Sensor Fusion on a mini Unmanned Vehicle

Integrating vision-based algorithms on an Parrot AR.Drone to autonomously follow linear shaped structures in a landscape.

Photo by: Parrot SA

A Bachelor Thesis by Camiel R. Verschoor

Sensor Fusion on a mini Unmanned Vehicle

Integrating vision-based algorithms on an Parrot AR.Drone to autonomously follow linear shaped structures in a landscape.

Camiel R. Verschoor
10017321

Bachelor thesis
Credits: 18 EC

Bachelor Opleiding Kunstmatige Intelligentie

University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

Supervisors
Dr. A. Visser

Informatics Institute
Faculty of Science
University of Amsterdam
Science Park 904
1098 XH Amsterdam

Drs. G. Poppinga

Defense Systems
Aerospace Systems
National Aerospace Lab
Anthony Fokkerweg 2
1059 CM Amsterdam

July 24th, 2012

Abstract

To be written.

Acknowledgements

I would like to thank my supervisors Arnoud Visser and Gerald Poppinga for their support and guidance. Furthermore, I am grateful to Nick Dijkshoorn for the distribution of his development framework, AR.Drone SLAM, and his support installing it. I also like to thank Robrecht Jurriaans for borrowing his AR.Drone. Likewise, I would like to thank Rob van Holstein for the construction of the 3-Dimensional printed frame of the mirror construction. I like to thank Christian Muller for helping collecting the dataset and learning me to fly a quadcopter manually. Furthermore, I am grateful to the National Aerospace Lab for the internship and the visit to the International Micro Aerial Vehicle Conference and Competitions. Lastly, I am thankful to my girlfriend and my family for their support and encouragement.

Contents

1 Introduction

In robotics one of the main goals is to develop mobile robots that can operate autonomously in the real world environment. These autonomous robots have various purposes and are used for a wide range of applications such as inspection, exploration and rescue. In rescue, robots are expected to operate in dangerous environments without putting human lives at risk. Even though reasonable developments have been made in the robotics field, robots cannot operate autonomously in the real world yet.

One of the main requirements of an autonomous robot is the ability of navigation in the operational environment. The traditional approach to navigate through the outdoor environment is via pre-planned paths based on a Global Positioning System (GPS). The main shortcoming of GPS is that it cannot be used in every environment as it needs to receive data signals from at least four different satellites [Bajaj et al., 2002]. Inside buildings and in several outdoor areas GPS is not convenient for navigation. In urban areas GPS is found to be especially unreliable. In order to navigate through these environments other sensors and navigation techniques need to be applied. Since there are several linear structures in the environment such as rivers, roads and power lines, line-following is one possible approach to navigate through an environment. Line-following is a classic technique in robotics as it has been successfully used for ground robots numerous times [Sampei et al., 1995, Dupuis and Parizeau, 2006]. For other robots and sensor configurations, open problems still remain. One of these is navigation for micro aerial vehicles (MAVs), which have a limited sensor composition due to their limited payload. For MAVs line-following is a greater challenge due to the extra dimension it can move in, comparing to the average ground robot. Where the sensors of a ground robot can rely on the stability of the ground, the sensors of a MAV have to count on the stability of the platform during a flight. Therefore, MAV are more likely to measure noisy data, which interpreting algorithms should resist.

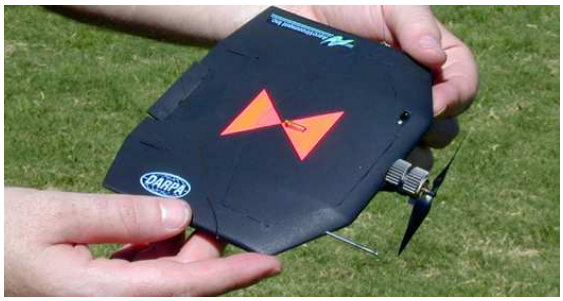

Figure 1: The Black Widow [Grasmeyer et al., 2001] was the first operating micro aerial vehicle system, a development which was finalized in 1999 by AeroVironment for Defense Advanced Research Projects Agency (DARPA). The Black Widow can fly for up to 20 minutes and carries a very small color video camera.

A micro aerial vehicle (MAV) is a subclass of the Unmanned Aerial Vehicles. Due to their small size, the MAV can operate in numerous robotic applications, for instance, search & rescue, inspection and exploration. AeroVironment Black Widow [Grasmeyer et al., 2001] (figure 1) is the first MAV operating in the field. The Black Widow has a fixed wing design with a single rotor. Another type of MAV is the quadcopter, which is controlled by four rotors. Quadcopters provide manoeuvrability and stability, which is suitable for indoor and urban flights. As a result of recent developments, small quadcopters with on-board stabilization have become more accessible,

both in terms of price and programming interfaces. Due to this, the research regarding this platform is moving towards intelligent applications, which demand information of the surrounding environment. Nevertheless, the fast movements and the limited amount of sensor combination mean that it is still a challenge to develop navigation methods for these platforms.

1.1 International Micro Aerial Vehicle Conference and Competitions

The International Micro Aerial Vehicle (IMAV)[1] conference and competitions is an initiative that attempts to share and demonstrate new MAV technology. The competitions emphasizes on flight dynamics and autonomous flight. The IMAV consists of an indoor and outdoor competition, where these aspects are extensively tested in the various challenges. The high level of autonomy is stimulated in this competition as the rules give significantly more points to teams that operate autonomous flights. One of the problems participants have to solve is autonomous navigation through the environment. Although teams are allowed to used visual aids (ie. markers) problems in this area remain. The possible contribution of this thesis to the IMAV competition is a vision-based navigation technique for following linear-shaped objects. This navigation technique can aid autonomous flights during the challenges of the IMAV.

1.2 Platform and Framework

In this thesis, the Parrot AR.Drone (see appendix C) and Ascending Technologies Pelican (see appendix E) are both considered as a platform. However, the Pelican had technological difficulties with the wireless connection. For this reason, the Parrot AR.Drone was chosen to perform during experiments. The AR.Drone (figure 2) is a wireless controlled flying quadcopter built by the French company Parrot SA[2]. The quadcopter is made of plastic and foam and is about 30 centimetres long. It carries a horizontal and a vertical camera opening the door for the development of various visual applications. The inertial measurement unit in combination with optical flow and a ultrasound sensor provide on-board stabilization during flights allowing the quadcopter to hover in the same place.

Figure 2: The Parrot AR.Drone is equipped with two cameras and several inertial sensors. The development is driven by commercial and research purposes. The small quadrotor allows remote observation of hazardous environments inaccessible for humans and ground robots.

AR.Drone SLAM [Dijkshoorn, 2012] is a development framework for the Parrot AR.Drone developed and proposed by N. Dijkshoorn. This framework contains a real-time Simultaneous Localization and Mapping (SLAM) implementation based on a down-pointing camera. Therefore, it allows a MAV to know its position and movement in the environment by generating a feature

[1] http://www.imav2012.org/
[2] http://www.parrot.com

map of the environment so the MAV can localize itself on this map. Furthermore, the framework contains a 3D mouse controller, a keyboard controller, a visual map and an elevation map. Due to the framework the robot acquires more information of the environment compared to the software of the manufacturer. This information can aid the robot in navigation. In appendix C and appendix D the platform and framework will be explained.

1.3 Research questions and objectives

In robotics one goal is to develop mobile robots that can advance robustly and truly autonomously in real world situations. One of the main requirements is the ability to navigate autonomously. Since there is no sufficient GPS signal available in urban and indoor environment, robots need to rely on other sensors.

