Stereo Vision for Flapping Wing MAVs

Design of an Obstacle Avoidance system

S. Tijmons

October 17, 2012

Stereo Vision for Flapping Wing MAVs

Design of an Obstacle Avoidance system

Master of Science Thesis

For obtaining the degree of Master of Science in Aerospace Engineering
at Delft University of Technology

S. Tijmons

October 17, 2012

Faculty of Aerospace Engineering · Delft University of Technology

Delft University Of Technology
Department Of
Control and Simulation

The undersigned hereby certify that they have read and recommend to the Faculty of Aerospace Engineering for acceptance a thesis entitled **"Stereo Vision for Flapping Wing MAVs"** by **S. Tijmons** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: October 17, 2012

Readers:

Prof.dr.ir. J. A. Mulder

Dr.ir. G. C. H. E. de Croon

Dr.ir. E. van Kampen

Ir. B. D. W. Remes

Abstract

In the field of Micro Air Vehicle (MAV) research the use of flapping wings attracts a lot of interest. The potential of flapping wings lies in their efficiency at small scales and their large flight envelope with a single configuration. They have the possibility of performing both energy efficient long distance flights as well as hovering flights. Most studies on Flapping Wing MAVs (FWMAVs) have focused on the design of the airframe and making them able to fly. Currently, the state-of-the-art permits investigation of the necessary autonomous flight capabilities of FWMAVs. Most previous studies have made important preliminary steps by using external cameras or an onboard camera with the FWMAV flying in a modified environment. However, since autonomy is most useful for flight in unknown areas, it will be necessary to use an onboard camera while flying in unmodified environments. Research in this direction has been performed on the DelFly. In particular, the well-known cue of optic flow was found to be rather unreliable for the determination of 3D distances, and it was complemented by a novel visual appearance cue. Since the combination of these cues may still not be sufficient for robust and long-term obstacle avoidance, this study focuses on a different well-known method to extract 3D information on the environment: stereo vision. The potential advantage of stereo vision over optic flow is that it can provide instantaneous distance estimates, implying a reduced dependence on the complex camera movements during flapping flight. The goal is to employ stereo vision in a computationally efficient way in order to achieve obstacle avoidance. The focus of this study is on using heading control for this task.

Four main contributions are made:

The first contribution comprises an extensive study on literature in the field of computational stereo vision. This research has been done for decades and a lot of methods were developed. These mainly focus on optimizing the quality of the results, while disregarding computational complexity. In this study the focus was on finding one or more time efficient methods that give sufficient quality to perform robust obstacle avoidance. It was concluded that Semi-Global Matching is a good candidate.

The second contribution is that for the first time it has been investigated what the requirements are for a stereo vision system to do successful stereo vision-based obstacle avoidance on FWMAVs. In order to achieve accurate stereo vision results, both hardware and software aspects are found to be of importance. FWMAVs can carry only a small amount of payload and therefore there is a large restriction on sensor weight.

The third contribution is the development of a systematical way to use the 3D information extracted by the stereo vision algorithm in order to find a guaranteed collision-free flight path. The focus was on dealing with the limited maneuverability of the MAV and the limited view angle of the camera.

The fourth contribution is in giving an indication on the usefulness of stereo vision based on multiple experiments. These focus on determining the accuracy of the obstacle detection method as well as on validating the functionality of the obstacle avoidance strategy.

The designed system proved to be successful for the task of obstacle avoidance with FWMAVs. The DelFly II successfully avoided the walls in an indoor office space of 7.3×8.2m for more than 72 seconds. This is a considerable improvement over previous monocular solutions. Since even reasonable obstacle detection could be performed for low-textured white walls, the experiments clearly show the potential of stereo vision for obstacle avoidance of FWMAVs. In combination with existing methods for speed and height control the proposed system has the potential of making fully autonomous (flapping wing) MAVs possible.

Acronyms

BRIEF	Binary Robust Independent Elementary Features
DARPA	Defense Advanced Research Projects Agency
FAST	Features from Accelerated Segment Test
FWMAV	Flapping Wing MAV
IMAV	International Micro Air Vehicle Conference and Flight Competition
IMU	Inertial Measurement Unit
MAV	Micro Air Vehicle
SAD	Sum of Absolute Differences
SGM	Semi-Global Matching
SIFT	Scale-Invariant Feature Transform
SKB	Semantic Kernels Binarized
SSD	Sum of Squared Differences
SURF	Speeded Up Robust Feature
UAV	Unmanned Air Vehicle

Contents

List of Figures

Chapter 1

Introduction

MAVs form a class of Unmanned Air Vehicles (UAVs) that is characterized by its small size. MAVs potentially have many new applications, especially because they can operate indoors. Therefore it has the interest of researchers, industries, governments and military and development is fast.

The small size and the ability to fly indoors allows MAVs to perform tasks that humans can or should not do. They can reach places with narrow passages as well as places that are too dangerous to enter for humans. Several important tasks could potentially be carried out by MAVs instead of humans, such as investigating buildings that might collapse or performing reconnaissance for military personnel in hostile territory. Though these applications are promising, research is currently still needed to develop MAVs further to put these promises into practice.

One major facilitator for any serious applications of MAV is the ability to fly autonomously in unknown environments. This explains the large interest for this field of research. For example, the United States Defense Advanced Research Projects Agency (DARPA) has been running projects on this subject (Benchergui, 2009). Every year the International Micro Air Vehicle Conference and Flight Competition (IMAV) is held in different countries where participants from all over the world disseminate and demonstrate advances in the development of MAVs (`http://www.imav2012.org`).

MAVs come in various configurations with the most common types having multiple rotors to enable high maneuverability and controllability. Several autonomy tasks, including obstacle avoidance, have been performed successfully with these platforms. This was mainly done by using laser range scanners for obstacle detection in combination with an Inertial Measurement Unit (IMU) for pose estimation. Acoustic range sensors have also been applied, mainly in combination with other sensors.

Research in MAVs is focused on getting to insect-sized systems. It has therefore been tried to copy the flight techniques of insects. Flapping wings gain a lot of interest because it is known that they can be applied efficiently at small scales and they allow for a large flight envelope with a single configuration. They enable energy efficient long distance flights as well as hovering flights. These reasons give FWMAVs a lot of potential and are an incentive for the DelFly project.

DelFly started as a project where a group of students managed to design the DelFly I as shown in figure 1-1, a small ornithopter (aircraft with flapping wings) that was capable of flying and also carrying a camera system to send onboard video to the ground. The idea behind the DelFly project was to start with a flying platform that should gradually be improved in order to size it down further and further. This approach let to the design of DelFly II, being much smaller (see figure 1-1) and lighter. With camera and transmitter on board this ornithopter weighs only 16 grams and fits in a 30 cm-diameter sphere. Depending on its configuration it can hover, fly forward with 15 cm/s or fly backward with 0.5 m/s. More recently the DelFly Micro was developed, an even smaller version (see figure 1-1) weighing only 3 grams and having a wingspan of 10 cm.

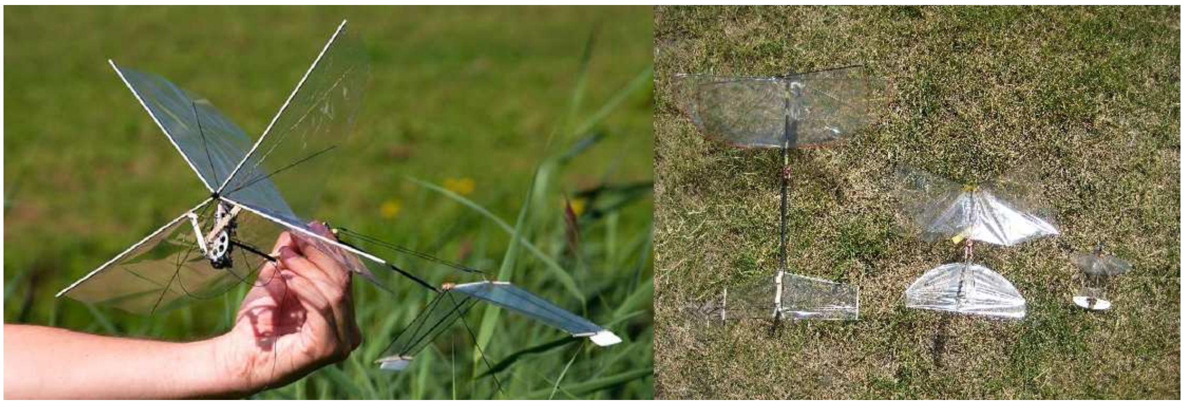


Figure 1-1: Achievements in the DelFly project. Left: One of the earliest versions of DelFly I. Right: Size comparison of the DelFly family, from left to right: DelFly I, DelFly II, DelFly Micro

1-1 Motivation

In many MAV applications teleoperation is not an option, especially when flying indoors. The signal is easily obstructed by walls or the distance between the operator and the vehicle is too large. Autonomy is in such cases a prerequisite. Many MAVs are meant to operate in indoor environments, which are complex because they can be unknown, unstructured and compact and in general they are GPS-denied. Autonomous obstacle avoidance is therefore necessary to let MAVs perform their tasks in a successful way and to prevent damage due to collisions.

While progression is made in making FWMAVs small, this has the disadvantage that they can carry only a small amount of payload. Together with the flapping motion it becomes more challenging to perform autonomous tasks. Sensors should therefore be used efficiently: they should give a lot of information compared to their size and weight. For that reason cameras are often used since a lot of information can be extracted from them while they are energy efficient and can have small size and weight.

There are several techniques to perform autonomy tasks that have been applied successfully to MAVs. Some techniques rely on laser or acoustic range sensors, instruments that are either too heavy or inaccurate to be carried onboard FWMAVs. It will be discussed in the next section that other methods, that make use of camera's, proved to be less useful because of limited camera quality and unwanted platform motion and vibrations. Optic flow, a method that can also be applied for obstacle detection, is still not usable for FWMAVs. To perform good optic flow measurements the camera images should be noiseless and rotation rates should be known, requiring three gyroscopes that can measure the vibrations of the vehicle. Measurements should be performed onboard, but the amount of onboard processing power is currently too limited. Therefore the video signal is sent to a ground station, which implies a low frame rate. This results in large image distortions that affect the optic flow quality. These problems are circumvented when using stereo vision. For optic flow image sequences are used, while stereo vision uses images taken at the same time. Vehicle motion has therefore a far more limited effect on the calculations and the video frame rate is of no importance for the quality. Furthermore, it gives an instant overview of obstacles in sight of the camera.

The main research question can therefore be formulated as:

How can stereo vision be employed in an efficient way to achieve obstacle avoidance on a flapping wing MAV?

For this study the current version of DelFly II will be used. Its design will be discussed in chapter 4.

1-2 Previous Research

Autonomous flight of FWMAVs has been studied before in various ways. Some approaches were only tested in computer simulations and were not implemented in real systems. More significant research where real tests were performed will be discussed here. (Hines, Arabagi, & Sitti, 2011) designed a FWMAV that is currently not able to fly on its own, but experiments showed it is able to control its pitch and roll angle by using actuators that change the wing shape. These angles can be controlled individually since there is no coupling. The roll control performance showed satisfying results. For pitch control it was demonstrated that enough torque could be generated in both directions for control, but the overall system response was quite poor. (Lin, Hsiao, Chen, & Shen, 2009) show the altitude control of the 10 gram FWMAV called *Golden Snitch*. No onboard processing or sensing was done. Using an external stereo camera the position of the vehicle could be determined, and further control was done by a ground station to successfully control its altitude. (Hsiao, Hsu, Chen, Yang, & Shen, 2012) is a follow up article about an 8 gram *Golden Snitch* where external stereo cameras are used to obtain flight information. The authors state that DelFly II is also only controlled using external cameras. This statement is false as will be shown in the next paragraph. (Duhamel, Perez-Arancibia, Barrows, & Wood, 2012) present an experiment with a 101 milligram flapping wing microrobot called *RoboBee*. Using an onboard optic flow sensor and a well textured screen, the altitude could successfully be controlled offboard in a closed-loop experiment, with only small oscillations and a slight drift. A video of the experiment can be found at `http://micro.seas.harvard.edu/ICRA2012/S1.mp4`. (Baek & Fearing, 2010) performed closed-loop altitude control on a 12 gram ornithopter by using an external camera. In a follow up on this research (Baek, Garcia Bermudez, & Fearing, 2011) a 13 gram ornithopter was able to fly autonomously to a target, using an onboard infrared sensor for target tracking and 3-axis gyroscopes for attitude estimation. During 20 trials a success rate of 85% was reached. (Garcia Bermudez & Fearing, 2009) performed optic flow measurements on a 7 gram ornithopter. Heavily down sampled onboard camera images were stored onboard during flight, and offloaded to a computer afterwards to compute optic flow. The main finding was that there is a strong coupling between body motion and the sensed optic flow. (Tedrake, Jackowski, Cory, Roberts, & Hoburg, 2006) realized autonomous flight of an ornithopter with a 2-meter wingspan. Only pitch control was tested successfully using IMU data. Yaw control was not implemented and altitude control using a sonar range sensor was too unreliable.

With DelFly II several autonomy experiments have been performed dealing with various control tasks. During the first experiment an external camera was used to successfully perform closed-loop altitude control (De Croon, De Clerq, Ruijsink, Remes, & De Wagter, 2009). In the next experiment both height and direction control were tested by making use of a predefined trajectory that was clearly visualized using white sheets of paper on the ground. Onboard camera images were processed by a ground station. Using the ground track and optic flow computations the altitude and the relative position and heading could be estimated to control both altitude and heading. The third experiment focused on height control only, but now without preparing a track or being in need of a room that satisfies certain conditions. The algorithm needed to be applicable to any room big enough to fly around the DelFly. However, it was necessary to perform a training round in order to 'teach' DelFly what it can expect to see at different heights. During flight the images captured using a forward-looking camera were used to perform a so called *texton* method; an image is classified as belonging to a certain height based on comparing its texture with the images from the training round. The tests performed turned out to be successful: the DelFly was able avoid the ceiling an the floor up to the point where the battery ran out.

In (De Croon, De Weerdt, Ruijsink, Remes, & De Wagter, 2012) a novel appearance cue for obstacle avoidance is introduced. It is based on the principle that when an object is approached, it colors and detailed texture become more and more visible, while other objects move out of sight. Since color and detailed texture of one object typically vary less than the colors and textures of many different objects, the variation in colors and/or textures decreases towards impact. It was shown that this cue is a useful complement to optical flow for detecting obstacles with the DelFly. The method makes use of random sampling (De Croon, De Wagter, Remes, & Ruijsink, 2011), where only a subset of all image samples is processed. This leads to a significant reduction in computational effort at the cost of only a slightly lower accuracy. This combination of methods mainly focused on obstacle detection. In (De Croon, Groen, et al., 2012) this combination of methods was extended with an algorithm to decide how detected obstacle needed to be avoided by either turning left or right. Experiments showed that the DelFly was able to fly around in a small room (3.20×2.70×2.80 m) for half a minute without colliding to the walls. Eventually a wall collision occurred during the experiments because of a wrong decision. This was mainly caused by the small size of the test space, in which the DelFly spends a lot of time in making turns. Decision making is the most reliable when the DelFly is flying straight. Right after a turn, measurements might be considered reliable by mistake, leading to wrong decisions. Apart from the obstacle avoidance task also closed-loop height control was tested using an onboard pressure sensor. Variations in the pressure measurements due to wing flapping were successfully filtered. The height of the

DelFly could be kept within a range of ± 40 cm by the controller.

1-3 Stereo Vision

In this study it is tried to give FWMAVs more autonomy by making them able to detect and avoid obstacles. As discussed previously, several methods that have been applied to MAVs can not be used on FWMAVs. The sensors used for these methods are either too heavy or inaccurate, or their performance proved to be insufficient, because of the flapping motion.

Another way of detecting obstacles is widely seen in nature: using two eyes, or two cameras, that look in the same direction but from different view points. This is also one of the ways in which humans perceive depth. When looking at a scene, we see objects in this scene from two different view points. Inversely, our eyes see these objects at two different locations. This principle is illustrated in figure 1-2. The tree and the blue pion are bot shifted when comparing the left and right camera view. Because the pion is closer to the cameras, its seems to be shifted further than the tree. The amount of shift is therefore an indication of the distance to an object.

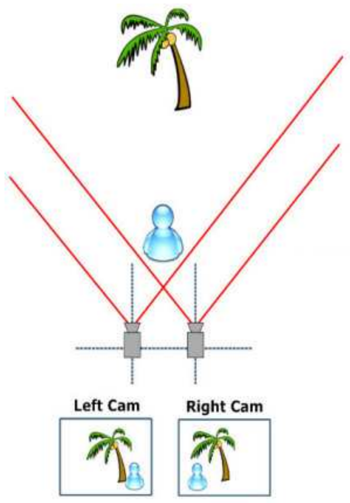


Figure 1-2: An illustration of stereopsis. The distance to an object determines how far it shifts in the view. (IMEC, 2007)

Computer stereo vision is the extraction of 3D information from digital images. In general this implies that images from two or more cameras are evaluated by an algorithm that tries to compute which pixels correspond to the same physical object. This principle is illustrated in figure 1-3. In the case of dense stereo vision, matching is done for all image pixels. When this matching is done, it is known for each pixel how large it is shifted in other images. By knowing the characteristics of the cameras these shifts, which are in literature are denoted as 'disparities' (as in figure 1-3), can be converted to real xyz-coordinates. By using all image pixels together a 3D reconstruction of the scene can be constructed.

The task of stereo viewing is easy for humans, but remains challenging for computers. A lot of studies have been done on stereo vision and this resulted in several methods to do robust

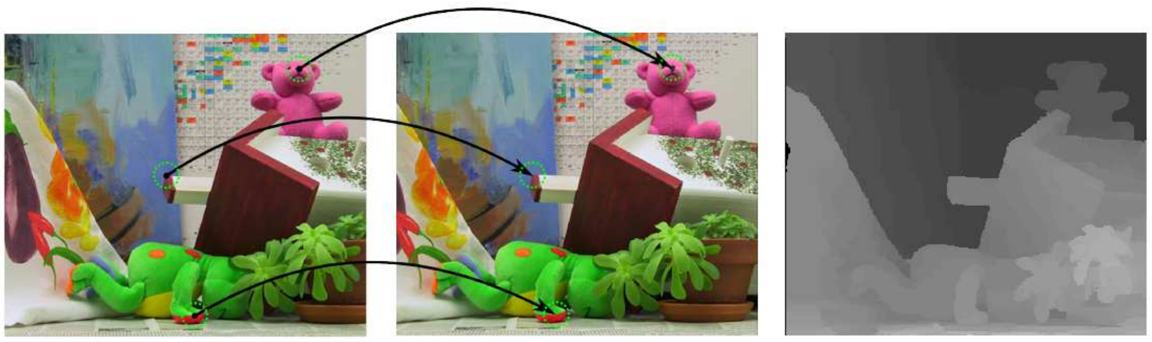


Figure 1-3: The principle of stereo vision: matching corresponding pixels between different images. The matching results produce the disparity map as shown on the right. Brighter pixels represent closer object points. (`vision.middlebury.edu/stereo`)

stereo matching. The quality that has been reached is promising. Still, low textured scenes, occlusions and poor camera performance (noise, resolution) remain problematic issues. As with a lot of other image processing techniques, when doing stereo processing it is not only the resulting quality that is important, but also the speed at which this result is obtained. Humans can quickly react to dangerous situations by using the information from their eyes. Also in robotics reaction speed is often a key factor in succeeding a task. Real-time methods have therefore been one of the focuses in stereo vision research.

1-4 Thesis Outline

This thesis is divided into two parts. The first part deals with stereo vision for obstacle avoidance. In Chapter 2 the technique of stereo vision is discussed. An overview is presented about the several types of stereo vision methods that are available and it is explained which method has been chosen for this project. Chapter 3 follows with the basic theory about camera calibration for the purpose of stereo vision. Chapter 4 gives a detailed description of the DelFly test platform. Both the design of the air and ground system are discussed in terms of specifications and limitations. In Chapter 5 the performance of the stereo vision system in combination with the DelFly is shown based on static tests and flight tests.

The second part deals with obstacle avoidance. Using the system described in the first part several experiments were conducted to test its applicability for obstacle avoidance. Chapter 6 describes how the experiments were performed and which different strategies were tried to perform reliable avoidance. The results are presented in Chapter 7, the conclusions are drawn in Chapter 8. Finally recommendations for future work are given in Chapter 9.

Part I

Stereo Vision algorithm and implementation

Chapter 2

Stereo Vision

As described in the introduction, computer stereo vision is based on a relatively simple principle: matching of image pixels from two or more images that are taken from different view points. When it is known which pixels match, the difference in pixel coordinates from these matches is a measure for the distance to the object these pixels correspond to. A considerable amount of research has been done for decades on the problem of computational stereo vision, with the first publications dating back to 1970s (Sobel & SCIENCE., 1970). Research is still being performed to improve the properties of quality and speed of computational stereo vision algorithms. Of course, these two properties are at a tradeoff with each other, but they are also limited individually. This will be explained in the next section. Further on in this chapter a concise overview of computer stereo vision methods that have been developed over the years is presented in which the focus is put on implementability in real-time systems. This overview is based on the literature research presented in the Preliminary Thesis which the reader is referred to for more detailed information. Based on these findings it will be explained which methods have the potential of being used in this project and which method has been selected for further research in this project.

2-1 Challenges

Several aspects complicate the process of stereo vision. These will be explained in short here.

Real-Time Performance
Obstacle avoidance on a flying vehicle requires real-time performance. For example, when the vehicle is making a turn it is essential to obtain quickly new information about obstacles in the changing direction of flight. This limits the amount of total computations that can be done on a stereo image pair. Some research in computational stereo vision is therefore focused on finding the most efficient strategy for matching.

Noise
Image noise is a complicating factor since it changes the appearance of image features or even makes them disappear. Several sources of noise can appear in images. In cameras electronic

noise occurs, leading to variations in brightness and color. This also strongly depends on the light conditions the cameras are used in. A more severe type of noise can occur when the stereo images belong to a video signal that is transmitted wireless. This type of noise depends mainly on the frequency of the video signal. If this frequency is close to that of other sources nearby, complete (bands of) image rows can become unrecognizable, as is shown in figure 2-1. Some stereo vision methods are specifically designed to give robustness to certain types of noise. Preprocessing techniques, such as filtering and noise detection might also be applied to reduce the effect of noise.

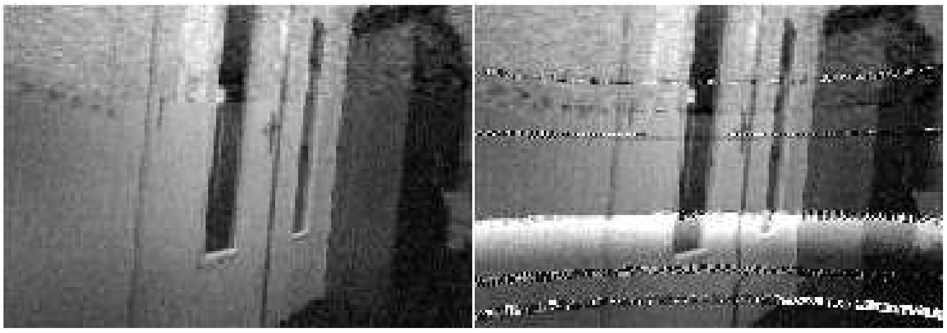


Figure 2-1: Example of noise due to an interfering source.

Sensitivity to light
Cameras are in general calibrated such that image brightness is regulated. However, when using two (identical) cameras for stereo vision, there might still be a small difference in brightness for the same image features (see figures 2-3 and 2-4).

Reflections
Reflections of light can lead to matching errors. Dealing with reflections is hard since it changes the appearance of objects drastically and locally. Figure 2-2 shows an example of reflections on a floor. In this case the floor acts as a mirror and the reflections make the floor appear far away.

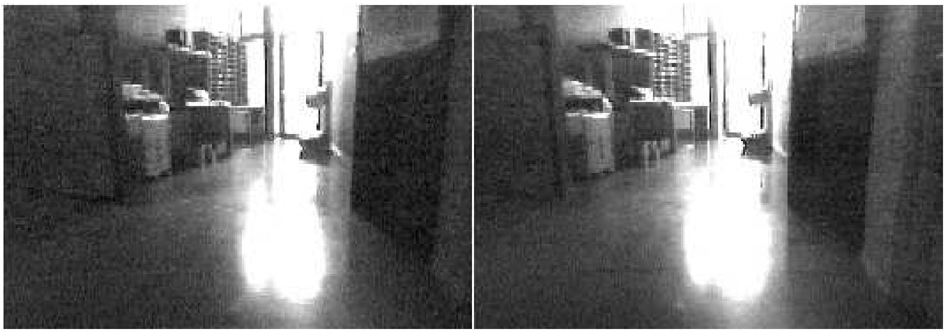


Figure 2-2: Example of light reflection on a floor.

Perspective distortions
Image features can have different proportions depending on the point of view, especially when they are close to the cameras. Cameras looking at a circle might show it as an ellipse. Cameras at different view points will show ellipses with different eccentricity. This type of distortion is also called 'foreshortening'. A more extreme example: an opened door that points into the

direction of the camera will be seen from its side and appear as a thin edge. Another camera a few centimeters to the side will also see the front of the door as a panel. This panel will have no match in the image from the other camera. This is illustrated by figure 2-3.

