TAMPEREEN TEKNILLINEN YLIOPISTO

Joonas Melin

Surface reconstruction using high resolution stereo vision in a micro air vehicle

Master of Science Thesis

Examiner: Risto Ritala
Examiner and topic approved in the council meeting of the
Faculty of Engineering Sciences
on 9.4.2014

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Automaatiotekniikan koulutusohjelma
Joonas Melin:Surface reconstruction using high resolution stereo vision in a micro air vehicle
Diplomityö, 63 sivua, 0 liitesivua
Huhtikuu 2014
Pääaine: Systeemien analyysi
Tarkastajat: Risto Ritala
Avainsanat: UAV, multirotor, stereo camera, point cloud, mapping, hexacopter

Maarobottien autonominen operointi ja tehokas reitinsuunnittelu vaatii informaatiota ympäröivästä alueesta. Informaatiota on helppoa hankkia robotin lähiympäristöstä mutta suurempien alueiden kartoitus on haastavaa.

Tämän työn tarkoitus oli rakentaa laitteisto, jolla voidaan tuottaa pistepilvi tyyppistä dataa ilmasta käsin. Työssä keskityttiin alueisiin, joissa ei ole selkeitä piirteitä, kuten hiekkakenttiin ja metsiin. Stereonäkö on paljon tutkittu alue mutta monet nykyisistä menetelmistä eivät toimi suuriresoluutioisella datalla tai hyödyntävät vain osan saatavilla olevasta informaatiosta.

Stereokamerat rakennettiin kahdesta digitaalikamerasta. Kamerat kiinnitettiin hexakopterin pohjaan. Kameroiden välinen etäisyys pidettiin pienenä, jotta kamerat saatiin sovitettua kopteriin kyytiin. Tästä johtuen stereonäön tarkkuus kärsii pidemmillä etäisyyksillä.

Työssä kehitetty algoritmi perustuu korrelaatiopohjaiseen hakuun, jolla etsitään kuvista vastaavat alueet. Laitteistolla saatiin hyviä tuloksia alle $5m$ etäisyyksiltä. Kohtalaista dataa onnistuttiin saamaan vielä $30 - 40m$ korkeudesta. Suurimmat virheet aiheutuivat kameroiden epätarkasta laukaisusta.

Työ tarjoaa hyvän pohjan järjestelmän jatkokehitykselle. Suurin osa työssä kohdatuista ongelmista on ratkaistavissa laitteistovalinnoilla.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Automation Science and Engineering
Joonas Melin : Surface reconstruction using high resolution stereo vision in a micro air vehicle
Master of Science Thesis, 63 pages, 0 Appendix pages
April 2014
Major: Systeemien analyysi
Examiner: Risto Ritala
Keywords: UAV, multirotor, stereo camera, point cloud, mapping, hexacopter

The autonomous operation of an unmanned ground vehicle requires map information to be able to plan its route efficiently. It is possible to obtain information from the vicinity of the machine but ground based machinery is not ideal for quickly mapping large areas of unknown terrain.

The aim of this thesis is to make a proof of concept implementation of an aerial system that provides point cloud data to be used for mapping areas. The focus is in areas which do not have many distinctive features, such as dirt fields or forest. Stereo vision is not a new concept but many of the existing methods cannot process high resolution data or are utilizing a small part of the available information.

Stereo cameras are constructed from two consumer grade digital cameras which are mounted below the hexacopter. Cameras are placed close to each other because of the payload and size constrains, this makes the stereo camera more compact but reduces the accuracy of the stereo vision.

The algorithm developed in this thesis uses correlation based block matching to determine the corresponding features from the stereo pair. The algorithm is able to find details from low feature surfaces when the altitude is below $5m$. Useful data can be obtained from altitudes up to $30 - 40m$. The largest errors are caused by the consumer grade cameras having inaccurate triggering.

This thesis serves as a good starting point for developing more efficient hardware and software for small aerial vehicles. Many of the problems encountered in this thesis can be mitigated with different hardware.

PREFACE

This work has been done in the department of Automation Science and Engineering. Major part of the software development and writing was financed with a grant from Tampereen teknillisen yliopiston tukisäätiö, the rest of the copter development was done for O3-SAM and FAMOUS projects.

I would like to thank all the people in the Department Automation Science and Engineering who have been kind to provide me with all the guidance and support. No matter what kind of challenges I faced, there was always someone to point me in a right direction.

First, I would like to thank Risto Ritala for providing all the means for working with this thesis and tirelessly proofreading my thesis, which was no easy task. Equally, I would like to thank Heimo Ihalainen for always providing his expertise and keeping me focused on the right questions.

Secondly, I would like to thank Mikko Lauri and Matti Jukola for always answering patiently to my questions that came at odd times and without warning. Making this thesis would have been much more challenging without their support.

Lastly, I would like to thank Riku Hellevaara for his expertise in localization. Without his enthusiasm many of the field tests would have been much more difficult to perform, especially because some of the tests were performed in less than ideal conditions.

Last edits to this document were done on the 15th of April 2014.

CONTENT

ABREVATIONS

MEMS	MicroElectroMechanical System
MP	MegaPixel
INS	Inertial Navigation System
IMU	Inertial Measurement Unit
LCD	Liquid Crystal Display
GPS	Global Positioning System
RTK-GPS	Real Time Kinematic GPS
MAV	Micro Air Vehicle
RGB	Red Green Blue
PCA	Principal Component Analysis
EKF	Extended Kalman Filter
CHDK	Canon Hack Development Kit
HSV	Hue Saturation Value
USB	Universal Serial Bus
WLAN	Wireless Local Area Network
SGM	Semi-Global Matching
SNR	Signal to Noise Ratio
RC	Radio Control
GPU	Graphics Processing Unit
NED	North East Down

1. INTRODUCTION

The objective of this thesis was to measure terrain with a stereo camera being carried aboard a multirotor aerial vehicle. This is a small part of a larger project in which the ultimate goal is to provide autonomous capabilities to mobile machines. This Chapter provides the background information on methods of getting the terrain information and how this information can be acquired. Justification is given as to why stereo cameras and multirotors were chosen for this thesis.

After the introduction, there will be a short review of stereo vision. Theory is discussed on such a level that when combined with the implementation Chapter and the source material, implementing all the algorithms of this thesis should be possible.

The implementation Chapter will describe the workflow that is required for the actual process of searching the 3D-point cloud of terrain. The algorithms and software has been developed for this thesis, but references are made when similar implementations are found in the existing literature.

1.1 Unmanned Aerial Vehicles (UAV)

Recent advancements on MEMS-sensor technology have brought down the prizes for small and lightweight gyro and accelerometer sensors. These have fueled the development of small and lightweight micro air vehicles (MAV). Term MAV refers here mainly to multirotor helicopters, but MAV in general are a much larger group of vehicle types that are generally lightweight and able to move in the air. MAV is a subgroup of UAV vehicles. MAVs differ from UAV mainly by their low weight and relatively small payload capabilities.

In this thesis, we are using MAV to transport the needed payload to the desired position. This work does not study the autonomous capabilities of the MAV's. MAV used in this work is a part of a larger project where the goal is to provide supporting information for route planning and decision making on ground based machines. This thesis also presents how the MAV is constructed and what kind of software is running onboard. The goal is not to provide a full documentation but to give a general idea of the system as a whole.

1.1.1 Multirotors

Multirotors are a relatively new type of MAV that uses multiple motors with fixed pitch propellers. The definition of a multirotor is very loose as anything that is flying and has more than two propellers can be considered multirotor. Traditionally, multirotors are distinguished by their rigidly mounted motors with fixed pitch, but several exceptions to both of these criteria exist.

Advantages on using multirotors are their mobility, fault tolerance and versatile structure. Their mobility is good as they can control their speed freely in all directions. However, they are not suited for high speed applications since their maximum speed is limited to roughly $80km/h$. Multirotor's body structure is highly advantageous for mounting measurement and computation hardware, because of the space available in the middle of the body. Large and oddly shaped payload can be fitted easily because all the moving parts are on the edges of the multirotor. The only moving parts on the multirotor are the motors and propellers and if the copter has 4 or more motors with enough lift, the multirotor can survive losing one motor. In case of a quadcopter with four motors, losing one motor means that the copter needs to abandon yaw control, but landing safely is still possible even without the yaw control. In case of a hexacopter with six motors, losing one motor will mean that there is less thrust, but it can still be flown as normal.

Weak points on multirotors are their high reliance on software stabilization. This means that software bugs can cause serious issues in flight, compared with winged planes that will keep on gliding if there is a momentary glitch in the software.

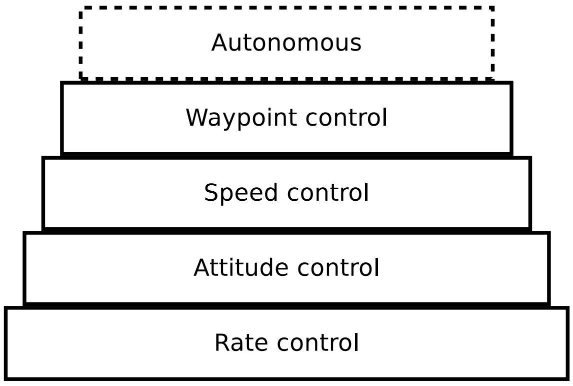


Figure 1.1: Hierarchy of control provided to a pilot by the multirotor. The Lowest level of the control is the rate control which feeds inputs straight to the MAV's rotational rate PID controller. Higher control levels rely on the lower level controllers. A fully autonomous control is not supported on available autopilots up to this date as it would require complex path finding algorithms on top of knowledge of the surrounding environment.

Multirotors are usually flown by a pilot operating a remote controller on the ground. In principle multirotors are operated by controlling their attitude and thrust. Levels of control are illustrated on Figure 1.1. The way that the pilot interacts with the multirotor is defined by the onboard software. The Lowest level of the control is rate control, which will bypass the attitude controllers and the pilot inputs are interpreted as inputs to the attitude rate controller. Giving input directly to the rate controllers enable the pilot to freely control the attitude of the copter and the software will not limit the attitude to a safe range.

One step up from rate control is attitude control. This means that the pilot inputs are interpreted as the desired attitude. Attitude inputs are input to higher level controllers which will in turn feed lower level rate controllers. Usually, attitude control modes restrict the angles to $+-45deg$ as this ensures that the multirotor will produce enough lift to maintain its altitude. When speeds are kept low, the attitude inputs translate to acceleration inputs. Tilting the multirotor will induce acceleration in a direction of the tilt.

The next control level is speed control, which will rely on a GPS onboard to estimate speed and position. The pilot inputs are interpreted as speed commands. Software will control the attitude to reach these speed goals. Flight controller software supports mixing the speed, rate and attitude controls, e.g. having attitude control on roll and pitch axis, speed control on height and rate control on the yaw axis.

The Highest level of the control is the waypoint control, where the multirotor gets only waypoints from the pilot, and the multirotor will autonomously fly through these waypoints. It is common for waypoints to have time and speed goals. Even though this type of flight mode does not require constant input from the pilot, it is not yet completely autonomous. A fully autonomous vehicle would need to have the capability of building a map of its environment and planning its route through this constructed map, avoiding obstacles that are detected. There is a lot of research to make this feasible, but these systems usually work under very limited circumstances, for example relying on a ready-made map in a limited area, or systems that only work indoors on a fairly controlled environment.

Different control levels are based on the lower level controllers. For example, when the pilot controls speed, the speed controller gets the set point from the pilot input, which the speed controller converts to an attitude goal for the attitude controller. The attitude controller then translates the given attitude goal as an input to the rate controller.

All the control levels are accessible from software with serial communication. The platform chosen for this thesis is open source so it is possible to modify the controllers to suit specific tasks.

1.1.2 Other types of MAV's

MAV's are not restricted to being only multirotors. Winged models are capable of flying planned routes and performing self-stabilizing flight. Winged models allow substantially longer flight times compared with multirotors and they can move with a high speed. These qualities are due to the use of a static wing surface which will generate lift by maintaining adequate airspeed as opposed to multirotor which requires constant power to their propellers to maintain sufficient airflow over the lift generating surface. Wings also provide reliability as it is usually possible to land the plane safely even without the motors. The downside of static wings is that the winged model needs to be moving constantly to maintain sufficient lift, as opposed to the multirotor which can hover stationary in the air. Winged models also need large wingspans to carry payload. Even though they are more efficient at carrying the payloads, they are not as versatile.