One possible solution to solve this navigation problem is to use a line-following algorithm since there are various linear structures in the operating environment. There are various approaches for line-following navigation. Edge and motion detection have proven to be strong algorithms for the detection of objects [Bills et al., 2011, Gerke et al., 2011]. Linear structures (ie. power lines) have a specific brightness and height, therefore, edge and motion detection are suitable algorithms. Edge detection finds sharp brightness changes in a image allowing it to find edges. Motion detection finds the disparity map, the apparent motion in a image. The disparity map shows the foreground objects brighter, denoting greater motion and lesser distance. However, these detection algorithms have their weaknesses. In order to overcome these weaknesses this thesis proposes a two stage approach, which is a common approach for object recognition [Frew et al., 2004, Katrasnik et al., 2010, Gerke et al., 2011]. In the first stage edge and motion detection are combined to strengthen each other. In the second stage a Probabilistic Hough Transform (PHT) is performed to extract the line from the preprocessed image. This is a common feature extraction technique used in the field of robotics [Frew et al., 2004, Li et al., 2008, Bills et al., 2011]. Based on the output of the approach the platform will navigate according to the instructions. Furthermore, this thesis is a survey for the experimental circumstances to fairly test both algorithms for a line following task.

Therefore, the main research question is to examine how edge and motion detection can be combined to strengthen each other in a line following task. This main research question is divided up in the following sub-questions:

- What is the optimal configuration for the optical sensors of the platform to follow a line?
- What are the experimental settings that demonstrate the strength and weaknesses of both algorithms clearly?
  - What are the weaknesses of the algorithms?
  - What kind of experiments show these weaknesses?
- What is the performance and robustness of different vision-based methods to navigation over a linear structure in a indoor environment?

In this thesis the quality of an edge and motion detection algorithm will be separately examined to determine where these can strengthen each other when combined. This in order to provide an essential component to autonomous behaviour using one camera.

1.4 Outline

Chapter 2 gives an overview is given over the related research regarding line-following on unmanned aerial vehicles. Chapter 3 gives an overview of the the theory this thesis relies on. First, the Computer Vision algorithms this thesis is based on are explained and then the navigation techniques will be discussed. The approach of this thesis is discussed in chapter 4. In chapter 5 the experiments are illustrated and in chapter 6 the results will be presented and discussed. In chapter 7 the conclusion of this thesis will be presented and directions for future research will be proposed. Finally, additional information about the platform, framework and Computer Vision techniques can be found in the appendices A, B, C, D and E.

2 Related Work

This chapter gives an overview of the related research that has been done regarding autonomous navigation for unmanned aerial vehicles. Small stable quadcopters have become affordable and research on this platform is moving towards more intelligent and autonomous applications. Various autonomous navigation methods have already been investigated in the field. First the various related autonomous applications are listed and briefly described and then a argumentation is given for the approach proposed by this thesis.

2.1 Overview

The following work is related to autonomous investigation for unmanned aerial vehicles:

Corridor and Stair Following Corridor following is a task performed by robots for autonomous navigation in indoor environments (ie. the office). Recently, a new navigation method for the MAV [Bills et al., 2011] was introduced by navigating in the indoor environment based on single image perspective cues. This is in contrast with previous approaches where a 3D model is built before planning and control. This method first classifies the type of indoor environment and then the MAV navigates through the environment using vision algorithms based on perspective cues to estimate the desired direction. The environment is detected through a confidence classifier, where estimates of the stair and corridor algorithms are used to compute the confidence values. The highest confidence value, above a threshold, is deemed as the current environment. The corridor and stair algorithm both use the the Canny edge detector and Probabilistic Hough Transform to acquire the line segments as done in this thesis. The corridor algorithm then tries to determine the vanishing point in the image. The stair algorithm on the other hand tries to determine the middle of the stairs by looking at horizontal line segments of the stairs.

These vision-based methods allow MAVs to traverse corridors, stairs and corners on the basis of a small, light-weight camera, which makes them suitable for MAV with payload and power restrictions.

Power Line Inspection Power line inspection is an essential task for the maintenance of the electric grids, which is difficult due to the range of grid distributions. Over the last years rapid development has been made owing to the need for fast, accurate, safe and low-cost power line inspection [Katrasnik et al., 2010]. Important requirements for power line inspection robots are to maintain position above power lines and to navigate over them. A vision-based power line following method has already been proposed [Golightly and Jones, 2005]. This algorithm makes use of the Hough Transform to detect the line segments in the image. On the basis of the vertical line segments the position of the system is adjusted. The main advantage of power line inspection is that the MAV can make use of the redundancy in power line design as it can follow three parallel lines. Therefore, it is easier to navigate over power lines as when one of the conductors is not detected, the MAV can adjust its position according to the other two.

Another recent approach [Li et al., 2008] in power line following is to apply a Pulse-Coupled Neural Network (PCNN) to remove the background in the image. Thereafter use the Hough Transform to detect the line segments and then k-means clustering is employed to discriminate power lines from other straight lines. The PCCN filter is a biological inspired approach based on the understanding of visual cortical models of small mammals. The PCCN filter is a great potential due to the low computational costs.

Road-Following Road-Following is a task performed by robots for autonomous navigation in outdoor environments. Small autonomous aircraft [Frew et al., 2004] have already been able to follow a road based on real-time road detection and localization. The algorithm uses multiple vision-based method to detect the road and the lane markings. First the Bayesian Pixel Classifier, a advanced alternative of the HSV-colour filter, is applied on every pixel to see whether it belongs to the road. The classifier makes use of a database of RGB values of over 20000 pixels. Then connected-component analysis is used by labelling pixels in order to remove noise in the image and detect connected regions. After detecting the road in the image the lane markings are detected by the Bayesian pixel classification algorithm. Then a Hough Transformation is applied to test multiple candidate lanes. This result in a rough discretization of the lanes. Lastly, robust line fitting, least-trimmed square, is applied to finalize the position and orientation of the center lane markings. This resulted into encouraging results, however, improvements of the algorithm can still be made.

Elementary Motion Detectors A approach presented last year during the IMAV competition by the BioMAV team [Gerke et al., 2011] was to combine motion information provided by Elementary Movement Detectors (EMDs) with edge detection. EMDs are useful for UAVs due to the detection of temporal and motion effects caused by the flight of the platform. The motion information is generated by the EMD implementation [Zhang et al., 2008], which is less dependent on differences in contrast and colour. In this approach EMDs are applied to improve the image segmentation as borders of relevant objects produce higher responses. Rotational movements of the drone will lead to edge enhancements and translational movements to larger apparent motion of objects closer to the drone. The constant self motion will provide steady and a reliable source of information. Although the EMD is useful, it rarely gives a complete picture of the environment as they rely on contrast differences. Therefore, the EMDs alone are not enough to segment the image and provide the vehicle with sufficient information. However, the EMDs provide an abundance of additional information to the traditional approach. For this reason, a combination of edge detection and motion detection is used to detect objects in the environment. EMDs filter out noise cause by the edge detection algorithm due the contrast differences in the image (see figure 3). From this cleaned image objects were detected by using a Standard Hough Transform.

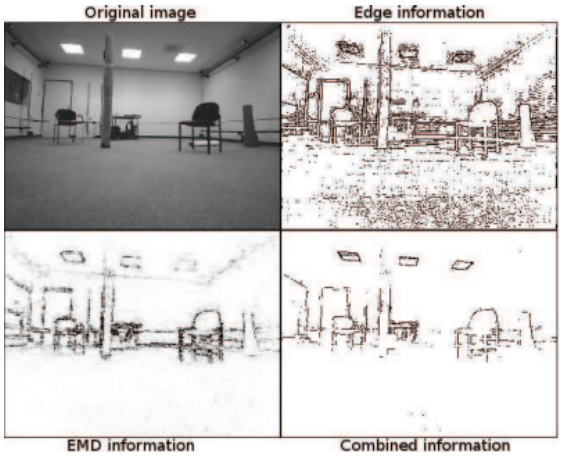

Figure 3: Example [Gerke et al., 2011] of EMD for the removal of spurious edges.