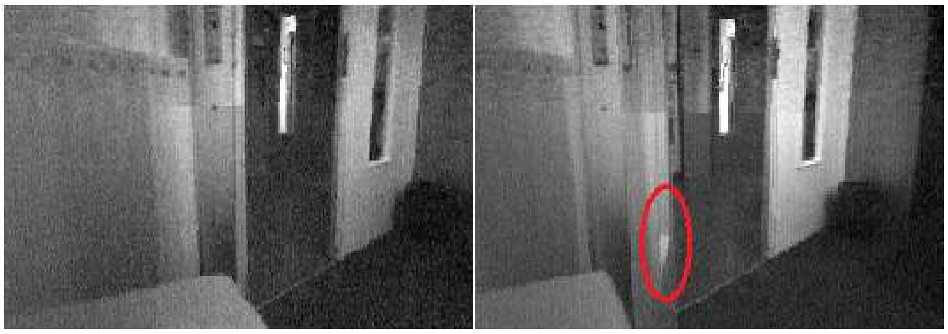


Figure 2-3: Example of perspective distortion. The red circle indicates a window frame in the opened door that is only visible for the right camera

Lack of features
Without image features, no matches are found. For example, a plain wall is typically problematic. While humans would still see very fine details, the resolution of cameras is in many cases not able to detect these. Especially not in cases where the resolution is kept low to keep processing time under certain limits. Except for this extreme condition, also relatively structured environments might contain texture poor regions. In these cases it is impossible to determine precisely which pixels in these regions actually correspond to each other. Only groups of pixels can be matched in these cases. It is mainly the strategy to this problem that distinguishes stereo matching methods. More on this will follow in the next section.

Repetitive features
Similar image features close to each other might be mixed up. If two stereo images would both show only part of a chessboard containing a lot of very small fields, even a human would not be able to decide which fields in the two images would correspond.

Occlusion
If one object is placed before another object and this situation is seen from two different view points, the following effect can occur: a feature on the far object seen from one view point might be blocked by the close object when seen from another view point. In this case some image features will only appear in one image. Some stereo vision methods use a specific strategy to handle this problem. Figure 2-4 shows an example of an occluded region.

Discontinuity
Pixels in an image might have neighboring pixels that belong to other objects at a different distance. The difference in disparity value between these pixels will then be relatively large. While the pixels are neighbors in one image, they will have a larger separation in the other stereo image. In between these pixels is now an area that is not present in the first image. This problem is related to occlusion (see figure 2-4). A common effect due to this problem is that pixels belonging to the far object will be matched as pixels that belong to the closer object. This is mainly the problem with methods that use windowing techniques.

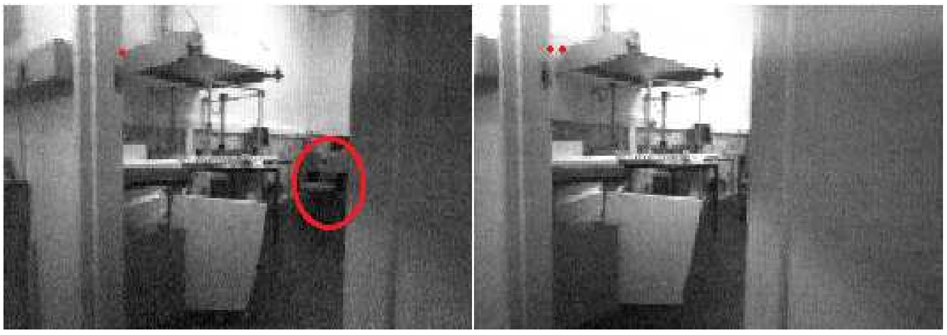


Figure 2-4: Example of occlusion. The chair in the left image is not visible in the right image. Also note the red dot in the left image. This point corresponds to two different points in the right image. This is the effect of discontinuities.

2-2 Overview of stereo vision methods

Because of the large and increasing amount of stereo vision methods that can be found in the literature, (Scharstein & Szeliski, 2002) published a taxonomy in which they compared individual algorithm components. It was meant to *"assess the performance of both established and new algorithms and to gauge the progress of the field"'*. Next to their taxonomy they also launched an online test bed to compare new methods with existing methods: the Middlebury Stereo Database. By using a standard set of test images and performance measures, a new method can immediately be ranked. Users can register their performance results, such that each new method can be compared with an up to date list of existing methods. The test bed can be found at: `vision.middlebury.edu/stereo`. Although the test bed became successful and is still being used, it only compares 'dense' two-frame algorithms. Dense means that a full disparity map is calculated: a disparity value is assigned to each individual pixel. But *"it can be seen from literature that it is not sufficient to generally categorize stereo algorithms"'* (Oon-Ee Ng & Ganapathy, 2009). They proposed the Modular Stereo Vision Framework (MSVF) *"to describe general stereo vision problems in a more general manner than the preceding taxonomy.* A graphical representation of the MSVF is shown in figure 2-5. The authors believe that their framework is much more complete in its ability to describe general two-frame stereo vision algorithms. They divide the stereo matching process in four general stages and each stage consists of modules. These modules represent classes of methods used at this stage of the stereo matching process. This is why the word modular is used in the framework description, as it is the key feature of the framework. The framework will be used in the next sections as a guide to give a summary of stereo vision methods found in literature with the focus on 'real-time implementability'. Real-time performance can be obtained in two ways: by using efficient algorithms or by using special hardware implementations. In this study the focus lies on efficient algorithms. Using for example a Graphical Processing Unit(GPU), Field Programmable Gate Array (FPGA), Application-Specific Integrated Circuit (ASIC) or Digital Signal Processor (DSP) allows the use of optimized computation strategies that are very specific and have a limited applicability. Since the aim of the DelFly project is to converge to full autonomy, on board processing is also a topic of interest. It is believed that if algorithms cannot be implemented on a CPU in real-time, they will also be no candidate for on board processing in future systems. The focus in this study is therefore further limited to methods that enable real-time performance on CPUs.

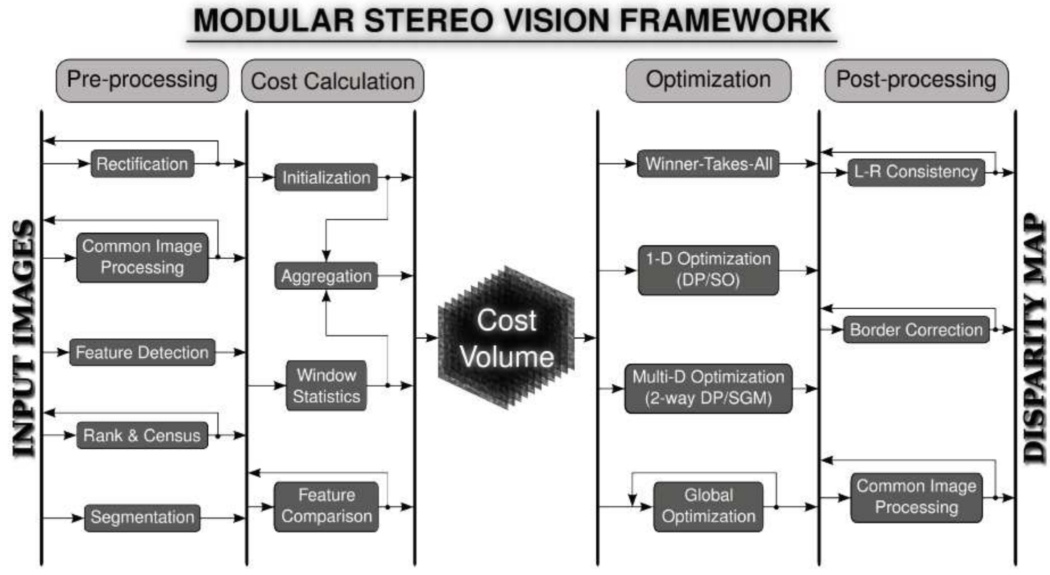


Figure 2-5: The Modular Stereo Vision Framework (Oon-Ee Ng & Ganapathy, 2009)

2-2-1 Pre-processing

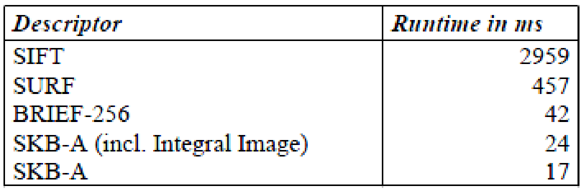
In the first stage the images are prepared. The images are treated as individual images; no data from other images is used.

Rectification
Rectification of images is essential to do stereo matching. By undistorting and rectifying (aligning) the images the search for correspondences is reduced to a one-dimensional problem. This process will be described in more detail in chapter 3.

Common Image Processing
To reduce the effect of noise or differences in brightness between input images, techniques such as filtering, normalization and contrast enhancement can be applied. However, according to (Oon-Ee Ng & Ganapathy, 2009) most of the common techniques destroy data such that the images become less suitable for stereo matching. They say that image normalization is an exception, since it does not perform smoothing.

Feature detection
Special methods exist to detect interesting features in images. These methods are in general used for localization and recognition/tracking tasks. These methods can also be used to match features between images. A 'sparse' disparity map is then obtained: only the pixels that belong to a feature are assigned a disparity value.

Feature matching is a process that consists of three steps: feature detection, feature description and feature matching. The last step belongs to another stage in the MSVF. Some methods both do both detection and description, others do one of them. The most popular one is the Scale-Invariant Feature Transform (SIFT) (Lowe, 2004), which at the time of publication outperformed the other available methods. The method was followed up by the more time efficient Speeded Up Robust Feature (SURF) method (Bay, Tuytelaars, & Van Gool, 2008). It was found that this method outperforms both SIFT in terms of speed (354ms com-



Descriptor	*Runtime in ms*
SIFT	2959
SURF	457
BRIEF-256	42
SKB-A (incl. Integral Image)	24
SKB-A	17

Table 2-1: Runtime of different descriptors for 8375 feature points on a single 3.33GHz CPU (Zilly et al., 2011)

pared to 1036ms) and recognition rate (83.8% compared to 78.1%) on a 800x640 test image. This is a significant improvement, but does not give real-time performance. Features from Accelerated Segment Test (FAST) (Rosten & Drummond, 2006) is a pure feature detector that efficiently detects if a pixel is a corner or not. On a 2.8GHz pc the method detects 100 features from a 752x480 image in 3.3ms (Mattmann, 2010). FAST has become a popular feature detector in combination with efficient feature descriptors. An example is the Binary Robust Independent Elementary Features (BRIEF) descriptor (Calonder, Lepetit, Strecha, & Fua, 2010) which creates very compact descriptions and also creates them in an efficient way. It was reported that in a test with 512 features, BRIEF needed around 15ms while SURF needed almost 370 ms for matching. Under certain conditions (BRIEF is not rotation invariant) it also outperforms SURF in terms of recognition rate. Binary Robust Invariant Scalable Keypoints (BRISK) is a combined feature detector, desriptor and matcher based on FAST (Leutenegger, Chli, & Siegwart, 2011). In a test with two 800x640 stereo images, it took BRISK 30ms to match around 1000 features, while SURF took 195ms to match 1500 features. In terms of quality the methods are comparable: they outperform each other depending on the image features. Another method, Semantic Kernels Binarized (SKB), is also a combined detector, descriptor and matcher which aims at real-time perfromance on stereo images. On a 3.33GHz system with 6 cores and 6 extra cores via hyper threading, matching approximately 3000 features took 37ms on 960x450 images. In another test the descriptor runtime was compared to other descriptors. The results are shown in table 2-1. It can be concluded that there are a few feature matching methods that deliver real-time performance. Note that the actual performance depends on the type of CPU and how the cores are used and divided.

Rank and Census

Some stereo vision methods first apply the Rank or Census transform in order to reduce the effect of differences in lighting (radiometric gain). In case of the Rank transform, each pixel is compared to its neighboring pixels in a surrounding window. It is counted how many of the surrounding pixels are brighter than the center pixel. This number then serves as a descriptor. In contrast to feature matching, each pixel is given a descriptor. Because this process reduces the discriminatory power of the pixel, the Census transform is used in some cases. The only difference with the Rank transform is that it is remembered which of the surrounding pixels are brighter. This results in more extensive descriptors, leading to more robust matches but also to more computations. A real-time implementation of the Census transform in a stereo vision algorithm was published by (Zinner, Humenberger, Ambrosch, & Kubinger, 2008). They report a performance of 42Hz on 320x240 images for a disparity range of 30 at a 2GHz CPU. This was achieved by special performance optimization techniques, resulting in an implementation that is up to 112 times faster than a generic version of the source code. The authors state that all other real-time implementations with Census were done with FPGA's ans ASIC's. By making *"intensive use of Streaming SIMD Extensions (SSE) and the multi-core architecture of state-of-the-art CPUs"'* the Census transform can be implemented in real-time. The authors also state that the Census transform outperforms the more standard approaches such as Sum of Absolute Differences (SAD) and Sum of Squared Differences (SSD).

Segmentation
Segmentation is the process of grouping neighboring pixels that contain similarities, such as (almost) equal color. This grouping can be used to reduce the matching problem, but it can also help in reducing ambiguity. Image regions that lack features because the region has a homogeneous color can then be treated as a single object in the matching process. *"However, segmentation itself is a wide field of research, and real-time segmentation is still an open problem for general images"* (Oon-Ee Ng & Ganapathy, 2009). Segmentation *"at real-time is a challenge in itself"'* (Gupta & Cho, 2010). Nevertheless (Baha & Larabi, 2011) published a stereo vision method in which color segmentation was used. They report a performance of 390ms with a 2.2GHz CPU on 450x576 images. Another segmentation approach (Tombari, Mattoccia, & Di Stefano, 2008b) tested on the same images takes only 200ms on a 2.4GHz CPU. According to (Oon-Ee Ng & Ganapathy, 2009) these methods rely on *"interesting trade-offs between accuracy and computational efficiency"*.

2-2-2 Cost Calculation

In this second stage matching costs are computed.

Initialization
This process is most often a pixel-to-pixel comparison where the similarity between them is measured. Similarity in brightness is a common cost factor, but other measures can also be used. Most methods do not use single pixel comparisons for stereo matching because these are vulnerable to any sort of noise. This is why in most general cases aggregation or window statistics is performed.

Aggregation and Window Statistics
To reduce vulnerability to noise and to reduce the effect of ambiguities, the matching cost of a pixel is in most methods based on the matching cost of surrounding pixels. The most common methods use a rectangular window around the pixel of interest to compute the matching cost. The most common methods are shown in table 2-2. SAD and sum of squared differences are two methods that are very common because they are less complex (faster to compute) than the others. Note that the Rank and Census transforms are also in this list. In this stage of the stereo process it is calculated how much the descriptors differ from each other. In case of Census, this means that is counted how many surrounding pixels have the same relative brightness as the center pixel. Segmentation can also play a role in this stage of the stereo matching process. Instead of using rectangular windows (as in table 2-2 for example), the window shape can also be defined by the shape of the segments (Tombari, Mattoccia, & Di Stefano, 2008a). Another approach is to select from a rectangular window only those pixels that belong to the same segments (Gupta & Cho, 2010).

Feature comparison
In case of feature matching methods, the descriptors of features from two different images

MATCH METRIC	DEFINITION
Normalized Cross-Correlation (NCC)	$\dfrac{\sum_{u,v}\left(I_1(u,v)-\bar{I}_1\right)\cdot\left(I_2(u+d,v)-\bar{I}_2\right)}{\sqrt{\sum_{u,v}\left(I_1(u,v)-\bar{I}_1\right)^2\cdot\left(I_2(u+d,v)-\bar{I}_2\right)^2}}$
Sum of Squared Differences (SSD)	$\sum_{u,v}\left(I_1(u,v)-I_2(u+d,v)\right)^2$
Normalized SSD	$\sum_{u,v}\left(\dfrac{\left(I_1(u,v)-\bar{I}_1\right)}{\sqrt{\sum_{u,v}\left(I_1(u,v)-\bar{I}_1\right)^2}}-\dfrac{\left(I_2(u+d,v)-\bar{I}_2\right)}{\sqrt{\sum_{u,v}\left(I_2(u+d,v)-\bar{I}_2\right)^2}}\right)^2$
Sum of Absolute Differences (SAD)	$\sum_{u,v}\lvert I_1(u,v)-I_2(u+d,v)\rvert$
Rank	$\sum_{u,v}\left(I_1'(u,v)-I_2'(u+d,v)\right)$ $I_k'(u,v)=\sum_{m,n}I_k(m,n)<I_k(u,v)$
Census	$\sum_{u,v}HAMMING\left(I_1'(u,v),I_2'(u+d,v)\right)$ $I_k'(u,v)=BITSTRING_{m,n}\left(I_k(m,n)<I_k(u,v)\right)$

Table 2-2: Common window statistics (Brown et al., 2003). *I* means image, the subscript number is used to discern from the left and right stereo image. The bars on top indicate 'mean'. *u,v* and *m,n* denote pixel coordinates, *d* is disparity, *HAMMING* is the operation of counting how many bits from two different bitstrings have the same value.

are compared. Descriptors can have any form and size, depending on the method they are calculated by. For example, BRIEF takes certain randomly predefined pixels around a center pixel. Like for the Census transform, the brightness of these pixels is compared to the brightness of the center pixel and for each pixel the descriptor indicates of this pixel is brighter than the center pixel or not. In case of BRIEF the length of the descriptor (and thus the amount of comparisons per pixel) and also the positions of the comparison pixels are optimized for speed and discriminability. The matching cost for BRIEF descriptors is computed in the same way as for Census. BRIEF can be regarded as an optimized version of the Census method.

2-2-3 Cost Volume

The result of the Pre-processing and Cost Calculation stages is the Cost Volume. Basically it is a volume where all matching costs are stored. In case of dense two-frame stereo vision, this volume is also known as the Disparity-Space Image (DSI). It is a three-dimensional image, defined by the two-dimensional pixel coordinates and the disparity. For each pixel the DSI stores the matching cost for each disparity. As discussed in the previous subsection, this matching cost can be computed in various ways. In the case of feature matching, the cost volume will consists of a list of feature matching costs. In some segmentation methods, the cost volume stores a list of segment matching cost.

2-2-4 Optimization

When the Cost Volume has been generated, the matching cost per pixel and per disparity is known. The next step is to search the cost volume for an optimal set of disparity values to obtain the two-dimensional disparity map. This can be done straightforward, but also by applying some types of constraints: ordering (the order in which pixels appear should be the same in the left and right images), surface continuity/smoothness (no (big) disparity jumps for neighboring pixels belonging to the same object), and point visibility (e.g. no pixel is matched with the same pixel from the other image). These constraints can be combined into optimization techniques. Such techniques optimize the matching result, but at the same time can add extra complexity to the problem, which leads to more computation time.

Winner-Takes-All
This is the most straightforward method to obtain a disparity map. The method looks in the DSI for the disparity value with the lowest matching cost, for each individual pixel. This is actually no type of optimization, but just a simple selection. It is a method that is normally used in real-time methods. Using this method in combination with simple pixel-to-pixel matching costs would result in very bad disparity maps. When using the winner-takes-all strategy, a good Cost Calculation strategy is essential. The main challenge for these strategies is to handle image regions that contain no features. Simple window techniques such as SAD have been used to obtain real-time performance, but for images with a small number of features they produce bad results (Bradsky & Kaehler, 2008), (Point Grey Research, 2000). Some methods try to improve specific matching errors. For example, (Hirschmüller, 2001) proposes a method that reduces the foreground fattening effect[1] by using multiple overlapping windows. As a result, the effect of this specific error is reduced, but the effect of other errors increases at the cost of a higher computational complexity.

The main issue with these types of methods is that on one hand rectangular windows are relatively easy to compute, but on the other hand their shape is not ideal. Either a window overlaps an image region belonging to another object, or the window is not large enough to cover a full object. For this reason segmentation approaches have been proposed (Tombari et al., 2008b), (Gupta & Cho, 2010). As discussed before these methods can still have (almost) real-time performance while they are better in handling texture-poor image regions.

One-dimensional Optimization
In all optimization techniques, the disparity value of a single pixel does not only depend on matching costs, but also on the disparity value of surrounding pixels. One-dimensional optimization is in general done for horizontal image lines. A popular form is Dynamic Programming. Near real-time implementations of this technique have been proposed: (Van Meerbergen, Vergauwen, Pollefeys, & Van Gool, 2002), (Salmen, Schlipsing, Edelbrunner, Hegemann, & Lke, 2009)), (Bradsky & Kaehler, 2008), (Forstmann, Kanou, Ohya, Thuering, & Schmitt, 2004). The last one of these three is the fastest and it can run at 30Hz on 384×288 images for 32 disparities (AMD Athlon XP 2400+). The main idea behind this technique is disparity continuity. Two neighboring pixels on the same image line are assumed to have the

[1]The foreground fattening effect results in image features with fattened edges. Pixels belonging to a far object are mistakenly matched as a pixel that belongs to an object close by. This occurs when trying to match pixels close to depth discontinuities using window matching. The larger the window size, the stronger the effect.

same disparity value (or the difference in disparity is very small). If the pixels are given different disparities, they are also given a matching cost penalty. Only if this penalty outweighs the reduction in matching cost corresponding to the different disparity, this different disparity is chosen. This way pixels in texture-poor image regions tend to have similar disparity values, while pixels containing image features show large disparity jumps.

Dynamic Programming methods can handle texture-poor regions much better than most real-time implementations of winner-takes-all methods. However, the methods have some imperfections. In the first place, they use the ordering constraint. It states that the order in which pixels appear in the left image should be the same as in the right image. If a pixel in the left image is assigned a wrong disparity, this might invoke that its neighboring pixel cannot be matched with the right pixel anymore (as this would contradict the ordering constraint). Mistakes made during matching can thus result in matching mistakes for other pixels on the same image line. Another side effect is *streaking*: large inconsistency between scan lines. The scan lines themselves show continuous behavior, but neighboring scan lines may give different results, even though they cover the same image features/objects. Some methods therefore use a refinement step to reduce this effect, for example by vertical filtering (Forstmann et al., 2004).

Multi-dimensional Optimization
Performing one-dimensional optimization in multiple directions is called multi-dimensional optimization. This is possible for Dynamic Programming as shown by (Ohta & Kanade, 1985). Multi-dimensional optimization improves consistency between image-lines (compared to one-dimensional optimization) but at the cost of a higher complexity (and thus computational demand). Nevertheless, an efficient type of algorithm was introduced: Semi-Global Matching (SGM) (Hirschmüller, 2005) (Hirschmüller, 2008). The method does not apply the ordering constraint. Basically, the method *"performs an energy minimization in a dynamic-programming fashion on multiple one-dimensional paths crossing each pixel and thus approximating the two-dimensional image* (Gehrig & Rabe, 2010). A more detailed description of the method follows in section 2-4. The original implementations do not have real-time performance (1.0s for a 450×375 image and 32 disparities) but *SGM is commonly regarded as the most efficient algorithm* among the top-performing algorithms on the Middlebury Stereo Database (Gehrig & Rabe, 2010). They show how the method can be implemented on a CPU to obtain real-time performance. In their implementation they use the Census transform which they can efficiently apply using multiple-processing techniques. They also use down-sampled images strategically: full resolution images are used for far away objects (small disparities), down sampled images are used for close objects (large disparities). As a result the method takes 63ms to compute over 16 disparities on 640×320 images using a 3.3GHz Intel Core i7-975ex (four cores). According to (Mattoccia, 2010) the SGM method is the best performing *"DP based approach on the Middlebury evaluation site and it has a running time of few seconds"*. The authors proposed the locally consistent (LC) method to be used in combination with SGM. This is a non-iterative technique that enforces the local consistency of disparity fields by explicitly modeling the mutual relationships among neighboring pixels. They claim that they can improve methods like SGM drastically using this technique, but at the cost of more computation time: 15 seconds for 450×375 images using a 2.49 GHz Intel Core 2 Quad CPU.

Global Optimization
Global optimization techniques use two-dimensional optimization. Information from the com-

plete image is used in this case. The computational complexity of these methods is significantly higher compared to the other optimization methods and real-time implementations on CPUs do not exist. These types of algorithms are either categorized by the theoretical models they are based on (Bayesian network, Markov random field (Tardon, Barbancho, & Alberola, 2011)) or by the method in which the optimization problem is solved (Graph Cuts citehong, Belief Propagation (Klaus, Sormann, & Karner, 2006)). The complexity of solving the optimization problem using these models is too high to be used on CPUs in real-time. By making use of a GPU (Yang et al., 2006) or FPGA real-time implementations are possible.

2-2-5 Post-processing

When the disparity map has been computed, there are some ways to correct or improve the result.

Left/Right Consistency
This method is costly, since it requires the stereo matching process to be run two times: from left to right and vice versa. The disparity maps can then be projected onto each other to check if disparity values from the two images correspond. False matches can be filtered out this way. This is mainly done to detect occluded regions.

Border Correction
These methods detect image borders in the disparity image and try to improve the area around these borders by using filters (Hirschmüller, Innocent, & Garibaldi, 2002). This way foreground fattening effects can be reduced.

Common Image Processing
Filtering techniques can be used to smoothen the disparity images, in order to obtain a 'nicer and cleaner' result. Accurate algorithms normally do not use these techniques since they destroy information, while the amount of corrections is marginal.

2-3 Comparison of stereo vision methods

In this section the main findings from the previous overview are checked by comparing three different algorithms that are based on the three important types of optimization techniques used in real-time stereo vision methods. These optimization techniques are: Winner-Takes-All, One-Dimensional Optimization and Multi-Dimensional Optimization. The methods are all implementations from the OpenCV library (`http://www.opencv.org/`):

Block Matching (Winner-Takes-All)
This method is called by the function *FindStereoCorrespondenceBM*. It is a 'very fast' algorithm bas on SAD. The method makes simply use of rectangular windows, so no improvements are included (such as using segmentation). The method allows to pre-filter the input images, to define the window size and disparity range and it allows to choose some post-processing techniques. A texture threshold can be applied such that regions with low texture are ignored. Also a uniqueness ratio can be chosen such that only matches are kept that have a matching cost that is much lower compared to the other disparity values. Finally speckle filtering can be applied. This method filters out matches such that the amount of variation within image regions is within an acceptable range.

Dynamic Programming (One-Dimensional Optimization)
This method is called by the function *FindStereoCorrespondence*. It allows to chose the disparity range and to set five parameters: an occlusion penalty, a reward for constant matches (same disparity) and three reliability values for high, moderate and slight reliable regions. The matching cost are computed by pixel-to-pixel comparison. Consistency between image lines is improved by removing pixels that have neighboring pixels containing different vertical gradients.