Performance of the traditional helicopters with main and tail rotor falls between multirotors and airplanes as they are more efficient than multirotors but not nearly as efficient as airplanes. Helicopters can also hover and have roughly the same speed limitations as multirotors. The better efficiency of the helicopter is due to a single large propeller with a variable pitch for generating lift. Variable pitch enables the helicopter to use technique called autorotation; in the case of a motor failure, the main rotor pitch angle can be set at negative value to convert height into rotation speed for the rotor. This slows down the fall. When the helicopter is close to the ground, the pitch can be set at a positive value and momentarily the helicopter has enough lift to slow down the fall, minimizing damages to the helicopter. The downside of using a helicopter is its complexity, vibrations and large rotating mass. Complexity is caused by the mechanism that is used in controlling the main rotor blade angles to direct the thrust to only part of the main rotor, which requires many push rods and bearings. Failures in these parts are usually critical for the operation. Vibrations are an issue in some applications, such as photography: as these vibrations are on low frequency they are harder to filter out mechanically. Large rotating masses on the main rotor make helicopters hazardous to their environment.

1.2 Stereo vision with high resolution

Stereo vision is based on using two images that are taken in known geometric relation to each other and having an overlapping image area. It is not mandatory to use two different cameras as long as the transformation of the camera between the images is known. The difference in the optics of the cameras is compensated by camera calibration. However, it is common to have two identical cameras with a fixed distance between them in order to make calibration and calculation easier. Stereo vision algorithm extracts distance information from the image pair.

Two images are required for the distance information. Distance calculation from a single image is possible with previous knowledge about the scene, for example some target of known size. Distance to a feature can be calculated from the images when it occurs in both images. The difference of the feature placement in two images is called disparity, which has the units of pixels. Disparity is then used to calculate the distance when the geometry of the cameras is known.

The original idea for this thesis was to test if high resolution can be useful in stereo vision. One of the main problems in traditional stereo vision is the difficulty in finding matching features. Large and high contrast features are needed. When using higher resolution, the feature size can be smaller and contrasts also smaller since window sizes can be larger in pixels. This leads to more information about the corresponding areas. Differences between 12 MP and 0.3 MP resolution images are illustrated in Figure 1.2 where the same image has been downsampled to 0.3 MP resolution for comparison. It is good to note that the full resolution image is real data imaged from onboard multirotor with no stabilization, so there is also notable blurring from multirotor movement.

Higher resolution also provides a wider variety of methods for image analysis, such as texture analysis or marker based localization. In short, the higher resolution will open many more development paths compared with low resolution stereo cameras. The downside is the drastically increased computation times, since the amount of data is considerably larger when there are more pixels on the images. Computation time can be considered as $O(n)$ where n is the amount of pixels.

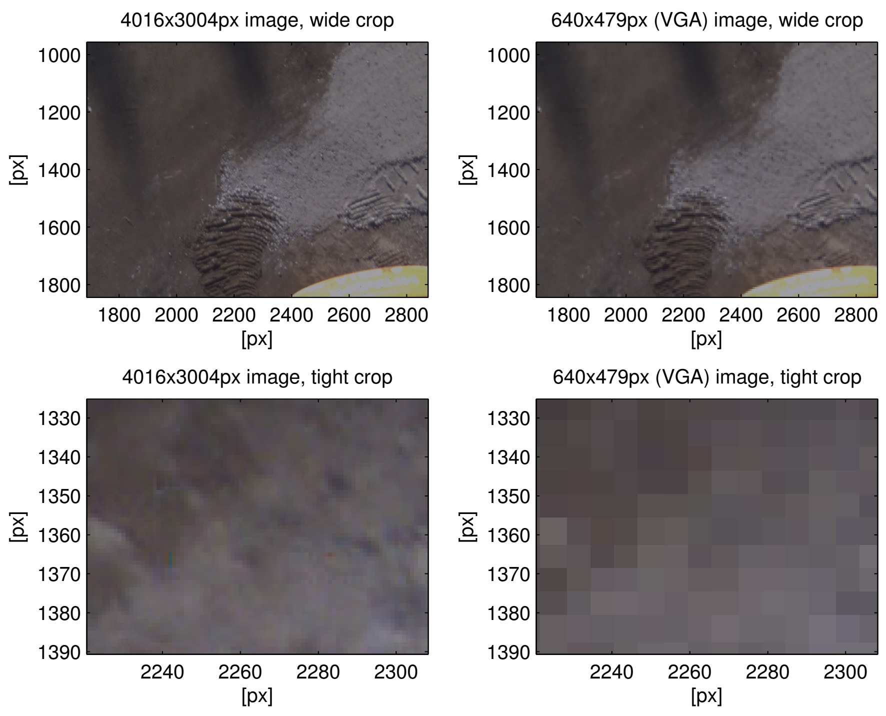

4016x3004px image, wide crop
640x479px (VGA) image, wide crop
4016x3004px image, tight crop
640x479px (VGA) image, tight crop
[px]
1000
1200
1400
1600
1800
1800 2000 2200 2400 2600 2800
1330
1340
1350
1360
1370
1380
1390
2240 2260 2280 2300

Figure 1.2: Top,left: original image. Top, right: original image downsampled to VGA resolution. Bottom, left: a detail of the original image. Bottom, right: a detail of the downsampled image. Detail images show that there are still distinct features found on the full 12 MP image where VGA resolution does not provide much of any information. 12 MP image shown is affected by the movement of the MAV and low quality lenses.

1.3 Other depth perception methods

This Chapter discusses briefly other methods of depth perception and reasons why they are not used in this thesis. This is not an exhaustive list but ought to provide the general idea of the working principle of each sensor type and their advantages and disadvantages.

1.3.1 Laser scanners

Laser scanners or laser range finders work with the time of flight principle in which a laser pulse is transmitted and received as a narrow beam, and the phase difference between the transmitted and reflected beam is then converted to distance. To scan several points, rotating mirrors and multiple lasers can be utilized to get data similar to a depth image.

Laser scanners are generally highly accurate with distance uncertainties of roughly few millimeters even at long distances. Laser-scanners are not used in this thesis because they are expensive and usually too heavy to be carried on the MAV used in this thesis.

1.3.2 Time of flight

Time of flight (TOF) cameras are another possibility for depth perception. TOF cameras have similar operation principle as the laser range finders as they detect the phase differences between the transmitted and received light, but instead of a single measurement point the TOF cameras utilize 2D array, so the resulting depth image is captured at once, without rotating or moving elements.

At the time of this writing there has been a push to get cheaper and more lightweight units. However, these have a very limited range and are mainly designed for indoor use.

1.3.3 Structured light

Structured light sensors e.g Microsoft Kinect, are a type of depth perception sensor. Structured light sensors work by projecting some known pattern on to the surface and then using camera to find the known points from the pattern. Surface shape will result on disparity between the pattern detected by the camera and the known original pattern. Distance is calculated similarly to stereo vision. These sensors can be considered as a kind of assisted stereo camera where the sensor itself will create texture on the surface.

Structured light sensors have very attractive properties: lightweight and a low price, but the downside is that their optimal measurement range is limited, e.g. in case of Kinect to roughly four meters, and operation on sunlight is not possible. On the other hand, these sensors can detect distance on surfaces that have no texture, such as uniformly colored walls or floors.

1.4 Why is distance information useful?

Distance information is essential for representing real world structures in three dimensions. Once the sensor pose is known in global orientation, the distance information can be translated to the real world coordinates. 3D-point information is called point cloud data, which can then be used to represent map information of the area. This is useful for visualization purposes and essential for autonomous robot operations, since avoiding obstacles and planning routes is not possible without information about the surroundings.

Point cloud data imaged from air is even more essential as MAV can cover large areas quickly with little concern for obstacles compared with rovers on the ground which only see a small portion of their surroundings and need to react to new information constantly. In case of obstructing terrain such as walls or hills, higher vantage point aids in mapping the area efficiently. Autonomous operation for land based vehicle without information about the surroundings is very challenging, as route is planned with high uncertainty.

1.5 Objectives of this thesis

The objective of this thesis is to study the possibilities of low cost and high resolution stereo vision onboard an MAV. This objective is part of a larger project where the goal is to get autonomous capabilities for ground based vehicles. This concept is illustrated on Figure 1.3 where the MAV is providing information for the ground based vehicle to assist route planning.

The system built in this thesis is for testing the concept with consumer grade cameras in a small and lightweight configuration. This is a proof of concept, not a real time system that can be used for control and route planning.

To achieve the goals in this thesis, the MAV was modified. A large part of this thesis focuses on what is needed to make the stereo vision functional on a MAV.

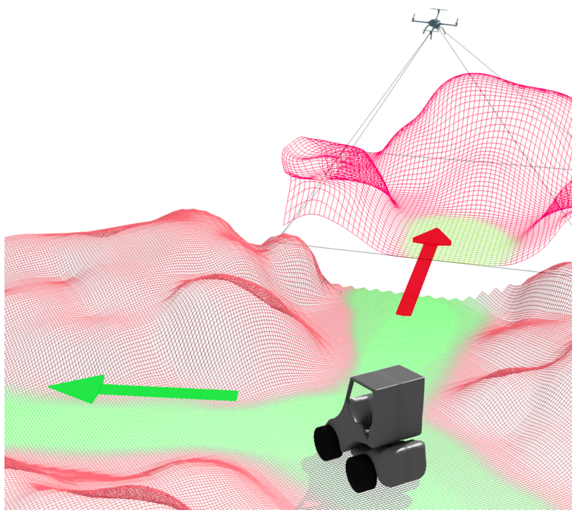


Figure 1.3: Long term objective for the MAV to provide supporting information to assist the operation of larger autonomous machines.

1.6 Structure of this thesis

Chapter 2 of this thesis discusses the theory of stereo vision. Sections 2.1-2.1.1 describes how cameras work and how to model and calibrate them. Section 2.3 illustrates the general idea of stereo vision: The method for calculating 3D-points out of the disparity image is presented here.

Chapter 3 describes the implementation of the algorithms and the construction of the MAV and stereo camera. Chapter 3 presents algorithms in the order that that they are used when extracting the point cloud from the images. It begins by presenting the hardware used in this thesis, which is followed by the description of the software running onboard the MAV. From Section 3.3 onwards the focus is on the software running offline.

The key results of the point clouds and disparity images are presented in Chapter 4. Sections on this Chapter are divided according to the different data sets. Results are further discussed on Chapter 5 which analyses error sources, and possible methods of mitigating their effects.

The system built in this thesis is for testing the concept with consumer grade cameras in a small and lightweight configuration. This is a proof of concept, not a real time system that can be used for control and route planning.

To achieve the goals in this thesis, the MAV was modified. A large part of this thesis focuses on what is needed to make the stereo vision functional on a MAV.

2. IMAGING FOR DEPTH PERCEPTION

The goal of this Chapter is to provide some background to how stereo matching is done and how different parameters are related to each other. The references in this section provide in-depth explanation of the concepts.

First we will begin by explaining how cameras operate and what kind of errors they can cause. This is important as the operation of the camera is the foundation for all the algorithms. We will shortly discuss how and why to calibrate cameras. Lastly we provide the idea of the epipolar geometry which defines how we can map features seen on images sensor to a real world position.

2.1 How cameras work

Camera is an old innovation and the basic principles have remained the same for a long time. This section describes how cameras are modeled for computational methods as simple devices. At the beginning, we discuss how the photons are captured by modern cameras and how we perceive colors with a camera.

This does not serve as a guide to cameras, since that topic would require a book of its own, but this Chapter will merely discuss the most widely adopted conventions to provide enough background information about the basic operation principles of cameras.

2.1.1 Sensors

Basic function of digital image sensor is detecting photons. The amount of photons that have hit the pixel is converted to an analog signal which combined with the signals from all the other pixels is converted to a digital representation of the image.

Image sensors can be thought of as an array of photon detecting cells. This leads us to the concept of resolution, which describes how many pixels there are in the image area. Sensor resolution and optics used determine the minimum size of targets that can be seen with the camera. Physical width of sensors typically range from few millimeters to $35mm$, this results on typical pixel size of less than ten μm wide. The middle point of the sensor defines the principal point of the camera. It is common to use a coordinate system where the principal point of the camera is seen as the point $(0, 0)$.

To detect colors, there needs to be a way of separating photons with different wavelengths. The common method of doing this is by placing filters on top of the pixels, see Figure 2.1. Sensors on most of the consumer cameras contain some configuration of Bayer filter. The important part in Figure 2.1 is that it has two green filters but only one filter for red and blue. The order of the filters is commonly different between sensor models and more exotic sensors may have radically different patterns as well.

The important part to note in Figure 2.1 is that to get the full resolution color image, the missing color values need to be interpolated, as for example red and blue pixels have information only of every other pixel, this means that three out of four pixels for this color need to be interpolated from the close by pixels.