Obstacle avoidance Obstacle avoidance is a task of satisfying some control objective subject to non-collision position constraints. In obstacle avoidance the UAV requires to navigate around the obstacle to its objective. A recent approach uses a motion detection based algorithm to avoid obstacles in a corridor [Jurriaans, 2011]. The approach is based on monocular stereo vision that computes the disparity map between two images. The disparity map shows the difference in motion in the environment. Obstacles that are close by have a greater motion than objects that are further away. Due to the different distances of objects the platform is able to detect and avoid them. The proposed motion detection algorithm is discussed in section 3.2.

The above related research shows that the Hough Transform is a suitable algorithm for feature extraction as nearly all approaches use a Hough Transform stage. Furthermore, edge detection and motion detection have been suitable pre-processing techniques for successful navigation [Bills et al., 2011, Jurriaans, 2011]. In last year's IMAV competition, the combination of edge and motion detection gave promising results [Gerke et al., 2011]. Given the above facts, this thesis proposes a two stage approach, where edge and motion detection will be combined to strengthen each other and the Probabilistic Hough Transform will be used for object recognition. The next chapter will describe the theory of the techniques.

3 Theory: Vision and Navigation

In this chapter, the basics of the theory this thesis relies on is explained. The work presented in this thesis is a continuation and extension of previous research regarding vision-based algorithms on the AR.Drone [Jurriaans, 2011] and the Open Source Computer Vision Library (OpenCV) [Bradski and Kaehler, 2008], which provides real-time computer vision implementations. Lastly, the navigation techniques will be discussed that interpret the results of the vision techniques.

Over the years various Computer Vision techniques have been developed to detect objects in images. One of the fundamental problems is finding the most applicable algorithm for the defined problem. In order to follow a line in flight, fast feedback from the camera is required in order for the system to adjust its trajectory. For this reason, a real-time algorithm is favourable. Furthermore, the algorithm should be able to handle movement noise caused by the motions of the flying platform. This thesis will investigate how edge and motion detection approaches can strengthen in each others weaknesses. Both approaches performance will be tested in combination with a Probabilistic Hough Line Transform [Kiryati et al., 1991], a real-time pattern recognition algorithm to detect lines.

This chapter provides an overview of theory of the Computer Vision algorithms this thesis is based on. A clear explanation will be given over the various algorithms applied in this thesis.

3.1 Edge Detector Algorithm

Edge detection is a tool in Computer Vision that is applied to detect identifying points in a image, where the image brightness discontinuities (see figure 5). A line or linear structure is one of the elements in a image that causes these discontinuities. Due to the wide research over the years to edge detectors there are several implementations [Ziou and Tabbone, 1998]. This thesis will focus on the Canny Edge detector as it is a suitable algorithm for edge detection. The edge detector will be applied to the images in combination with a Colour Filter, which filters out the colours not in range of the filter. An alternative that can be applied instead of the Colour Filter is the Texture Filter, which filters the texture of the infrastructure. This is also a possible approach as the platform flies at a low altitude and is able to detect infrastructure. A colour or texture filter is applied for the reason that only edges containing the same colour or texture as the line need to be detected.

3.1.1 Colour Filter

To filter the colour the colour-space is converted to a different representation, namely, Hue, Saturation and Value (HSV). The HSV colour-space (in figure 4) is a simple transformation of the Red, Green, Blue (RGB) model and a intuitive colour-space, where separate colours can be easily filtered. The Hue stands for the visual sensation according to which an area appears to be similar to one of the perceived colors: red, yellow, green, and blue, or to a combination of two of them. Saturation is the colourfulness of a stimulus relative to its own brightness. Value is the brightness relative to the brightness of a similarly illuminated white. The three parameters have the following ranges: Hue has a range of 0°-360°, Saturation a range of 0-100 and Value a range 0-100.

For filtering a certain colour in a image we define the ranges of the colour that has to be filtered. Every pixel is checked whether it is within this range, if so the pixel in the result image is set to 1 otherwise to 0. This result image is a binary image, where the colours in range of the colour are set to white. Colour filtering reduces the amount of noise caused by other colours in the image.

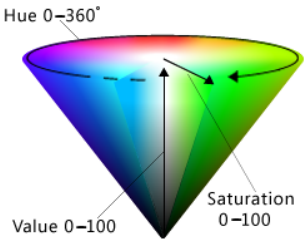

Figure 4: The HSV model, a cone having three parameters: Hue, Saturation and Value.

3.1.2 Canny Edge Detector

The Canny Edge Detector [Canny, 1986] is an edge detection operator that uses a multi-stage algorithm to detect edges. The aim of Canny was to develop a edge detector algorithm that is optimal for the following aspects:

Good detection by marking the most real edges in the image.

Good localization by marking the edges as close to the real edge.

Minimal response by only marking edges once.

Calculus of variations, a mathematical technique that finds the optimal function, was applied to fulfil these requirements. The algorithm consists of the following four stages in order to reach edge detection: Noise reduction, finding the intensity gradient, non-maximum suppression and hysteresis threshold. These stages will be explained in the next sections.

Noise reduction
Noise reduction is applied to raw image data as the edge detector is sensitive to noise. Noise reduction can be realized by using convolution with a suitable kernel. The Canny edge detector uses a Gaussian kernel as this is a suitable kernel. This phenomenon is called Gaussian Smoothing (see appendix B) and filters out the small distortions so these are not detected by the edge detector. Gaussian Smoothing results into a smoothed image.

Finding the intensity gradient
After the noise reduction the directional change in intensity, the gradient, is calculated for every pixel in the image. The gradient of every pixel in the image may point in a variety of directions. Therefore, the Canny Edge Detector uses four filters to detect horizontal, vertical and diagonal edges in the smoothed image. In order to find the gradient in the image the algorithm uses a edge detection operator, which takes the first derivative of the image that results in the gradient in horizontal direction Gx and vertical direction Gy. There are a variety of edge detection operators and the one applied in this thesis is the Sobel operator. The Sobel operator convolves (see appendix B) two 3×3 kernels with the image A to find the horizontal Gx and vertical Gy derivative, which is described in equation 1.

$$Gx = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * A \qquad Gy = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * A \tag{1}$$

From the derivatives the gradient G and direction Θ can be determined, this is illustrated in equation 2.

$$G = \sqrt{G_x^2 + G_y^2} \qquad \Theta = \arctan(\frac{G_x}{G_y}) \tag{2}$$

The directional angle is rounded to one of the four angles representing vertical, horizontal or the two diagonals in order to reduce computational costs of the algorithm.

Non-maximum suppression
Non-maximum suppression is the process of scanning a image along the image gradient direction. The pixels that are not part of the local maxima are set to zero. This suppresses all image information that is not part of the local maxima. Given the image gradient and direction estimates the algorithm seeks for the local maxima in the gradient direction by looping over the pixels of the image. For example, if the rounded gradient angle of the pixel is zero degrees, the pixel pointing downwards, the gradient magnitude is compared to the gradient magnitude of its left and right neighbour. If the gradient magnitude of the pixel is greater than its neighbours it will be considered a possible edge. Non-maximum suppression removes pixels that are not considered to be part of an edge. Therefore, only thin lines, candidate edges, will remain. This stage results in a binary image containing a set of possible edge points.

