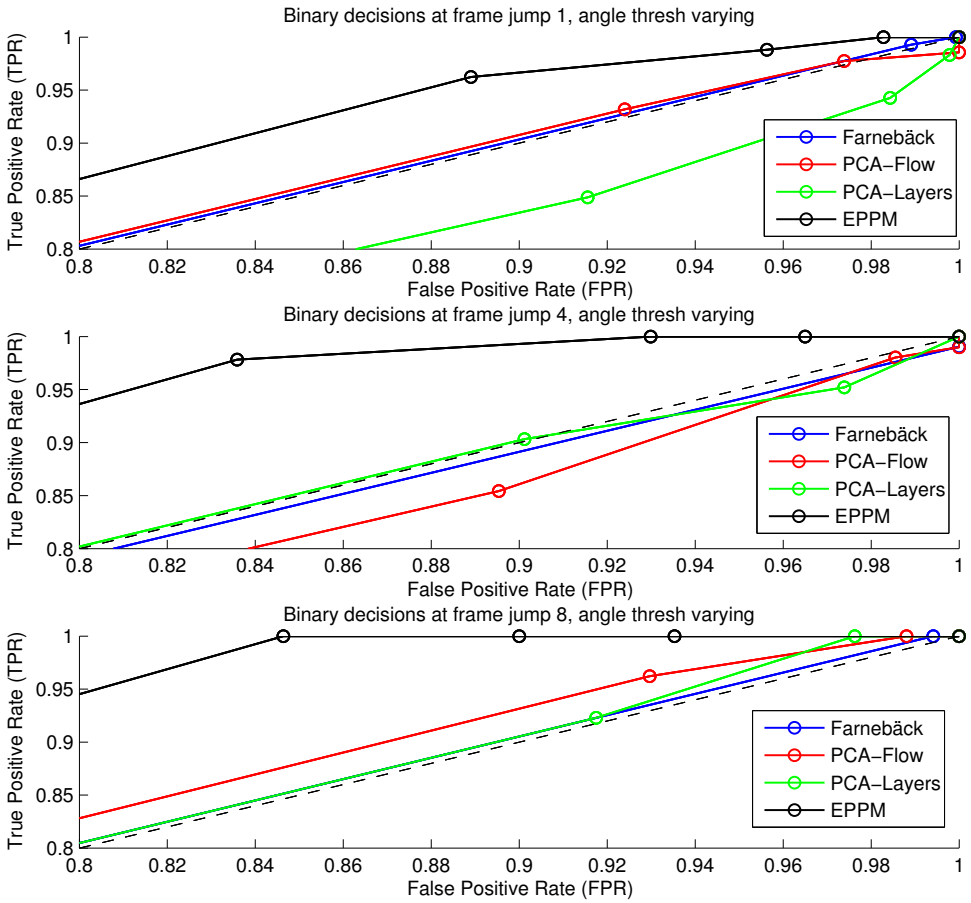

Figure 4.4: ROC curves of binary decisions, illustrating what happens when the angle threshold varies and the frame jump is kept constant. The top figure shows frame jump 1, the middle figure shows frame jump 4, and the bottom figure shows frame jump 8. In all three figures, angle threshold are varying through the values $\pi/2, \pi/3, \pi/4$. Note that the values shown are only between 0.8 and 1 for both x-axis and y-axis.

5
Discussion

In this chapter, a discussion of the presented methods and results is presented. Both practical aspects and limitations of the route recording and theoretical notes regarding optical flow and focus of expansions calculations are discussed. Finally, the future work needed in order to make the algorithm work on a real autonomous vehicle is examined

5.1 Recording Routes

The recording of the routes is done with equipment that is easily available to anyone, which fits well into the problem description as the point of the thesis is an autonomous system which utilizes low-cost hardware. The result of this is however that the videos are more or less shaky, resulting in optical flow fields that sometimes do not provide much useful information in regards to finding objects. This is because if the camera shakes violently between two given frames (for example because of a rock on the road, or unsteady handling of the bicycle from which the recording is taken place), the dominant motion is more to the side than the camera motion forward. This results in the motion model presented in section 2.3.1 being faulty, which makes the segmentation of objects of bad quality and almost always resulting in an over-segmentation of objects. The evaluation of the algorithm would probably entirely different if more focus was put on image stabilization, both in the cases of better cameras and better equipment for holding the cameras in place.