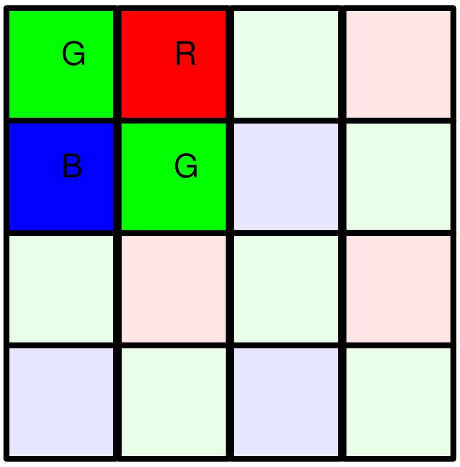


Figure 2.1: The construction of camera sensors Bayer filter. Each square represents the actual pixel located on the sensor.

The downside of using filters in front of pixels is that the photons that do not match the filter wavelength, are discarded. This results in lower efficiency, meaning that longer shutter times need to be used to achieve similar brightness levels compared to monochrome sensors without Bayer filter.

2.1.2 Image noise

Image noise consist of several components that are combined to the resulting image. Resulting intensity $p_{n,m}(t)$ at pixel n, m is composed of components described in the Equation 2.1.Here the number of photons hitting the pixel is described as $v_p(t)$ and the efficiency of the reading is presented as q. The analog noise $v_a(t)$ describes the offset caused by the analog to digital conversion when reading from the sensor. Fixed pattern noise that is a constant error on the pixel brightness is described as $v_{f;n,m}$. Term t denotes the dependency to time, meaning that the results will change as a function of t.

$$p_{n,m}(t) = v(t)_p q + v(t)_a + v_{f;n,m} \tag{2.1}$$

The reason why the variation to the amount of photons hitting the sensor can also be thought as a noise source is that images taken from identical scene will have different amount of photons hitting the sensor because of the discrete nature of the light. This phenomenon can be modeled as Poisson distribution [1]. The efficiency q is used to model the fact that not every photon hitting the sensor produces a response.

In this thesis, the camera calibration process addresses the removal of the fixed pattern noise from the images as this is the simplest noise type to detect and compensate. Fixed pattern noise can be detected by observing multiple images and calculating changes in the pixel values. As fixed pattern noise is produced by pixels that are not producing the same results as other pixels, they can be detected either by different signal levels or lack of variance on the values between images in case the pixel does not react to input at all.

2.1.3 Modeling of cameras

Cameras are usually complex systems consisting of several lens elements and aperture. The simplified model of a real camera system of this kind is described in Figure 2.2. However, this model can usually be simplified as a simple model of a camera that consists of a pinhole and a sensor. This kind of simplification is described in Figure 2.3.

The simplified pinhole camera is a geometrical model of the camera where the optical center is at distance equal to focal length from the sensor plane [2, p. 35]. This model is often enough to describe any calibrated camera.

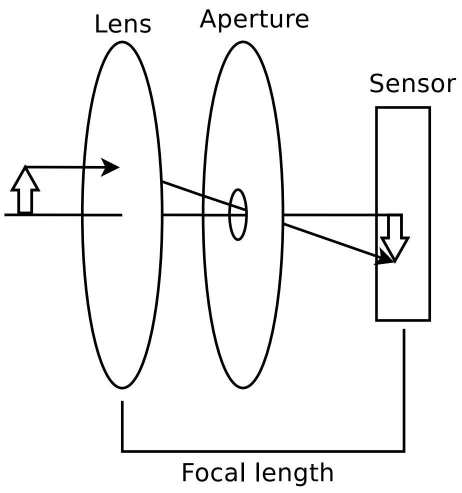


Figure 2.2: Illustrating the elements present in a real camera. The lens will distort the incoming rays. This is done to compensate for the large aperture. Aperture is used to control the amount of light coming in and the width of the area in focus.

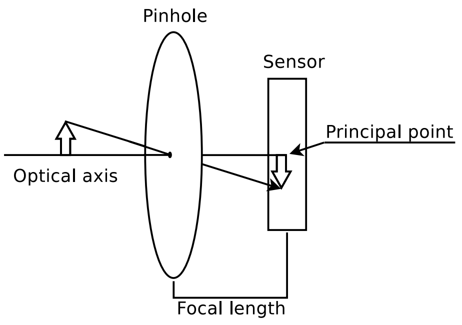


Figure 2.3: Simplified camera model where the light is assumed to arrive through one focal point, which is infinitely small. This causes the objects in front of the pinhole to be drawn upside down on the sensor plane. The optical axis goes through the focal point to the middle of the sensor plane.

2.2 Coordinate systems and camera calibration

Camera calibration is essential for many algorithms. The main goal in camera calibration is to ensure that features on the image are not distorted. If an image of a square box for example is taken, the calibration ensures that all the sides are planar and at correct angle relative to each other. Camera calibration corrects typical errors such as barrel distortion on images. Systematic errors caused by optics or sensor can be compensated with camera calibration but random errors like analog sensor noise cannot be corrected with calibration.

Image and camera coordinates are presented in Figure 2.4. Image coordinates are defined in pixels and have their origin at the principal point. Camera coordinates have their origin at the focal point, with the Z-axis pointing away from the sensor plane. Camera coordinates are usually defined in meters. The direction of the axes depends on the author but the convention presented in Figure 2.4 is the convention used in this thesis.

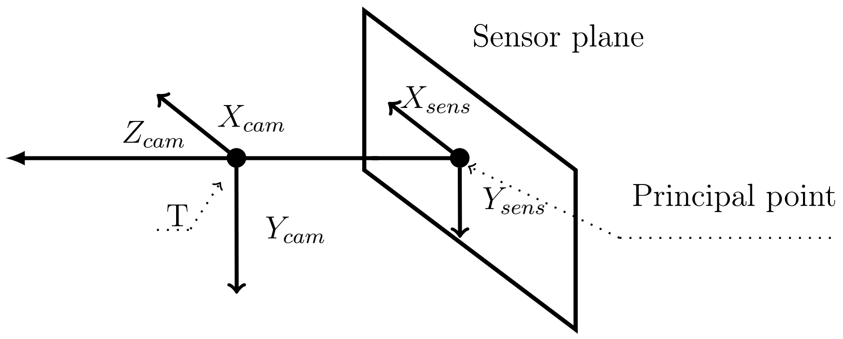


Figure 2.4: Presenting the image and camera coordinates. Coordinates X_{sens} and Y_{sens} represent the image coordinates while X_{cam} and Y_{cam} define the camera coordinates. Z-axis points away from the sensor plane on both cases. Origin for the camera coordinates is located at the focal point which is at the location of the pinhole in the simple camera model.

Matrix operations for coordinates, are best described in the homogenous coordinate system. The idea is that when we need to apply a $3x3$ rotation matrix to the 2D points, $\mathbf{X} = \begin{bmatrix} x & y \end{bmatrix}^T$ we can add an additional coordinate. This way we end up with the form $\mathbf{X} = \begin{bmatrix} x & y & 1 \end{bmatrix}^T$ or in more general form $\mathbf{X} = \begin{bmatrix} wx & wy & w \end{bmatrix}^T$. Geometrically this means that a point in Cartesian coordinates is presented as a line in homogenous coordinates. In practice, this makes operations like affine transformations and projections easier.

To get real world coordinates based on the image coordinates $\mathbf{X_{sens}}$ we need rotation, translation and calibration parameters of the camera. When the simplest pinhole camera model is considered, the parameters can be arranged as a matrix $\mathbf{K}$ seen on 2.2 [3, p. 157]: The matrix $\mathbf{K}$ is called camera calibration matrix, f describes the focal length of the camera. p_x and p_y terms are the principal points of the sensor, which is assumed to be the origin of the sensor coordinate frame [3, p. 155]. Generally the principal point is located roughly at the center of the sensor. The term s is a skew parameter which is usually zero for most of the normal cameras [3, p. 157]

$$\mathbf{K} = \begin{bmatrix} f & s & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

3D camera frame coordinates $\mathbf{X_{cam}}$ can be coverted to 2D sensor coordinates $\mathbf{X_{sens}}$ using equation 2.3 [3, p. 155]. The vector $\mathbf{X_{cam}}$ has the form $\begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T$. This is used to describe coordinates on the real world coordinates where the Z-axis points away from the sensor plane, see Figure 2.4. Consequently, the vector $\mathbf{X_{sens}}$

has the form $\begin{bmatrix} X & Y & 1 \end{bmatrix}^T$ which defines coordinates on the sensor plane. Equation 2.3 relies on the fact that there is no rotation or translation.

$$\mathbf{X_{sens}} = \mathbf{K} \begin{bmatrix} \mathbf{I} | 0 \end{bmatrix} \mathbf{X_{cam}} \tag{2.3}$$

The next step is to take into account that the camera has rotation $\mathbf{R}$ and translation $\mathbf{T}$. This can be written in a form of an Equation 2.4 where $\mathbf{R}$ is a $3X3$ rotation matrix and $\mathbf{T}$ is the camera center [3, p. 156].

$$\mathbf{X_{cam}} = \begin{bmatrix} \mathbf{R} & -\mathbf{RT} \\ 0 & 1 \end{bmatrix} \mathbf{X_{glob}} \tag{2.4}$$

When we combine the Equation 2.4 with the Equation 2.3 we get the form 2.5 which maps the global world coordinates to the sensor plane [3, p. 155]. This takes into account all the parameters of the simple pinhole camera model in addition to camera rotation and translation.

$$\mathbf{X_{sens}} = \mathbf{KR} \begin{bmatrix} \mathbf{I} | -\mathbf{T} \end{bmatrix} \mathbf{X_{glob}} \tag{2.5}$$

To take into account the distortions caused by the lens, the radial distortion is often the largest source of error. Radial distortion makes straight lines appear curved on the image. Radial distortion can be corrected by fitting a polynomial to the curvature of the image, this way the original image can be corrected by interpolating the new values for the changed coordinates. Often it is better to do calculations on the original image and correct for the results with the polynomial, this way the whole image does not need to go through the correction.

2.3 Epipolar geometry

The Figure 2.5 describes a case of unrectified stereo image with two image planes which both see the 3D-point $\mathbf{U}$. The point $\mathbf{U}$ is projected onto image planes as a 2D-point $\mathbf{x}$ and $\mathbf{x}'$. The epipolar points $\mathbf{e}$ and $\mathbf{e}'$ are points where the baseline intersects the image plane. The plane which is formed from the camera centers and the point $\mathbf{U}$ is called the epipolar plane. Camera centers $\mathbf{T}$ and $\mathbf{T}'$ are physically located at the focal point of the lens. For convenience, they are drawn here behind the image planes so that the image is not upside down.

Epipolar geometry can be thought as a way to visualize geometry between image planes. This enables us to find the 3D-location of a 2D-point visible on both image planes assuming that the epipolar geometry between the image planes is known and there is a non zero baseline (line from one camera center $\mathbf{T}$ to the other cameras center $\mathbf{T}'$) between image planes. The problem with a single camera is that points

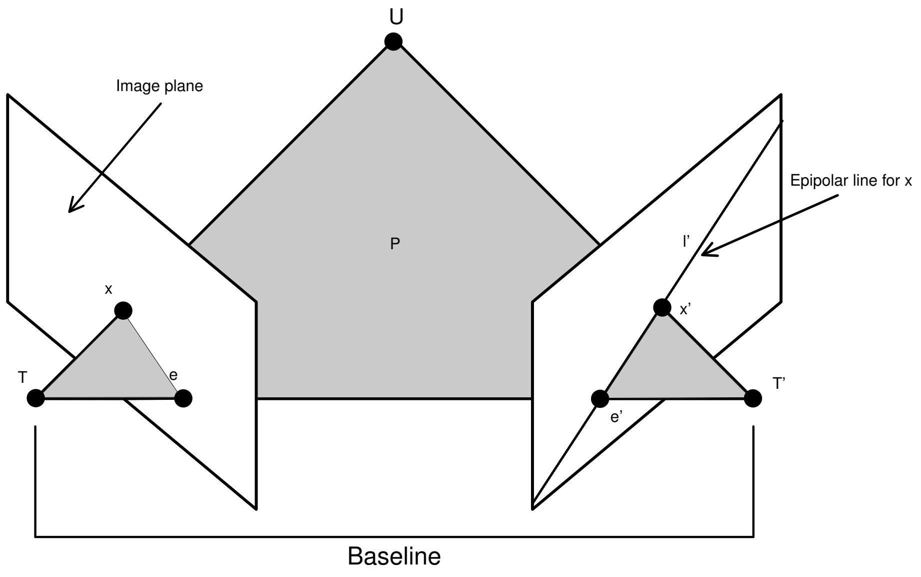


Figure 2.5: Epipolar geometry between two image planes that both are registering point $\mathbf{U}$ at 2D points $\mathbf{x}$ and $\mathbf{x}^{'}$. The epipolar plane P is a plane that is defined by camera centers $\mathbf{T}$, $\mathbf{T}^{'}$ and 3D-point $\mathbf{U}$. Epipolar points $\mathbf{e}$ and $\mathbf{e}^{'}$ are created when the P intersects with the image planes.

on the image plane correspond to lines in the 3D world. When we add another image taken with a non zero baseline, we can constrain the depth axis and two lines in the 3D coordinates become a point.