Hysteresis thresholding
The large intensity gradients are more probable to correlate to edges than small intensity gradients. For this reason, it is impossible to specify a threshold to determine whether a certain gradient is a edge or not. Therefore, the Canny Edge Detector applies another technique, namely, thresholding with hysteresis. In thresholding with hysteresis a upper and lower threshold is defined. The algorithm makes the assumption that important edges should be along continuous curves in the image, which allows the algorithm to follow a faint section of a given line and to ignore a few noisy pixels that have large gradients. Thresholding with hysteresis selects edges on the following basis:

- All candidates above the upper threshold will be selected as edges
- All candidates below the lower will be refused as edges
- All candidates between the upper and lower threshold will only be selected if it is connected to a pixel that is above the upper threshold.

This is the final step of the Canny Edge Detector and results into a binary image representing the edges in the image.

Figure 5: Left: Original image. Middle: Smoothed image. Right: Canny Edge Detector

3.2 Motion Detection Algorithm

Motion detection is a tool in Computer Vision that is applied to detect whether an object has changed its position relative to its surroundings or the surroundings have changed relative to an object. Motion can be detected by mechanical and electronic methods. This is done by various sensors such as: sound, ultrasonic and opacity. Motion can be detected electrically via optical or acoustical detection. This thesis focuses on optical motion detection from the front camera as the platform has only a limited sensor suite (see chapter C). This paper will focus on a algorithm that has been investigated for obstacle avoidance for the AR-Drone [Jurriaans, 2011]. In this approach optical flow is calculated to determine the disparity map (see figure 7) with monocular stereo vision. The disparity map indicates whether objects in the environment are close to the camera. If a object is close to the camera it will show up brighter on the disparity map as the object has greater motion indicating lesser distance. Due to the fact that in case of this thesis the platform follows a certain linear structure in the environment it can be assumed that this structure is at a certain distance. For instance, power lines hang above the ground and will be detected closer if the platform is flying over them. The disparity map is determined by using the resulting vectors of optical flow to calculate the fundamental matrix, which is needed to rectify images without calibration, using RANSAC. The fundamental matrix serves as the input for the Hartley algorithm that rectifies images. Finally, stereo matching is performed on the rectified images to generate a disparity map. This approach is explained in the following sections.

3.2.1 Shi-Thomasi

To find the optical flow in a set of images features have to be extracted from the current frame and be found in the next frame. A possibility that can provide decent features is the Shi-Tomasi algorithm [Shi and Tomasi, 1994]. It tracks corners and defines a good feature in a similar approach as Harris [Harris and Stephens, 1988], which depends on second-derivatives of image intensities. Calculating the autocorrelation of the second derivative over small windows around each point give a description of the window. According to Shi and Tomasi a good corner could be described by their eigenvalues as a good corner has eigenvalues that are greater than a minimum threshold.

As the images have a low resolution (see appendix C), the sub-pixel corner locations are calculated as the intensity peak is almost never centred on a pixel. By fitting a parabola on the intensity of the window and find the maximum of this parabola to determining the sub-pixel location.

3.2.2 Optical Flow

Optical flow is the apparent motion between an observer and the environment. A example of calculated optical flow can be seen in figure 6. The algorithm tries to find the location of each pixel from one frame in the second. However, it is a challenging task to find corresponding pixels as many have the same colour and similar surrounding pixels. In order to calculate the optical flow mostly the assumption is made there is only a small movement between frames. In sparse optical flow, the most identical features are tracked across the images. Lucas-Kanade is an algorithm that makes this assumption, nonetheless it is able to track features over any distance due its pyramidal implementation.

Lucas-Kanade Pyramidal approach
The Lucas-Kanade [Bouguet, 2000] algorithm makes the assumption that there is only small

Figure 6: Left: Frame one. Middle: Frame two. Right: Optical flow.

movement in frames in order to hold equation 3 when small windows are taken.

$$\begin{aligned} I_x(q_1)V_x + I_y(q_1)V - y &= -I_t(q_1) \\ I_x(q_2)V_x + I_y(q_2)V - y &= -I_t(q_2) \\ &\vdots \\ I_x(q_n)V_x + I_y(q_n)V - y &= -I_t(q_n) \end{aligned} \tag{3}$$

In this equation q_i stands for the pixels, $I_x(q_i)$, $I_y(q_i)$ and $I_t(q_i)$ are the partial derivatives of the image I for positions x, y and time t for q_i. These equations can be put in matrix form as is shown in equation 4.

$$\begin{aligned} Av &= b \\ A &= \begin{bmatrix} I_x(q_1) & I_y(q_1) \\ I_x(q_2) & I_y(q_2) \\ \vdots & \vdots \\ I_x(q_n) & I_y(q_n) \end{bmatrix} \\ v &= \begin{bmatrix} V_x \\ V_y \end{bmatrix} \\ b &= \begin{bmatrix} -I_t(q_1) \\ -I_t(q_2) \\ \vdots \\ -I_t(q_n) \end{bmatrix} \end{aligned} \tag{4}$$

This system is usually over-determined because it has more equations than unknowns. Lucas-Kanade algorithm obtains a compromise solution given by the least-square fit (see equation 5).

$$v = (A^T A)^{-1} A^T b \tag{5}$$

The least square fit gives the same importance to each pixel in the window. However, practice has shown that generally it is more efficient to increase the weight of the centre pixels in the window. This is done by adding a weight matrix W, which is a matrix containing all the weights for each pixel. This results in equation 6.

$$v = (A^T W A)^{-1} A^T W b \tag{6}$$

By gradually increasing the size of the window the optical flow over larger distances can be calculated as a least squares fit is found. A example of optical flow can be seen in figure 6.

3.2.3 Monocular Stereo Vision

In order to compute the disparity map with only one camera, Monocular Stereo Vision is required. Stereo vision with only one camera is possible when the assumption is made that environment is static. This means that the time between the frames is not important and that the frames have to behave as if they were taken by two cameras. This assumption limits the algorithm as any moving object will violate this. To compute the disparity map the fundamental matrix has to be estimated. In stereo vision with two cameras the configuration is fixed and it is only necessary to calculate the fundamental matrix once. However, in the case of Monocular Stereo Vision there is only one camera and because pf that it is required to find transformation between the frames for every new frame. This means that the fundamental matrix has to be found for every two frames. This is a computation complex task and difficult as for finding the fundamental matrix it is required to find points in both frames and link them together to estimate the transformation. Finding corresponding points and using them for stereo vision is known as structure from motion [Varga, 2005]. A possible approach of finding these corresponding points is optical flow, which finds the position of a pixel in the next frame.

3.2.4 Fundamental Matrix with RANSAC

The fundamental matrix relates to the correspondence of each pixel location in two frames. This means that it can give the location of each pixel in one frame to the location of that pixel in the other. Together with the physical relation between two frames, the fundamental matrix can be used to calculate the essential matrix [Hartley and Zisserman, 2004] that provides information about the physical coordinates. The essential matrix is a specialization of the fundamental matrix.

In order to determine the fundamental matrix RANdom SAmple Consensus (RANSAC) is used to discriminate outliers. RANSAC takes a subset of the points and fits a model on this subset. The model is evaluated by calculating whether the remaining points are inliers or outliers. When a sufficient amount of inliers is found the model is re-determined based on all the inliers of the model. This model is then evaluated by comparing it to other models of different iterations. The model with the smallest error is kept and will be used as model. The advantage of RANSAC is that it performs well with outliers in the data.

3.2.5 Hartley's algorithm

After determining the fundamental matrix the images can be rectified without calibrations by using Hartley's algorithm. Hartley's approach attempts to find homographies [Trucco and Verri, 1998] that map epipoles to infinity while minimizing the computed disparities. This achieved by matching point between the two frames, which implicitly contain the camera intrinsics. The limitation of Hartley's approach is that it does not calculate scale. This causes 3-Dimensional reconstructions to be determined by a projective transformation. This means that different projections of an object can appear the same to the human eye, when looking at the configuration of feature points.