Semi-Global Block Matching (Multi-Dimensional Optimization)
This method is called by the function *StereoSGBM*. It is a modified implementation of (Hirschmüller, 2008) where the amount of optimization directions can be chosen to be 5 or 8. The user can also specify to use pixel-to-pixel matching costs or SAD matching costs. Sub-pixel interpolation is used and a few pre- and post-processing techniques are implemented: pre-filtering using a Sobel filter, a uniqueness check and speckle filtering as in Block Matching and quadratic interpolation to obtain sub-disparity values.

The methods have been compared on images from the stereo vision camera used on the flapping wing MAV from this project. Below, a test on a single stereo image pair made by the stereo camera setup is shown. The input images were resized per stereo algorithm such that the performance would be real-time (40ms). The parameters for the methods were tuned such that the disparity output would give the best result 'on sight'. The test image and the output from the three methods are shown in figure 2-6.

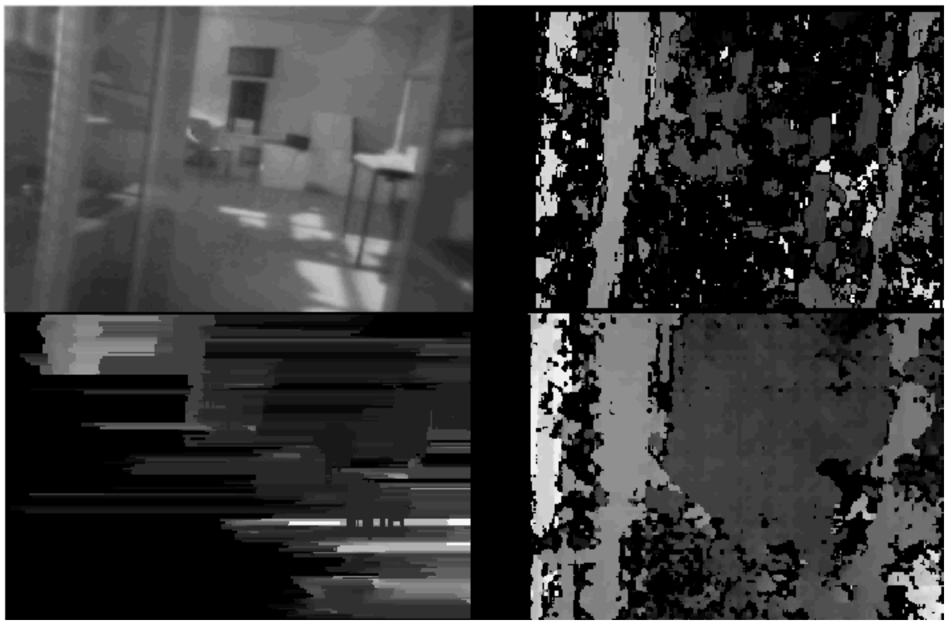


Figure 2-6: Comparison of three different types of stereo vision methods. **Top-Left** test image **Top-Right** Block Matching **Bottom-Left** Dynamic Programming **Bottom-Right** Semi-Global Block Matching

The figure expresses the main findings from the overview in the previous section. The Block Matching method shows a relatively sparse result. Dominant features, such as vertical lines, are matched quite well, but in between these features a lot of unknown regions are left empty. Even the shadows on the ground apparently do not provide enough texture for good matching.

The information from this method is partly useful, in that it provides information on obstacles close by. But this information would be much more useful if the method would be able to indicate that the center zone of the image contains only obstacles far away. Note that the center zone even contains blobs of white pixels that indicate non-existing close objects.

The Dynamic Programming method performs even worse. The main structures in the image can not even be distinguished. This result might not be fully representative for dynamic programming algorithms since these perform better than winner-takes-all methods in general. In the top-left corner of the image the streaking effect is visible: the image lines appear as if they are a little bit randomly shifted horizontally. The bottom-left part of the image is almost empty (no reliable matches) and the right part of the image does not show clear objects. This illustrates the short-coming of Dynamic Programming: matching errors influence the results for the remainder of the image lines. The bad matching results in the left part of the image spoil the results in the right part of the image. The fact that this implementation uses pixel-to-pixel matching costs might have a large negative influence of the final result.

Compared to the other two methods, Semi-Global Block Matching gives significantly better results. The main structure of the scene is clearly visible in the disparity map: two cabinets close by on both sides and in between there is space with obstacles much further away. Also here some regions are left empty but the amount of known disparities is substantially larger. False matches are also visible but their number is also small. This method gives the most useful information, and this information potentially useful enough for obstacle detection. The result is also notable because SAD was not used, but the simple pixel-to-pixel matching cost.

Based on the findings of the overview from the previous section and the above results that support these findings it was decided to use the Semi-Global Matching method for further testing and for implementation in the obstacle avoidance strategies that were developed and tested in this project. The theory behind Semi-Global Matching and more details on the used implementation will follow in the next section.

2-4 Semi-Global Matching

In this section the theory behind the Semi-Global Matching method (Hirschmüller, 2005) is explained.

The cost calculation method is taken from (Birchfield & Tomasi, 1998). This method does not compare pixel intensities directly. Instead, the cost is dependent on the intensity of neighboring pixels. The method is visualized in figure 2-7.

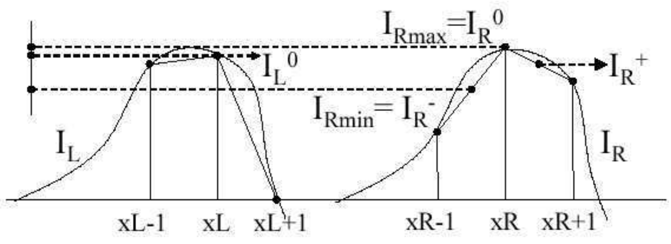


Figure 2-7: Visualization of he sampling insensitive cost measure (Birchfield & Tomasi, 1998)

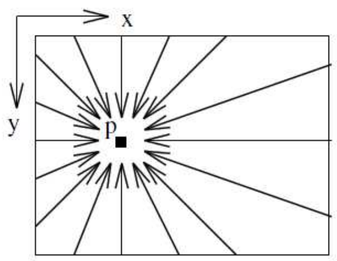


Figure 2-8: 16 paths from all directions going to pixel *p* (Hirschmüller, 2005)

The cost to match pixel *xL* from the left image with pixel *xR* from the right image depends on the neighbors of *xR*. First intensity interpolation is done around this pixel using the neighbors *xR-1* and *xR+1*. Halfway between both pixels the interpolated intensities x_R^+ and x_R^- are calculated. From these two intensities and the intensity of the center pixel, I_R^0, the minimum and maximum value are selected which form an intensity band. The intensity of pixel *xL* is now compared with this intensity band. If the intensity of *xL* falls inside the band, the matching cost is defined as zero. Otherwise, the cost equals the difference in intensity between *xL* and the closest boundary of the intensity band.

By computing this matching cost (indicated from here as *C(p,d)*) for each pixel-disparity combination, the DSI is obtained. The DSI indicates for each pixel how much it costs to assign a certain disparity value to it. Simply selecting the disparities with the lowest costs would give bad results. *"Pixelwise cost calculation is generally ambiguous and wrong matches can easily have a lower cost than correct ones, due to noise, etc.* (Hirschmüller, 2005). Therefore he proposes an optimization strategy that aggregates matching costs in multiple one-dimensional directions. The aggregated matching cost S for a certain pixel p and disparity d is defined as:

$$S\left(p,d\right)=\sum_r L_r\left(p,d\right) \tag{2-1}$$

It is the summation of the costs of all paths L coming from r different directions. These paths are visualized in figure 2-8. 16 paths are shown, but this number can be reduced to speed up the algorithm. The cost of each path is defined as:

$$\begin{aligned}L_r\left(p,d\right)=C\left(p,d\right)+min\Big\{&L_r\left(p-r,d\right),\\&L_r\left(p-r,d-1\right)+P_1,\\&L_r\left(p-r,d+1\right)+P_1,\\&\min_i L_r\left(p-r,i\right)+P_2\Big\}-\min_k L_r\left(p-r,k\right)\end{aligned} \tag{2-2}$$

So the cost for path $L_r(p,d)$ depends on the matching cost *C(p,d)* and the cost of the most optimal path up to the previous pixel *p-r* for this specific direction. In theory, the current pixel p can choose from i different optimal paths that all end at the previous pixel (i is the number of disparities that is contained in the DSI). The current pixel can easily choose the path with the lowest cost. However, to support smoothness, cost penalties are added if a path corresponding to a different disparity is chosen. If this difference in disparity equals 1, the cost penalty is *P1*. If the disparity jump is larger than 1, the penalty *P2* is added. Using a smaller penalty *P1* for small disparity changes permits an adaption to slanted or curved surfaces. The last term in the equation is used to prevent the path cost to keep permanently increasing. It has no influence on the path through the disparity space since its value is the same for disparities for this specific pixel.

Note that each path is independent from any other path, but that the amount of paths is very large (much more than for dynamic programming where the amount of paths equals the number of image lines). After all paths have been optimized individually, equation 2-3 combines all paths that go through pixel p. Note that this is done for each disparity value individually. The most optimum disparity value is now selected by taking the minimum:

$$\min_d S(p, d) \tag{2-3}$$

So the optimum disparity value is that value for which the combined optimal path costs are the smallest.

2-4-1 Implementation

Since the Semi-Global Matching algorithm turned out to be producing the best results at real-time it was selected as the stereo vision method for further testing in this project. The OpenCV implementation as shortly discussed in section 2-3 was used without any adaption. (Hirschmüller, 2005) proposes two different cost functions: the Birchfield-Tomasi sub-pixel metric (as discussed before), and a Mutual Information (MI) based matching cost (Viola & Wells, 1997). The latter matching cost is defined by the entropy of the two stereo images as well as by their joined entropy. This method is insensitive to recording and illumination changes. Tests show that SGM in combination with MI is 30% slower than the combination with the Birchfield-Tomasi metric. For this reason the latter has been included in the OpenCV implementation.

Figure 2-8 shows the 16 paths along which optimization can be performed. Because 16 paths results in high memory demands and is not time efficient, the OpenCV implementation allows two options: 5 or 8 paths. The reason behind selecting 5 paths is that the algorithm becomes much more efficient, since it allows a single-pass evaluation. If only the 5 directions in the top left corner from figure 2-8 are evaluated all calculations can be performed by going once through the image in a left-right top-down fashion. The 8 path option was used during this project since it improves the output quality significantly.

Even though the OpenCV implementation allows SAD block matching, it was decided to use single pixels for the matching cost (window size equals one). This is not only more time efficient but also results in better disparity maps.

In this implementation the disparity range must be divisible by 16. The larger the disparity range, the smaller the minimum distance at which objects can be detected. Based on the findings as reported in Chapter 3 it was concluded that only 16 disparities is enough to detect obstacles at a range smaller than 60cm. By increasing the disparity range up to 32, the minimum detection distance decreases under 30cm. This is not necessary since obstacles will need to be detected earlier in order to avoid them in time.

The relation between detectability range and disparity range depends on the resolution of the stereo images. If the images are sub-sampled to a lower resolution, the detectability range increases. To achieve real-time performance, the stereo images are down-scaled to 160×112 pixels. The resulting frame rate for image rectification (chapter 3) and stereo processing is over 25Hz. After down-scaling a disparity range of 16 enables obstacles to be detected at distances as close as 60cm.

The values for parameters *P1* and *P2* have been chosen based on experiments with the camera images from the stereo system used in this project (as discussed in chapter 4). The values used throughout all experiments are: *P1* = 50 and *P2* = 500. *P2* should have the largest value and it was indeed observed that this delivered the best performance. No ground truth comparison was used in tuning these parameters. Instead it was done by observing how well different obstacles could be distinguished from the disparity maps. Detectability was regarded as the most important criterion.

Table 2-3 lists all parameters including those for the pre- and post-filtering techniques. *uniquenessRatio* is the margin in percents by which the best (minimum) computed cost function value should win from the second best value to consider the found match correct. *speckleWindowSize* is the maximum size of smooth disparity regions to consider them noise speckles in order to invalidate them. *speckleRange* is the maximum disparity variation within each smooth disparity region. *preFilterCap* truncates the x-derivative of pixel intensities that are used for the Birchfield-Tomasi sub-pixel metric. *disp12MaxDiff* is the maximum allowed difference (in integer pixel units) in the left-right disparity check. *fullDP* is either true or false and determines if the full 8 path method is used or not.

Table 2-3: Parameter values for the OpenCV implementation of Semi-Global Matching

Parameter	Value
SADWindowSize	1
minDisparity*	9
numberOfDisparities	16
P1	50
P2	500
uniquenessRatio	20
speckleWindowSize	10
speckleRange	32
preFilterCap	100
disp12MaxDiff	10
fullDP	TRUE

* not a fixed value. This value depends on the calibration parameters.

Chapter 3

Camera Calibration

In this chapter it is discussed how the stereo matching algorithm from chapter 2 can be used to obtain three-dimensional vision. First it is explained how the raw camera images are undistorted and rectified such that stereo correspondences can be found by the algorithm. Then it is explained how these correspondences can be translated into three-dimensional coordinates.

3-1 Camera Model

As camera model, the pinhole camera model is used as illustrated in figure 3-1. The pinhole lets through only those light rays that intersect a particular point in space. Only a single ray enters from any particular point in space. This point is then projected onto an image plane. As a result an image appears on the image plane that is always in focus.

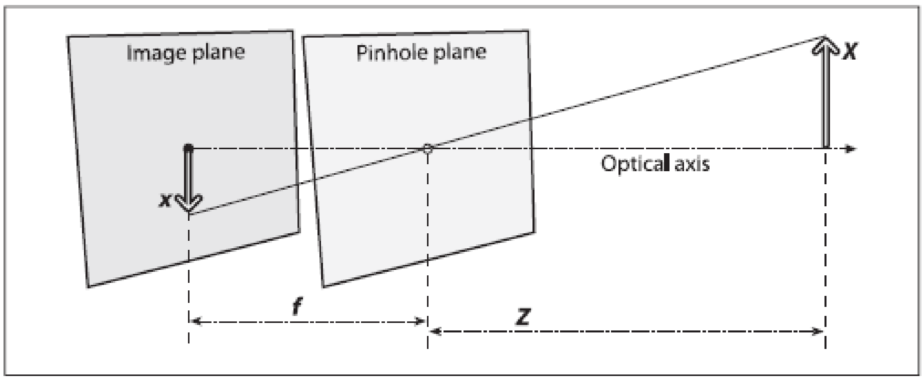


Figure 3-1: The pinhole camera model (Bradsky & Kaehler, 2008)

The position X of a point in space and its projection x on the image plane is given by the following relation:

$$-x = f\frac{X}{Z} \tag{3-1}$$

Z is the distance from the object to the pinhole camera and f is the focal length, which equals the distance between the image plane and the pinhole camera. For mathematical convenience, the pinhole camera is in general rearranged as shown in figure 3-2. The image plane is now put in front of the pinhole. Physically this is not possible, but objects on the image plane now appear right side up. As can be seen in this figure, the optical axis goes through the center of the image plane. However, in real life a camera chip will never be perfectly aligned with the optical axis (of the lens) so the pinhole camera model is extended with a displacement parameter c to deal with this misalignment. Furthermore the model uses two separate equations for the two dimensions of the image. This is because in a typical camera the individual pixels are not square but rectangular. While the actual focal length has units of millimeters, the model uses the product of focal length and imager element size s (the size of the sensor pixels) which has the unit of pixel per (milli)meter. This means that the focal lengths f_x, f_y are defined in pixels. The displacement parameters c_x, c_y can also be defined in pixels. A point with position X, Y, Z is then projected onto a screen with pixel locations x_{screen}, y_{screen} using the following relations:

$$x_{screen} = f_x\left(\frac{X}{Z}\right) + c_x, \qquad y_{screen} = f_y\left(\frac{Y}{Z}\right) + c_y \tag{3-2}$$

When using this projective transform it is convenient to use homogeneous coordinates. These are typically expressed as a vector that has one more dimension than the associated point in the projective space. In case of an image plane, the projective space has two dimensions, so these points will be represented in three dimensions. By dividing with the third dimension, the actual pixel coordinates can be retrieved. This way the camera intrinsic parameters can be defined in a single matrix M, and the projective transform is given as:

$$q = MQ, \quad with \quad q = \begin{bmatrix} x \\ y \\ w \end{bmatrix}, \; M = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \; Q = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{3-3}$$

Since q is in homogeneous coordinates, its third vector element q_3 can be used to get the actual pixel coordinates:

$$x_{screen} = \frac{x}{q_3} = \frac{x}{w} = \frac{x}{Z}, \quad y_{screen} = \frac{y}{Z} \tag{3-4}$$

3-2 Distortion

A pinhole, as introduced in the pinhole camera model, will let through only a small amount of light, which would require long exposure periods for a camera to give good results. Therefore

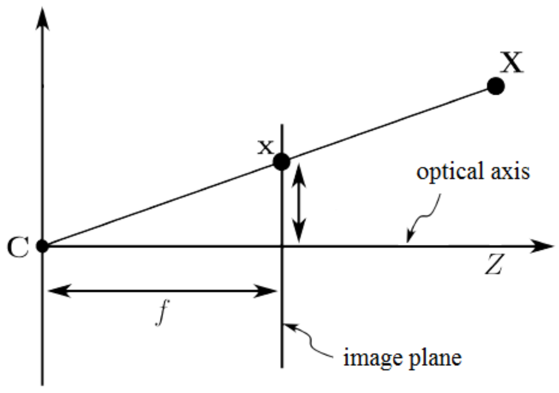


Figure 3-2: The pinhole camera model with the image plane in front of the pinhole

lenses are used in order to catch and focus more light on the camera chip. Lenses normally have several distortions because of misalignment and manufacturing inaccuracies. The two main distortions that are modeled are radial distortion and tangential distortion. Both will be briefly discussed.

Radial distortion has to do with the shape of the lens. Inexpensive lenses normally have a larger radial distortion, because their shape is not perfected to keep costs low. Because of the simple lens shape, light rays farther away from the center are bent more than the rays close to the center. In case of radial distortion the distortion is zero at the optical center and increases when going further away from the center. The distortion can be approximated using a Taylor series expansion around the center based on the distance to the center, defined as r. For most lenses only two terms are used, in case of heavy distortion a third term is used. The relation between the original image location and the new undistorted location is then given as:

$$x_{corrected} = x(1 + k_1r^2 + k_2r^4 + k_3r^6), \quad y_{corrected} = x(1 + k_1r^2 + k_2r^4 + k_3r^6) \tag{3-5}$$

The second main type of distortion, tangential distortion, comes from the fact that lenses are never positioned perfectly parallel to the camera chip. The distortions are normally characterized as:

$$x_{corrected} = x + [2p_1y + p2(r^2 + 2x^2)], \quad y_{corrected} = x + [p1(r^2 + 2y^2) + 2p_2x] \tag{3-6}$$

The parameters in these two equations are normally combined in the distortion matrix, which is defined as:

$$D = \begin{bmatrix} k_1 & k_2 & p_1 & p_2 & k_3 \end{bmatrix} \tag{3-7}$$

Other types of distortion have a much weaker effect and are therefore not taken into account.

3-3 Epipolar Geometry

When using stereo cameras, it is not only necessary to compute the intrinsics parameters and to correct for distortions, but it is also necessary to align the two images to make stereo matching possible. For this task the epipolar constraint is essential, which will be discussed in this section.

The epipolar geometry, which is the geometry of a stereo imaging system, is visualized in figure 3-3. Two cameras are shown, both having their own center of projection (O_l, O_r, which correspond to the locations of their lenses). Two points in the figure are of special interest: the epipoles e_l, e_r. As can be seen, an epipole is the image projection of the center of projection (O) of the other camera. Considering a point P, the plane between this point and the epipoles is defined as an epipolar plane. The projections of P on the image planes, p_l, p_r, and their corresponding epipoles can be connected by lines, which are called epipolar lines. In fact, the epipolar lines are the image projections of the epipolar plane. As illustrated in figure 3-4 any point in the physical world will be part of an epipolar plane, and each epipolar plane has its own epipolar line. The epipolar lines intersect at the epipoles.

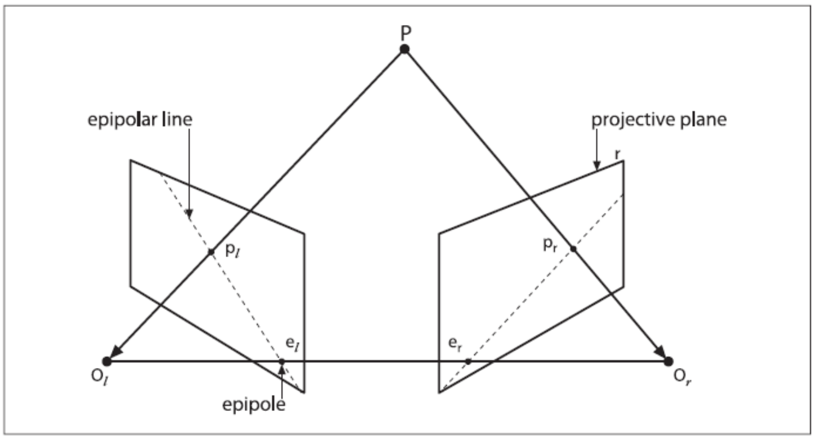


Figure 3-3: The epipolar geometry (Bradsky & Kaehler, 2008)

Now it is interesting to look at point P in figure 3-3 again. Its projection in the left camera image is p_l. From this projection it is known that P is somewhere on the line defined by p_l and O_l, but this does not define the actual location of P on this line. However, the projection of this line in the right camera image is the epipolar line defined by e_r and p_r. From this it can be concluded that every point in one camera image is part of the corresponding epipolar line in the other camera image. This called the epipolar constraint. It means that the search for matching features in two different cameras is a one-dimensional problem when the epipolar geometry of the camera system is known. Looking at figure 3-4 again, one can see that each epipolar line in the left camera image has its corresponding epipolar line in the right camera image. The process of stereo matching can be regarded as finding corresponding features between every set of epipolar lines. The illustrations in figures 3-3 and 3-4 show camera systems where the optical axes are turned to each other. In real cases one will try to make the optical axes parallel. In such a case the image planes of both cameras will also be parallel. This means that the epipoles will be far outside the actual image plane, at infinity, and all epipolar lines, both left and right, will appear parallel. This is the ideal case when trying to compute stereo correspondence, because features will appear at the same horizontal line in both camera images. Because in real life there will always be a small misalignment between the two cameras, the calibration process is not only used to compute the epipolar geometry of the system, but also to rectify the images such that the epipolar lines appear parallel.

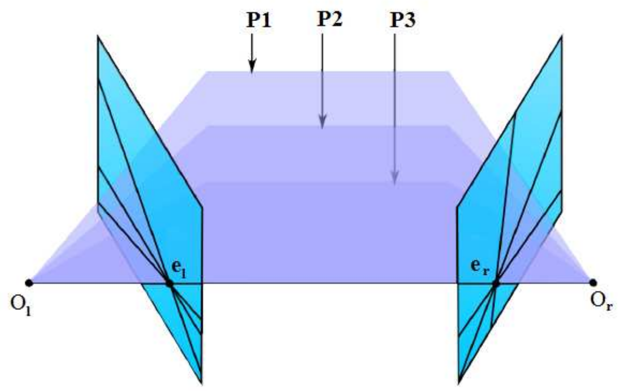


Figure 3-4: Projections of epipolar planes as epipolar lines

3-4 Camera Calibration steps

The camera calibration for the stereo system used in this project was done using the *Camera Calibration Toolbox for Matlab* ® (Matlab, 2005). For this procedure a chessboard is used for convenience. The relative positions of the chessboard corners are known and structured and their corners can be easily found in images because of the high contrast between the black and white fields. For the calibration it is required to make multiple images of the chessboard with the cameras from different view points. In theory it would be sufficient to use two images taken from different view points to solve the calibration problem (Bradsky & Kaehler, 2008). In practice one requires at least ten images with a minimum size of 7-by-8 internal corners, and preferably even more because of noise and numerical stability. There should also be enough movement between the views to obtain good calibration results. In the case of a stereo camera system, it is required that the two cameras contain the same set of images, so their calibration images need to be taken at the same time. Calibration is done in the following way:

Single camera calibration In the case of stereo camera calibration, the two cameras are first calibrated individually to obtain their intrinsic parameters and their distortion coefficients. Next to that also information on the relative position of the camera with respect to the chessboard is estimated for each image individually.

Stereo camera calibration Having the intrinsic parameters (M) and distortion coefficients (D), it is calculated what the relative position of both cameras is. This is done by combining the relative positions of the two cameras for each image pair. As a result one gets a translation vector (T) which is the position of the right camera expressed in the coordinate system of the left camera, and a rotation vector (R) which is the relative rotation of the right camera with reference to the left camera. With this information, the right camera is put in the same plane as the left camera: the two image planes become co-planar.

As discussed in section 3-3 it is required to align the image planes of the two cameras such that the epipolar lines are parallel and corresponding image lines belong to corresponding epipolar lines. Therefore a rectification step is needed to compute the rotation matrix R. This matrix rotates the left camera about its center of projection such that the epipolar lines become horizontal and the epipole goes to infinity.

Figure 3-5 shows an example result of the calibration steps. The top row shows the raw left and right camera images, the middle row shows the undistorted and rectified result, and the bottom row shows the area indicated by the white rectangle in the middle row. The white horizontal lines can be regarded as epipolar lines and show the quality of the rectification step. As can be seen the left and right images are nicely row aligned. The clear distortions in the raw images (curved edges) have been removed significantly and the chessboard fields now appear to be more rectangular.