The line from $\mathbf{T}$ to $\mathbf{U}$ is projected on the other image as a line $l^{'}$ which is called the epipolar line [3, p. 240]. This is very convenient for searching the corresponding point from the other stereo image, as the search can be limited to the epipolar line $l^{'}$ and there is no need to search the whole image. The search algorithms can be simplified by rectifying both images so that the epipolar lines are horizontal on both images, this way the search is limited to a horizontal direction.

The epipolar geometry defines the fundamental matrix $\mathbf{F}$ seen on the Equations 2.6 and 2.7. The geometric meaning of a fundamental matrix is that it describes point $\mathbf{x}$ on the image plane as a line $l^{'}$ on the other image plane [3, p. 242]. This makes it possible to search for the corresponding point on the line $l^{'}$ for the point $\mathbf{x}$, the point $\mathbf{x}^{'}$ is one possible match for the point $\mathbf{x}$. This relation is described on equation 2.6 [3, p. 243].

$$\mathbf{Fx} = l^{'} \tag{2.6}$$

$$\mathbf{F} = [\mathbf{e}^{'}] \times \mathbf{H}_{\pi} \tag{2.7}$$

The fundamental matrix $\mathbf{F}$ is composed from terms described on equation 2.7 where the $\mathbf{H}_{\pi}$ represents the homography that describes the translation from points

x to point $\mathbf{x}^{'}$. To describe the translation with a homography matrix, we need to define an arbitrary plane π, see Figure 2.6. The plane π passes through the point **U** but does not pass through either of the camera centers. The existence of plane π is not required for the **F** to exist. The plane π is used there only as a mean to define the relation between two points.[3, p. 242-243]

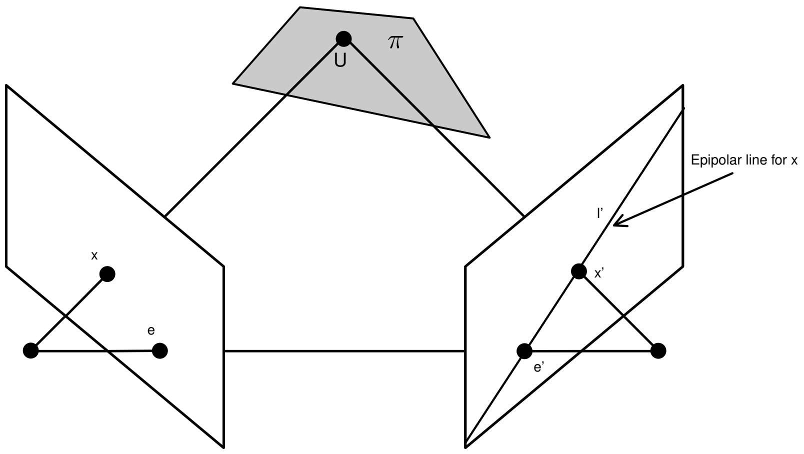


Figure 2.6: The fundamental matrix **F** describes the transformation from point **x** to the line $l^{'}$. This is presented on equations 2.6 and 2.7. The plane π is used to describe the homography that maps the point **x** to the point $\mathbf{x}^{'}$.

$$d_t = \sum \sqrt{(\mathbf{T} - \mathbf{T}')^2} \tag{2.8}$$

$$z = \frac{d_t f}{d} \tag{2.9}$$

In a special case where the camera planes are parallel, the depth coordinate z can be solved with equations presented in 2.9 [2, p. 175]. The baseline d_t presented on equation 2.8 is the distance between the camera centers, when this information is combined with the focal length f and disparity d we can calculate the metric depth. The disparity is defined as the distance of the matched features in pixels along the epipolar line. This equation is the basis for converting the pixel units of the disparity to the real world units.

3. IMPLEMENTATION

This Chapter illustrates all the details of the implementation. This includes the description of the hardware used for capturing the data. First the construction of the stereo camera rig is outlined and other hardware of the MAV is presented. The construction of the Inertial Measurement Unit, IMU, is described. The system and measurement models used for estimating the attitude with the Kalman filter are shortly described.

After the hardware, the software and algorithms for calculating the surface estimate are described. The entire analysis is described in Figure 3.1. The phases on the activity diagram point loosely to sections on this chapter.

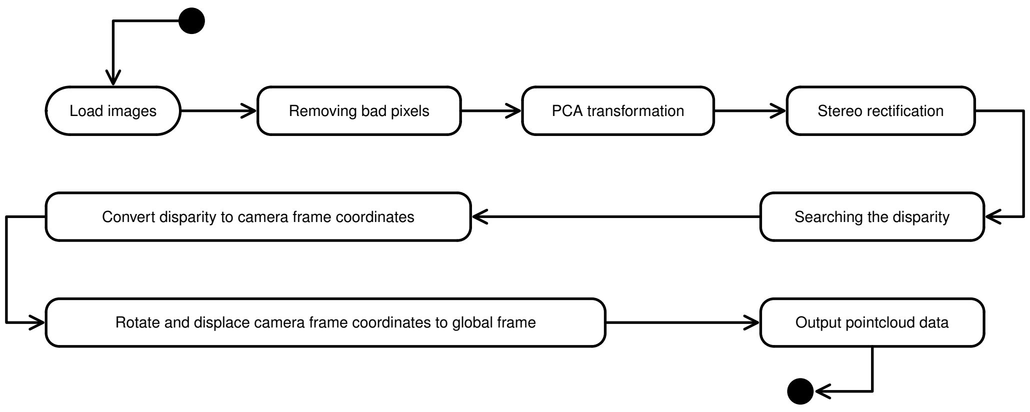


Figure 3.1: This describes the process of point-cloud calculation from loading the images to outputting the point-cloud. Steps on this diagram are discussed more in the following sections.

Most of the software and hardware discussed in this chapter have been written specifically for this purpose. Most of the development was done on MATLAB. The original idea was to use a ready-made toolbox for stereo calculation, but it quickly became obvious that readily available solutions, such as Peter Corke's robotics toolbox [4], could not process our data due to memory constraints. Preexisting tools have been used when possible. Camera calibration was done with algorithms from Bouquet's toolbox for MATLAB [5]. Software running inside the MAV has been developed with Python on top of the Robot Operating System (ROS).

3.1 Hardware

This section describes the hardware used in this thesis.The Kalman filter responsible for fusing the IMU data is described on this section. The implementation of the actual Kalman filter is not explained, only the models for standard extended Kalman filter equations.

Most of the hardware is consumer electronics modified to meet the requirements of this thesis. The following sections will briefly discuss the performance and limitations of this hardware. Hardware has been built with a low budget. Better performance could be achieved in many parts by using specialized hardware made for the task.

3.1.1 Stereo imaging hardware

This system uses two standard consumer cameras mounted on a sandwiched honeycomb composite plate. This setup is similar to one used in [6] where they are using small baseline-to-depth ratio. Our stereo rig has a smaller baseline of 219 mm,compared with baseline of 700 mm in [6]. All though in [6] depths from $20m$ to $100m$, whereas here depths range from $10m$ to $30m$. However, it can be concluded that our system fits into a category of small baseline-to-depth ratio which Warren et al. do cite as " metric visual odometry for longer range stereo remains an open problem in robotics."[6].

The small baseline-to-depth ratio is due to space and weight constraints, even tough the MAV is relatively large. Larger baseline could be obtained if cameras were placed on the arms, but this was not done as cameras under the propeller downwash are shaken by the air flow and disturb the MAVs efficiency because of the turbulence caused by them. The camera rig construction is shown in Figure 3.2. Cameras are Canon IXUS 220HS running CHDK firmware to provide software triggering, raw capturing and scripting capabilities.

Cameras capture their data to their own memory cards as the cameras don't have capability to transmit full quality images. As raw data was captured, the storage time is long compared with storing compressed JPG files. Images are taken once every 7 seconds, to allow enough time for saving the data and refocusing the image after each shot.

Software triggering quality was tested with the setup described in Figure 3.4 where multiple image pairs were taken from the LCD screen running a timer, where the smallest number represent milliseconds. The differences of ten image pairs were collected and are represented on a histogram at Figure 3.3. It can be seen that the majority of the images are captured within 100 ms of each other, but some images might have the difference of as much as 300 ms. This is a problem because there is

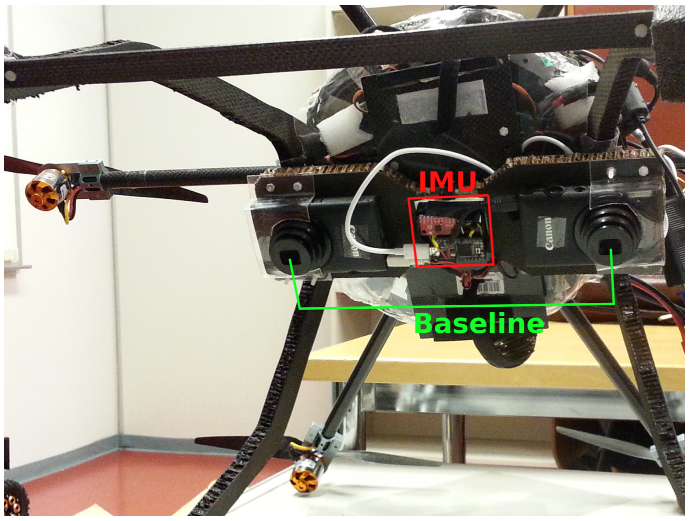


Figure 3.2: Onboard camera rig illustrated. The Baseline described here is 219 mm. Noteworthy is the camera mounting orientation which results in an increased baseline and reduced space usage, but affects the shutter sweep direction. The IMU unit consists of MPU6050 gyro and accelerometer and ARM cortex M4 microcontroller.

no way to know when the images are captured relative to each other.

Cameras triggering at different time create errors as the MAV rarely stays in place when capturing images: as cameras are triggered at different times, their geometry in relation to each other is unknown. When the geometry of the cameras is not known accurately, the calibration procedure becomes invalid. This results in incorrect distance calculated from the disparity and baseline. The effect on the baseline at a modest $0.5\frac{m}{s}$ speed and with the 100 ms difference in triggering will result in up to $50mm$ error at baseline distance. Movement in a direction perpendicular to the baseline axis will result on difficulties in image matching as the calibration will not bring the features to the same horizontal line (this is not considering changes in the attitude that will affect the result). According to the Equation 2.9 and with $0.219m$ baseline, focal length of $2269px$ and disparity of $30px$, we get the resulting distance of $16.6m$. When we add the $50mm$ error caused by linear movement, we will be using base line of $0.219 + 0.05m$, which results on distance of $3.79m$. This gives us an error of $12.8m$ for triggering difference of $100ms$.

The Figure 3.5 presents the relationship between the disparity and distance. Distances under $10m$ will cause significant variation on disparity but at $30m$ distance, the disparity changes significantly slower. This results on higher sensitivity to errors when the objects are further away from the camera, as calculated earlier.

Unknown triggering time will also affect the location and orientation estimates for the images. In the previous example, the $100ms$ difference from the assumed triggering time would result on the same $50mm$ offset error in the location estimate. This error is negligible in comparison with the accuracy of the GPS position estimate.

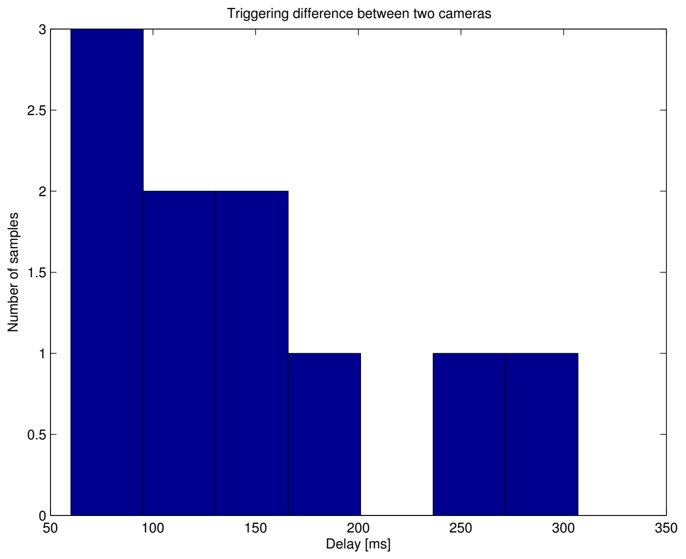


Figure 3.3: Histogram of triggering-time differences measured from 10 image pairs. One source image pair is presented in Figure 3.4.