Since the route recording took place in central Gothenburg, the environment was uncontrolled and the camera had to stop moving sporadically to make place for pedestrians, cars or other moving objects. Standing entirely still provides problems with the presented algorithm, as there is no FOE present in the image in that case. This leads to either no FOE being found at all, or a faulty FOE being found, which most likely results in an over-segmentation of objects. Neither of these cases are preferable, which shows that the algorithm in its current state is not adapted to images where the optical flow vectors are all close to zero (i.e. when there is close to no camera motion).

5.2 Optical Flow

In this thesis, four different optical flow algorithms are used as basis for the recognition of when objects are present. It is clear that the results vary based on which

optical flow algorithm is used, and exactly how they vary is analyzed here.

One issue that is a clear problem in the field of optical flow is *large displacement flow*. The large displacements are caused by both fast motion of the camera and usage of higher frame jump (i.e. using a frame jump of 8 at a constant velocity would yield an average of 8 times larger magnitude of the flow field, compared to using frame jump 1). It is important that the provided optical flow algorithms can handle high velocities of the camera, but it is also important to be able to use higher values of frame jump than 1. Using a higher frame jump helps reduce the effect of shaky recordings, as the shaking of the camera is high-frequency but low-magnitude motion, while the forward motion of the camera is in most cases rather steady. This makes it more likely to find the FOE, which increases the chance of a well-performed object segmentation.

5.3 FOE Calculations

Calculating the focus of expansion using a matched filter can take a lot of time, especially if the given optical flow has many vectors with magnitude less than the magnitude threshold (cf. Algorithm 1). The filter size w also greatly impacts the time used, however in this thesis the main focus of the calculations have been to find the best FOE possible, rather than perform time-optimal computations (i.e. this report uses a rather large w to make sure the algorithm is as correct as possible). The optical flow algorithms used can also make an impact in the choice of the threshold values t and $mag\_threshold$ used in the FOE calculations, however for this report these parameters have had constant values.

5.4 Binary Decisions

The binary decisions, which are introduced in section 2.4, can be seen as the main output of the program. It is a simplified way to look at how an autonomous vehicle should avoid the obstacles that have been recognized to be present – if there are any objects present at all, the vehicle should be ordered to stop altogether. This approach may seem naïve, but the main focus of this report has been to evaluate how optical flow and FOE calculations can be used in order to recognize presence of objects, rather than to generate control input for the autonomous vehicle in itself.

As the thesis can be characterized to be about safety, the most important aspect of the different parameter values (angle threshold and frame jump) are that they produce results that indicate safety of the vehicle in first hand, and liveness of the vehicle in second hand. In this case, safety means to avoid collisions, while liveness means that the vehicle should keep moving forward. By looking at Figure 4.4, it can be seen that both the TPR and FPR of the binary decisions are close to 1 for all algorithms. The important characteristic is that the binary decisions have a high TPR, meaning that when there are object present, the algorithm manages to find them. In terms of the curves, it is therefore important that the points found are in

the top of the coordinate system (meaning a high TPR). The further to the left (i.e. the lower the FPR), the more live the vehicle will be as the decisions will not as often be ”STOP” when there in reality are no objects to avoid. In this sense, Figure 4.4 shows that using angle threshold $\pi/2$ and frame jump 8 provides the best binary decisions for the EPPM algorithm – however, since the FPR is still around 0.85, it is clear that the model used is perhaps to simple to be useful in a real situation involving a control input for an autonomous vehicle.

An example of when the algorithm manages to find an object (in this case a bicyclist) can be found in Figure 5.1. The manual analysis does in this case show and object at approximately the same position as the one found in the segmentation. An example of when the algorithm finds objects that are not there in the manual analysis (e.g. false positives) is shown in Figure 5.2. Both of these examples are created using PCA-Flow for the flow calculations, frame jump 1, and angle threshold $\pi/2$.

Figure 5.1: An illustration of when the algorithm gives a satisfactory result. Top left and top right shows frame one and frame two, respectively. Bottom left shows the optical flow field as calculated by PCA-Flow, where the FOE has been marked with a red circle with black border. Bottom right shows the segmentation result when using an angle threshold of $\pi/2$. As the frames are subsequent in the video used, the frame jump in this example is 1.