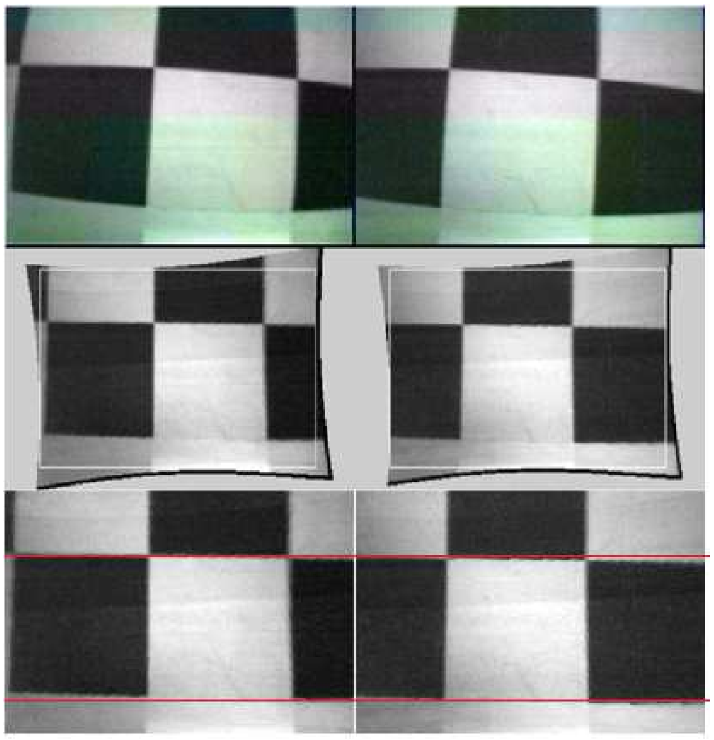


Figure 3-5: Calibration example showing undistorted and rectified images. **Top**: raw images **Middle**: undistorted and rectified images. **Bottom**: cropped images. The red lines illustrate that the images contain the same features at the same scanlines which allows horizontal matching.

3-5 Triangulation

The calibration steps described so far allow the stereo matching process, as described in chapter 2, to compute a disparity map. The disparity map indicates for each pixel from the left image how far it has been shifted in the right image, expressed in pixels. The pixel position in combination with its disparity value defines the three-dimensional position of the feature this pixel belongs to. This is based on the principle of triangulation which is visualized in figure 3-6. In this simplified situation the projections of a point P on the left and right image are indicated as x^l and x^r. The disparity d is then defined as the difference between these two. Two triangles can be seen. The first one is the triangle P,O_l,O_r, which has width T and height Z, the second one is the triangle P,x^l,x^r, which has width $T - x^l + x^r$ and height $Z - f$. These triangles have the same proportions, so the following triangulation rule is obtained:

$$\frac{T - x^l + x^r}{Z - f} = \frac{T}{Z} \tag{3-8}$$

This relation can be rewritten as follows:

$$\frac{T - \left(x^l - x^r\right)}{Z - f} = \frac{T - d}{Z - f} = \frac{T}{Z} \quad \Rightarrow \quad Z = \frac{fT}{d} \tag{3-9}$$

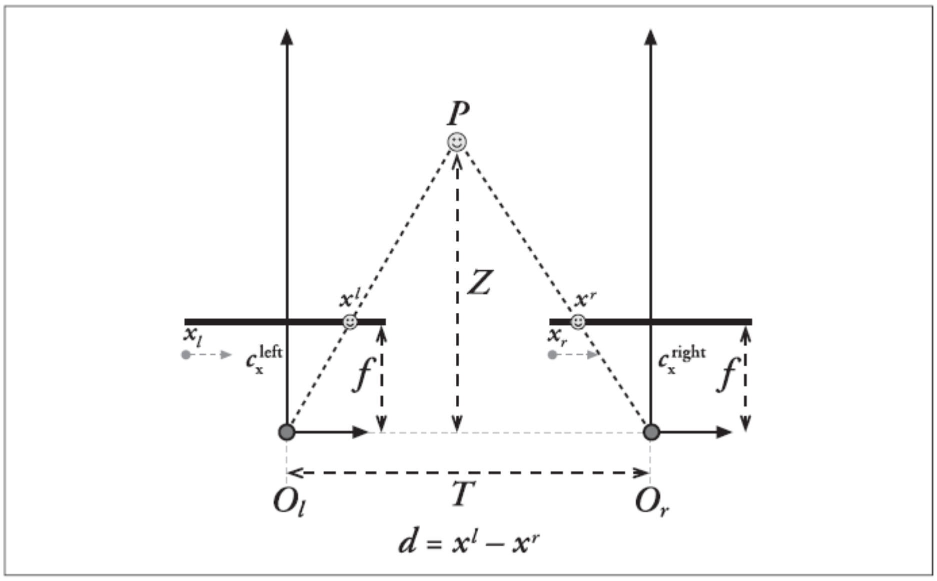


Figure 3-6: Triangulation principle (Bradsky & Kaehler, 2008)

It can be seen that the depth is inversely proportional to disparity. As will be shown, this relation only holds in case the principal rays (the rays going through the center of projection and the principal points) intersect at infinity. In other words, only if the image planes are perfectly parallel this relation holds. Calculating the three-dimensional points back from pixel coordinates and disparity values is called reprojection. This is done by the following relation:

$$\begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = Q \begin{bmatrix} x \\ y \\ d \\ 1 \end{bmatrix} \tag{3-10}$$

$$Q = \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & -1/T_x & \left(c_x - c'_x\right)/T_x \end{bmatrix} \tag{3-11}$$

In the equation c'_x is the x coordinate of the principle point in the right image, the other terms belong to the left image. T_x is the x component of the translation vector T, which is parallel to the horizontal image lines. The three dimensional coordinates are now defined as *(X/W,Y/W,Z/W)*. Note again that the last term of matrix Q becomes zero when $c'_x = c_x$. This is the case when the principal rays would intersect at infinity.

When the intrinsic parameters of both cameras are known, as well as their distortion parameters and the rotation and translation matrices, the Q matrix is also known. The relation between distance Z and disparity d can then be obtained as shown in figure 3-7 by the dashed line. In this figure also the result of an experiment is shown. In this experiment the camera was placed at several known distances (100,150,200,250 and 300 cm) from a textured wall, and using the stereo matching algorithm the corresponding disparity values were measured. The results from this experiment are indicated by the dots in the figure. From the results one can see that there is a very strong mismatch, especially for small disparity values. It was therefore tried to adjust the parameters in the Q matrix by hand. The result is the solid line, which matches nicely with the measurements. For all experiments that have been done in this project, the Q parameters were always tuned this way. The mismatch error was not this big for all calibration runs. The matching quality depends on the sample images used for the calibration routine, which in some cases do not allow for good estimations of the rotation and translation matrices. As a result the images are rectified correctly, but the triangulation data can be inaccurate.

From figure 3-7 a few observations can be made. One can see an asymptote at a disparity value of 13.5 pixels. This indicates that even for objects at infinity, the disparity value will never be smaller than this number. In other words, for the stereo matching process, it is useless trying to check for matches with a lower disparity value. The position of the asymptote is a measure for the horizontal offset between the principal rays of the cameras. It should be noted that this is a pure effect from camera misalignment and not from camera offset (baseline). Another important observation is the high nonlinearity in the distance-disparity relation. The larger the distance between an object and the camera, the larger the difference in measured distance will be for small variations in the disparity value. This an important aspect for the purpose of obstacle avoidance, and it will get more attention in chapter 5. The camera baseline plays an important role in the shape of the curve. As an example the red line has been drawn. It corresponds to the same Q-matrix as was found from the measurements fit, but with the assumption that the baseline would be halve the size. The matrix elements containing the term T_x then have a value that is two times larger. From the result one can see that the curve is steeper for small disparity values. This indicates that the distance to far away objects will be estimated with worse accuracy. The green markers in the plot indicate points in the curves that have the same slope. From this it can be concluded that a larger baseline allows for better long distance estimates. On the other hand, having a smaller baseline, a smaller range of disparity values is needed for measuring a certain distance range. However, this is only true for small distances. Still this is an important factor in terms of computational

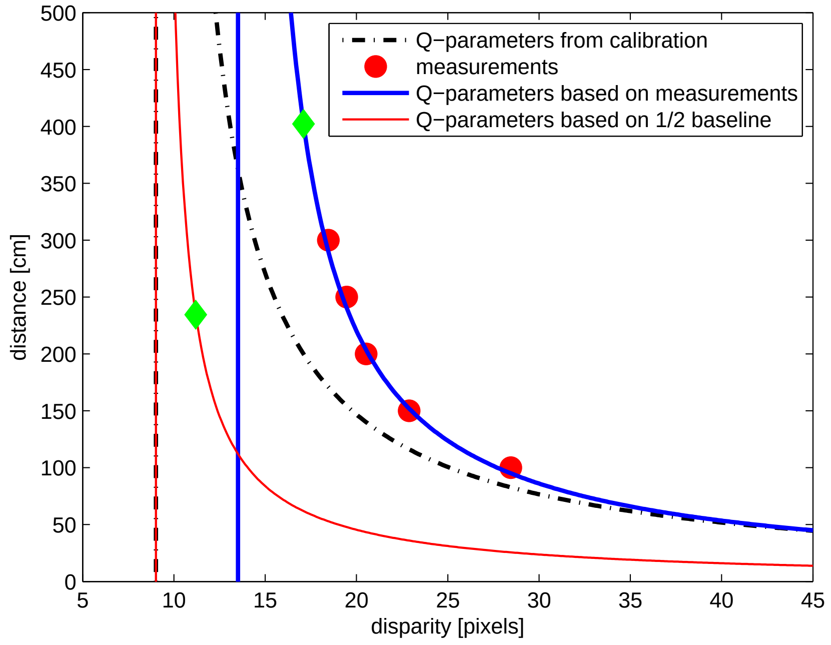


Figure 3-7: Relation between disparity and distance for the calibrated camera system. The asymptotes indicate the minimum disparity value.

demand. The larger the baseline, the more disparity values will have to be calculated for measuring the same distance range.

From figure 3-7 it can be deduced what the minimum distance is at which obstacles can be detected. The calibration parameters in the figure correspond to the rectified images as shown in the middle row of of figure 3-5. From these rectified images only the region between the white squares is useful. The images are therefore reduced to those in the bottom of the figure. These have a reduced resolution of 271×189 pixels. The images are then further sub-sampled to a resolution of 160×112 pixels to obtain real-time stereo matching performance. The image dimensions are thus reduced by a factor of 1.69. According to section 2-4-1 the stereo matching algorithm checks for a disparity range of 16 pixels. Because the images have been reduced in size, the actual disparity range has to be corrected by the resize factor and then becomes 27.1. Adding this to the minimum disparity value of 13.5 (according to the asymptote), the maximum disparity that the stereo matching algorithm evaluates is 40.6. This corresponds to a distance of 52.3cm. These numbers can vary for different calibration runs but the minimum distance was always smaller than 60cm.

Chapter 4

System Overview

The very first experiments in this project were done on the Parrot AR.Drone to demonstrate the potential of stereo vision for the task of obstacle avoidance on an MAV. Since the research in this project is focused on FWMAV, further testing was performed with the DelFly II. In this chapter the setup of the complete system used to perform these tests is explained in detail. Some changes that were made in the system design during preparation and testing are explained and motivated.

4-1 Overview

The complete system with all components and their interactions are illustrated in the diagram of figure 4-1. The logic behind the system can be summarized as follows: the stereo camera onboard the DelFly is connected to a transmitter which creates a video stream. A receiver on the ground captures the image stream and sends it to the ground station via an analog-to-digital converter. The ground station processes the video stream in order to detect obstacles and to decide which control input is required to avoid obstacles. Using a Bluetooth connection the control input is received by the onboard transceiver and passed on to the CPU. This processor delivers the control input to the servos and motor regulator. The CPU also receives angular rate information from two onboard gyroscopes. Furthermore, it can send information back to the ground station via the transceiver. More details about each of these components will follow in the next sections.

4-2 DelFly II

For this project the DelFly II has been used as test platform as shown in figure 4-2. It mainly consists of the following components: wings, cranck mechanism, fuselage rod and tail. The wings (**1**) are made of very thin Mylar. In combination with carbon rods that make them stiff and form their shape the wings are lightweight and have a good flexibility. The wings

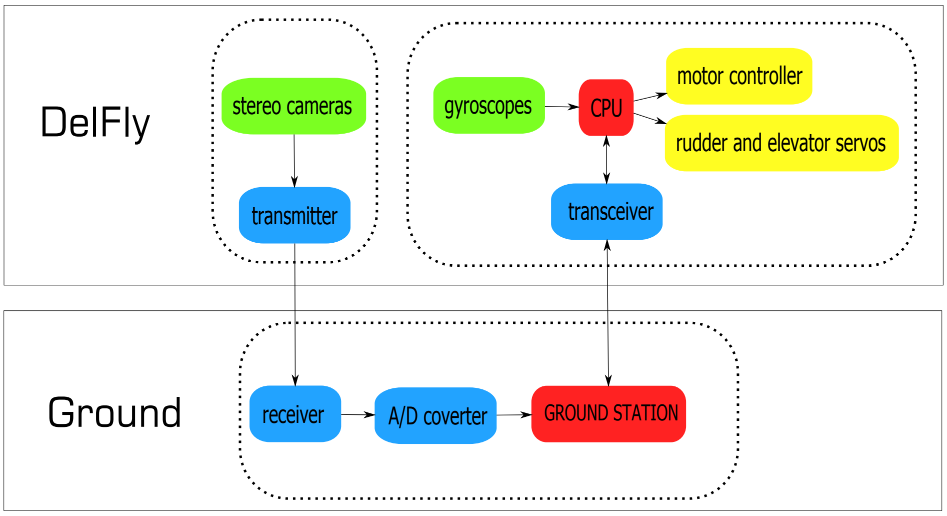


Figure 4-1: Diagram of the interaction between all system components

are placed on top of each other. The cranck mechanism (**2**) design has been changed several times. In the current design all rotating parts have their axes of rotation parallel to each other and the two wings are always in phase with each other. A special housing (**3**) for the crack mechanism was designed to minimize any type of unwanted friction between the rotating parts. The cranck mecahnism is driven by a brushless motor (**4**) specifically designed for DelFly II. A motor controller (**5**) regulates the motor such that it can cope with variations in the load due to the flapping motion of the wings. A carbon rod with sufficient bending stiffness connects the cranck mechanishm with the tail (**6**). It is also used to connect the wing and to attach any type of payload. The tail (**7**) used to be made of Mylar and carbon rods, but in the new design it is made of Polyethylene. This is a slightly heavier solution which is more robust to impacts and also easier to produce. The servos (**8**) for controlling the rudder and elevator are small enough to be embedded into the tail.

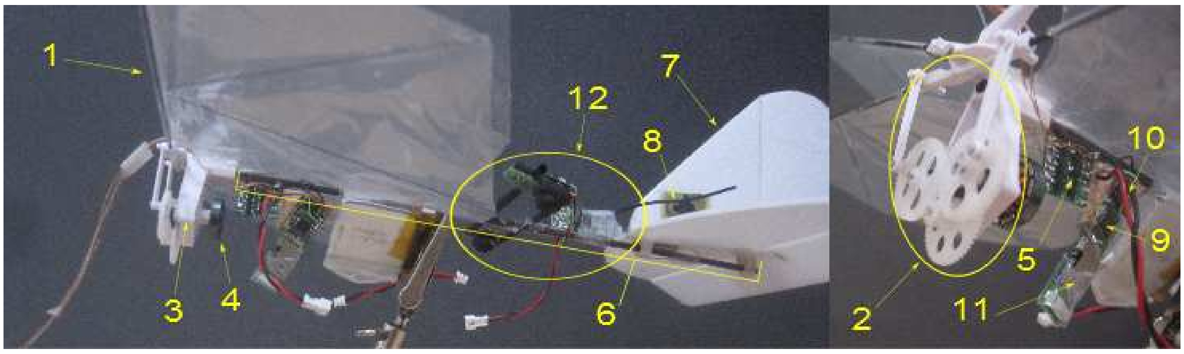


Figure 4-2: The DelFly II model used for this project. (**1**) wings, (**2**) cranck mechanism , (**3**) gear housing , (**4**) motor , (**5**) motor controller , (**6**) fuselage rod , (**7**) tail , (**8**) servo , (**9**) CPU , (**10**) gyro , (**11**) Bluetooth transceiver , (**12**) stereo camera system

4-2-1 CPU

Onboard processing is performed by an Atmel AVR ATmega88PA microcontroller (**9**) which is running on 8 MHz. It is connected to a bi-directional analog gyro (**10**) via an analog-to-digital-converter. The gyro is the InvenSense IXZ-500. It has a sampling rate of 9 kHz and the axes of rotation are positioned such that pitch and yaw rates can be measured. The gyroscopes have not been used for any test in this project.

4-2-2 Transceiver

The CPU, bi-directional gyro and the transceiver form the heart of the DelFly. The CPU controls the motor and servos, the transceiver provides a communication link to the ground. The transceiver (**11**), a Panasonic PAN1321, is capable of communicating at a speed of 2.1 Mbit/s via Bluetooth. This speed is reduced if the distance to the other Bluetooth-module is too large or other sources at 2.4 GHz lead to noise. The transceiver was replaced by a receiver later on in the project. It enables radio control (RC) communication with a Graupner TX transmitter on the ground.

4-2-3 Stereo Camera

The stereo camera system (**12**) is the main sensor on the DelFly that is used in this project. Its components can be seen in more in detail in figure 4-3. The setup consists of two synchronized CMOS 720×240 cameras running at 30 Hz and a 2.4 GHz NTSC transmitter. Because there is only one transmitter, the video streams from both cameras have to be combined as one. In the initial setup this was done as follows: an NTSC frame consists of an even field and an uneven field. When using just one camera, first the even lines are scanned and then the uneven lines. To combine two synchronous NTSC cameras, the even lines of the first camera are scanned first, and then the camera source is switched and the uneven lines of the second camera are scanned. The resulting frame consist of image lines from the left and right camera alternately, as shown in figure 4-4. The resulting frame size is still the same (720x480) but the resolution for each camera has now been reduced to 720×240 pixels.

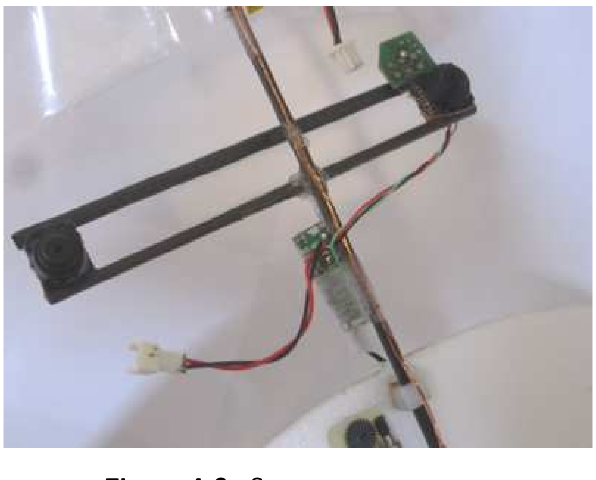


Figure 4-3: Stereo camera system

During early tests with the camera system mounted on the AR.Drone (described in the Preliminary Thesis), a shortcoming of this setup was noticed. The result from the stereo matching process was strongly affected by the motion of the drone. During hovering the results were promising and seemed reliable due to the stable behavior of the drone, but during maneuvers the results would become distorted. This effect is explained by looking at figure 4-4 again. Since all even lines (bright colors) are scanned before the uneven lines (dark colors), there is a time difference between the scan lines from the left and the right image. The first line of the 'transmitted' image comes from the right camera, the second line comes from the left camera. When the camera is at rest, it can be roughly assumed that these two lines are observing the same features. When the camera is in motion, this assumption does not hold anymore because there is a time difference of approximately 20ms (half the time between two frames) between the lines. During this time the cameras might have changed orientation and the left and right image lines cover different view directions. As a result, the epipolar constraint is not valid anymore and the output from the stereo matching process becomes distorted.

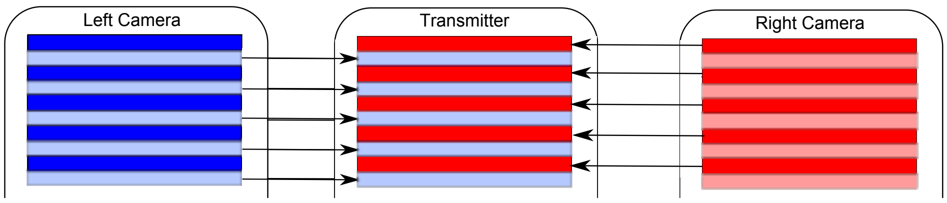


Figure 4-4: Initial method of switching between left/right camera

As a solution to this problem cameras with CCD chips could be used. These operate as 'global shutter' scanners: all pixels are scanned at the same time. However, no CCD cameras are available that have the same combination of performance, size and weight as the CMOS cameras used in the current stereo camera system. Another solution that has been thought of is using feature matching techniques to detect motion between the frames such that the images can be compensated for that. This could form a solution for maneuvers like yawing or pitching, but for vibrations this is not very realistic. The flapping motion and vibrations of the DelFly lead to image deformations that are not even consistent for a single frame.

The final solution proved to be the most suitable. The hardware of the camera system was changed such that each time after a scan line has been scanned, the system switches to the other camera. This is illustrated in figure 4-5. Note that the cameras still operate in the same way: the even lines are scanned first, and then the uneven lines. As a result the frames coming out of the transmitter now consist of two sets of two left/right images that have been taken at different times (bright and dark colors from both cameras). Two images (one from the left camera and one from the right camera) are captured on the even lines first, and after that another set of stereo images is captured on the uneven lines. The images on the uneven lines are always the most recent stereo images, and these are used for stereo processing. Each individual image now has a resolution of 720×120 pixels. The benefit of this approach is that the time difference between the stereo images has been reduced significantly. Instead of switching after 240 lines have been scanned, switching is now done after each single scanline. So the time difference has been reduced with a factor 1/240 to roughly 83μs.

The impact of this modification is shown in figure 4-6. A small test was performed where

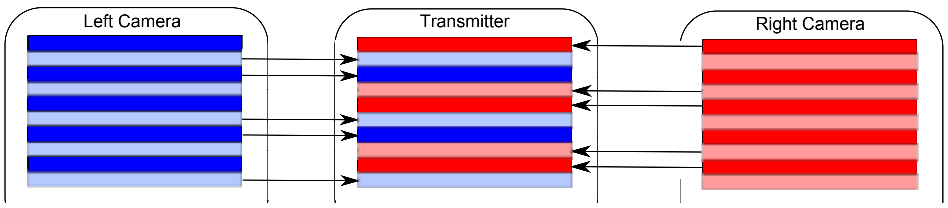


Figure 4-5: Implemented method of switching between left/right camera

the stereo camera setup was positioned at a fixed height above a large chessboard (to assure texture). A record was made of the camera stream while during the first few seconds the scene was static. After a few seconds, the chessboard was slid back and forth (left-right in the camera view) to introduce motion. The disparity was then computed for both types of camera implementations to see the effect of motion on the output. From the figure it is clear that the 'initial' system (top plot) performs significantly worse as soon as the scene starts to move. From the data one can see the left-right motion of the chessboard. When the chessboard slides to the left, the images appear to move towards each other. Hence a smaller disparity is measured. When sliding in the other direction, larger disparities are measured. From the bottom plot it can be seen that this motion is not visible from the measurements with the new setup. But it should be noted that the measurements show smaller deviations during the first seconds of the experiment (while there was no motion).

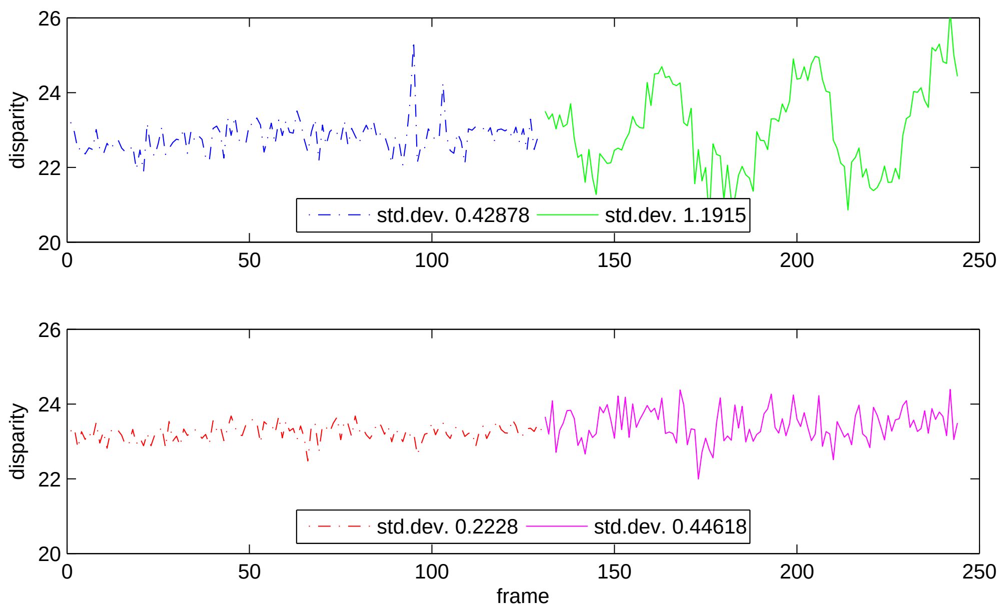


Figure 4-6: Comparison between the camera reading methods. **Top** initial method **Bottom** implemented method

4-2-4 DelFly weight

The weight of the DelFly is of large influence on its performance. Most of the DelFly II models that were built had a total weight of under 18 gram including all onboard electronics and batteries. During the first tests with the DelFly during this project it was not capable of keeping its altitude during flight for more than 30 seconds. Its weight was therefore measured in order to find a way of making it lighter. Table 4-1 gives an impression of the weight of different important components from the DelFly.

Table 4-1: Overview of DelFly weight

Component	Including	Weight (g)
DelFly	airframe motor controller autopilot	11.9
Battery pack	main battery	4.1
	camera battery	1.1
Camera system	stereo cameras transmitter	4.1
Total		21.1

As can be seen, the DelFly in this configuration is heavier than normally. This is due to a larger camera system, and the need of a separate batteries for feeding the motor and the camera system. Using only the main battery the DelFly can easily stay in the air for a few minutes. However, when the camera system is using the same battery, the battery can not deliver enough current to the motor, and as a result the lift force generated by the wings is not sufficient. Therefore an extra battery was added which adds quite some weight. The original transmitter that was used on the camera system has been replaced by a lighter version, saving almost 0.9 gram. In the current configuration the DelFly can stay in the air for two to three minutes, depending on the quality of the battery.