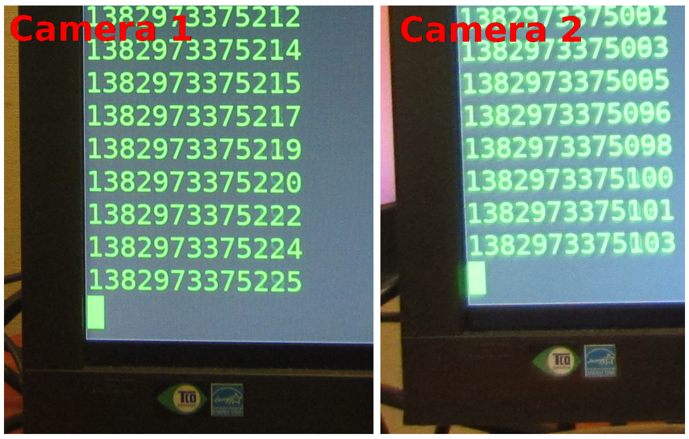


Figure 3.4: Measurement setup for the camera triggering difference. Screen is showing milliseconds. Difference is deduced by deciphering the numbers visible on screen. The results of this test can be seen in Figure 3.3.

Effects caused by the angular rate of the MAV can be larger. One degree error in the attitude estimate in the height of $10m$ will cause $17.5cm$ offset to the location estimate.

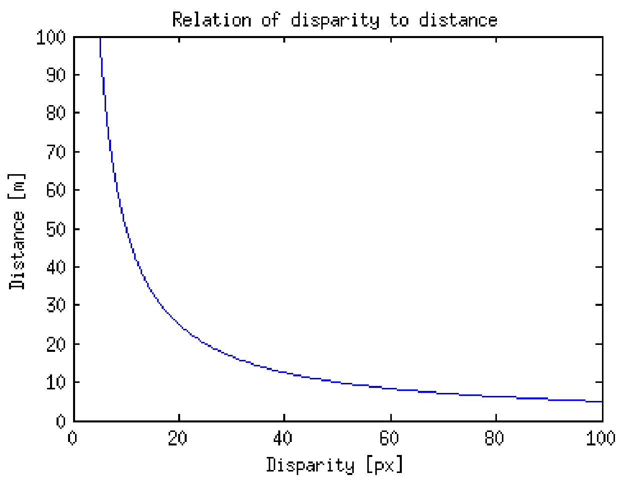


Figure 3.5: The relation between the disparity and distance with the parameters of the stereo rig used in this thesis. Equation for calculating the distance is presented at Equation 2.9. The baseline used was $0.219m$, the focal length was $2269px$.

3.1.2 Copter hardware

The MAV is built mainly from carbon fiber honeycomb plate and carbon fiber tubing. The arms and their connections come from a commercial kit but the rest of the hardware has been cut from carbon fiber plates according to the design made in collaboration with the Department of Materials Science. The MAV is shown in Figure 3.6.

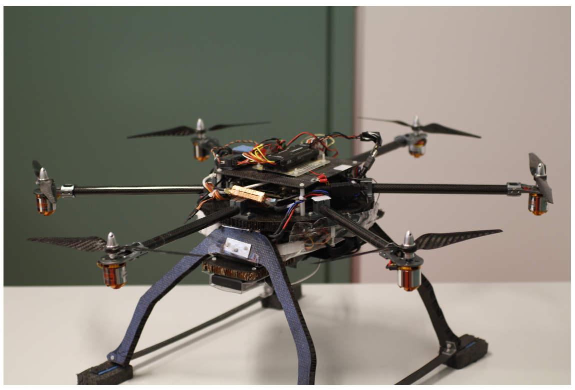


Figure 3.6: The MAV used in this thesis.

The motors have been modified to have their shafts reversed to raise their natural frequency higher to prevent oscillations on the arms as it has been noted that the particular carbon fiber tubing has its resonant frequency at roughly 200 Hz region, which coincides with the frequency produced by the imbalanced propellers. This has been verified by motor testing and Solidworks simulations, which both produced similar results. Mounting the propellers closer to the arms does solve the resonance problem. This mounting scheme is presented in Figure 3.7.

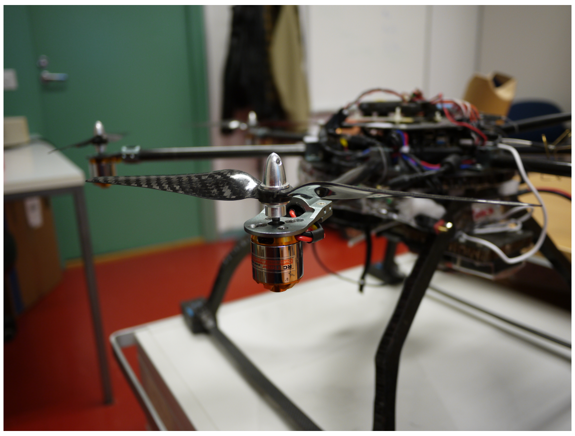


Figure 3.7: Illustrating the reversed motor mount which moves the motor and propeller to the opposing sides of the mounting plate, decreasing the moment of the force caused by imbalance propellers.

The hardware inside the MAV is presented in Figure 3.8. Actual processing hardware consists of the main PC and Arducopter flight controller. Supporting hardware includes the GPS and RC receiver, with the GPS receiver connected to the main PC and RC receiver connected to the flight controller. The detail description of the hardware connections can be seen on the Figure 3.9.

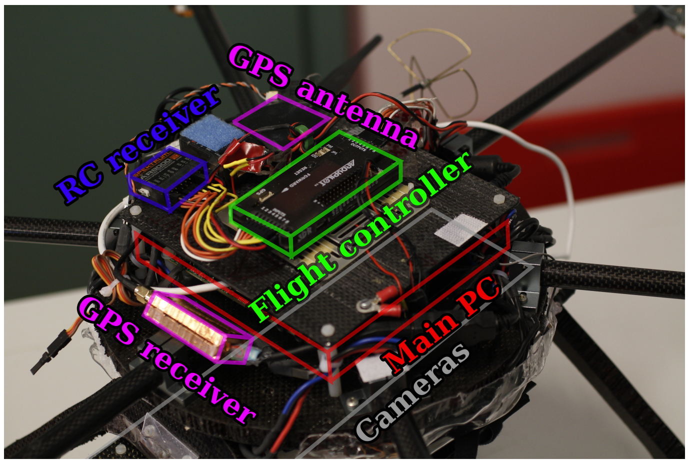


Figure 3.8: Components of the MAV illustrated.

RC receivers operate at 2.4GHz. Their constant transmission prevents reliable WLAN connections. There are plans to move to a 400 MHz receiver to reduce interference with the WLAN. However, so far there has not been a need to transmit large amounts of data to the ground station while flying.

The Arducopter flight controller provides control levels up to waypoint control (see Figure 1.1). Arducopter is running on an ATmega 2560 microcontroller with very modest processing power which results in suboptimal waypoint and speed control algorithms. Actual automated speed or waypoint control requires the main PC to take advantage of the MavLink protocol to input commands to the flight controller. It has been tested to input a better location estimate to the flight controller through serial port emulating a GPS receiver, transmitting NMEA messages. At the time of the tests, the Arducopter's NMEA class had errors, which were suspected to momentarily result on drastically different position estimates being received. This resulted on MAV navigating in a random direction at high speed.

3.1.3 System layout

The system consists of several processors which each handle their specific tasks. The main PC coordinates all the sub processes. The layout and communication hierarchy is presented in Figure 3.9.

Arducopter is basically a modified version of Arduino that is running an ATmega 2560 microcontroller. Arducopter takes care of all the low level flight controls such as attitude control. Interfacing with Arducopter is possible with MavLink protocol. MavLink is used to stream sensor data to the main PC. Arducopter's magnetometer is used for heading information on the IMU. Barometer from the Arducopter hardware is used for relative height information.

The GPS receiver was Yuan10 with RTKlib open source RTK-GPS software. RTKlib connects to the receiver through USB emulated serial port. RTKlib uses the base station receiver data streamed from the laptop on the ground. Base station data was streamed through a wireless serial port most of the time. WLAN was tested on early tests and was found unreliable because the RC controller transmits at the same frequency. RTKlib outputs its solution internally through virtual serial ports which are used as an input to a node on the Robot Operating System (ROS).

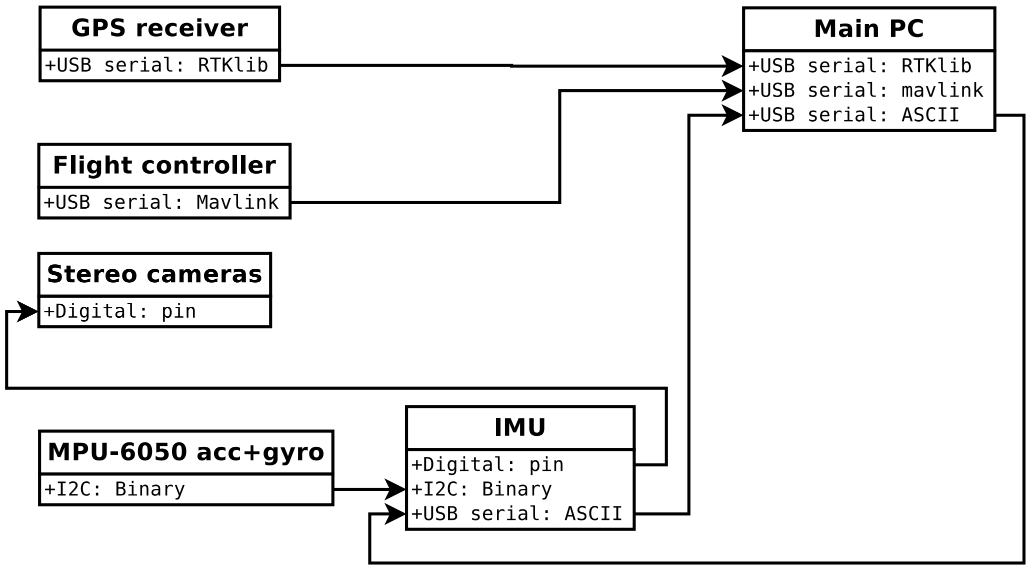


Figure 3.9: Different parts of the on-board hardware illustrated. The internal structure of the ROS running inside the MAV is more accurately described in Figure 3.13. The physical layout of different parts can be seen in Figure 3.8.

3.1.4 Positioning system

Good position information for the MAV is required automating any tasks on the multirotor. Furthermore, the accuracy of matching image data to the environment depends on the accuracy of the positioning information.

For this thesis, only GPS position estimate was considered because the accuracy in the earlier tests conducted in Hellevaara's thesis [7] deemed GPS accurate enough for our purposes. As Hellevaara notes in his thesis, the performance of the system depends on a large number of variables; at best, the accuracy can be 10 cm and at worst it may be in order of tens of meters. Some attempts were made to resolve MAVs position based on marker positions from the image but the recorded GPS positions for the markers were found too inaccurate. The measured positions for the markers can be seen in Figure 3.11. The GPS performance while airborne is not necessarily this bad because the accuracy is usually better while in the air as the reflected GPS signals cause less problems.

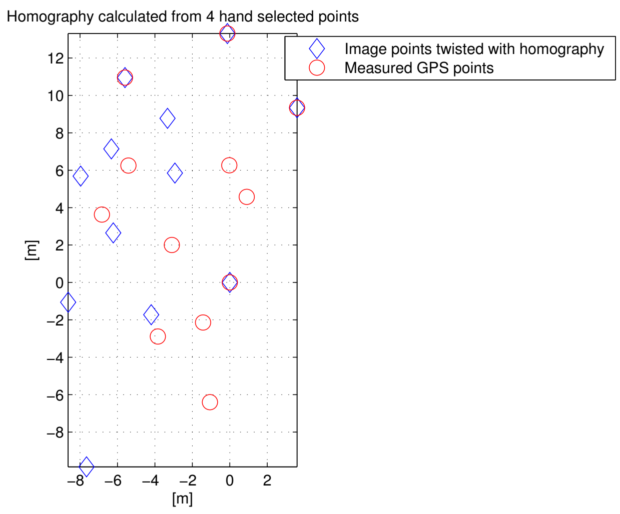


Figure 3.10: Markers found in the image were twisted with an homography to match four of the measured GPS points. This describes the difference of GPS and image based location measurement.