5.5 Individual Pixels of Segmentation

In Figures 4.1, 4.2 and 4.3, the characterization of the segmented pixels can be shown. For ROC curves, the ideal point for a certain parameter point to result in is the top left corner of the graph (i.e. TPR equal to 1 and FPR equal to 0). It is evident from looking at the ROC curves and bar graphs that the data presented is

Figure 5.2: An illustration of when the algorithm finds objects that are not there in the ground truth (e.g. false positives). Top left and top right shows frame one and frame two, respectively. Bottom left shows the optical flow field as calculated by PCA-Flow, where the FOE has been marked with a red circle with black border. Bottom right shows the segmentation result when using an angle threshold of $\pi/2$. As the frames are subsequent in the video used, the frame jump in this example is 1.

far from ideal. For safety aspects however, it is once again important that the TPR is high for a given combination, as this means that there is a low probability that true objects are missed in the automatic segmentation. This seems to propose that the ideal angle threshold of the ones presented is $\pi/4$ (cf. Figure 4.3), however this also shows evidence of quite much over-segmentation. The fact stands that to arrive at a decent TPR, there will be much over-segmentation due to the nature of the simple motion model presented, and the fact that only angles of optical flow vectors (and not magnitudes) are considered.

5.6 Future Work

The base assumption of the work presented in this thesis is a rather simple motion model, which also fails to deliver the results necessary to prove useful in a hypothetical environment with an autonomous vehicle. However, the work presented can be considered a basis of further work, where more ideas can be used in collaboration with the approach of using optical flow and FOE calculations to reach a desired safety system for e.g. self-driving cars.

The work in its current form makes no attempt to guess where the FOE is in the image, based on where it was in the previous calculation. Such an *a priori* estimation of the FOE could be done, if previous optical flow and FOE positions are used in conjunction with more sensors than just a camera (e.g. accelerometer and gyroscope). This might also help decreasing the time needed to perform FOE calculations, as they in the algorithms current form can take several seconds depending mostly on the magnitudes of the optical flow vectors.

Using more sensors of different kinds would also be convenient to make the algorithm able to more efficiently consider situations where there is no FOE available in the image, for example when the vehicle is standing still or it has a high turning-rate compared to the forward speed. Such an identification is crucial when aiming to construct a system that not only creates binary decisions such as ”STOP” and ”GO”, but more complex ones such as indicating in which direction potentially dangerous objects currently are moving.

Another factor which is possible to evaluate is the usage of different cameras. As the approach is centered on the safety of a vehicle, it could be interesting to use more wide-angle cameras that capture more of the environment. As of right now, the impact of different types of camera on the performance of the algorithm is still unclear.

Finally, the parameters used for this thesis could be varied through more extensive values, given the time for a more complete evaluation. The angle threshold and frame jumps parameters have many more potential values that should be examined, and parameters that have been static in this thesis (area threshold and resolution) could also be varied to see the impact it has on the final segmentation results.

6
Conclusion

In this thesis, a new approach to detecting when obstacles are present in front of an autonomous vehicle was presented. The approach utilizes optical flow and focus of expansion calculations to try and detect when the vehicle should keep moving or stop, solely based on a single camera pointing forward. The algorithm presented uses a simple motion model which characterizes flow vectors as objects if they deviate too much from a certain expected angle. The expected angle is based on the assumption that all flow vectors should diverge from the focus of expansion, meaning that objects in this case are regions in the images that move in a different way than their environment.

The presented algorithm was evaluated on routes that were recorded with standard cameras, using four different optical flow algorithms and varying parameters. The parameters used to vary the results were angle threshold, which defined how much each flow vector could deviate from its expected angle, and frame jump, which specifies with respect to how many frames in the past the optical flow calculations should be made.

The results were presented with binary classification and ROC curves, where it can be shown that more effective ways of recognition of objects were connected with lower angle threshold and higher frame jump. However, the calculations very often result in finding many false positives (i.e. finding objects where there actually are none), meaning that in reality the algorithm would not allow an autonomous vehicle to drive much at all due to it finding potentially dangerous objects in the way at almost all times.

The conclusion of these results is that the algorithm as presented in its current form is not yet ready for implementation on a real autonomous vehicle. Instead, it can be seen as a basis for future work in development of safety systems for e.g. self-driving cars. Some proposals are made regarding what are the most obvious potential improvements, such as optimizing FOE calculations for time and using more sensors than just a camera.