Hartley first calculates the epipoles using the relations $Fe_1 = 0$ for the left epipole and $(e_r)^T F = 0$. for the right epipole. Homography H_r will map the right epipole to the 2-Dimensional homogeneous point at infinity. The homography has seven contraints since scale cannot be computed. The four degrees of freedom require only three constraint to map to infinity. The four degrees have to be selected cautiously because most choices result in distorted images. To calculate translate T that will take a selected point of interest to the origin of the

right image and the rotation R that will take the epipole to $(e_r)^T = (f, 0, 1)$ can be calculated as shown in equation 7.

$$H_r = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1/k & 01 & \end{bmatrix} RT \tag{7}$$

The homography of the left epipole can be determined by aligning the rows. Aligning by minimizing the total distance between all matching points in the frames.

3.2.6 Stereo Matching

After calculating the fundamental matrix and rectifying the images, the disparity map shown in figure 7 can be computed between the two frames. In order to compute the disparity map, a stereo matching algorithm has to be performed. This paper uses a fast one-pass stereo matching algorithm [Gutmann and Konolige, 2000]. The algorithm uses sliding sums of absolute differences between pixels in the left image, while the pixels in the right image are shifted by some varying amount of pixels.

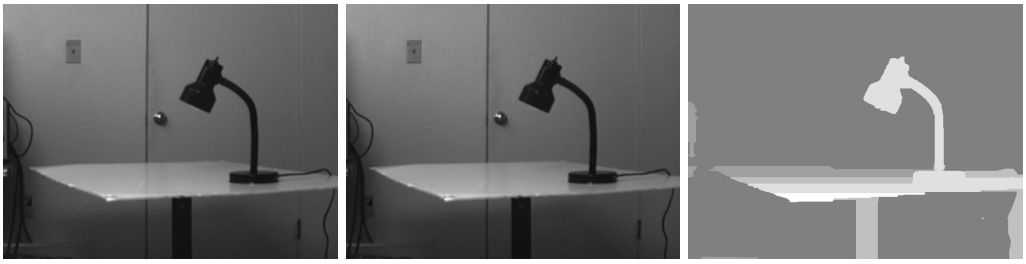

Figure 7: Left: Left frame. Middle: Right frame. Right: Disparity map. Images by Stanford University

3.3 Probabilistic Hough Transform

The Probabilistic Hough Transform (PHT) [Kiryati et al., 1991] is a feature extraction technique, which is similar to the Standard Hough Transform (SHT). This method is applied after the pre-processing stage, which is in this case is done by a edge detector or a motion detector. Due to noise caused by the image data or pre-processing stage there may be missing points on the desired curve in the image. The goal of PHT is to address this problem by grouping edges into object candidates by performing a voting procedure over a set of parametrized image objects. In PHT lines are described in polar coordinates, distance r and angle θ. In the polar coordinate system r is the length of the vector and θ is the angle between the origin and the vector pointing to the closest point on the line. The representation of the line in the polar coordinate system is as follows:

$$y = -\frac{\cos\theta}{sin\theta}x + \frac{r}{sin\theta} \tag{8}$$

Every line is associated with a pair (r, θ), which is unique when $\theta \in [0, 2\pi]$ and $r \geq 0$. For every point (ie. (x_0, y_0)) in the image the lines that are go through are the following pairs (r, θ):

$$r(\theta) = x_0 \times \cos\theta + y_0 \times \sin\theta \tag{9}$$

This results in a unique sinusoidal curve for every point in the (r, θ) plane. The sinusoidal curves of points that lay on the same line will have a joint intersection at the parameters for that line. All line candidates have to pass a threshold of intersections to be selected. Contrary to the SHT, the PHT takes only a subset of the points found at the pre-processing stage. For this reason, PHT is computationally faster.

3.4 Tracking

Tracking is the process of locating a object in multiple images. In line-following navigation it is important to receive feedback from the environment. If the visual system gives rapid feedback the system is able to react faster. For instance, when the line changes direction the platform has to adjust its angle. In order to obtain rapid visual feedback the detection method of the object should be agile.

In order to detect the line in a agile way a Region Of Interest (ROI) window is placed, once the line is found. A ROI is a selected subset of the image. The ROI is placed at the most probable place the line will appear in the next frame on the basis of the orientation of the line and motion of the platform. Since the platform is required to follow the line the ROI is frequently placed in the middle. A ROI decreases the computational costs of an algorithm as it minimizes the search window.

3.5 Navigation

After detecting a line in the image the platform is required to navigate towards the line and follow it. This process of monitoring and controlling the movement of the vehicle is called navigation. In order to navigate towards the line the trajectory has to be calculated. In order to not lose the line it is kept in the middle of the screen while navigating over it. This gives the system the following tasks:

1. Move towards the line in the correct orientation
2. Navigate over the line while making adjustments to keep it in the middle of the screen.

The line that is found gives coordinates (x_1, y_1) and (x_2, y_2). The linear equation $y = mx + b$ can be determined from these coordinates. Given the formula of the line the adjustment can be calculated, which results in a angle θ and a translation x. However, flying towards the line can be done in several ways. In this thesis the platform first flies on top of the line and then correct its orientation parallel to the line. When the platform is hovering in the correct orientation above the line it will move forward, while correcting its orientation accordingly to the line. If the platform moves to far from the line the platform will do the above step again. This is the control strategy for line-following navigation.

4 Algorithm

In this chapter the various algorithms that are used for edge and motion detection for this thesis are discussed.

4.1 Main Approach

The main approach of this thesis for line-following navigation consists of three phases. These are the following phases:

Pre-processing In the pre-processing phase, the edge and motion detection algorithms, Canny Edge Detector and Monocular Stereo Vision, are applied to the image in preparation for the feature extraction stage.

Feature extraction The feature extraction phase extracts objects, in this case lines, from the pre-processed image by using a Probabilistic Hough Transform.

Navigation In the navigation phase, the lines found are interpreted and controls are given to the platform.

The system is a closed loop system, as shown in figure 8, and will loop through these phases in order to keep on following the line. The approach is integrated in the AR.Drone SLAM [Dijkshoorn, 2012] development framework (see appendix D) and can be activated by a press on the button. In this thesis the edge and motion detection were independently implemented in order to examine their strengths and weaknesses. In the following sections these phases will be described.

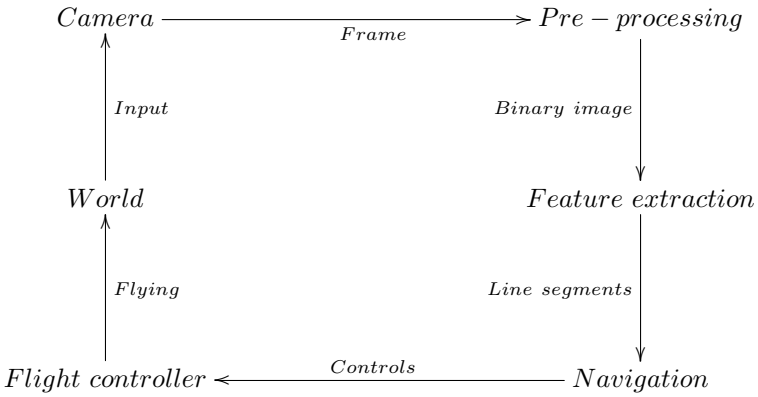

Figure 8: Schematic overview of the approach

4.2 Pre-processing

There are two types of pre-processing method implemented in this thesis. In this section the implementation of the edge and motion detection, Canny Edge Detector and Monocular Stereo Vision, will be described.