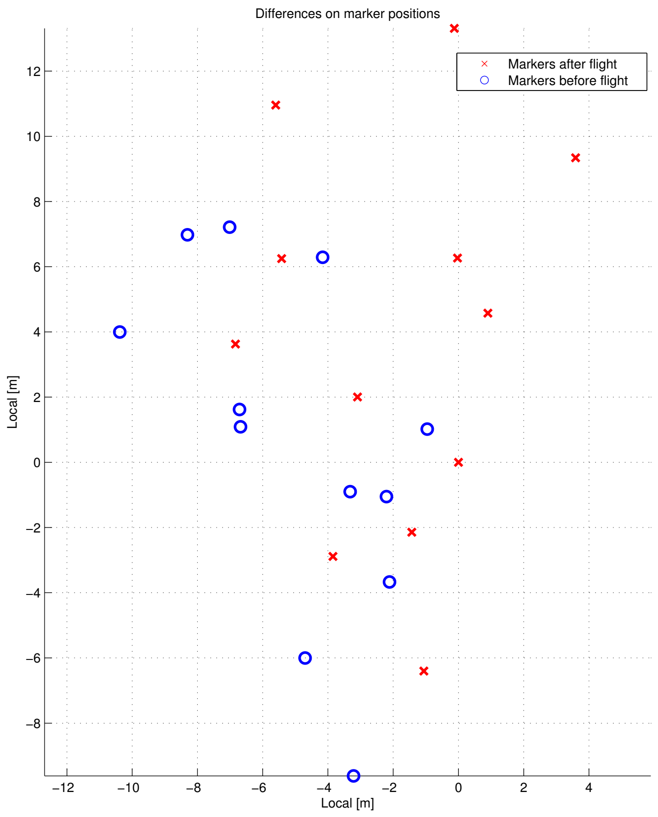


Figure 3.11: Attempted measurements on markers placed on the ground. The positions of the markers were measured before the flight by keeping the MAV on top of the marker and averaging the position for roughly 30 sec. The later position estimate is closer to the truth though significant errors still remain.

3.1.5 Inertial measurement unit

Orientation of the camera is essential for orienting the images on the 3D-space. Without the orientation information, it would not be possible to convert distance information from the local camera coordinate frame to the global world coordinate frame. Matching different image pairs to each other would be more challenging.

Inertial Measurement Unit (IMU) is composed of an accelerometer, a gyroscope and sometimes additional sensors such as a barometer and a compass. The purpose of IMU is to provide sensor data that attitude and acceleration of the unit. The IMU unit used in this thesis is MPU-6050 chip by Invesense which has the accelerometer and gyro on the same chip. MPU-6050 has a simple motion fusion algorithm on the same chip that combines the measurements from accelerometer and gyro sensors to form attitude and global linear acceleration estimates. However preliminary testing revealed that movement will affect the attitude estimate considerably as moving the unit on a flat surface by hand could generate errors as large as 5-10 deg. This error is caused by the assumption that accelerometer measures the angle by measuring the gravitational acceleration, which is assumed to be pointing straight down. This assumption is correct in case where the IMU is stationary. When the IMU unit is moved, the acceleration from movement will be mixed with gravitational acceleration.

Effects of acceleration on the IMU unit can be compensated by using the gyro sensors to estimate the attitude. The caveat of using gyros is that they only measure rotational speed, not the attitude so the speeds need to be integrated to get the measurement of the attitude. This results on errors on the speed measurements being integrated, which means that any bias will accumulate over time, corrupting the results completely. This drifting by integrating errors can be compensated by using direct measurements of the attitude. Accelerometers measure total acceleration, which includes the gravitational acceleration. This means that 3D accelerometers measure the attitude when the IMU is not changing its speed. Accelerometer angle measurement and the angle produced by integrating gyros need to be combined with technique called the Extended Kalman Filter (EKF).

As this thesis is not focused on Kalman Filtering, we will only present models required to the implement EKF. The more in depth explanation of the equations and their proofs can be found in e.g. [8, p. 178].

The direction and orientation of the local and global reference frame can be seen in Figure 3.12. This describes the rotation and movement conventions used in this thesis. The same rules apply to the local and global frame, where X-axis is pointing to the north at the global frame and in the local frame the nose of the multirotor is in line with X-axis. The angles $\alpha\beta$ and γ correspond to multirotors roll, pitch and yaw movements in the local frame. These angles are so called Euler angles which describe the orientation of an object by rotations around three axes. The order of rotation affects the resulting orientation so care must be taken to adopt consistent rotation order convention.

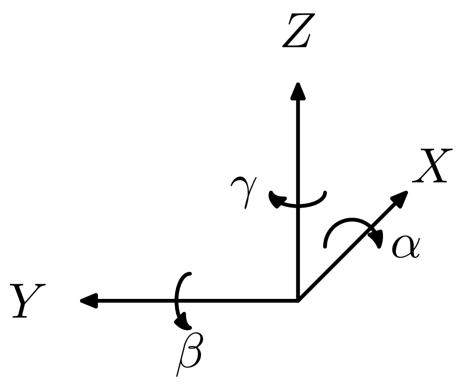


Figure 3.12: The local IMU frame described. Axes point in the direction of positive movement and rotation direction is for positive rotation. The multirotors nose is pointing in the direction of X axis, which points to the north on the global frame.

$$\mathbf{R} = \mathbf{R}_\alpha \mathbf{R}_\beta \mathbf{R}_\gamma \tag{3.1}$$

$$\mathbf{R} = \begin{bmatrix} \cos(\beta)\cos(\gamma) & -\cos(\beta)\sin(\gamma) & \sin(\beta) \\ \cos(\alpha)\sin(\gamma)+\cos(\gamma)\sin(\alpha)\sin(\beta) & \cos(\alpha)\cos(\gamma)-\sin(\alpha)\sin(\beta)\sin(\gamma) & -\cos(\beta)\sin(\alpha) \\ \sin(\alpha)\sin(\gamma)-\cos(\alpha)\cos(\gamma)\sin(\beta) & \cos(\gamma)\sin(\alpha)+\cos(\alpha)\sin(\beta)\sin(\gamma) & \cos(\alpha)\cos(\beta) \end{bmatrix} \tag{3.2}$$

When describing the rotations with a rotation matrix $\mathbf{R}$, the rotation order is defined by Equation 3.1. The resulting rotation matrix is given by Equation 3.2.

The state vector $\mathbf{f_k}$ in EKF is described on Equation 3.3. This consists of global frame Euler angles defined on 3.12 and global frame angle rates described in Figure 3.12. The prediction of the next state is presented on Equation 3.5 where the matrix $\mathbf{P}$ described in Equation 3.4 is used to integrate the angle rate measurements to produce the next state estimate $\mathbf{f_{k+1}}$. The term Δt is the time between the measurements, which is measured by the IMU separately for each measurement. The Equations for the prediction step come from traditional linear Kalman Filter since the prediction step is linear, so linearization steps are not necessary.

$$\mathbf{f_k} = \begin{bmatrix} \alpha & \beta & \gamma & \Delta\alpha & \Delta\beta & \Delta\gamma \end{bmatrix} \tag{3.3}$$

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ \Delta t & 0 & 0 & 1 & 0 & 0 \\ 0 & \Delta t & 0 & 0 & 1 & 0 \\ 0 & 0 & \Delta t & 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

$$\mathbf{f_{k+1}} = \mathbf{f_k} P \tag{3.5}$$

Equation 3.6 describes how the accelerations are represented inside the EKF equation. The measurement model $\mathbf{H_{acc}}$ which is used to convert the global frame acceleration $\mathbf{a}$ to correspond to local frame values measured by the IMU unit is described in Equation 3.7. The acceleration used in this case is $\mathbf{a} = \begin{bmatrix} 0 & 0 & 9.81 \end{bmatrix}$. This assumption works as long as the accelerations caused by the movement are relatively short and have zero mean. This works because accelerometers have high uncertainty given at the covariance model matrices, which means that they are used to correct the slow attitude drifting caused by the integration of gyro error.

$$\mathbf{a} = \begin{bmatrix} a_x & a_y & a_z \end{bmatrix} \tag{3.6}$$

$$\mathbf{H_{acc}} = \mathbf{aR} \tag{3.7}$$

Similar equations are presented for the local frame gyro values $\mathbf{g}$ where Equation 3.8 describes the order of the measured gyro rotations. Gyro measurement model $\mathbf{H_{gyro}}$ is presented in Equation 3.9 which describes how the global frame gyro values in the state are translated to the local frame with the measurements.

$$\mathbf{g} = \begin{bmatrix} \Delta\alpha & \Delta\beta & \Delta\gamma \end{bmatrix} \tag{3.8}$$

$$\mathbf{H_{gyro}} = \mathbf{gR} \tag{3.9}$$

The Jacobian matrices for the update step are calculated by taking partial derivatives from the models presented at Equations 3.7 and 3.9. Jacobian matrices perform linearization of the measurement model at the estimated state.

3.2 Robot Operating System

ROS is running in the main PC and handles of all the higher level functions such as motion fusion, sensor logging and image triggering. ROS is an open source package that provides features, such as multithreading sensor communication and support for generalized inter-process communication. ROS works by creating a framework where different programs can communicate with each other. Therefore, different parts of the program can be programmed with different programming languages such as Python or C++ and they communicate with each other through subscribing and publishing to ROS topics which operate as couriers between ROS nodes. Nodes and topics are the main elements of ROS. Nodes are the actual program code, executed on their own threads, and they can receive data through subscribing to a topic. Nodes can output data by publishing a topic. Topics have their own definition of what data they store, so that they usually have information about the time of the message and possibly the coordinate frame which the data is from.

ROS nodes and topics that are in the MAV are described in Figure 3.13. This configuration consists of data input and processing nodes. Data is inputted to Kalman Filter running at "/motionFusion" -node through "/localAccGyro" and "/mavSensors" -node. "/localAccGyro" -node is reading the data from the IMU unit. IMU data consists of acceleration and gyro values, which are coming through serial port on a raw sensor format. "/localAccGyro" node then converts this data to metric values. At this point, there is only bias subtracted from gyros which will be calibrated at the beginning of the node. Accelerometer bias is compensated at "/motionFusion" node. After metric conversion, "/localAccGyro' node publishes the information on "/copter/ImuLocal" topic. The "/mavSensors" -node feeds compass and barometer altitude data from Arducopter connected through serial port using Mavlink protocol to "/motionFusion" node. Measurements come at roughly $0.5Hz$ to avoid overloading the Microcontroller.

The process of triggering the cameras is described in Figure 3.14. This describes the operation of the script running on the ROS that triggers the image capturing when requirements for reliably capturing images are met. The script will capture the "/copter/stereoGrab" topic which has information regarding the camera orientation and position and the quality of these estimates. This information is written to the disk by the "/imgInfoWrite" node after the "/imgGrabber" node has broadcast the "/copter/stereoGrab" message. The message will be broadcast at the time of the triggering and after a set delay when the images are expected to be taken; this way it is possible to compensate for the triggering delay in post processing if a better estimate for the camera triggering delay is obtained.

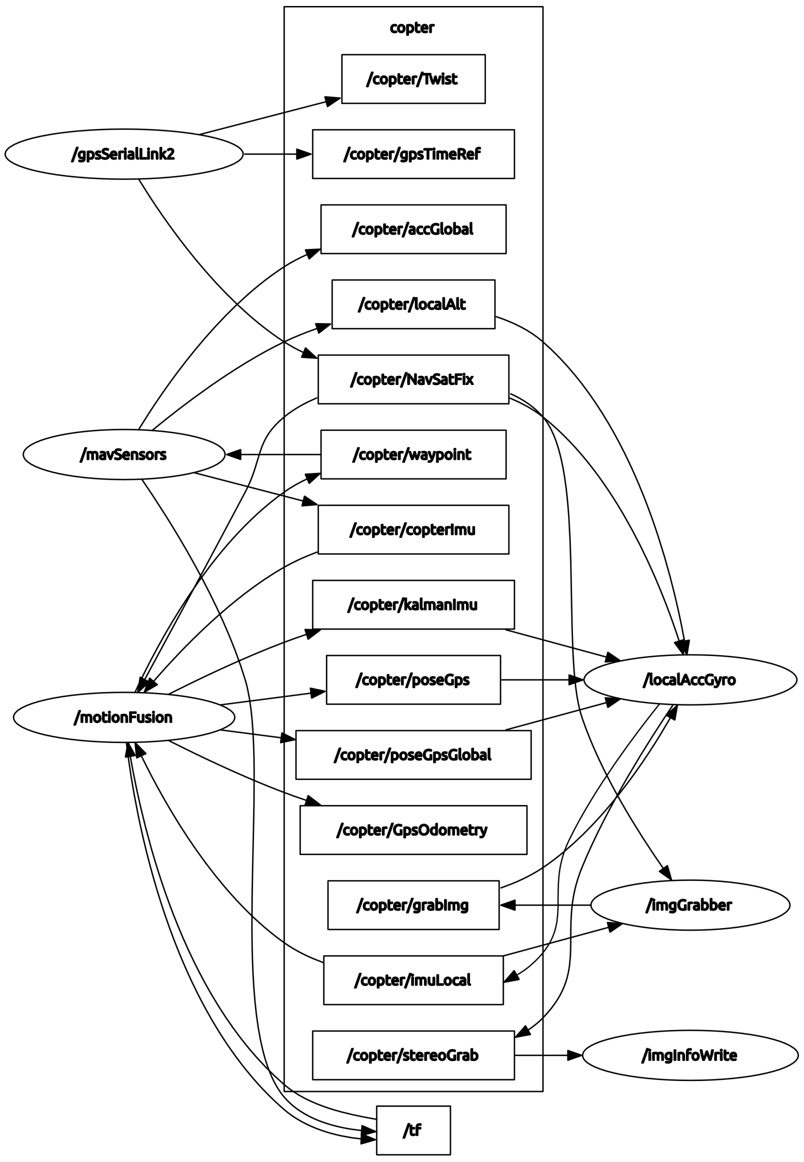


Figure 3.13: Topics published on ROS are listed inside the box labeled copter. Nodes that use and publish topics are listed on the ovals outside. "/motionFusion" -node is running the Kalman Filter which gets the sensor data from IMU unit at "/localAccGyro". "/mavSensors" node is receiving information from the Arducopter hardware and feeding it for the" /motionFusion" through "/tf" node.