4-2-5 Interference

Both Bluetooth and the camera transmitter operate at a frequency of 2.4GHz. As a result, the signals interfere. The main consequence is severe noise in the video stream caused by the onboard Bluetooth antenna. To minimize this unwanted effect, no data is transmitted from the DelFly to the ground, unless this is required for the obstacle avoidance task (no data logging).

4-3 Ground system

On the ground the video stream is captured by a CCS-707 2.4 GHz receiver. The analog signal coming from the receiver is transformed into a digital signal by a Terratec analog-to-digital video converter, which on its turn is connected to the ground station laptop. This is a

2.30 GHz dual-core system running on Windows 7. All processing is done within SmartUAV, a software program developed by MAVLab which is written in C++. It is able to capture the video stream from the analog-to-digital converter via a USB port. It processes the video stream (split images in left and right image, rectify images, perform stereo matching, check for near obstacles, define control signal) and sends control signals to the DelFly. Communication with the DelFly is performed via Bluetooth. The laptop has an inbuilt Bluetooth antenna which proved to be unsuitable for this task. Except for very small ranges ($< 1m$) the delay in the signal can easily increase up to several seconds. Therefore a Class I Bluetooth USB adapter is used which allows communication over more than $20m$ when the signal is unobstructed. It also enables communication through walls, though this reduces the maximum range.

4-3-1 System delays

The system is subject to several types of delay that influence the reactivity of the DelFly. Delay occurs in the following three processes: air-to-ground video streaming, video processing and ground-to-air control communication. In the first process the analog video signal is received on the ground and first converted into a digital signal before it is passed on to the ground station. To determine the total delay for this process a simple experiment was conducted. The cameras were placed such that the mouse of the laptop was in its field of view. No onboard processing was done, the digital video stream was only received. As soon as the system detected a mouse-click, the system would show the last video frame it had received. The mouse click was given in such a way that the finger of the person who was clicking would be above the mouse before clicking and below the mouse after clicking. This way it was observed that while the mouse-click was given, the current video frame indicated that the finger was still above the mouse. It was then tested if one frame later the video frame would show the finger below the mouse. It was observed that this was also not the case: after two frames, it could be seen that the finger was below the clicking point. It can be concluded that the delay in the video signal equals two frames. At a frame rate of 30Hz, this means that the delay in the video signal is 60ms.

The delay from the video (stereo) processing can be obtained directly from the processing time/frame rate. According to the experiments in chapter 2 the processing time is 40ms.

The time to send a command to the DelFly is estimated in the following way: a signal is sent to the DelFly that instructs to send back an answer signal. The time between sending the signal and receiving the answer indicates the round trip time. This delay is variable and depends on factors like distance and other noise sources. Under good conditions the round trip time was measured to be 120ms. Assuming that this is double the time for a one way signal, the delay is estimated at 60ms under good conditions. It was noticed that during tests this delay sometimes increased up to 200ms. This brings the total delay from image capture to command to a range of [180,320]ms. A delay check can be performed during flight in order to obtain live delay times. This might be useful to estimate at which moment a control signal should be send. To limit noise in the video stream, this check should be performed at strategic moments, for example right after the detection of a close-by obstacle. As explained in section 4-2-2 the Bluetooth link was replaced by a radio control (RC) link. The reason for this was the size of the test room used for the final tests. Even though the ground station was in the middle of the 7.3×8.2 test room, the delay could increase up to more than half a second (mainly close to the walls). This makes robust obstacle avoidance impossible. The RC link is also operating at 2.4GHz but it is designed for communication over longer ranges. The delay of this link was not tested accurately but it was observed that it performed faster (instantly for a human observer) and more robust than the Bluetooth link.

Chapter 5

System Performance

In this chapter the performance of the stereo vision system is evaluated. Based on static tests and flight tests the accuracy of the system is determined.

5-1 Static Accuracy Measurements

An important performance measure for the stereo vision system is the accuracy with which distances to objects can be measured. To measure its actual performance without the influence of platform vibrations, a static test was done. For this test the camera was fixed at several distances (100,150,200,250,300,400,500 and 600cm) from a screen. The screen was a chess matt that was hanging vertically in the field of view of the camera. The stereo vision system was used in the same way as it is during flight. Disparity maps were computed from 1100 frames per measurement point. From each disparity map a small patch of 10x10 pixels was taken from the center of the map to compute the mean disparity. This disparity was used for calculating the distance from the camera to the screen. The results are shown in figure 5-1.

The figures show for each measurement point the mean of the measured distances from all image frames. The left plot gives detailed results for the measurements up to 500cm. As one would expect, the standard deviation is larger for larger distances. The measurement point at 400cm forms an exception to this observation and it has the largest standard deviation: 17cm. The mean values show some variation with a maximum error of 12.6cm for the measurement point at 300cm. The right plot also shows the measurements point at 600cm. At this point the mean error suddenly grows to 63cm and the standard deviation to 70cm.

From the results it can be observed that, at least for the static case, the stereo camera system is capable of measuring the distance to obstacles up to 500cm with a mean error of less than 50cm. For the task of obstacle avoidance this can be regarded as an acceptable performance. Obstacles that are even farther away will be detected with a lower distance accuracy. The mean error is larger than 140cm in these cases. The sudden drop in accuracy for this distance can be explained by the triangulation theory from section 3-5. Figure 3-7 in this section shows

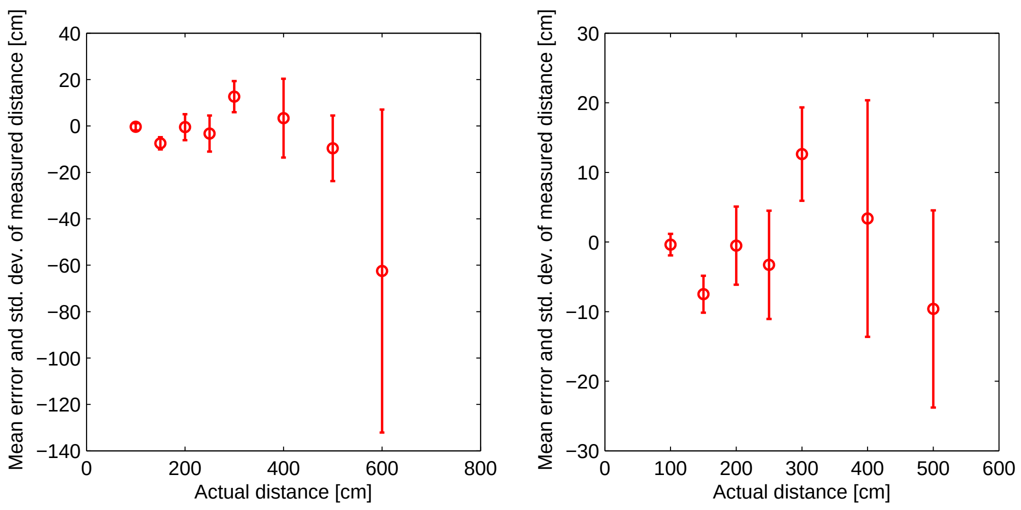


Figure 5-1: Distance measurement accuracy for static camera. The right plot shows the same results with more detail.

that the distance-disparity curve becomes very steep. Small variations in measured disparity result in large variations in calculated distance.

5-2 Accuracy Measurements during flight

The accuracy of the stereo vision system has also been measured in flight. The experiment was performed using a free-flying DelFly at a speed of approximately 60cm/s. The DelFly was flying in the direction of the chess matt. Two external cameras were used to track the position of the DelFly. Tracking was performed as follows: two video cameras were positioned such that the chess matt would be in their field of view and also the area in front of the chess matt (around 5m). The cameras were positioned on both sides of the flight path of the DelFly. By using a powerful background subtraction routine (Barnich & Van Droogenbroeck, 2011) and blob tracking, a special small marker positioned under the DelFly could be tracked. By using triangulation routines from OpenCV, the three-dimensional flight path (w.r.t the chess matt) of the DelFly was determined. The measurements from the onboard camera and the external cameras were synchronized by looking for specific features in the recorded videos. For example, the moment the wings start to flap is audible in the video streams of the external cameras. This is also the moment that the onboard cameras start to vibrate. The experiment was repeated several times under different light conditions. These conditions are shown in figure 5-2.

The results will show experiments with different lengths. This is because during the experiments the DelFly needed to fly in a straight line to the chess matt. During the experiments this was in most cases only realized during the last few meters.

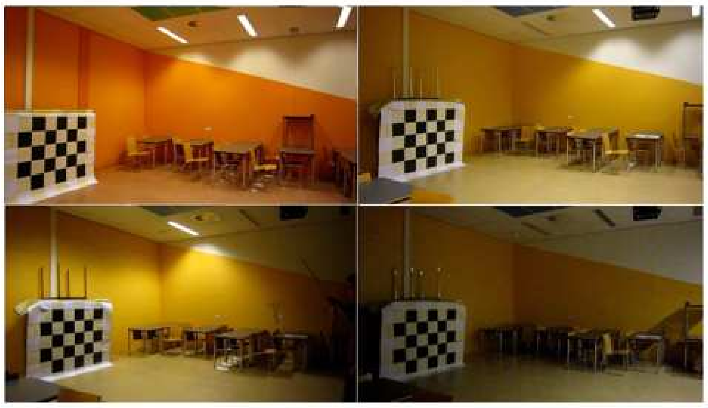


Figure 5-2: Light conditions for the flight tests. **Top Left** 5-2-1 **Top Right** 5-2-2 **Bottom Left** 5-2-3 **Bottom Right** 5-2-4

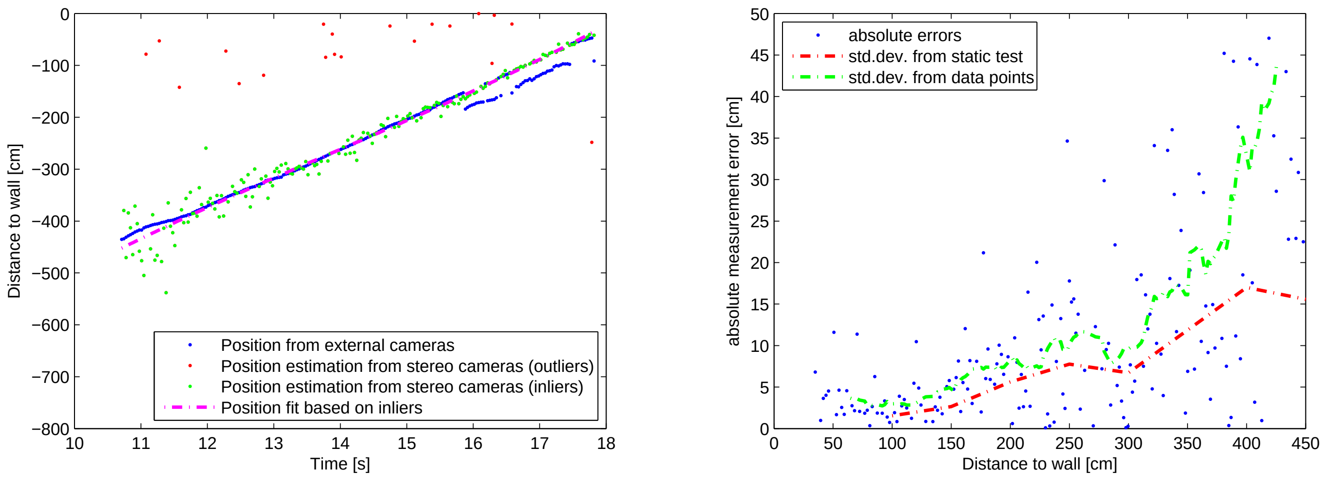


Figure 5-3: Distance measurement accuracy in flight (1)

5-2-1 All lights on

Figure 5-3 shows the result from the first flight test. The blue points in the left plot indicate the distance between the DelFly and the matt, based on measurements from the external cameras. At small distances the blue points show some discontinuities. This is a result from the background subtraction. At small distances the DelFly flies between the cameras and the matt. The white marker on the DelFly will at some points not be noticed when it is in front of a white chess board field. The tracking routine will then find another point on the DelFly, leading to triangulation errors. These measurement errors should therefore be ignored.

The red and green dots are onboard distance measurements. As can be seen from the plot, most of these points are concentrated around the blue points. However, some very clear outliers (red dots) are visible. These measurements result from a hardware problem. In some cases the video frames received by the ground station are mixed-up. The order of the scan-lines is then different from the normal case and the left-right images going to the stereo processing routine contain wrong combinations: two images from the same camera, or swapped left-right images. This results in corrupt disparity maps. Another problem is a typical haze effect which results in images that are a mixture of two images. This effect is shown in figure 5-4.

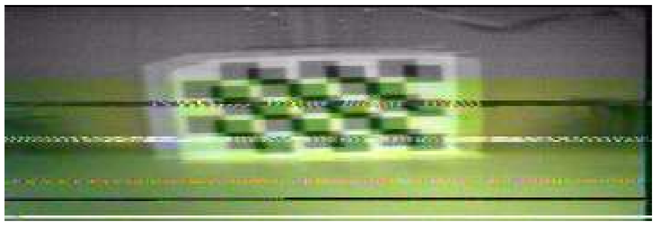


Figure 5-4: Haze effect in onboard stereo camera images.

Because the stereo vision method performs a uniqueness check, the resulting disparity maps from these corrupt images contain a lot of pixels with 'undefined' disparities. By counting the amount of undefined disparities and setting a threshold, these corrupt stereo measurements can effectively be ignored. From the results it can be seen that these disturbing effects occur irregular and infrequent.

Image noise is also a factor of influence during the experiments. Two very remarkable types of noise are shown in figures 5-5 and 5-6. The first type results in a big amount of black pixels, surrounded by bright pixels. The second type results in images than overlap each other. This has to do with a timing error in the hardware.

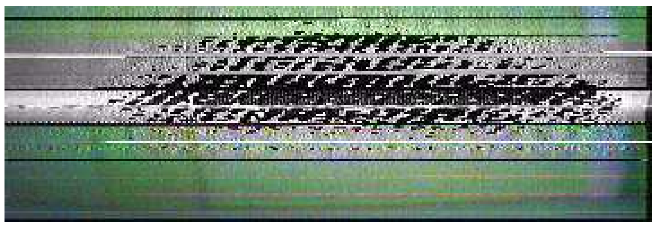


Figure 5-5: Aggressive type of image noise

In figure 5-3, and also for the other figures that will follow, these bad results (red dots) were left out. The curve fit is based on the good measurements (green dots). The right plot in the figure shows the deviation of the measurement points based on the curve fit. A running

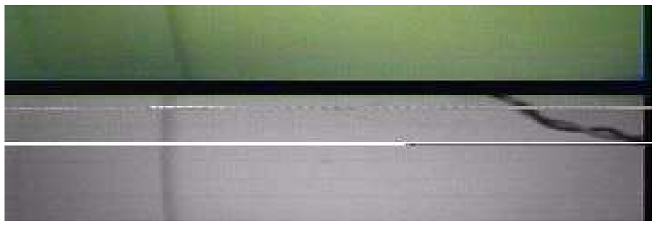


Figure 5-6: Overlapping images as a result of a timing error

average (green dashed line) was computed based on the average error with a windows size of 21. Also the standard deviation for the static case is shown in the figure for comparison.

From the left plot of figure 5-3 it can be observed that the tracked distances and the measured distances show a very good correspondence. The main observation from the right plot is that the onboard measurements have a larger standard deviation than those obtained during the static test. For distances larger than 350 cm the error seems to grow rapidly, but this at a moment that the DelFly is still turning towards the matt.

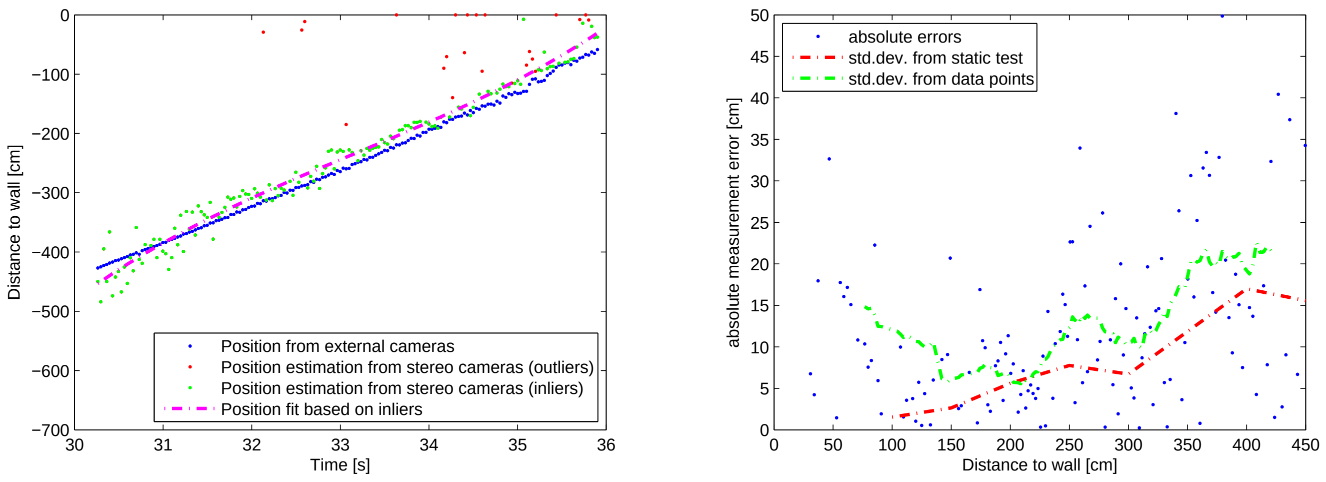


Figure 5-7: Distance measurement accuracy in flight (2)

Figure 5-7 shows the results from the second test flight. During this flight the DelFly was not flying straight into the direction of the matt, but at an angle. This might explain the difference in measured distances between the fit and the measurements from the external cameras. One would expect that the onboard distance measurements would indicate a larger distance, but this is not the case. A reason for this might be a misalignment of the stereo camera. Note that also in this case the plot on the right shows an increase in the standard deviation for the flight case. At small distances the standard deviation is relatively large. This has to do with the width of the window averaging function in combination with very bad distance measurements at small distances. Because the stereo vision method computes disparities only for a limited range, too large disparities will lead to wrong matches. This is the case at small distances (less than 60cm). This effect if visible in most plots in this chapter.

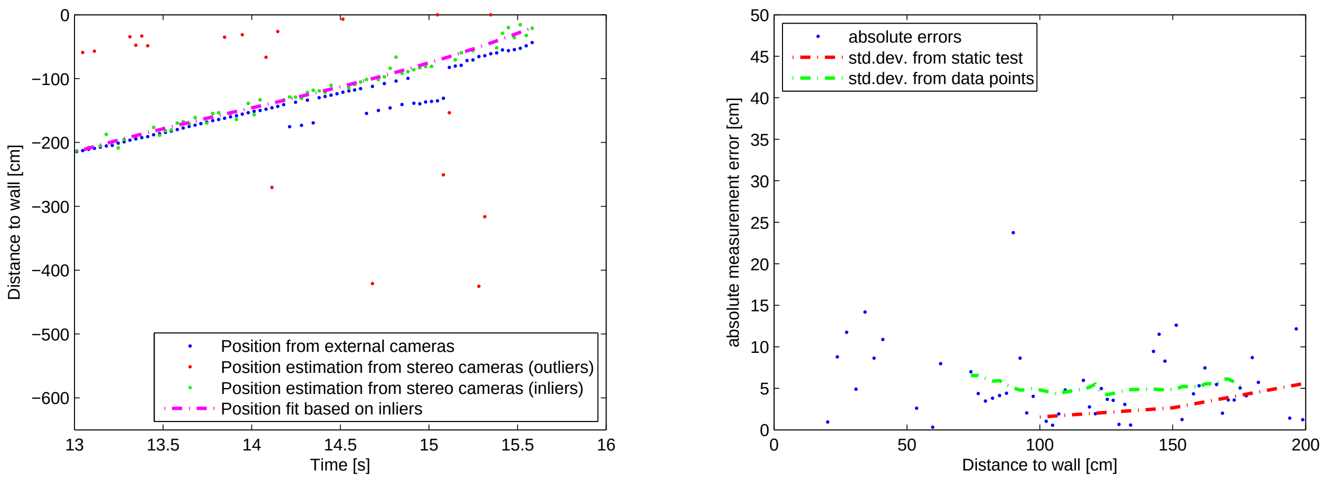


Figure 5-8: Distance measurement accuracy in flight (3)

Figure 5-8 shows the result from the third test flight. The fitted line seems to drift away from the externally measured path. This is mainly the result from the inaccurate measurements at small distances. The plot on the right shows the same trend: relatively large deviations at small distances, and a standard deviation that grows with distance and that is always larger than for the static case.

5-2-2 Reduced light (1)

The tests were performed in a room with different light groups. To measure the effect of light conditions on the performance, different light groups were selected during the tests. The following two experiments were performed under slightly reduced light condition.

Figure 5-9 shows an initial offset between the fit and the track. This was caused by a yawing motion of the DelFly during this test; the measured distance was larger because the cameras were looking at an angle. Also in this case the misalignment at small distances can be ascribed to the bad measurements at small distances.

Figure 5-10 also shows an offset (around 10cm for the main part). The right plot shows that the standard deviation nicely follows the standard deviation for the static test with an added error of a few centimeters.

5-2-3 Reduced light (2)

For the next two experiments, the amount of light was reduced further. In this case, the lights were right above the chess matt, which results in large reduction in light reflected by the chess matt.

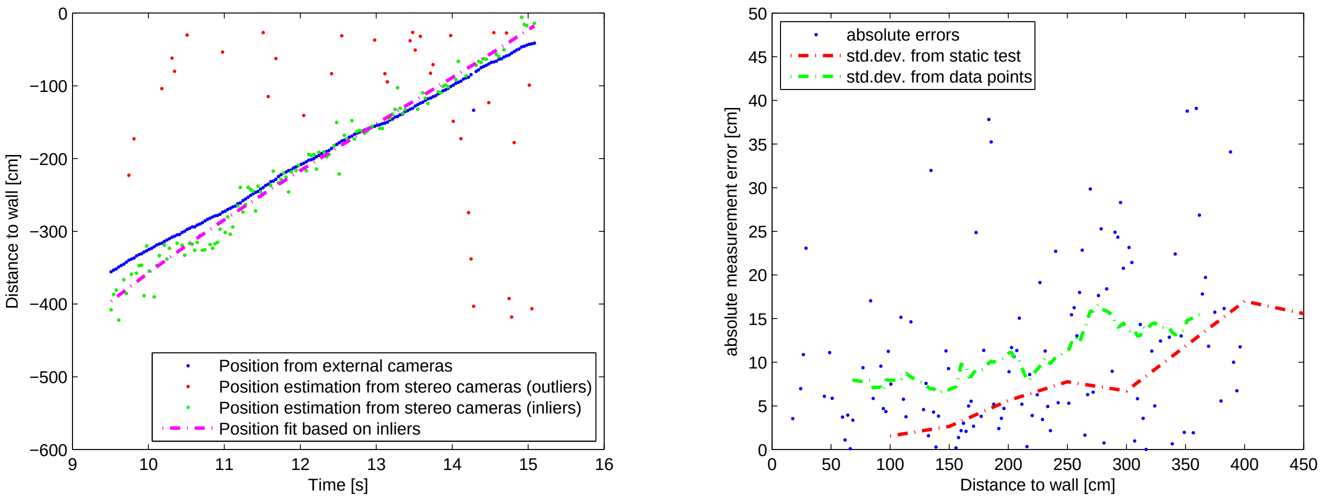


Figure 5-9: Distance measurement accuracy in flight (4)

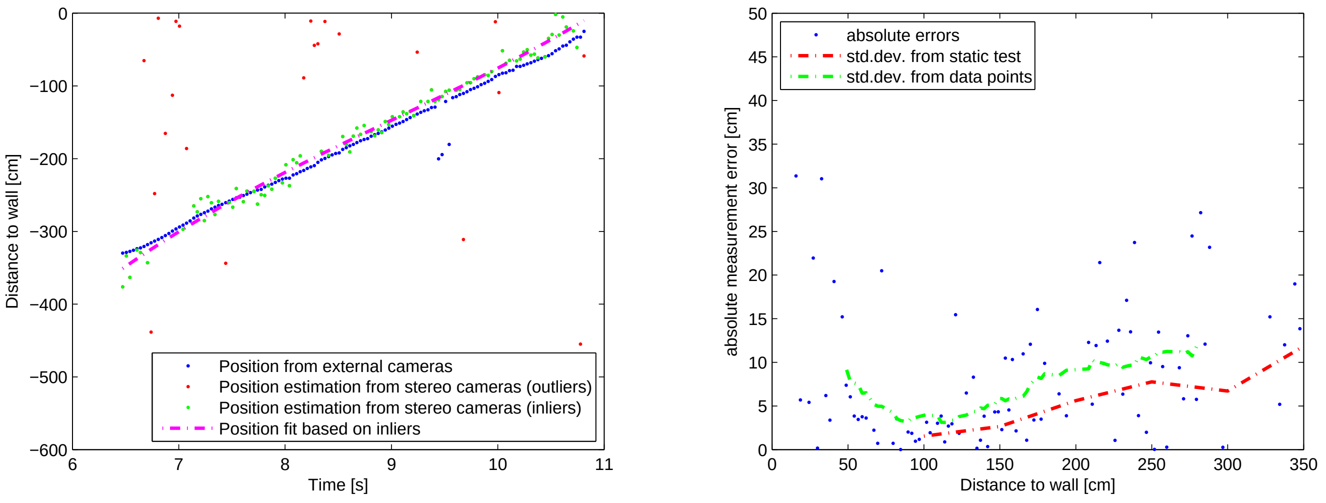


Figure 5-10: Distance measurement accuracy in flight (5)

Figure 5-11 shows that the difference in measured distance is for the main part around 20cm of. Also in this case the DelFly was flying at an angle in the direction of the wall. Despite the bad light conditions, the right plot shows a small increase in standard deviation for the measurement errors.