Bibliography

[1] Linchao Bao, Qingxiong Yang, and Hailin Jin. Fast edge-preserving patchmatch for large displacement optical flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3534–3541, 2014.

[2] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics-TOG*, 28(3):24, 2009.

[3] Didi Sazbon, Héctor Rotstein, and Ehud Rivlin. Finding the focus of expansion and estimating range using optical flow images and a matched filter. *Machine Vision and Applications*, 15(4):229–236, 2004.

[4] Kahlouche Souhila and Achour Karim. Optical flow based robot obstacle avoidance. *International Journal of Advanced Robotic Systems*, 4(1):13–16, 2007.

[5] Dong-Wan Yoo, Dae-Yeon Won, and Min-Jea Tahk. Optical flow based collision avoidance of multi-rotor uavs in urban environments. *International Journal Aeronautical and Space Sciences*, 12(3):252–259, 2011.

[6] Dirk Holz, David Droeschel, Hartmut Surmann, Stefan May, and Sven Behnke. *Fast 3D perception for collision avoidance and SLAM in domestic environments*. INTECH Open Access Publisher, 2010.

[7] Berthold K Horn and Brian G Schunck. Determining optical flow. In *1981 Technical symposium east*, pages 319–331. International Society for Optics and Photonics, 1981.

[8] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. In *IJCAI*, volume 81, pages 674–679, 1981.

[9] Thomas Brox, Christoph Bregler, and Jagannath Malik. Large displacement optical flow. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 41–48. IEEE, 2009.

[10] Frank Steinbrücker, Thomas Pock, and Daniel Cremers. Large displacement optical flow computation withoutwarping. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1609–1614. IEEE, 2009.

[11] Andrés Bruhn, Joachim Weickert, Timo Kohlberger, and Christoph Schnörr. A multigrid platform for real-time motion computation with discontinuity-preserving variational methods. *International Journal of Computer Vision*, 70(3):257–277, 2006.

[12] Christopher Zach, Thomas Pock, and Horst Bischof. A duality based approach for realtime tv-l 1 optical flow. In *Pattern Recognition*, pages 214–223. Springer, 2007.

[13] Andreas Wedel, Thomas Pock, Christopher Zach, Horst Bischof, and Daniel Cremers. An improved algorithm for tv-l 1 optical flow. In *Statistical and Geometrical Approaches to Visual Motion Analysis*, pages 23–45. Springer, 2009.

[14] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Image Analysis*, pages 363–370. Springer, 2003.

[15] Jonas Wulff and Michael J Black. Efficient sparse-to-dense optical flow estimation using a learned basis and layers. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 120–130. IEEE, 2015.

[16] Shahriar Negahdaripour and Berthold KP Horn. A direct method for locating the focus of expansion. *Computer Vision, Graphics, and Image Processing*, 46(3):303–326, 1989.

[17] Ignacio S McQuirk, Berthold KP Horn, Hae-Seung Lee, and John L Wyatt Jr. Estimating the focus of expansion in analog vlsi. *International Journal of Computer Vision*, 28(3):261–277, 1998.

[18] M Tistarelli, E Grosso, and G Sandini. Dynamic stereo in visual navigation. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on*, pages 186–193. IEEE, 1991.

[19] Chris McCarthy, Nick Barnes, and Robert Mahony. A robust docking strategy for a mobile robot using flow field divergence. *Robotics, IEEE Transactions on*, 24(4):832–842, 2008.

[20] Edward H Adelson, Charles H Anderson, James R Bergen, Peter J Burt, and Joan M Ogden. Pyramid methods in image processing. *RCA engineer*, 29(6):33–41, 1984.

[21] Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, 2011.

[22] David Fleet and Yair Weiss. Optical flow estimation. In *Handbook of Mathematical Models in Computer Vision*, pages 237–257. Springer, 2006.

[23] Xing Mei, Xun Sun, Mingcai Zhou, Shaohui Jiao, Haitao Wang, and Xiaopeng Zhang. On building an accurate stereo matching system on graphics hardware. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 467–474. IEEE, 2011.

[24] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(4):433–459, 2010.

[25] S Santhosh Kumar, Preetham Shankpal, KR Prashanth, Saima Mohan, Narasimha Reddy, Govind R Kadambi, and SR Shankapal. Simulation studies on horn and schunck optical flow algorithm and focus of expansion for autonomous navigation of unmanned vehicles.

[26] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[27] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.