4.2.1 Edge Detection

In this thesis edge detection is implemented in the following way:

Colour Filter The colour filter is applied to the image and filters for the colour of the line. The resulting binary image is combined with the result of the Canny Edge Detector. A colour filter was chosen as this thesis focuses on simple linear structures. However, a texture filter would also be a appropriate choice since the system flies at a low altitude, textures can easily be detected.

Gaussian Smoothing Gaussian Smoothing is applied to prepare the image for the Canny Edge Detector in order to remove small distortions. The resulting image is provided to the Canny Edge Detector.

Canny Edge Detector Subsequently, the Canny Edge detector is applied to the image in order to find the edges indicated in a binary image. The Canny Edge Detector was chosen as it already was successful in previous studies regarding MAV navigation [Bills et al., 2011]. For this reason, this method is suitable to be surveyed for use in combination with motion detection.

Combining The resulting images of the Colour Filter and Canny Edge Detector are combined in a image that indicates the place of the line. Combining the two techniques filters out noisy edges.

The result of this algorithm is a binary image containing the most probable place of the line.

4.2.2 Motion Detection

Motion detection is implemented in the following way:

Finding features Shi-Tomasi and Sub-Pixel corner.

Optical Flow Lucas-Kanade.

Fundamental Matrix RANSAC.

Rectification Hartley's algorithm

Stereo Matching

Threshold

4.3 Feature Extraction

4.4 Navigation

5 Experiments

This chapter describes the methods and experiments that lead to autonomous line-following. The proposed methods and experiments aid to answer the previously determined research questions (see chapter 1.3). The various designer choices, configurations, datasets and evaluation criteria will be discussed in this chapter.

5.1 Platform

For this paper, the Ascending Technologies Pelican (see appendix E) and Parrot AR.Drone (see chapter C) were both considered for the evaluation of the vision-based algorithms. To evaluate the algorithms a stable platform with on-board stabilization was required. The Pelican has several advantages such as: on-board processing, modular design and high payload. However, the system has no indoor on-board stabilization and had technological difficulties with the wireless connection. Therefore, the Parrot AR.Drone was chosen to perform during the experiments.

5.2 Optimal Camera Configuration

In order to perform vision-based line-following navigation the optimal camera configuration has to be determined. In the current configuration of the AR.Drone it has a bottom and front camera. For optimal line-following navigation the camera has to point obliquely to the ground in front of the MAV. This for the reason that the platform should look ahead of itself so it can adjust itself on time. Neither the bottom or front camera provide this view as the bottom camera has a small field of view and the front camera looks to far ahead. This makes it impossible for the drone to navigate over a line as the platform cannot adjust itself in time to changes or not even able to detect the line. Therefore, the current configuration of the cameras is not suitable for navigation.

The optimal solution for the camera configuration would be the pan-tilt camera of the Pelican. The pan-tilt camera can change the angle of the camera and therefore suitable for flying at various altitudes. Since the experiments are only indoors the platform only flies at altitudes between 1-2 metres. Therefore, a angle of approximately 45° is considered and tested in this paper. In order to change the angle this thesis constructed and tested the following two solutions to change the angle:

Mirror construction A mirror construction that changes the view of the front camera. This construction can be placed on top of the front camera. The mirror has an angle towards the ground, which gives the camera the mirrored ahead view.

Modification of AR.Drone Modify the position of the front camera. Due to the fact that the AR.Drone is made from styrofoam its simple to modify the angle of the front camera. By cutting away styrofoam its possible to set the camera under a different angle.

5.3 Experiments

To compare the two vision-based algorithms two types of experiments have been designed. The experiments (see figure 9) challenge both the algorithms. In experiment one the weakness of the motion detection algorithm is challenged by putting a orange line on the ground, while in experiment two the edge detection algorithm is challenged by hanging a blue coloured line in the air with a blue background.

Figure 9: Left: Setup of experiment one. Right: Setup of experiment two

5.4 Evaluation Criteria

5.5 Hypothesis

- Ontwerpbeslissing voor lijn vindenn
- Optimale configuratie camera
- Opstellingen
- Evaluatie criteria
  - Frames met lijn
  - Nauwkeurigheid directie van de lijn
- Vaste datasets om goed te kunnen evaluaren zodat ik niet onnauwkeurigheid navigatie meeneem.

6 Results and Discussion

- Tabellen met resultaten
- Gemiddelde afwijking
- Standaard derivatie
- camera kijkt vooruit
- waarom resultaten zo zijn

7 Conclusion

- Optimaal algoritme bestaat niet
- Frame als indicatie gebruiken

7.1 Future works

- Combinatie algoritme
- Testen in meer environments
- Meer algoritmes testen
- Meer Criteria
- Navigatie oplossen
- Tracking
- Evaluatie criterium navigatie

Appendices

A Source code

All the source code is available at the GitHub repository:
`https://github.com/camielv/ThesisVerschoor`.
The recorded datasets are available here:
`https://github.com/camielv/ThesisVerschoor/Datasets`
A research log can be found here:
`http://camielv.nl/thesis`

B Gaussian Smoothing

This chapter provides a explanation for Gaussian Smoothing. The convolution technique is discussed first and then the Gaussian kernel of convolution is explained.

Convolution
Convolution is a mathematical way to combine two functions. The mathematical definition of convolution for the two scalar functions $f(x)$ and $g(x)$ is:

$$h(x) = \int_{-\infty}^{\infty} f(u)g(x-u)du$$

The integral expresses the amount of overlap of function $g(x)$ due the shift over the other function $f(x)$. This causes one function to blend with the other. The convolution operator contains the following algebraic properties: commutativity, associativity and distributivity. Convolution is commonly applied in computer vision in order to detect edges in a image. In figure 10 a example of a convolution with the Gaussian Kernel is illustrated.

Figure 10: Left: original. Middle and right: Gaussian Blur of 11 and 22 pixels.

Gaussian Kernel
The most suitable kernel for an optimal convolution is the Gaussian function due to the smoothing, derivatives and separability. The Gaussian function has the best smoothing properties as it has existing derivatives to any order at every point. Due to this every function can be smoothed as the derivative is always non-zero. In image processing the intensity is smooth smoothed intensity of a image this can be done by convolving the original image with the derivative of the Gaussian function. Gaussian Smoothing is the result of blurring an image by a Gaussian Function. Smoothing is applied to reduce noise and detail in the image. Gaussian smoothing is a well know technique mostly used in computer vision algorithms as a pre-processing technique in order to enhance image structures at different scales. Gaussian Smoothing is the same as convolving the image with a Gaussian function. The 2D Gaussian smoothing function is represented by the following equation:

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

C Platform: Parrot AR.Drone

One of the basic steps for the development and testing of intelligent applications in robotics is to find an applicable robot platform for the defined problem. A common choice for a Unmanned Aerial Vehicle (UAV) is to use a quadcopter. The small size and manoeuvrability allows both indoor and outdoor flights. Moreover, quadcopters have a simple design due to the fact that they do not require mechanical connections to vary the pitch angle of rotor blade. As a result of technological developments in aerospace engineering of UAV's, a small quadcopter with on-board stabilization have become more accessible, both in terms of price and programming interfaces. Because of this, research regarding this platform is moving towards more intelligent applications, which demand information of the surrounding environment. The specific platform selected for the experiments in this research is the Parrot AR.Drone quadcopter. The advantages of this platform are its on-board stabilization, lightweight and the affordable price of the platform. The AR.Drone is carrying a front and bottom camera that provide live video streaming through a Wifi data link. The platform is controlled via the same data link, which allows the user to send commands and receive data of the platform. In this chapter, the AR.Drone platform is described. The operational part of te platform is discussed, the hardware the platform contains is described and the software development kit is briefly described.