Figure 5-12 shows another test flight under these conditions. The DelFly was yawing left and right a little bit during the flight. It seems as if the external tracking method has difficulties because of the bad light conditions. The onboard measurements show a clear trend, while the externally measured distance shows more variation. The variation in standard deviation is comparable to the previous tests.

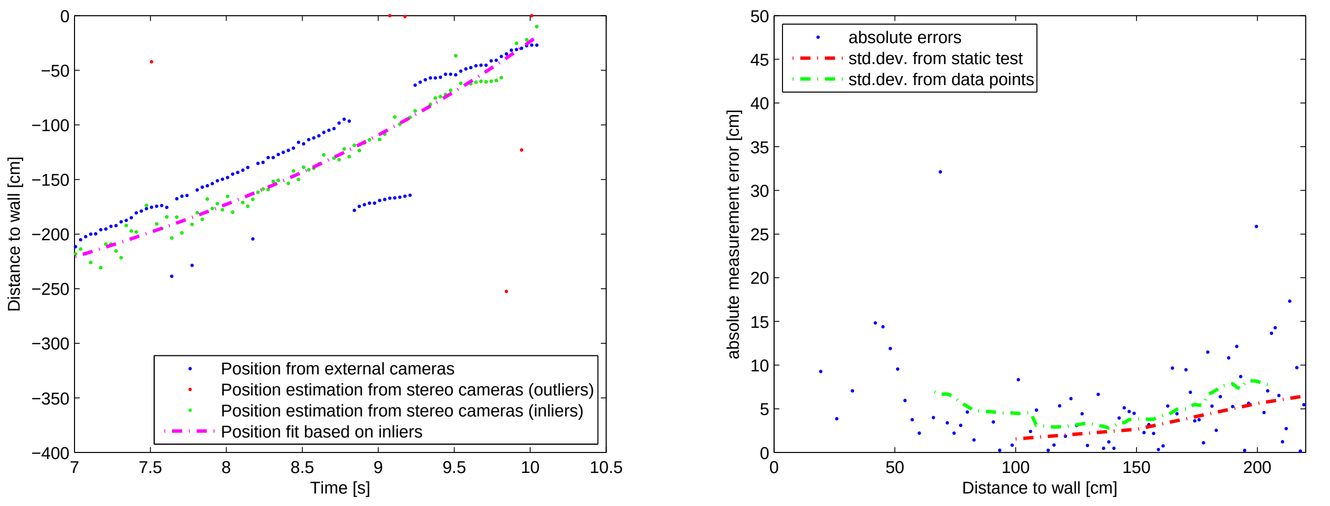


Figure 5-11: Distance measurement accuracy in flight (6)

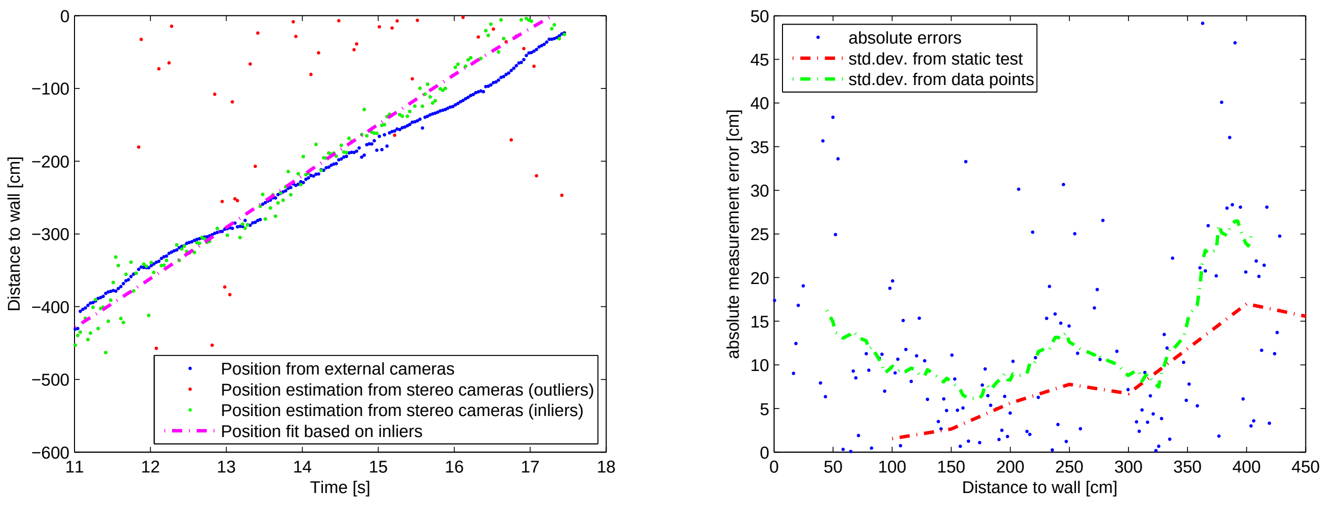


Figure 5-12: Distance measurement accuracy in flight (7)

5-2-4 Reduced light (3)

During this test the only light that was used was at the other side of the test room. Though the amount of light was small, it was better reflected by the test matt.

Figure 5-13 shows the result from the test. Despite a lot off useless images, a clear trend line can be found that nicely matches the externally tracked flight path. For the small range of distances covered by this experiment it is shown that the standard deviation in measured error is again larger than for the static test but not significantly larger than for the other previous tests.

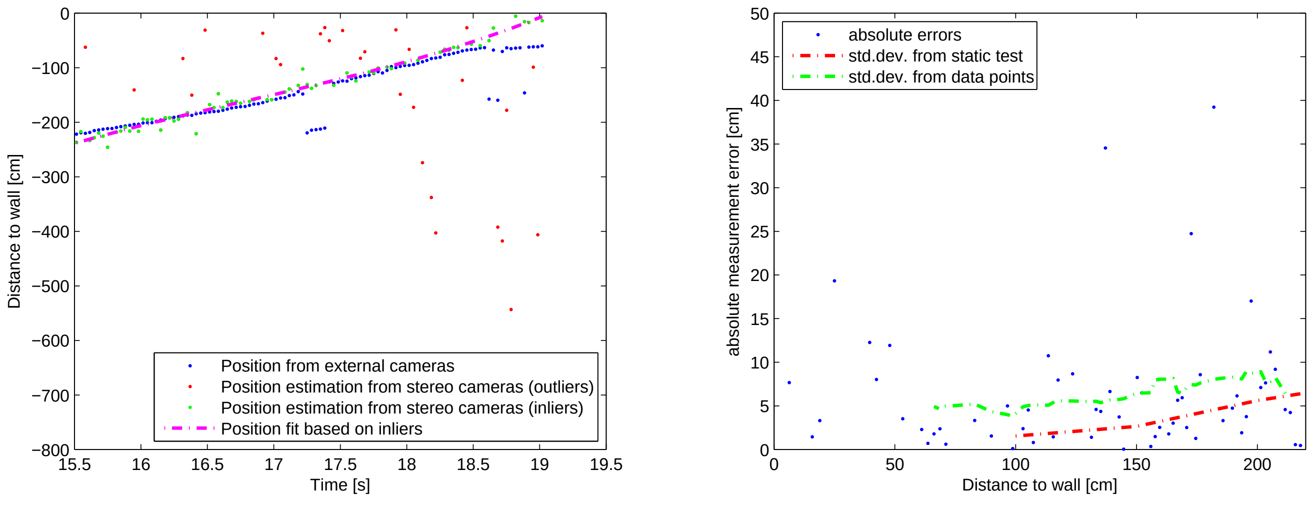


Figure 5-13: Distance measurement accuracy in flight (8)

5-2-5 Low-textured wall

A final experiment was conducted that is not fully comparable to the previous experiments, but which also gives a good performance measure. In this case the DelFly was not flying in the direction of the chess matt but in the direction of a wall with little texture. The wall is shown in figure 5-14.

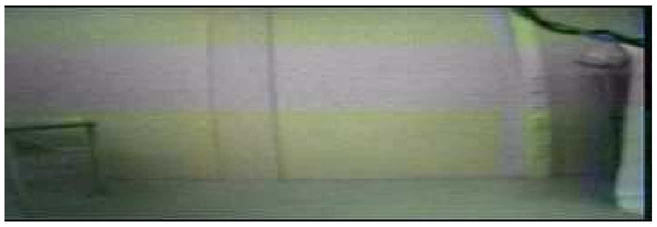


Figure 5-14: Low-textured wall used for the experiment of figure 5-15

From figure 5-15 it can be observed that the distance to the wall is measured with a higher inaccuracy compared to the other experiments. The error between the onboard and external measurements is more than 30 cm at small distances (less than 2m). From the right plot it is also observed that the standard deviation of the measured error is the largest for small distances and decreases with increasing distance. This can be explained by the fact that the closer the DelFly approaches the wall, the smaller the field of view and the amount of image features. Despite the bad results, the distance to the wall can still be estimated for distances larger than 2m.

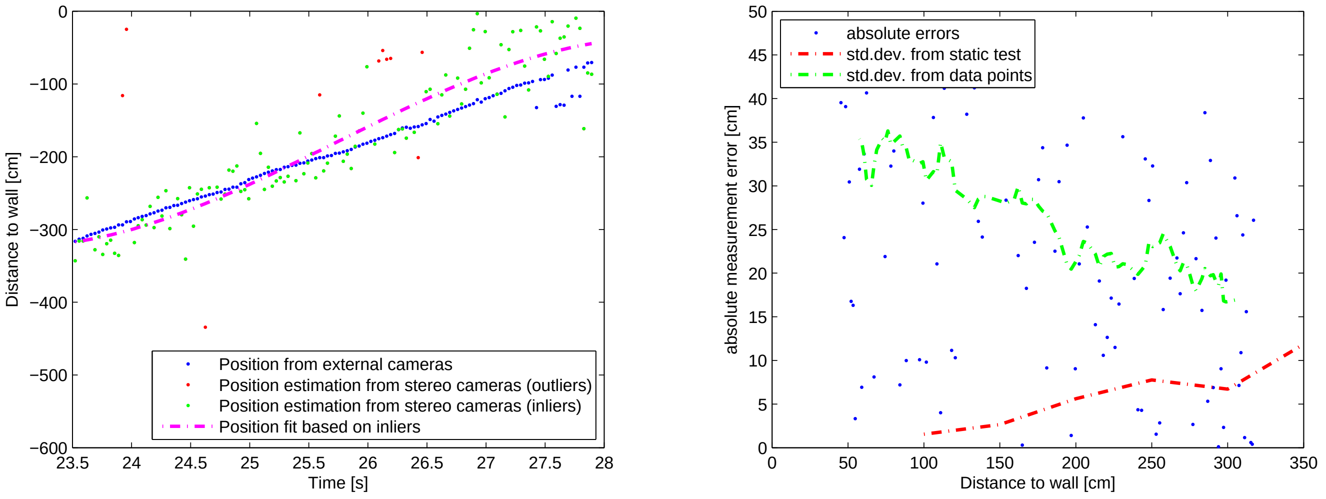


Figure 5-15: Distance measurement accuracy in flight (9)

Part II

Stereo Vision based Obstacle Avoidance

Chapter 6

Experiments

The first part of this thesis discussed how stereo vision can be used on an FWMAV (in particular the DelFly) and how accurate it can provide depth information about the environment. In the second part it is investigated how the stereo vision system can be used to perform successful obstacle avoidance with the DelFly.

In this chapter three different experiments are described. These experiments were conducted to test three different types of obstacle avoidance strategies. Both the strategies and descriptions of the experiment setups are discussed.

6-1 Direct Yaw control

The first experiment is about an obstacle avoidance algorithm that is based on the method described by (De Croon, Groen, et al., 2012). Based on the *appearance variation cue* and *optic flow*, time-to-impact estimates are performed which form a measure for the distance of obstacles. Basically, they compute time-to-impact τ for the left and right part of the onboard camera images independently. If τ_L is larger than τ_R, the DelFly makes a turn to the left. This rule can not be used directly. Instead it is used to integrate obstacle information over time while only using measurements that are sufficiently reliable. Their experiment was conducted in a room of ~3.20×2.70m. One of the observations was that that because of the small room size for most of the time the DelFly was performing turns (instead of flying straight, which is required for their optic flow measurements). One of the reasons the DelFly would eventually collide during their experiments was a bad turn decision based on few measurements right after a turn.

6-1-1 Turn Strategy

The strategy for deciding to make a left or a right turn is in principle the same as for the experiment discussed above. But instead of using time-to-contact estimates, the 'amount' of obstacles at short distance are determined based on the computed disparity map. Also in this strategy this is done for the left and right image parts individually. To perform this calculation efficiently, a reference disparity map is computed in advance, which indicates for each individual image pixel how large its disparity value may be before it counts as an 'obstacle'. The map calculation is based on known camera calibration parameters and a self-defined safety region. The safety region is a three-dimensional space defined in the camera reference frame that should be obstacle-free. For the current strategy this region as shown in figure 6-1 has the shape of a disk with a height of 40cm (20cm above and below the camera) and a radius of 1.10m. The radius was chosen experimentally such that it would be as small as possible (to reduce turning time) while providing enough turn space (to deal with delays and turn radius). The shape of the safety region is actually a slice of a disk with the cameras in the center and the slice width defined by the field-of-view.

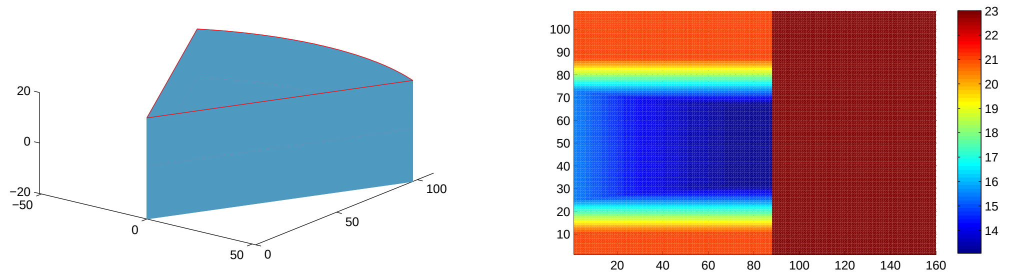


Figure 6-1: Impression of the safety region. This region should be obstacle free. The right plot indicates the allowable disparity values per pixel for the left camera half.

Two separate reference disparity maps were computed for this turn strategy: for both image halves individually. In figure 6-1 the map for the left camera is shown. During flight the disparity maps computed by the stereo vision algorithm are directly compared to the two reference maps. By summing the amount of pixels that have a larger disparity value than allowed by the reference map, two 'obstacle'-signals are obtained. These signals are low-pass filtered to suppress the effect of noisy computations. When the left obstacle-signal reaches a threshold, the DelFly turns right, and vice versa. The turn is initiated by giving a predefined step input to the rudder. This rudder input is a fixed value that can be set separately for left and right turns. Its value was chosen such that the turns are steady and symmetrical and the turn speed is not too fast to avoid spiral motions. The turn will end only as soon both 'obstacle'-signals become lower than another (and smaller) threshold. As soon as the lower threshold has been reached, the rudder will go back to its trim position. However, if one of the 'obstacle'-signals reaches the higher threshold again within a predefined safety-time, the DelFly will continue its previous turn, regardless of which of the two 'obstacle'-signals reached the threshold. This will prevent the DelFly to turn back into the direction it just turned away from, since this is most likely not a safe maneuver.

6-1-2 Experiment setup

The experiment was conducted in a room of ∼4.23×4.42m. Figure 6-2 shows a floor plan of the room. The images on the sides give a good impression of its appearance. Except for the walls the main obstacles are two black cabinets. The door on the left was closed during the experiments, and part of the window on the left was covered to prevent window collisions. It should be noted that the images in the figure only show a part of the scene (mainly the top part) while the onboard cameras of the DelFly could see more of the lower parts of the room. The lights were switched off during the experiments since they resulted in a flickering effect in the stereo cameras (due to interfering camera and light source frequencies). During the experiments the 'obstacle'-signals were logged, as well as turn events. Furthermore, an onboard image was captured at the moment a turn event (left/right turn or end of turn) occurred. The DelFly was connected to a stick by a fishing line. The experimenter held the stick in his hand and walked behind the DelFly while making sure that the line would hang loose. It was only used to prevent collisions from bad decisions and pulling the DelFly up to its flying height (after turns). This is needed since maximum power is required to fly in a straight line (after less than a minute, only during the first part of the flight the DelFly has a little over power). During turns, the DelFly tends to loose a little height. The elevator was given a constant input such that the forward speed would be around 0.6m/s during the test.

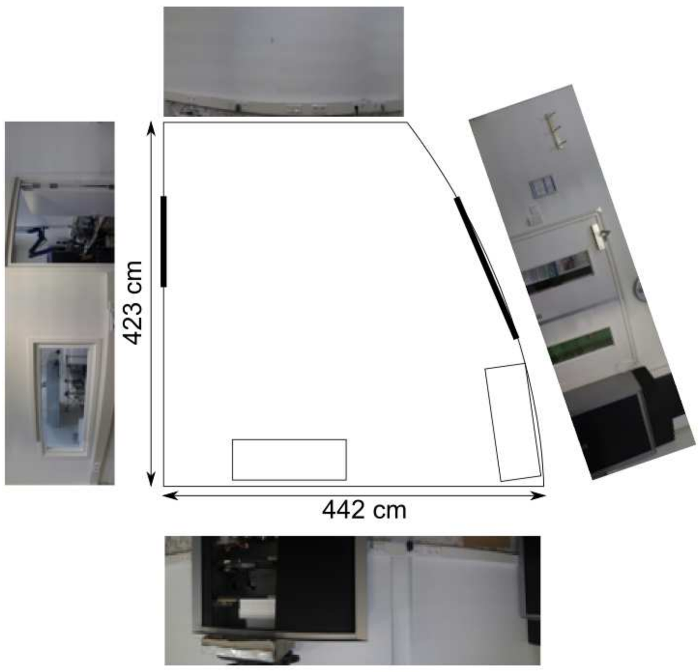


Figure 6-2: Floor plan of the test room. The images around show the walls, doors and cabinets in the room.

6-2 Direct Yaw and pitch control

The second experiment is a follow-up of the first one, and it was done the same way. The only difference is the addition of a simple pitch control rule. During unobstructed flight, the elevator is in its fixed position such that the DelFly will fly at a speed of around 0.6m/s. As soon as an obstacle needs to be avoided, a turn is initiated the same way as in the first experiment. At the same time the elevator input is changed such that the DelFly will loose its speed and start to hover. As a result the DelFly will change its heading (by yawing) while it keeps its position. Obstacles can be avoided without the risk of making a turn and colliding with another object out of the camera field of view.

Before this test was conducted, it was already known that the DelFly in its current configuration is too heavy for hovering. It will definitely loose height at the turning points. However, the experiment can be useful in demonstrating that this simple avoidance strategy is suitable for an (FW)MAV as long as it is able to hover. Future designs of the DelFly might be able to hover more efficiently and could use this strategy for maneuvering in small spaces.

As an alternative, the next section proposes another strategy that should allow the DelFly to fly at a constant speed while making sure that it does not collide with obstacles on the side. A drawback of this strategy is that it cannot be applied for flying in small spaces.

6-3 Look-Ahead Yaw control

As will be demonstrated by the second experiment, hovering is not possible with the DelFly in the configuration that is used for this project. The first experiment demonstrates that responding directly to obstacles in the field-of-view will result in collisions with obstacles that are outside the field-of-view. Therefore a new turn strategy has been devised to circumvent these problems.

6-3-1 Turn strategy

In this new strategy the DelFly continuously flies with a constant speed (fixed elevator setting). A turn is initiated when there too many obstacles (pixels with a large disparity value) detected in the safety region. The safety region is defined such that it covers an area large enough for the DelFly to turn around 360 degrees. Because of the limited field-of-view of the camera, this turn area will lie ahead of the current position of the DelFly. Figure 6-3 shows one of the ideas based on this definition of the safety region. The region is defined in the camera reference frame, with the x-direction positive to the right, the y-direction positive up and the z-direction positive in the direction of flight. Starting at the origin (position of the camera), two oblique lines define the camera-field of view. The dashed line is the trajectory the DelFly will follow as soon as too many obstacles are detected. In this case the DelFly turns in steps of 90 degrees. This offers the possibility to decide after each turn if the new flight direction is safe to continue with. If this is not the case, another turn follows. Because it takes time to decide if the new flight direction is safe and because of delays between ground and air, in between turns there are straight paths. Note that the turn area is shifted 178cm ahead such that it fits inside the field of view. Around the flight path extra space is left to account for the width of the DelFly and for uncertainties in distance measurements, signal delays and unpredictable DelFly responses. Also note that initially a left turn is performed such that the turn area is centered in the field-of-view. This way the length of the safety region (in z-direction) is kept as small as possible.

The reason for turning with angles of 90 degrees is that this allows obstacle detection for the actual new flight direction. Another option is to keep turning until a safe flight direction has been found. However, before the ground station has evaluated the new direction of flight

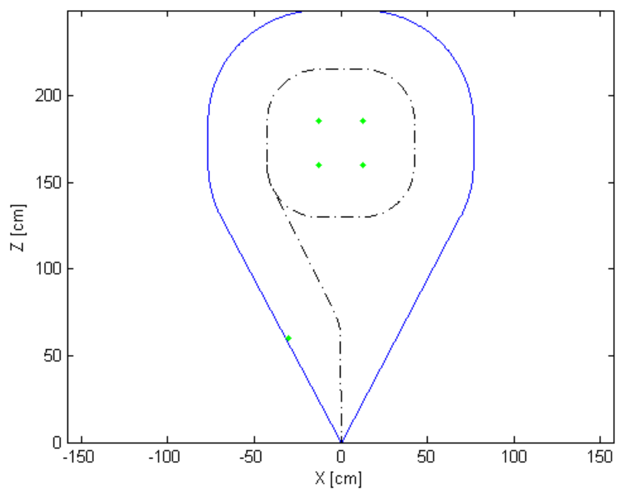


Figure 6-3: Turn strategy using fixed-angle turns

and has sent the command to stop turning, the DelFly will have turned further during this delay. On the other hand, a large drawback of turning in steps of 90 degrees is that it is not guaranteed that a safe direction of flight is found.

Another issue with this strategy is the turn performance of the DelFly. (De Croon, Groen, et al., 2012) demonstrated that with the current onboard hardware it is possible to perform turns with a predefined angle. However, experiments revealed that in the current configuration of the DelFly the gyroscope measurements were not reliable. This is because of the stiff connection between the gyros and the airframe. The fast flapping motion therefore has a disturbing effect on the measurements. The earlier successful results obtained by (De Croon, Groen, et al., 2012) were possible by hanging the autopilot board under the airframe using small wires.

Because of this issue with performing fixed turns and because it is more advantageous to be able to fly in any (undefined) direction, another idea for the safety region is proposed, as shown in figure 6-4. As soon as too much obstacles are detected in this region, the DelFly will fly the trajectory of the dashed line. After 225cm a right turn will be initiated. During the turn the same safety region is used to detect a new safe flight direction. As soon as it is found, the turn will be terminated. Because a turn might be terminated by mistake or an overshoot can occur due to delays, the turn will be continued immediately if the new direction of flight is not regarded as safe anymore. This is only possible within a fraction of a second after turn termination. This type of turn recovery has been taken into account the the safety region definition. This is why the outer circle has been drawn. Around this outer circle extra safety margin has been included to accommodate for the width of the DelFly and inaccuracies in range estimations. Note that the turn area has been centered in the image in order to minimize the size of the safety region. As a result, the flight trajectory towards the turn area is drawn as a slanted line. For this reason the stereo vision cameras were mounted on the DelFly with an offset angle to align the drawn flight trajectory with the flight direction of the DelFly. In other words, the cameras are pointed a little bit to the right.

In theory this strategy will direct the DelFly from turn point to turn point where it is

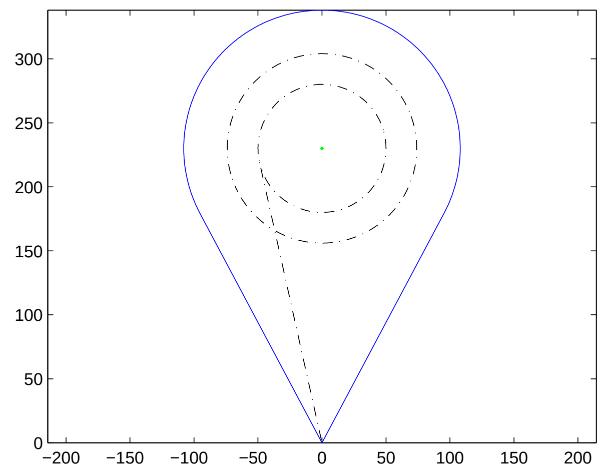


Figure 6-4: Turn strategy using continuous turns

guaranteed that in between the turn points there will be no obstacles. For the safety region a turn diameter of 50cm is assumed. The DelFly can do faster turns, but this makes it harder to find safe flight directions since the angular speeds become high. It was observed during experiments that the turn speed and radius vary while giving the rudder a fixed input. As a result the turn trajectory is not perfectly circular anymore, but assuming that the mean turn radius is 50cm suffices. For the outer circle it is assumed that the DelFly flies with a forward speed of 100cm/s. This number is higher than for the earlier experiments to increase the flight endurance. It is also assumed that if a turn is wrongly interrupted, the turn is continued within 250ms.