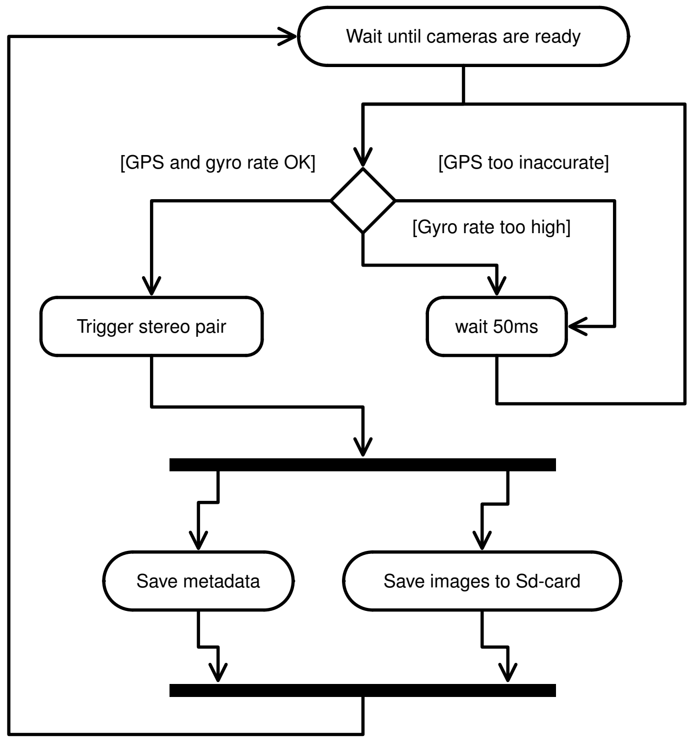


Figure 3.14: Cameras are triggered on the ROS with the algorithm described on this figure. Metadata includes data taken from the ROS topics at the moment of the capture. Most important data stored is the camera orientation and position estimates.

3.3 Camera calibration

This section details the methods used for correcting the errors in camera sensors and optics. Calibration is essential because the algorithm for calculating the disparity assumes that there is no fixed pattern noise on the images and the epipolar lines are assumed to be horizontal.

Camera calibration was done with planar checkerboards. Two calibration methods were tested. The First method used a laser etched aluminum board with 16x16 grid of 16 mm checkers see Figure 3.15. The Second method used a large paper sheet with 90x90 grid of 10 mm checkers, see Figure 3.16. The Larger board was used to get accurate calibration with fewer calibration images as there are more points on one image and they cover a larger area of the sensor. In addition, the calibration images can be taken further away so the cameras will be closer to the distance in actual measurements measurements. This is important because the focusing of the cameras affects their focal length, which is assumed to be known in distance calculations. The actual camera calibration was done with the Bouquet's camera calibration toolbox [5].

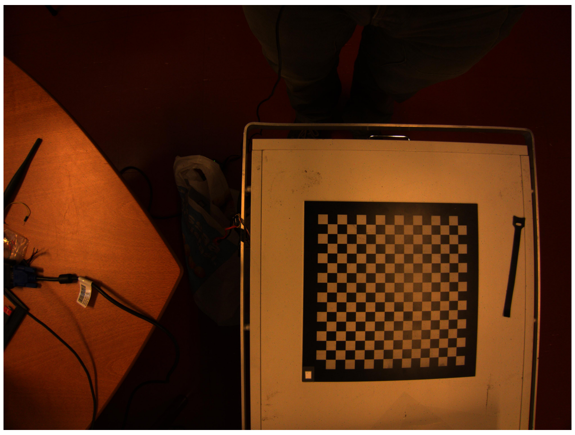


Figure 3.15: The small calibration plate etched on aluminum plate with the laser.

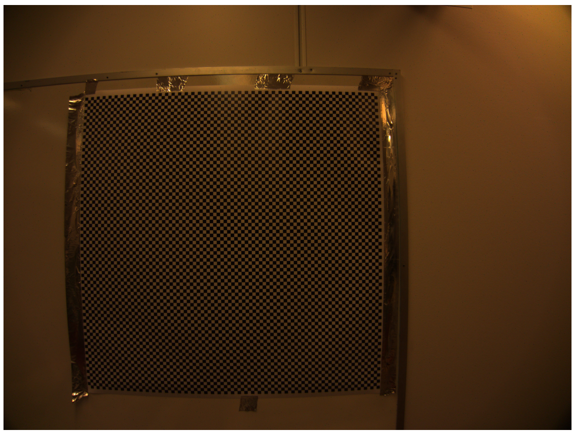


Figure 3.16: The Large calibration grid printed on a paper.

3.3.1 Removing faulty pixels

Faulty pixels are caused by manufacturing defects on the image sensors. Errors appear like salt and pepper noise, but they remain constant between images. This type of noise is called fixed pattern noise. These errors can be seen in Figure 3.17. It is important to note the difference between a fixed pattern noise and random noise as fixed pattern noise can be removed with a minimal loss of information but random noise cannot. Random noise depends on the temperature and lighting conditions and varies from image to image. Fixed pattern noise in the compressed images is corrected by the software provided by the camera manufacturer. However, we are using the raw sensor data from the camera and consequently have to compensate for these errors. Faulty pixels are especially harmful in our case when using correlation for finding corresponding areas on the images and when converting images to monochrome.

The faulty pixels were found by taking several images of a white wall on different orientations. In this way, the images can be stitched together to calculate gains for individual pixels. Reasoning behind this was that stitching several images together results on an average white image, where the differences in brightness are caused by the systematic errors of the sensors and optics.

Figure 3.18 shows that faulty pixels are not the only systematic errors affecting pixel brightness in the camera. Other errors in pixel brightness are caused by the optics. In this thesis, these errors are separated so that faulty pixels are detected by gain for the pixel exceeding the threshold. When faulty pixels are detected, the values on those pixels are replaced by taking average of the surrounding pixels. This operation is done separately for each color channel since the errors are not identical to red, green and blue channels. Remaining errors on pixel gain caused by optics and sensor variation are compensated with the gain matrix.

Gain matrix $\mathbf{G}_c$ calculation is presented in Equation 3.10 where $\mathbf{G}_c$ is the NxM

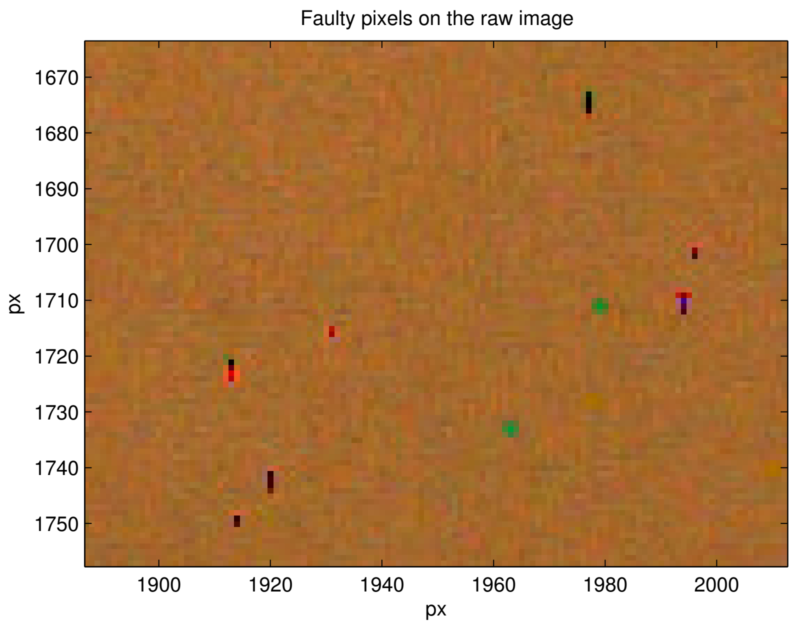


Figure 3.17: Faulty pixels presented on a cropped raw image. Color channels have different errors. Some of the pixels are not completely faulty, resulting in incorrect gain.

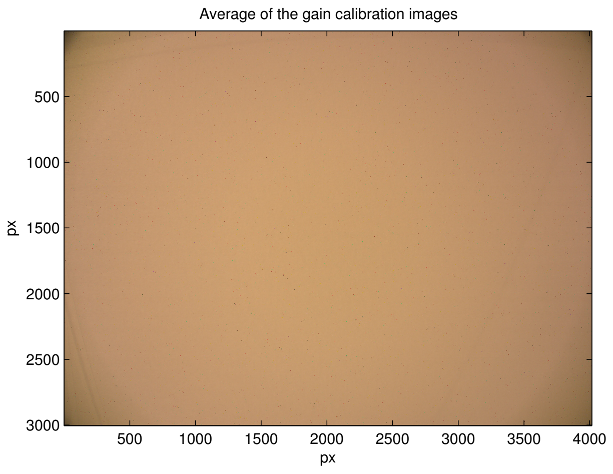


Figure 3.18: Mean calculated from all the calibration images. Systematic errors are clearly visible. A larger image set would reduce the amount of residual features from the original images.

gain matrix for each color and i_M is the median value from the stitched mean image. $\mathbf{I}_m$ is the N times M image matrix where each value is the mean from all the images. Median is used as the mid color value since mean value would be skewed by the faulty pixel values.

$$\mathbf{G}_c = \frac{i_M}{\mathbf{I}_m} \tag{3.10}$$

The resulting gain corrected image $\mathbf{I}_2$ is presented in Equation 3.11 where the original uncorrected image $\mathbf{I}_1$ is multiplied by the gain correction matrix $\mathbf{G}_c$. The notation $\mathbf{I}(i,j)$ indicates the matrix entry i,j.

$$\mathbf{I}_2(i,j) = \mathbf{I}_1(i,j)\mathbf{G}_c(i,j) \tag{3.11}$$

The whole process is presented in Figure 3.19. The mask was generated by thresholding the gain matrix for values that were too large to be caused by variations on pixels. Threshold was set at a gain of 2 for these test images. White-balance is not corrected here since images used are monochromatic.

This way the stitched image has equal brightness on each pixel and there is ideally only random noise left in the images. However, in practice sensor noise is dependent on many factors such as the temperature which makes the gain calibration less efficient. The sensor may also develop more faulty pixels overtime.

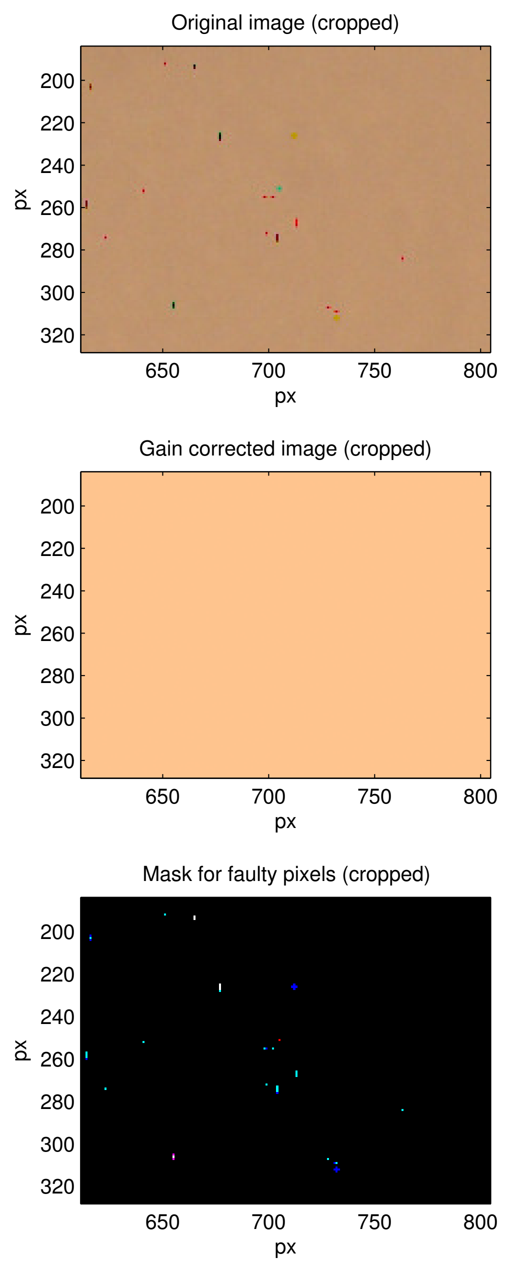


Figure 3.19: Different phases of the gain correction presented. Since the calibration image was used as the input image, the result is a prefect white with no noise. Color errors are caused by white balance not being corrected deliberately, since the calibration images were taken under fluorescent light, the slightly yellow tint is the correct representation of the ambient light.