C.1 Quadcopter

A quadcopter consists of four rotors that are attached to a main frame, which commonly has a cross-shaped form (see figure 11). Every rotor produces thrust T and torque τ over the center of rotation, whereas it also produces drag force D_b in the opposite direction of flight. Thrust T is the force that is generated by increasing and decreasing acceleration the mass in one direction. The acceleration of the mass will result in a force of equal magnitude but in the opposite direction of the platform. Torque τ is the force that rotates an object around its axis. Drag D_b is the force that is in opposite direction to the motion of the aircraft through air. This force is inclined on the velocity of the quadcopter and de-acceleration will take place if insufficient thrust is generated. The rotors together should generate sufficient thrust to stay airborne during flights.

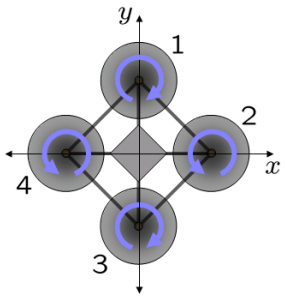

Figure 11: Diagram of the reaction torgues on each motor of the quadcopter, due to the rotors. Rotors one and three are spinning clockwise, whereas rotor two and four spin counter-clockwise, causing opposing force for control

In order to fly, the quadcopter relies on differences in thrust and torque. Pitch, roll and yaw (see figure 12) is the naming of flight dynamics to indicate the rotation angles in three dimension of the center mass of the quadcopter. The opposing rotor pairs (pair 1, 3 and pair 2, 4) turn in the same direction. One of the pairs is turning clockwise, while the other pair turns counter-clockwise. This causes the platform to have no angular acceleration, when all rotor pairs

have the same angular acceleration. Alternating the angular speed of the rotor pairs will cause angular acceleration about the yaw.

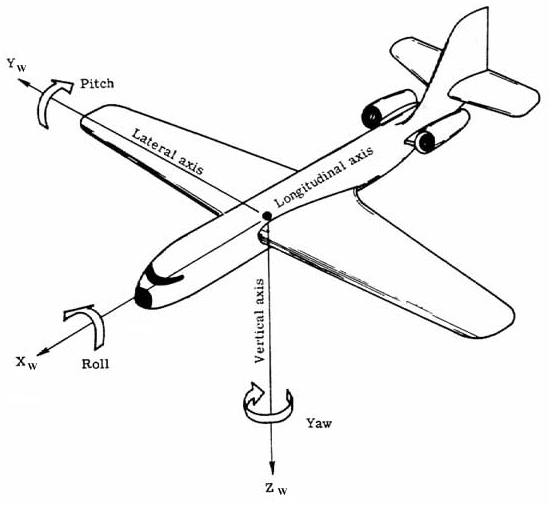

Figure 12: Diagram of pitch, roll and yaw rotations on an aerial vehicle

Vertical movements are accomplished by changing the thrust from each rotor alike, which is causing the resulting thrust to change and the differential torque to remain zero. Moreover, when the thrust is kept constant the vertical velocity remains the same. Horizontal movements are caused by changing the pitch and roll angle. Angular accelerations over the pitch and roll angles can be generated independently. Each pair of opposing rotors controls either pitch or roll rotation of the platform. A torque balance is kept for yaw stability during differential torque over the roll and pitch by increasing the speed of one rotor, while decreasing the speed of the opposing rotor.

C.2 Hardware

The AR.Drone is a remote-controlled quadcopter developed by the french company Parrot SA for consumer use. The main frame is made of carbon fibre and high resistance plastic. The AR.Drone has an indoor and outdoor hull, which is used for protection of the system. The rotors are powered by motors, whom are connected to a lithium battery allowing the system to fly approximately ten minutes.

The AR.Drone has a sensor suite containing an six degrees of freedom Inertial Measurement Unit (IMU), a bottom camera and a ultrasound altimeter used for automatic stabilization. The IMU consists of a three axis accelerometer, a two axis roll and pitch gyrometer and a single axis yaw gyrometer. The IMU reports on the system its velocity, orientation and gravitational forces. The ultrasound altimeter measure the altitude of the system and in combination with the bottom camera it calculates the optical flow of the system. All the above sensors contribute to the on-board stabilization module of the quadcopter, which allows the quadcopter to hover in one place.

C.2.1 AR.Drone 1.0

The AR.Drone has an on-board computer running a custom Linux operating system. A mini-USB connector is included on the system for software maintenance and additional external sensors (e.g. GPS sensor). The integrated wireless card provides network access for external devices that control the vehicle. The onboard software is proprietary of the manufacturer, however, an application programming interface is provided.

The horizontal camera has approximately 75° x 60° field of view and provides 640 x 480 pixel color images. The bottom camera provides color images with 176 x 144 pixels and its field of view is approximately 45° x 35° [Krajník et al., 2011].

C.2.2 AR.Drone 2.0

Recently, Parrot brought a new platform on the market, the AR.Drone 2.0. This new platform has a similar configuration as its predecessor, however, most of the components have been improved. The new on-board technology gives better stabilization and more precise sensor measurement comparing to their previous platform. The AR.Drone two carries a horizontal camera with a resolution of 1280 x 720 pixels and the vertical camera a resolution of 320 x 240 pixels. These improvements aid intelligent vision-based applications as more detailed images give more information about the environment.

C.3 Software Development Kit

Parrot created a open source Software Development Kit (SDK) providing developers the opportunity to create intelligent applications for their platform. The SDK comes with source code, multiplatform examples and documentation. The SDK does not provide software that is embedded on the AR.Drone itself. The SDK implements the following three channels of communications with the platform:

1. Configuration and control of the platform.
2. Status of the platform (ie. altitude, attitude and speed).
3. Video stream.

These four channels of communication can be used for designing intelligent applications for the AR.Drone.

D Framework: AR.Drone SLAM

In this chapter, the functionality and architecture of the development framework AR.Drone SLAM [Dijkshoorn, 2012] will be briefly described. The framework allows the performance of advanced tasks (ie. automated drone control) in real-time. The framework consists of an abstraction layer that allows the user to test methods in simulation and in real. The framework is built modular so new methods can be easily integrated into the framework.

D.1 Functionalities

The framework has the following main functionalities:

- Object-oriented: the robot and the various methods are represented by objects.
- Abstraction: the framework can be used in a real and simulated environment.
- Replay: the framework is able to record and replay datasets.
- Independent sensor and data processing.
- Various controllers: keyboard and 3D mouse.
- Autonomous way-point navigation.
- Real-time 3D map visualization of the environment.
- Simple communication with AR.Drone
- Access to Computer Vision library OpenCV.

The simulated environment the framework has access to is USARSim[3]. USARSim is is a high-fidelity simulation of robots and environments based on the Unreal Tournament game engine. In USARSim a model of the AR.Drone is available and new environments can be created easily. Therefore, it allows methods to be evaluated in various environments. Furthermore, the framework has access to the AR.Drone via the provided SDK. This gives full control over the drone.

Due to the functionalities of the framework it is suitable for developing and testing vision-based methods. This for the reason that a new module can simple be integrated into the framework. After the integration the method can then first be evaluated in simulation before testing it in real. This provides more certainty whether the developed method will work in real. Furthermore, the resources (ie. localization) the framework provides can aid the methods of navigation.

D.2 Architecture

The framework consists of several components, which are illustrated in figure 13. Here the various components will be described:

Main application this is the place, where the configurations (ie. controller and environment) are set of the programme.

Bot the main representation of a robot in the framework. The bot has access to all functionalities

[3] `http://usarsim.sourceforge.net`

Controllers the behaviour, 3D mouse and keyboard controller are implemented here and control the drone.

Interfaces the USARSim and ARDroneLib are the interfaces for the communication between the framework and the simulated or real AR.Drone.