In this strategy only rudder commands are used. Because the obstacle measurements are sensitive to noise, filtering is required to increase robustness. For this reason a logical diagram was developed as shown in figure 6-5 which decides upon rudder inputs. In this logic each turn is divided in phases. During Phase 1 the DelFly flies straight and the avoidance region as in figure 6-4 is used to detect obstacles. A threshold is used for this decision based on the amount of pixels that exceed the disparity constraint defined by the safety region. To suppress noisy measurements, filtering is applied as follows. Each time the threshold is exceeded (250 pixels), the current time is stored. It can then be checked if the threshold is exceeded ten times within one second. If this is the case, it is concluded that there is an obstacle. The earliest detection time of these ten detections is then used as reference time for the second phase. In Phase 2, the DelFly still flies straight and waits until it has reached the point in figure 6-5 where the turn needs to be started. This time to turn is just over 2 seconds. However, because the turn response of the DelFly to rudder inputs is sluggish initially, and because of communication delays, the time to turn was tuned experimentally and set a a value of 1500ms. After this time has elapsed the turn is initiated in Phase 3. During the turn it is checked if the current direction of flight is obstacle free. Because the DelFly also rolls due to the rudder input, the map with allowable disparities (figure 6-1 right) is rotated over several angles (10, 20, 30deg). The obstacle check is performed on each of these maps, and the smallest output value is used. As soon as this output value is below a threshold of 200 pixels, Phase 4 starts. No filtering is used here because this will result in unwanted delay. The turn speed of the DelFly is around 1 rad/s and small delays result in large flight direction differences. To compensate for the quick decision making and turn overshoots, it is checked in Phase 4 if the new flight direction is indeed a safe direction to fly. If within one second after turn termination the obstacle threshold of 250 pixels is exceeded again, the turn is resumed in Phase 3. Otherwise the new flight direction is regarded as safe and Phase 1 starts again. In Phase 1 also another check is performed to detect obstacles at short range. The avoidance region as in figure 6-1 is used here. A threshold of 500 pixels is used. If it is exceeded three times in row, Phase 3 is activated to start turning immediately. The main reason for including this rule is the sensitivity of the DelFly to wind disturbances which changes the flight trajectory to such an extent that obstacles initially out of the field-of-view will result in collisions. This rule is used to prevent unexpected collisions but does not guarantee that the DelFly will be able to continue safely.

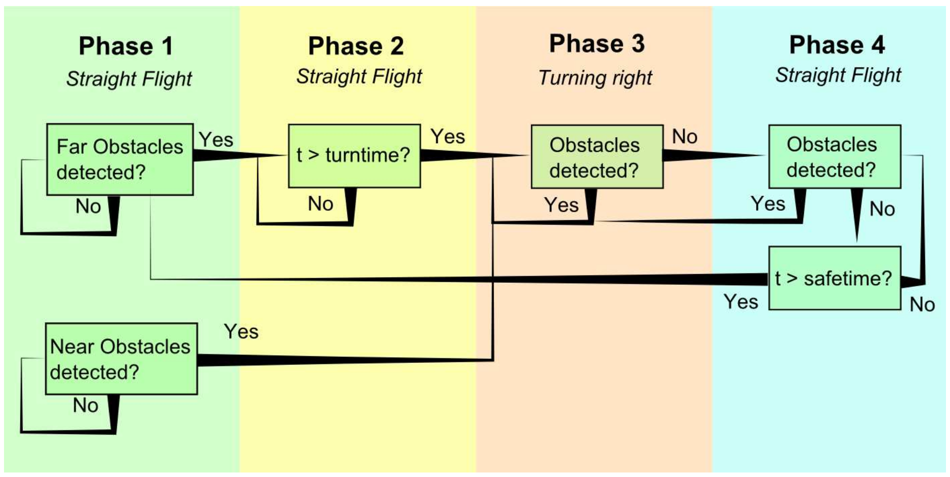


Figure 6-5: Flight phase diagram for rudder control

6-3-2 Experiment setup

Tests with this turn strategy were performed in a larger test room because of the size of the safety region. In the test room from the first experiment the DelFly would keep turning continuously. The test room is visualized in figure 6-6. It contains some large cabinets along the walls and smaller cabinets in the middle. The area on the right is large enough for the DelFly to enter. There are desks along the lower and left wall. The lights in this room could not be switched off. These lights do not result in a flickering effect as was the case in the smaller test room. However, the lights influence the stereo images since the cameras have different sensitivity to direct light. This will be further discussed in the result section. The forward speed of the DelFly was between 0.8 and 1.0m/s during the experiments by increasing the downward deflection of the elevator to increase the lift capability of the wings. This enabled longer flight times. During the test flights onboard camera images, disparity images and measurement data was stored: flight time, obstacle signals for the large and small safety region, and the phase of the controller (as in figure 6-5).

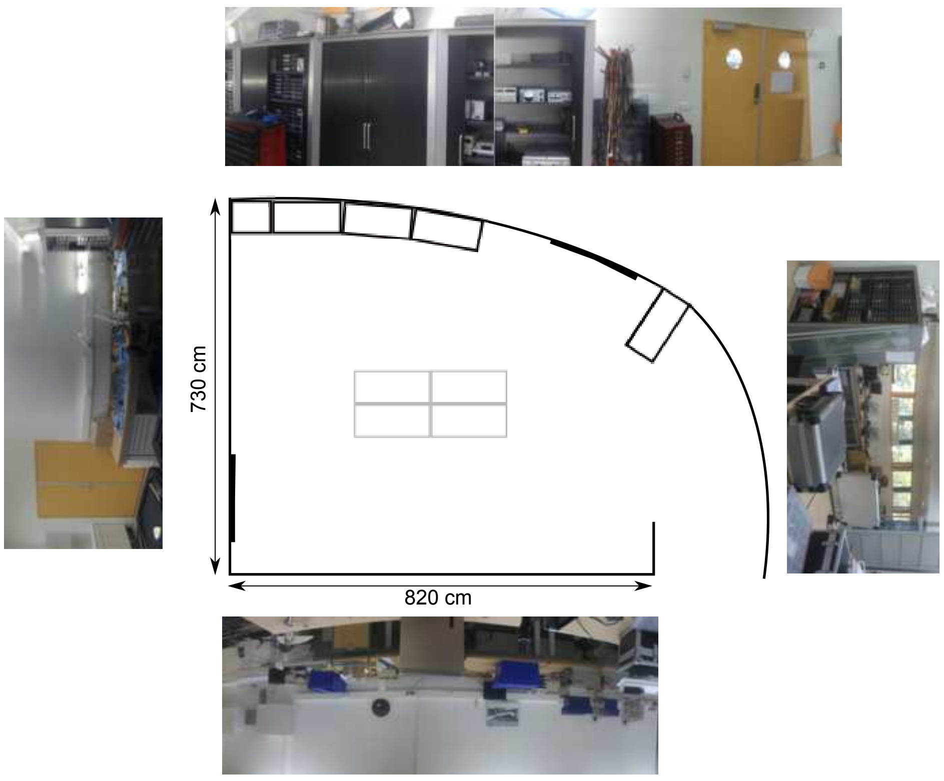


Figure 6-6: Floor plan of the test room. The images around show the walls, doors and cabinets in the room.

Chapter 7

Results

7-1 Direct Yaw control

This experiment was repeated several times and resulted in various observations. As a general result it can be stated that the obstacle detection performed well. The obstacle avoidance strategy showed some expected flaws. This will be illustrated by data recorded during the flights.

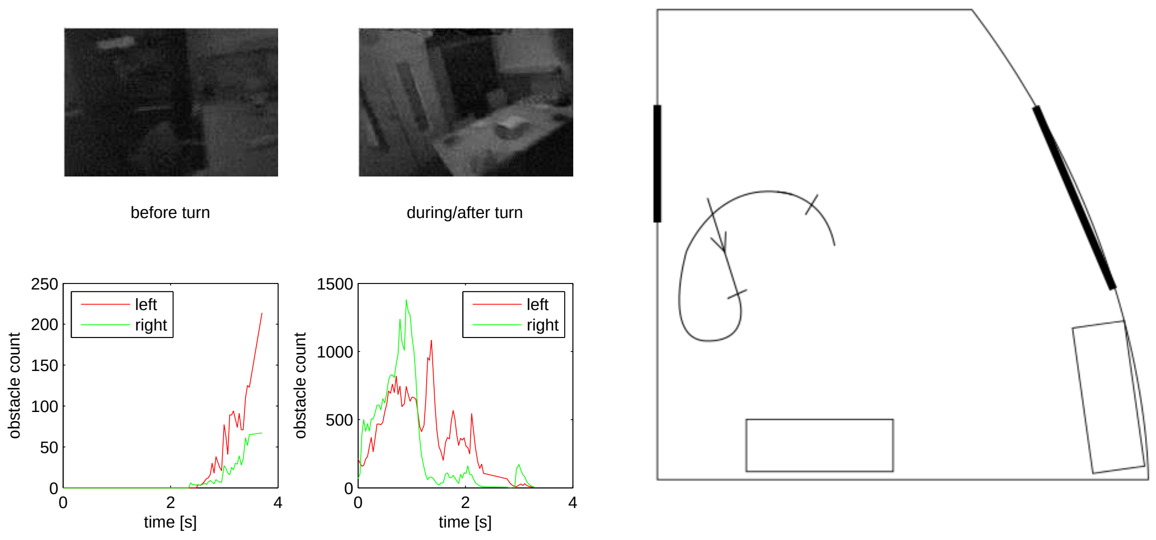


Figure 7-1: Obstacle detection and avoidance situation (1). **Top left** onboard cameras image at turn initiation and end of turn **Right** flight trajectory **Bottom left** obstacle detection signals before and during turn. Yellow regions indicate periods without obstacle detection due to bad camera images

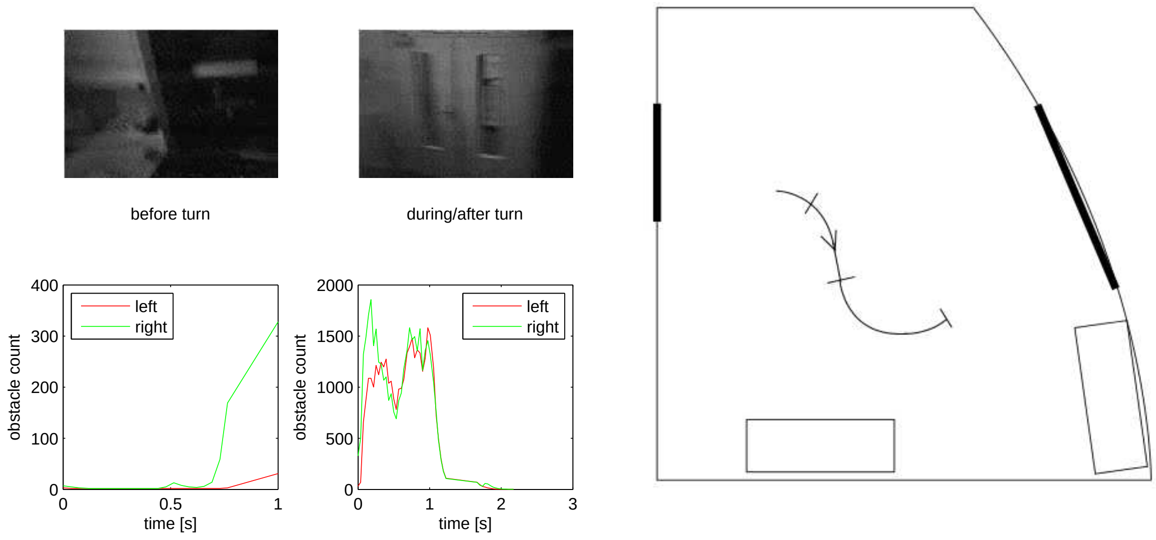


Figure 7-2: Obstacle detection and avoidance situation (2). **Top left** onboard cameras image at turn initiation and end of turn **Right** flight trajectory **Bottom left** obstacle detection signals before and during turn. Yellow regions indicate periods without obstacle detection due to bad camera images

Figure 7-1 shows a situation during the first seconds of one of the test flights. The sketch on the right indicates the position of the DelFly at the start of the flight and during the first turn. During the first seconds, it flies close to the wall. But, as can be seen in the left onboard image, the wall on the right is outside the field of view. The left bottom plot shows the 'obstacle'-signals during the last seconds before the turn. In this case these values are initially zero because the obstacle detection was not activated yet. From the plot it can be seen that the cabinet, (mainly) on the left side in the image, lets the left obstacle signal increase faster than the right signal, as expected. When the left signal exceeds the threshold (in this experiment set at 200), a turn to the right is initiated. As a result, the DelFly turned into the direction of the wall. It was prevented from colliding with the wall by pulling it aside using the fishing line.

The middle bottom plot shows the amount of obstacles detected during the turn. It can be seen that during the first half second of the turn, the amount of right 'obstacles' increases first. This is because the wall now enters the field of view. After one second the DelFly has turned around (with some help from the fishing line) and the right obstacle signal decreases. Since the wall is now in the left side of the view, the left obstacle signal is now very high. While turning away from the wall, the left obstacle signal decreases. It can be observed that it takes fairly long before a safe flight direction was found during the turn. First it takes two seconds before the left obstacle signal decreases below the threshold (set at 50). If this threshold would have been the same (also 200) the turn would have been ended approximately one second earlier. However, earlier experiments showed that for a threshold value of 200, turns would very frequently be ended too early (and then continued immediately, but with some delay). Also note that at the end of the turn, the left obstacle signal decreases below 50, but at the same time the right signal increases again. From the right onboard image in figure 7-1 it can be observed that the DelFly rolls while making a turn. The table in the image appears to be shifted up in the right side of the image. Apparently it is then detected as an obstacle.

In the sketch on the right it is indicated at which points the turn was initiated and ended. The end point corresponds to the location where the right onboard image (see figure) was taken. In the sketch it is indicated that after this moment the DelFly continued its turn a bit longer. This is a result from the delay between the ground station and the DelFly.

Figure 7-2 shows how the DelFly continued after the first turn. It is flying into the direction of the same cabinet as before, but now it is in the right side of the camera field of view (top left image). The left bottom plot indicates that indeed an obstacle is detected on the right side. During the first 0.2s after turn initiation (middle bottom plot), the 'obstacle'-signals increase quickly since the DelFly approaches the cabinet. Then, after some delay, the turn command is received onboard and the DelFly starts to turn to the left. Note that around one second later, both signals drop quickly. However, it takes another second before the signals drop below the threshold value. Apparently this is caused by the other cabinet in the corner.

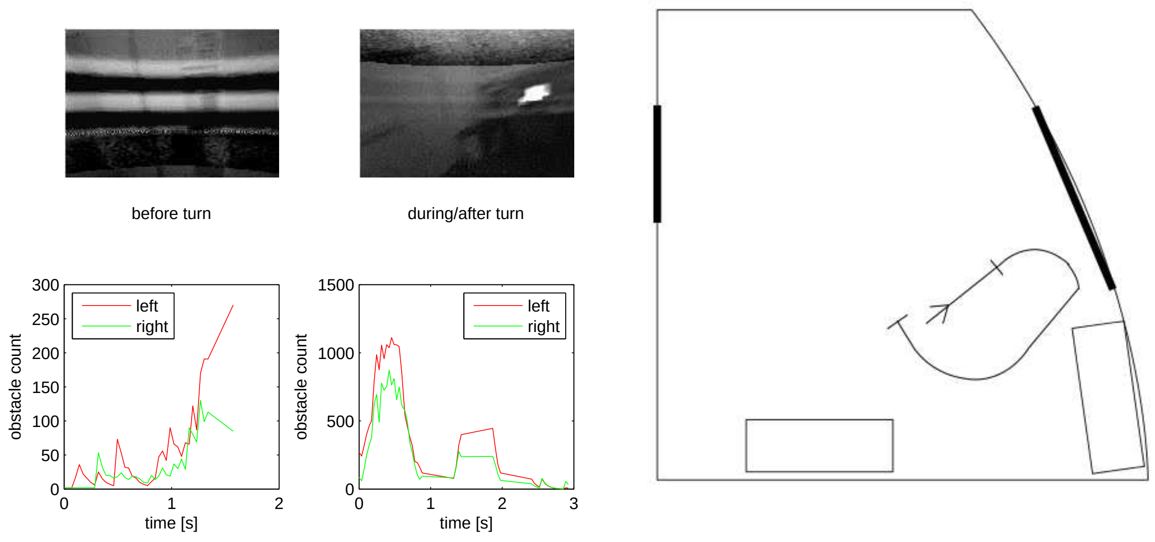


Figure 7-3: Obstacle detection and avoidance situation (3). **Top left** onboard cameras image at turn initiation and end of turn **Right** flight trajectory **Bottom left** obstacle detection signals before and during turn. Yellow regions indicate periods without obstacle detection due to bad camera images

Figure 7-3 shows what happened during the next turn. As a result the turn was initiated fairly late. The DelFly did not collide with the doors, but it would have collided with the cabinet if this was not prevented by the experimenter. This is why the trajectory in the right of the figure shows a jump at this point. The turn was continued after that because the other wall (at the bottom of the sketch) became close, as well as the other cabinet (mainly in the left signal, as expected). The turn ended later than expected, which is probably caused by noise.

During this experiment a lot of bad turn decisions were made by the system, as illustrated by the previous examples. Another observation that was made is that the response of the DelFly to rudder inputs is unpredictable. Because it results in a roll and a yaw motion at the same time, the response is different between situations. As a result the turn speed and turn radius are not constant. This is not a critical factor since this can be taken into account in the avoidance strategy.

During another experiment the amount of conflict situations (as described before) was fairly limited and several good turn decisions were taken in sequence. This demonstrates that the obstacle detection routine is performing well, but that the applied avoidance strategy does not satisfy under all conditions.

7-2 Direct Yaw and pitch control

As explained in the previous chapter, this second avoidance strategy is meant to demonstrate the benefit of making turns without forward speed. An example situation is shown in figure 7-4.

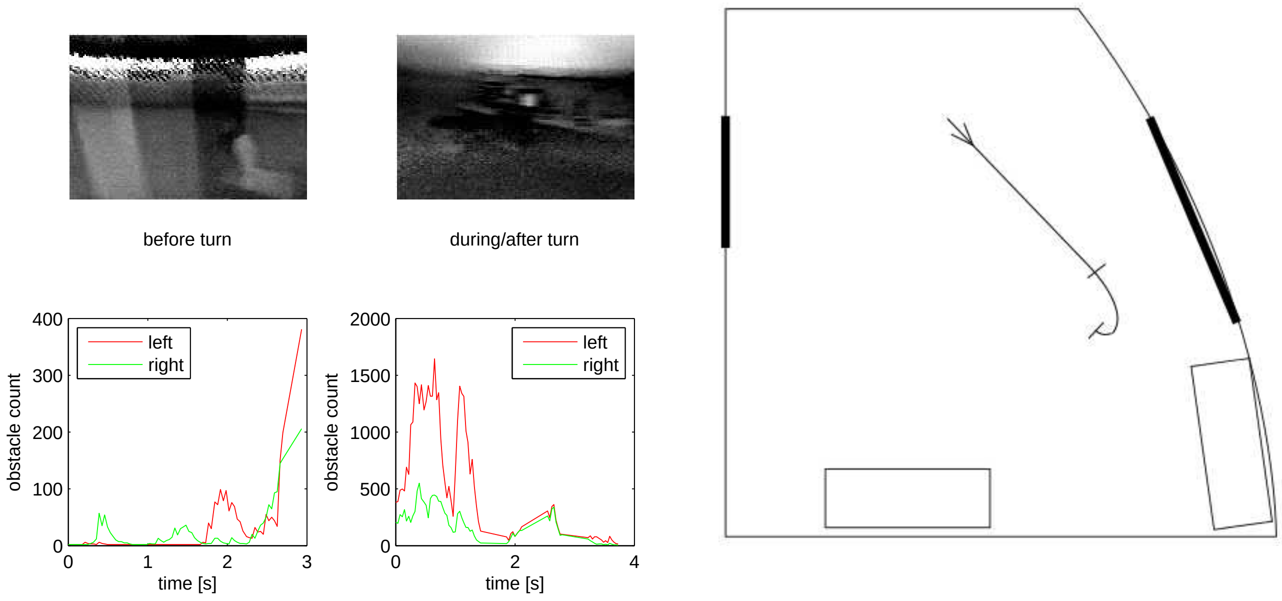


Figure 7-4: Obstacle detection and avoidance situation (4). **Top left** onboard cameras image at turn initiation and end of turn **Right** flight trajectory **Bottom left** obstacle detection signals before and during turn. Yellow regions indicate periods without obstacle detection due to bad camera images

The DelFly approaches the cabinet and at some point a turn is initiated. From the bottom middle plot it can be seen that initially the amount of left detected obstacles increases because of control delay and initial forward speed. The DelFly turns to the right and the obstacle signals decrease. From the right onboard image (top middle image) it can be observed (if one knows the appearance of the test room well) that the DelFly has lost some height and is now flying slightly above table height. As discussed earlier, this is an expected (but unwanted) result due to the bad hover performance of this specific configuration of the DelFly. Furthermore this results in the additional problem that there are now other obstacles detected, such as those that are on the tables. Also partly because of noise (and bad obstacle detection measurements for that reason) the turn is ended at a point much further than one would expect.

7-3 Look-Ahead Yaw control

In this section the results from the third avoidance strategy are presented. Multiple tests were performed to tune the system such that a satisfactory performance was obtained. Satisfactory performance means that the DelFly is able to fly without hitting obstacles for a reasonable amount of time (i.e. more than 60 seconds). Several tests were then conducted to determine the actual performance of the complete system. Figure 7-5 shows the result from the test with the best result that has been obtained. During this test the DelFly flew around for 72.6s without hitting any object. The experimenter did not have to pull the DelFly up to keep it at a constant height. It should be noted that the experiment was ended without reason. The DelFly was still performing autonomous flight and the total successful test time could have been longer than the reported length.

The experiment starts at a point where the DelFly is coming out of a turn that was performed early after start up. This is point *t=0* in Figure 7-5. The track colors indicate the flight phases the controller is in according to figure 6-5. During start of the experiment Phase 4 is active where the DelFly is ending its turn. During the next ten seconds Phase 1 is active and the DelFly should perform straight flight. From the flight track it can be observed that the flight is far from straight. Due to non-zero wind speeds in the test room, caused by ventilation and air conditioning systems, the DelFly swerves significantly. At this time this does not result in avoidance problems.

When the DelFly approaches the upper wall after 10 seconds, Phase 2 and 3 are activated. These phases are combined in the figure. The flight track during the subsequent Phase 4 goes right past the cabinet. However, at this point the DelFly was flying above cabinet height and for that reason it was not considered to be an obstacle. Only the wall needed to be avoided at this point. The flight is then continued in the direction of the lower wall. Note that in all cases where Phase 4 is active, the turn continues. In many cases the new flight direction is around 90 degrees further to the right. This is a result from all system delays, including video reception delay, processing time and radio control delay. After 20 seconds in the experiment Phase 1 is active again. Note that the flight track unexpectedly deflects to the left. As a result a new turn is triggered which directs the DelFly back to the upper wall. This sudden left turn can be explained by yaw/roll instability and is unpredictable. The cabinets in the top then force the DelFly to go back (at around 30 seconds). Note that back at the bottom wall the DelFly preserves a larger distance to the wall compared to other turns. Again the DelFly goes to the top cabinets and back. Just after 50s the lower wall is approached again. In this case an early turn is initiated which is ended too early. As a result Phase 4 is activated while the DelFly still continuous in the direction of the wall. Because the wall is detected again Phase 3 is active again after 684ms. The turn is then continued till the flight direction is now in the direction of the cabinets again. Again the DelFly unexpectedly turns quickly to

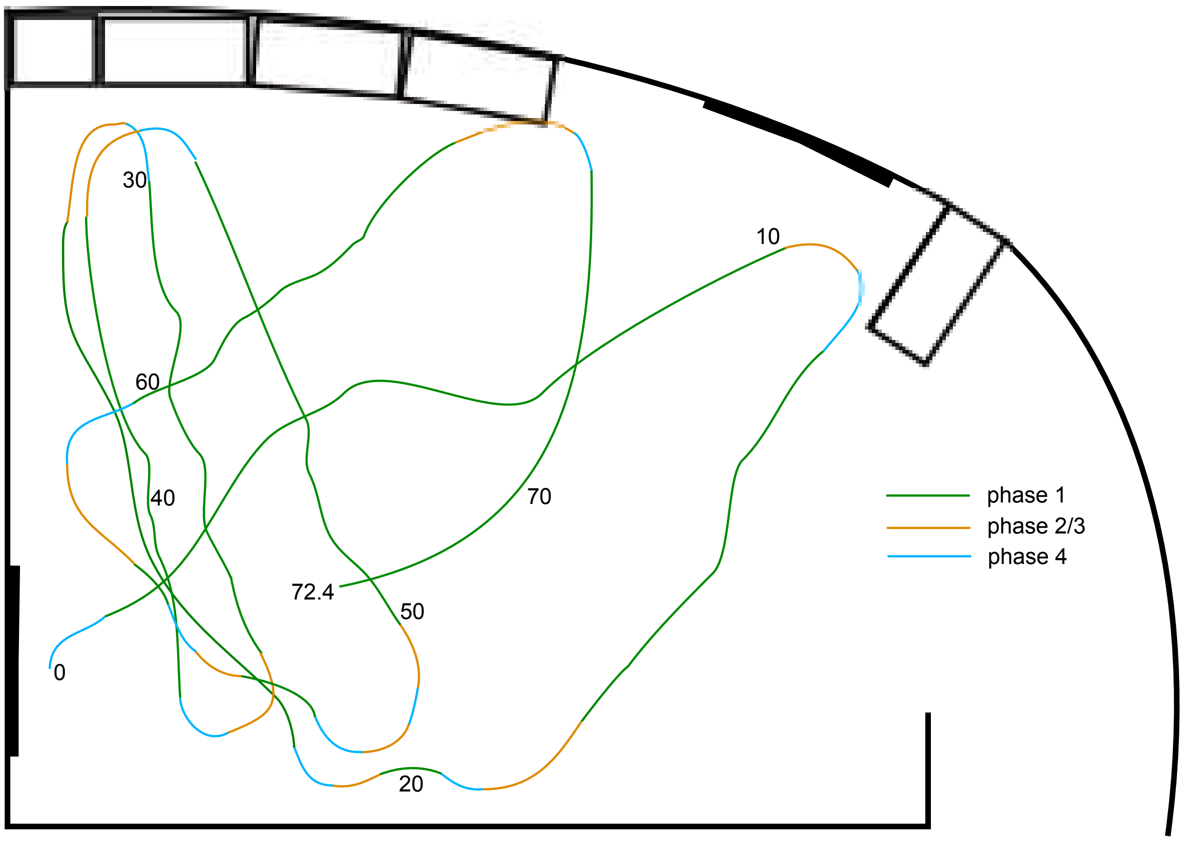


Figure 7-5: Flight track of the DelFly during the experiment. The numbers indicate the flight time, the colors represent the flight phases.

the left and flies in the direction of the doors on the left. These are detected early and a slight turn follows immediately. Another turn is then initiated 1517ms later to avoid the left wall. At *t=60* the DelFly is performing a straight flight in the direction of the top wall at a height above the cabinets. The wall behind the cabinets is then detected and avoided successfully. The flight ends after 72.6s without colliding with any obstacle. Note that during the last part of the flight the DelFly gradually but severely makes a turn to the right. Near obstacles (see bottom of Phase 1 in figure 6-5 were never detected. This means that the DelFly never tried to avoid obstacles that were detected late.

During the flight the short range safety region was never used to avoid undetected close obstacles.

Figure 7-6 shows the measured obstacles during the flight. It can be observed that several times during Phase 1 parts the threshold (red line) was exceeded. The applied filtering technique proved to be successful in these cases in ignoring the false obstacle detections. These incorrect detections and also the incorrect transition to Phase 4 at *t=51* prove that the obstacle detection method is not 100% reliable. The latter case (incorrect transition to Phase 4) is due to the low textured white wall that the DelFly approaches. A remarkable observation to be made is that white walls were in most cases detected because of mismatches. Close to these walls the disparity maps contain a large amount of pixels with unknown disparities.

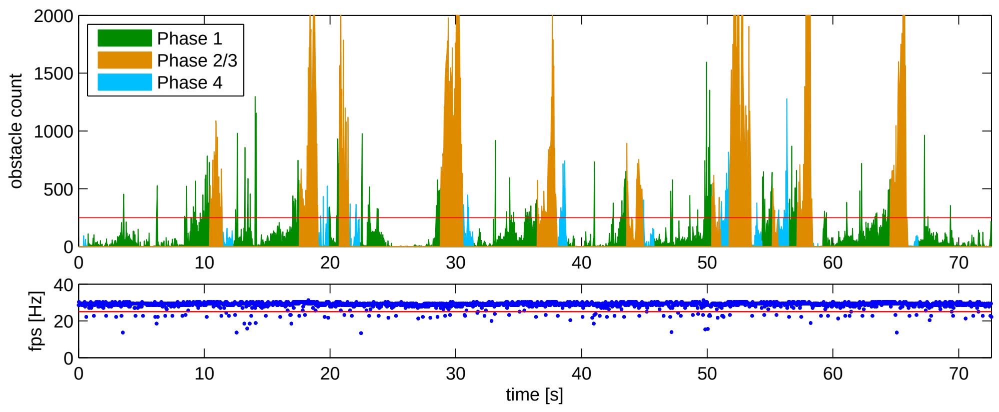


Figure 7-6: Data recordings corresponding to the test from figure 7-5. **Top plot**: amount of detected obstacles. The colors represent the flight phases. **Bottom plot**: frame rates during the flight.

Their disparity value is set to very small and they are therefore never counted as obstacles. Besides that the disparity maps also contain a lot of pixels with very high disparity that do not necessarily belong to correct matches. An example situation is shown in Figure 7-7 (*t=19*). On the left the onboard left camera image is shown, on the right the corresponding disparity map. Note both the amount of dark pixels (unknown disparities) and the bright pixels.

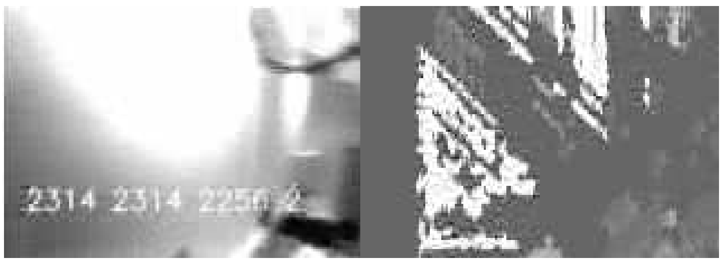


Figure 7-7: Onboard left camera image and corresponding disparity map taken at t=19s and close to the wall. A lot of pixels with unknown disparities can be observed, a well as pixels with a high disparity value.

The processing frame rate of the ground station during the flight is shown in the bottom plot of figure 7-6. A reference value of 25Hz is plotted as a red line. Only 5.6% of the flight time this frame rate could not be achieved. This is caused by the multiple processes that are running at the same time, such as image processing, radio control communication and flight data (image) recording. The mean frame rate during this test was 28.7 Hz.

During the test the DelFly was mainly flying in the left part of the test room. This behavior was even more apparent during two other test flights. During these tests the batteries that feed the onboard cameras were weaker. As a result the received images contain more noise due to the RC link and the cameras are more sensitive to light sources. Figure 7-8 shows how light sources seem to blind the camera. The corresponding disparity map is useless. The test room contains mainly white walls. Only the upper wall containing the cabinets with black doors is free from light reflected from the walls. As a result the cabinets 'attract' the DelFly. It is the only flight direction where (even in case if noisy images) robust obstacle detection can be performed. As a result the DelFly starts to fly mainly the same pattern: from the cabinets to the lower wall and then (via the left wall) back to the upper wall.

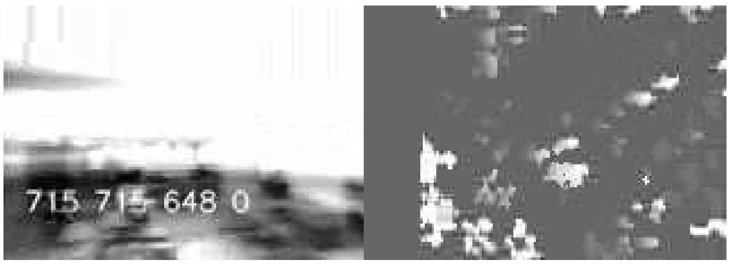


Figure 7-8: Onboard left camera image where the blinding effect is apparent. Note that the disparity map is useless.

Three other test flights were conducted but these were not as successful as the test that was previously discussed. The reasons for the several failures will be discussed next.

The first failure situation is shown in figure 7-9. During this part of the flight the DelFly flies at a height which is only 15cm higher than the cabinets in the middle of the test room. Initially the DelFly flies in the direction of the cabinets along the top wall (the track is not very straight). At some points the cabinets are detected and the DelFly starts to turn away. After a small turn the flight direction is safe, and Phase 4 (blue) is initiated. Due to delays the turn continuous further such that the DelFly flies in the direction of the cabinets again. The turn then takes too long to avoid the cabinets and a collision with objects on top of the cabinets is the result.

A second failure that occurred later on during the same flight is also shown in figure 7-9. In this case the DelFly approaches the doors in the left of the test room. Just before the turn is initiated the DelFly suddenly rotates to the left. A few moments later the cameras are blinded. As a result the controller switches to Phase 4. This error is corrected after 320ms but this does not prevent the DelFly from crashing against the door.

Later on during this test flight the DelFly could not maintain its height and crashed against a ground obstacle. The flight lasted 80.4s until this final collision and besides that only the earlier two discussed errors occurred.

During another test no collisions occurred. However, due to the large size of the cabinets in the middle of test room and the short fishing line between the DelFly and the experimenter,

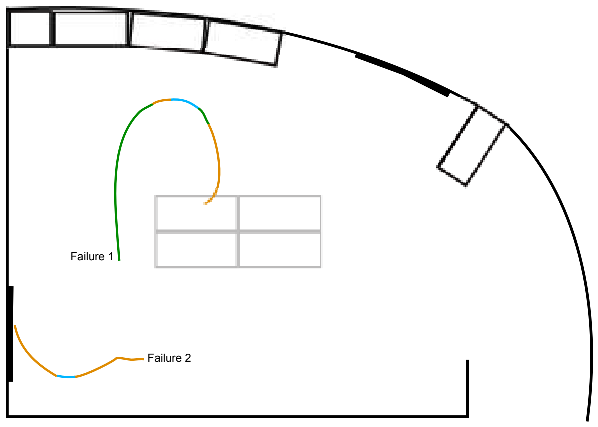


Figure 7-9: Flight tracks of two situations where a collision occurred.

the flight was a little bit disturbed at some points. Furthermore there was a lot of noise in the images (mainly blinding effects) which resulted in several early turns. The flight lasted for 78.1s. During this time the DelFly only flew around in the left part of the test room.

Finally a test is described where again some collisions occurred. Failure 1 in figure 7-10 shows a situation where the DelFly appears to be flying in the direction of the cabinets along the top wall (according to the onboard images), but from the flight track it can be observed that it approaches the wall on the left. The systems detects the wall at a very late instant at which time it is too late to avoid a wing-wall collision. Later on during this flight the situation as described by failure 2 occurs. The DelFly has lost height and it is the flying at around 0.5m. The door in the top of the test room is approached, detected and avoided successfully. The turn does not end after 90 degrees even though it looks like this would be a safe direction. However, on the right of the low cabinets some more obstacles were present. When the DelFly flies parallel to the cabinets Phase 4 (blue) is initiated, but the turn continues a bit further, in the direction of the cabinets in the top of the test room. Phase 1 then becomes active, and Phase 3 is reached shortly after, but too late to avoid the cabinet.

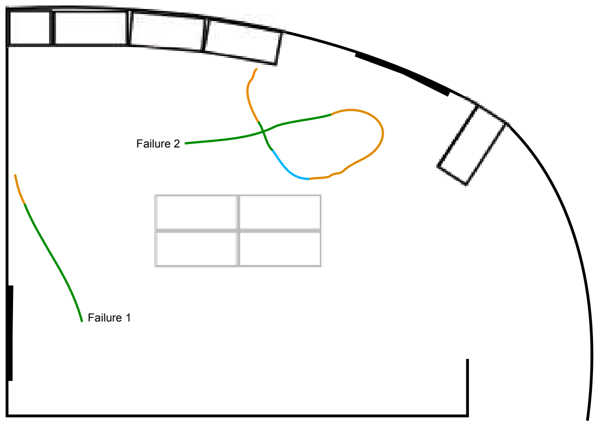


Figure 7-10: Flight tracks of two situations where a collision occurred.

In between these two collisions the DelFly flew around successfully autonomous for 77.3s.

After the second collision the performance was worse due to the low voltage of the batteries which resulted in multiple crashes like Failure 2 and collisions with obstacles while the DelFly descended during turns.

Chapter 8

Conclusions

The main conclusion is that stereo vision is a viable option for providing light-weight Flapping Wing MAVs with obstacle avoidance capabilities.

The stereo camera system used for this task provides stereo images with sufficient quality on which the flapping motion of the FWMAV has a minor influence. In combination with the efficient Semi-Globsl Matching stereo vision algorithm accurate and sufficient obstacle information is obtained at frame rates of over 25Hz.

The performance of the stereo vision system in measuring the distance to a textured wall proved to be appropriate for use on FWMAVs. The static test revealed that it is capable of measuring distances up to 5m with less than 20cm standard deviation. The flight tests show that for the onboard measurements the performance is slightly less. In flight, the system can measure distances of 4m with 30cm standard deviation. The system is not capable of detecting obstacles at a distance smaller than 60cm. These short and long distance characteristics are convenient for performing obstacle avoidance on a vehicle with a small forward speed since it allows looking ahead several meters and because obstacles at a small distance can not be avoided anyway.

From the flight test where the distance to a low-textured wall was measured, a few observations can be made. In the first place it demonstrates that the stereo vision system requires very little texture for detecting objects. Furthermore, the distance measurements are less accurate compared to a textured wall. The smaller the distance to the wall (for distances less than 2m), the higher the inaccuracy. However, apart from this high inaccuracy, the measurements prove to be highly useful since even at these distances it is apparent from the observations that there is a near obstacle. This is an important result since for the task of obstacle avoidance it is more important that obstacles are detected than that their distances are known accurately.

Based on these conclusions, the results from the obstacle avoidance experiments were to a great extent in line with expectations. The first experiment where obstacles were avoided at a small distance showed that the obstacles were successfully detected. However, avoiding these obstacles resulted in collisions with other obstacles that were initially out of the field-of-view. The second experiment, where pitch control was added, demonstrates that if the vehicle would be able to hover, obstacle avoidance could be performed successfully. It allows the vehicle to stop at a short distance and to turn with a small radius such that it will not hit obstacles on the side.

Since hovering is not an option with the DelFly II in the configuration that was used for these experiments (not in the last place because of the weight of the stereo vision system), the result from the last experiment is regarded as the most important. Apart from the observation that the DelFly was able to fly autonomously for 72.6 seconds and without hitting any obstacles, the most remarkable result is the capability of detecting and avoiding white walls. Even though the measurement data reveals that the walls are detected very inaccurately (as was observed earlier during the accuracy test with the low-textured wall), the stereo vision data at least reveals that the wall is at a very short distance. This proved to be sufficient for successful avoidance, though it has to be noted that the DelFly approached the wall very closely.

A remarkable observation is the tendency of the system to repeatedly fly the same pattern in the test room. This can be observed from the presented flight track and this effect was also noticed during other test flights. It should be noted that during a major part of this pattern the DelFly is flying in the direction of the most texture rich region of the test room (where the cabinets are). This indicates that the system tends to fly to unambiguous regions.

From the presented flight tracks important remarks need to be made. First, it should be noted that it is not safe to assume that the DelFly flies straight when the rudder is in trim position. It was demonstrated that this can lead to unavoidable collisions. In the second place it was shown that the turn precision is very poor. After transition to Phase 4 (where the DelFly should end its turn), the turn continued in some cases for another 90 degrees. It was shown that this also inevitably leads to collisions.

The tests have proved that the DelFly is able to autonomously avoid obstacles for longer than a minute. However, due to the total weight of the vehicle including the stereo vision system, the flight endurance is reduced to three to five minutes. Full throttle is required for a major part of the flight which means that obstacle avoidance by active height control is currently not an option.

Chapter 9

Recommendations

Testing environments Tests have only been performed in two different rooms. It is recommended that different types of rooms/areas would be used as test environment to see how well the stereo vision system performs under different conditions. It is useful to test with different types of texture, obstacles with varying sizes, obstacles with open structures, different light conditions, windows, open doors and while flying close to the ground or the ceiling.

Gyroscopes The gyroscopes onboard the autopilot board were not used for the experiments in this project. However, the tests revealed that the DelFly is very sensitive to disturbances due to air flow which has a large effect on its attitude and flight direction. Earlier experiments have shown that the onboard gyros can be used successfully to improve the forward flight performance of the DelFly. Furthermore they were also used to perform controlled turns. This required the autopilot board to be attached using wires in order to reduce the effect of airframe vibrations on the measurements. It is recommended that this method is also tried in combination with the stereo vision system. A barometer has been successfully applied before on other DelFly's for height control and this would also be useful addition to the stereo vision system.

Furthermore, accelerometers might also improve the success of the obstacle avoidance system. During one of the tests it was shown that the DelFly was blown aside against a wall. Because the heading did not change the cameras did not face the wall. Accelerometers might be used to avoid these situations.

DelFly A drawback of using the stereo vision system is the increased weight of the DelFly. This significantly reduces the flight endurance. Good quality batteries enabled flights of three to five minutes. Almost full throttle was required at the start of the flights and it had to increase to full to prevent the DelFly from gradually losing height, already after one minute. There not many options to reduce the total weight of the DelFly. The airframe itself has already been optimized, apart from the new Polyethylene tail. Some weight might be saved there since the surfaces are relatively large compared to the wings, but its weight is around

1 gram so there is not much to win on this part. Flying with separate batteries for the motor and the camera system is unavoidable. What remains is the stereo camera system. A smaller and lighter transmitter has already been used to save weight. Only the cameras and the connecting rods might be further optimized. The weights of the independent parts is not known, but smaller cameras have been used on previous DelFly's. These do not allow to use of the scan line switching feature that proved to be necessary for the stereo vision application. It would therefore be useful to look for other smaller cameras where this method can be applied.

A reduced weight would also enable to use height control for the task of obstacle avoidance. Ideally the weight of the DelFly would be reduced to such an extent that it can hover for a reasonable amount of time. In the current situation robust obstacle avoidance is only possible in relatively large rooms. It was demonstrated that hovering allows flying in much smaller rooms and probably even in very small spaces. Hovering capability would also be a useful extension of the current look-ahead avoidance strategy. Because of the large overshoot when making turns, collisions occur as shown in the results section. These collisions might be avoided if hovering would be an option in such critical situations. Furthermore the DelFly will only enter areas where it is assured that the vehicle can safely turn around. Therefore it will avoid narrow corridors and open doors. This limits the freedom of movement significantly. If hovering would be an option, more risks could be taken to extent the area that can be covered by the DelFly.

Communication link Communication proved to be very critical during the tests performed in this project. The Bluetooth link that was used initially proved to be useless for testing in the larger test room. Because the video transmission is carried out on the same frequency, (2.4 GHz) these two signals interfered each other significantly. This degraded both the link speed and video signal quality. The radio control link proved to be less sensitive to the interfering video signal and enabled robust communication. However, the other way around the video signal suffered from the radio control signal after the camera batteries had been used for a few minutes. It is therefore recommended that these two systems (video and control link) operate at different frequencies.

Another solution that should obviously be aimed for is onboard processing. Signal interference would be no issue anymore. Besides that there would be no delays between camera and processor and between processor and actuator. This way turns could be started and ended more accurately and this would certainly prevent collisions.

Bibliography

Baek, S., & Fearing, R. (2010). Flight forces and altitude regulation of 12 gram i-bird. *IEEE RAS and EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, 454-460.

Baek, S., Garcia Bermudez, F., & Fearing, R. (2011). Flight control for target seeking by 13 gram ornithopter. *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Baha, N., & Larabi, S. (2011). Accurate real-time disparity map computation based on variable support window. *Int. Journal of Artificial Intelligence & Applications (IJAIA)*, *2*, No. 3.

Barnich, O., & Van Droogenbroeck, M. (2011). Vibe: A universal background subtraction algorithm for video sequences. *IEEE Transactions on Image Processing*, *20, Issue: 6*, 1709 - 1724.

Bay, H., Tuytelaars, T., & Van Gool, L. (2008). SURF: Speeded Up Robust Features. *Computer Vision and Image Understanding*, *110*, 346-359.

Benchergui, D. (2009). The year in review: Aircraft design. *Aerospace America*, *47, Number 11*, 17.

Birchfield, S., & Tomasi, C. (1998). A pixel dissimilarity measure that is insensitive to image sampling. *IEEE Trans. Pattern Anal. Machine Intell.*, *20*, 401-406.

Bradsky, G., & Kaehler, A. (2008). *Learning opencv.* OReilly.

Brown, M. Z., Burschka, D., & Hager, G. D. (2003). Advances in Computational Stereo. *IEEE Transaction on Pattern Analysis and Machine Intelligence, VOL.25, NO.8 August*(IEEE 0162-8828).

Calonder, M., Lepetit, V., Strecha, C., & Fua, P. (2010). BRIEF: Binary Robust Independent Elementary Features. *In Computer Vision - ECCV*, *6314*, 778-792.

De Croon, G., De Clerq, K., Ruijsink, R., Remes, B., & De Wagter, C. (2009). Design, aerodynamics, and vision-based control of the delfly. *In the International Journal on Micro Air Vehicles*, *Volume 1, Number 2*, 71-97.

De Croon, G., De Wagter, C., Remes, B., & Ruijsink, R. (2011). Sub-sampling: Real-time vision for micro air vehicles. *Robotics and Autonomous Systems*, *Volume 60, Issue 2*, 167-181.

De Croon, G., De Weerdt, E., Ruijsink, R., Remes, B., & De Wagter, C. (2012). The appearance variation cue for obstacle avoidance. *in IEEE Transactions on Robotics, Volume 28, Issue 2*, 529-534.

De Croon, G., Groen, M., De Wagter, C., Remes, B., Ruijsink, R., & van Oudheusden, B. (2012). Design, aerodynamics, and autonomy of the delfly. *Bioinspiration and Biomimetics*, *Volume 7, Issue 2*.

Duhamel, P.-E., Perez-Arancibia, N., Barrows, G., & Wood, R. (2012). Altitude feedback control of a flapping-wing microrobot using an on-board biologically inspired optical flow sensor. *IEEE International Conference on Robotics and Automation*, 4228-4235.

Forstmann, S., Kanou, Y., Ohya, J., Thuering, S., & Schmitt, A. (2004). Real-Time Stereo by using Dynamic Programming. *Computer Vision and Pattern Recognition Workshop*, 29-29.

Garcia Bermudez, F., & Fearing, R. (2009). *Optical flow on a flapping wing robot* (Tech. Rep.). Department of Electrical Engineering and Computer Sciences University of California.

Gehrig, S. K., & Rabe, C. (2010). Real-Time Semi-Global Matching on the CPU. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 85-92.

Gupta, R. K., & Cho, S.-Y. (2010). Real-time Stereo Matching using Adaptive BinaryWindow. *3DPVT*, *18*.

Hines, L. L., Arabagi, V., & Sitti, M. (2011). Free flight simulations and pitch and roll control experiments of a sub-gram flapping-flight micro aerial vehicle. *IEEE International Conference on Robotics and Automation*, 1-7.

Hirschmüller, H. (2001). Improvements in real-time correlation-based stereo vision. *IEEE Workshop on Stereo and Multi-Baseline Vision*, *IJCV*, 141-148.

Hirschmüller, H. (2005). Accurate and Efficient Stereo Processing by Semi-Global Matching and Mutual Information. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2*, 807 - 814.

Hirschmüller, H. (2008). Stereo processing by semiglobal matching and mutual information. *IEEE Trans. Pattern Anal. Machine Intell.*, *30 no. 2*, 328-341.

Hirschmüller, H., Innocent, P., & Garibaldi, J. (2002). Real-time correlation-based stereo vision with reduced border errors. *International Journal of Computer Vision*, *47 no. 1*, 229-246.

Hsiao, F. Y., Hsu, H. K., Chen, C. L., Yang, L. J., & Shen, J. F. (2012). Using stereo vision to acquire the flight information of flapping-wing mavs. *Journal of Applied Science and Engineering*, *15*, 213-226.

IMEC. (2007). http://www.imec.be/scientificreport/sr2007/html/1384302.html.

Klaus, A., Sormann, M., & Karner, K. (2006). Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. *Proc. ICPR Conference on Pattern Recognition*, *3*, 15-18.

Leutenegger, S., Chli, M., & Siegwart, R. Y. (2011). BRISK: Binary Robust Invariant Scalable Keypoints. *Proceedings of the IEEE International Conference on Computer Vision*.

Lin, S., Hsiao, F., Chen, C., & Shen, J. (2009). Altitude control of flapping-wing mav using vision-based navigation. *IEEE International Conference on Robotics and Automation, 2009. ICRA '09.*, 3644 - 3650.

Lowe, D. (2004). Distinctive image features from scale invariant keypoints. *IJCV*, *60(2)*, 91-110.

Matlab. (2005). http://www.vision.caltech.edu/bouguetj/calib_doc/.

Mattmann, P. (2010). *Fast Feature Extraction for Visual Navigation.* Unpublished master's thesis, Eidgenssische Technische Hochschule Zrich.

Mattoccia, S. (2010). Improving the accuracy of fast dense stereo correspondence algorithms by enforcing local consistency of disparity fields. *3D Data Processing, Visualization and Transmission (3DPVT2010).*

Ohta, Y., & Kanade, T. (1985). Stereo by intra- and inter-scanline search using dynamic programming. *IEEE Trans. Pattern Anal. Machine Intell.*, *7*, 139-154.

Oon-Ee Ng, & Ganapathy, V. (2009). A Novel Modular Framework for Stereo Vision. *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 857-862.

Point Grey Research, I. (2000). Triclops, techincal manual, available at: http://www.ptgrey.com/products/triclopssdk/triclops.pdf.

Rosten, E., & Drummond, T. (2006). Machine learning for high-speed corner detection. *Lecture Notes in Computer Science*, *3951:430*.

Salmen, J., Schlipsing, M., Edelbrunner, J., Hegemann, S., & Lke, S. (2009)). Real-Time Stereo Vision: Making More Out of Dynamic Programming. *LNCS: Computer Analysis of Images and Patterns*, *5702/299*, 10961103.

Scharstein, D., & Szeliski, R. (2002). A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. *International Journal for Computer Vision*, *47*(1-3), 7-42.

Sobel, I., & SCIENCE., S. U. C. D. O. C. (1970). *Camera models and machine perception.* Defense Technical Information Center.

Tardon, L. J., Barbancho, I., & Alberola, C. (2011). Markov random fields in the context of stereo vision. *Advances in Theory and Applications of Stereo Vision.*

Tedrake, R., Jackowski, Z., Cory, R., Roberts, J., & Hoburg, W. (2006). *Learning to fly like a bird* (Tech. Rep.). Massachusetts Institute of Technology Computer Science and Artificial Intelligence Lab.

Tombari, F., Mattoccia, S., & Di Stefano, L. (2008a). Classification and evaluation of cost aggregation methods for stereo correspondence. *IEEE Conference on Computer Vision and Pattern Recognition*, 1-8.

Tombari, F., Mattoccia, S., & Di Stefano, L. (2008b). Near real-time stereo based on effective cost aggregation. *ICPR.*

Van Meerbergen, G., Vergauwen, M., Pollefeys, M., & Van Gool, L. (2002). A Hierarchical Symmetric Stereo Algorithm using Dynamic Programming. *International Journal of Computer Vision*, *47*, 275-285.

Viola, P., & Wells, W. (1997). Alignment by maximization of mutual information. *International Journal of Computer Vision*, *24(2)*, 137-154.

Yang, Q., Wang, L., Yang, R., Wang, S., Liao, M., & Nister, D. (2006). Real-time global stereo matching using hierarchical belief propagation.

Zilly, F., Riechert, C., Eisert, P., & Kauff, P. (2011). Semantic Kernels Binarized A Feature Descriptor for Fast and Robust Matching. *Conference for Visual Media Production*, 39-48.

Zinner, E., Humenberger, M., Ambrosch, K., & Kubinger, W. (2008). An optimized software-based implementation of a census-based stereo matching algorithm. *International Symposium of Visual Computing (ISVC).*