Modules the modules sensor, frame, modules and UI implement the various implemented functionalities of the framework.

Record/Playback allows datasets to be recorded and replayed.

Integrating a new module in the framework is simple as this basically expands the functionality of the robot.

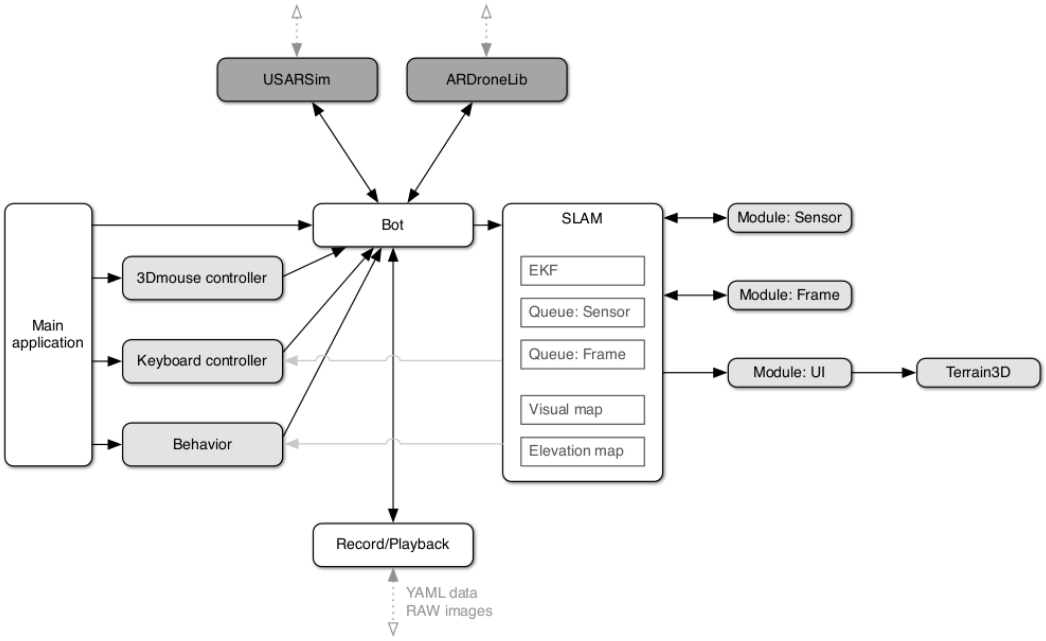

Figure 13: Schematic overview [Dijkshoorn, 2012] of the development framework AR.Drone SLAM

E Ascending Technologies Pelican

To be written.

8 Bibliography

[Bajaj et al., 2002] Bajaj, R., Ranaweera, S., and Agrawal, D. (2002). Gps: location-tracking technology. *Computer*, 35(4):92 –94.

[Bills et al., 2011] Bills, C., Chen, J., and Saxena, A. (2011). Autonomous mav flight in indoor environments using single image perspective cues. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5776 –5783.

[Bouguet, 2000] Bouguet (2000). Pyramidal implementation of the lucas kanade feature tracker. *Intel Corporation, Microprocessor Research Labs*.

[Bradski and Kaehler, 2008] Bradski, G. and Kaehler, A. (2008). *Learning OpenCV*. O'Reilly Media Inc.

[Canny, 1986] Canny, J. (1986). A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8(6):679 –698.

[Dijkshoorn, 2012] Dijkshoorn, N. (2012). Simultaneous localization and mapping with the ar.drone. Master's thesis, Universiteit van Amsterdam.

[Dupuis and Parizeau, 2006] Dupuis, J. and Parizeau, M. (2006). Evolving a vision-based line-following robot controller. In *Computer and Robot Vision, 2006. The 3rd Canadian Conference on*, page 75.

[Frew et al., 2004] Frew, E., McGee, T., Kim, Z., Xiao, X., Jackson, S., Morimoto, M., Rathinam, S., Padial, J., and Sengupta, R. (2004). Vision-based road-following using a small autonomous aircraft. (3).

[Gerke et al., 2011] Gerke, P., Langevoort, J., Lagarde, S., Bax, L., Grootswagers, T., Drenth, R.-J., Slieker, V., Vuurpijl, L., Haselager, P., Sprinkhuizen-Kuyper, I., Otterlo, M., and de Croon, G. (2011). Biomav: bio-inspired intelligence for autonomous flight. In *Proceedings of the International Micro Air Vehicle conference and competitions, 12-15 September (IMAV 2011)*.

[Golightly and Jones, 2005] Golightly, I. and Jones, D. (2005). Visual control of an unmanned aerial vehicle for power line inspection. In *Advanced Robotics, 2005. ICAR '05. Proceedings., 12th International Conference on*, pages 288 –295.

[Grasmeyer et al., 2001] Grasmeyer, J. M., Keennon, M. T., and Inc, A. (2001). Development of the black widow micro air vehicle. In *39th AIAA Aerospace Sciences Meeting and Exhibit*.

[Gutmann and Konolige, 2000] Gutmann, J.-S. and Konolige, K. (2000). Incremental mapping of large cyclic environments. pages 318–325.

[Harris and Stephens, 1988] Harris, C. and Stephens, M. (1988). A Combined Corner and Edge Detection. In *Proceedings of The Fourth Alvey Vision Conference*, pages 147–151.

[Hartley and Zisserman, 2004] Hartley, R. I. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition.

[Jurriaans, 2011] Jurriaans, R. (2011). Flow based obstacle avoidance for real world autonomous aerial navigation tasks. Bachelor's thesis, Universiteit van Amsterdam.

[Katrasnik et al., 2010] Katrasnik, J., Pernus, F., and Likar, B. (2010). A survey of mobile robots for distribution power line inspection. *Power Delivery, IEEE Transactions on*, 25(1):485 –493.

[Kiryati et al., 1991] Kiryati, N., Eldar, Y., and Bruckstein, A. (1991). A probabilistic hough transform. *Pattern Recognition*, 24(4):303 – 316.

[Krajník et al., 2011] Krajník, T., Vonásek, V., Fišer, D., and Faigl, J. (2011). AR-Drone as a Robotic Platform for Research and Education. In *International Conference on Research and Education in Robotics*, Heidelberg. Springer.

[Li et al., 2008] Li, Z., Liu, Y., Hayward, R., Zhang, J., and Cai, J. (2008). Knowledge-based power line detection for uav surveillance and inspection systems. In *Image and Vision Computing New Zealand, 2008. IVCNZ 2008. 23rd International Conference*, pages 1 –6.

[Sampei et al., 1995] Sampei, M., Tamura, T., Kobayashi, T., and Shibui, N. (1995). Arbitrary path tracking control of articulated vehicles using nonlinear control theory. *Control Systems Technology, IEEE Transactions on*, 3(1):125 –131.

[Shi and Tomasi, 1994] Shi, J. and Tomasi, C. (1994). Good features to track.

[Trucco and Verri, 1998] Trucco, E. and Verri, A. (1998). *Introductory Techniques for 3-D Computer Vision*. Prentice Hall PTR, Upper Saddle River, NJ, USA.

[Varga, 2005] Varga, M. (2005). *Practical Image Processing And Computer Vision*. Halsted Press, New York, NY, USA.

[Zhang et al., 2008] Zhang, T., Wu, H., Borst, E., Kühnlenz, K., Buss, M., and München, T. U. (2008). An fpga implementation of insect-inspired motion detector for high-speed vision systems. In *IEEE International Conference on Robotics and Automation (ICRA*, pages 335–340.

[Ziou and Tabbone, 1998] Ziou, D. and Tabbone, S. (1998). Edge detection techniques - an overview. *International Journal of Pattern Recognition and Image Analysis*, 8:537–559.