3.3.2 Stereo rectification

Stereo rectification was done with algorithms provided by the Bouquet's toolbox [5] for MATLAB. Modifications were done to speed up the rectification process. Normally, the coordinates for the stereo rectification are calculated every time, but the script was modified to recalculate coordinates only in case where they are not found stored.

Stereo rectification is not essential for disparity matching, but it was decided to use it for this thesis as strict real time constrains are not set for the algorithms and rectifying the images simplifies the disparity calculations as the epipolar lines can be assumed to be horizontal.

3.4 Converting color image to monochrome

Stereo vision algorithms are usually implemented on monochrome images as this reduces calculation time. Therefore, RGB data, needs to be interpreted as a single channel monochrome image. In many cases, a monochrome camera is used to produce a monochrome image on the source. Traditionally, the RGB image is converted to monochrome by multiplying different channels with constants and combining the channels, this results on visually pleasing results and is roughly equivalent to discarding hue and saturation information from the image. The actual constants used varies between different implementations and sources.

Our approach was windowed PCA over the whole image in order to find optimal multipliers for each color channel on each window so that local contrast would be maximized. Comparison of the traditional monochrome conversion and PCA transformation can be seen in Figures 3.20 and 3.21 the latter of which describes the worst case scenario for the traditional monochrome conversion. The PCA-loadings normalized to RGB channels can be seen in Figure 3.22.

PCA transform generates artifacts to the images, caused by the windowing. Artifacts are prominent close to sudden changes on the brightness or color. This problem has been mitigated by using median filtering on the PCA-loadings. This solution uses the window size of $3x3$ so the resulting coefficients are affected by their neighboring values in all directions. This will smooth the resulting changes on the coefficients and reduces the artifacts. An example of these artifacts can be seen in Figure 3.23 where high contrasts on the indoor scene make the artifacts caused by windowing of the PCA transform visible.

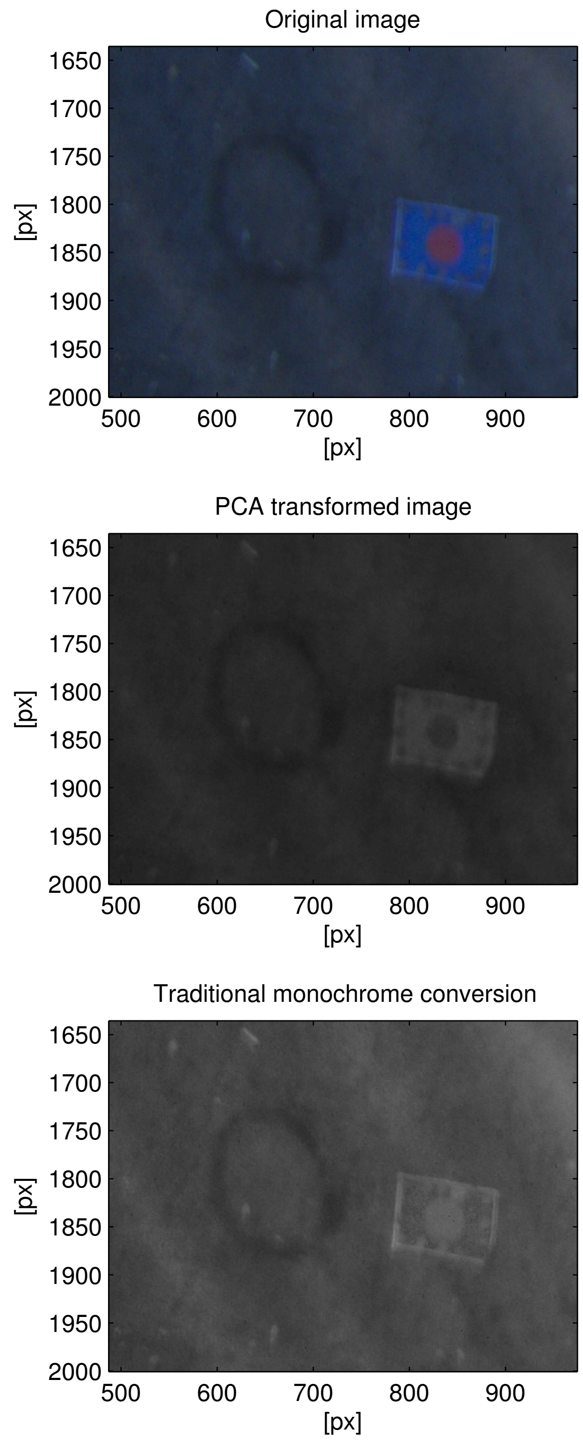


Figure 3.20: PCA transformation and traditional monochrome conversion compared on the marker image. Noteworthy is the different handling of the colored areas and how the red center of the marker is not bleeding at the edges in the PCA transformed image. Some slight artifacts caused by the PCA windowing are present in areas where there are large changes in color.

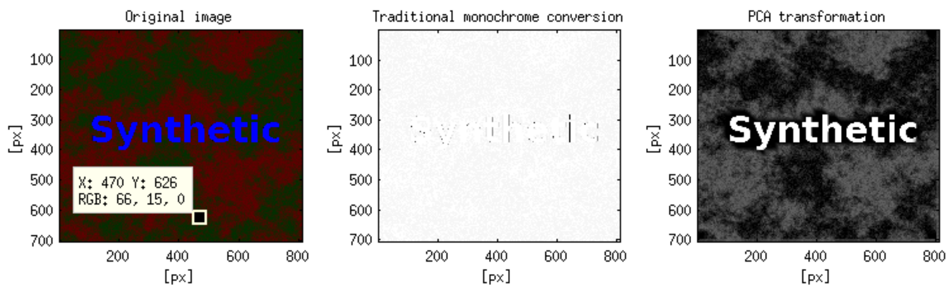


Figure 3.21: It is possible to construct an image where traditional monochrome conversion does not perform well when matching the RGB values to the multipliers used in the conversion. The resulting image is featureless as PCA transformation produces good results, this only proves that traditional monochrome conversion might not perform adequately on special cases.

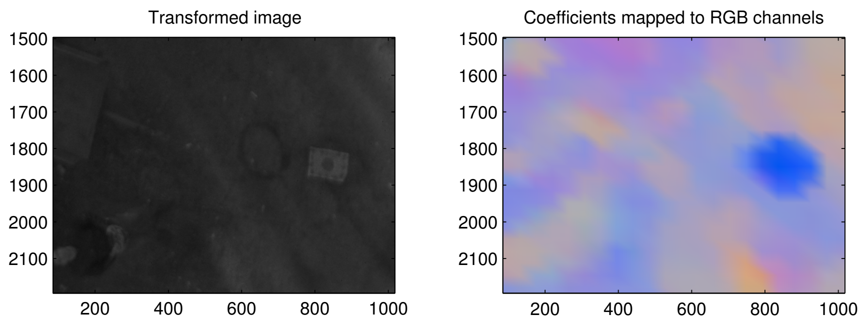


Figure 3.22: Illustrating the loadings used in PCA transform. Loadings are normalized on the RGB channels on the right image. The left image is the resulting monochrome image.

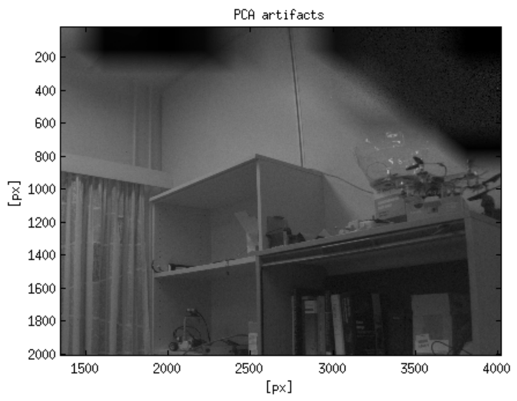


Figure 3.23: Illustrating the artifacts caused by the PCA algorithm when there are sudden changes in color. These artifacts are less pronounced on natural surfaces.

3.5 Identification of disparity

The algorithm for finding the disparity between each pixel works iteratively so that the amount of pixels is kept constant throughout the process. This means that resolution is dropped significantly for the larger windows. The reasoning is that the larger windows provide a rough estimate for the disparity by using the large features on the image. This rough estimate for the disparity is then input to the next round of the algorithm. Different template sizes and resolutions are illustrated in Figure 3.24. The algorithm itself is described more in detail in Figure 3.26 where the main elements of the algorithm are illustrated. The actual block matching is done with the 2-dimensional cross-correlation. This finds the best horizontal match of sub-pixel accuracy by fitting a 2nd order polynomial to the point of maximal correlation.

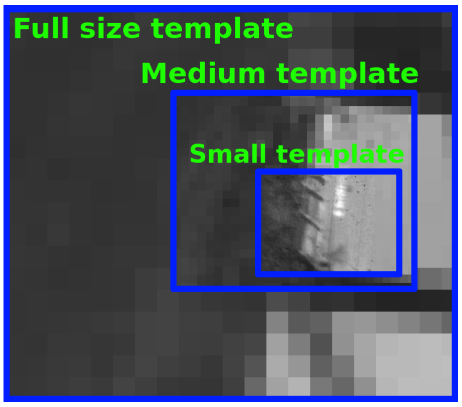
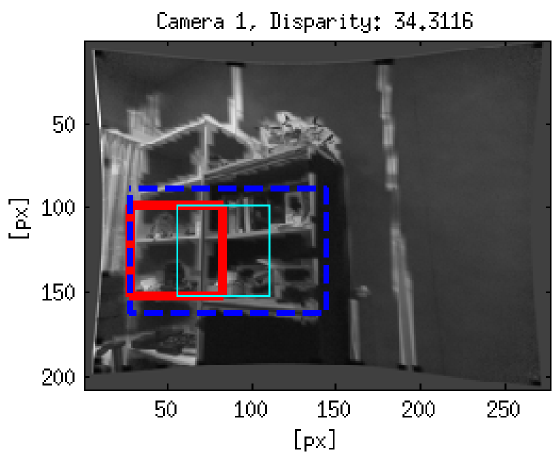


Figure 3.24: Presenting how different template sizes are implemented. The idea is to vary the template size, but keep the amount of pixels the same. This means that there is the same amount of pixels on different template sizes, large templates are used to recognize the large features and the small windows get more accurate estimates for fine detail.

The algorithm used in this thesis is called block matching. The principle is described in Figure 3.25 where blocks used for matching are drawn on both images. The difference in the horizontal position of the original block and the found block position on the other image is the resulting disparity in pixels. The movement is assumed to be nearly horizontal since the images are rectified beforehand though small movement in a vertical direction is assumed to compensate for errors.

The algorithm is similar to one described by Hirschmuller et al. in [9]. The are key differences are that Hirschmuller et al. use only local standard deviation to enforce smoothness of the resulting depth estimate whereas this thesis employs multiple cost functions to describe the match quality, see subsection 3.5.1. Hirschmuller et al. do not rectify images and use the knowledge of the epipolar geometry to pick the appropriate search areas. Hirschmuller et al. use similar hierarchical structure to speed up calculations, however, they only calculate a rough estimate for the depth



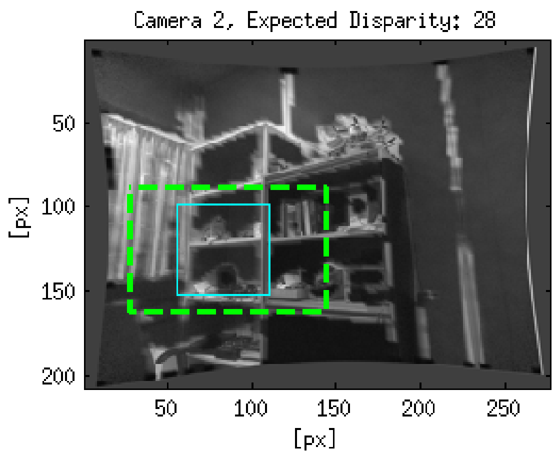


Figure 3.25: Illustrating the actual matching algorithm where the red square represents the area where the template is taken. Green and blue rectangles represent the search areas used for searching the template. Cyan square represents the area where the red square was found on the other image. The difference between red and cyan squares is the disparity found.

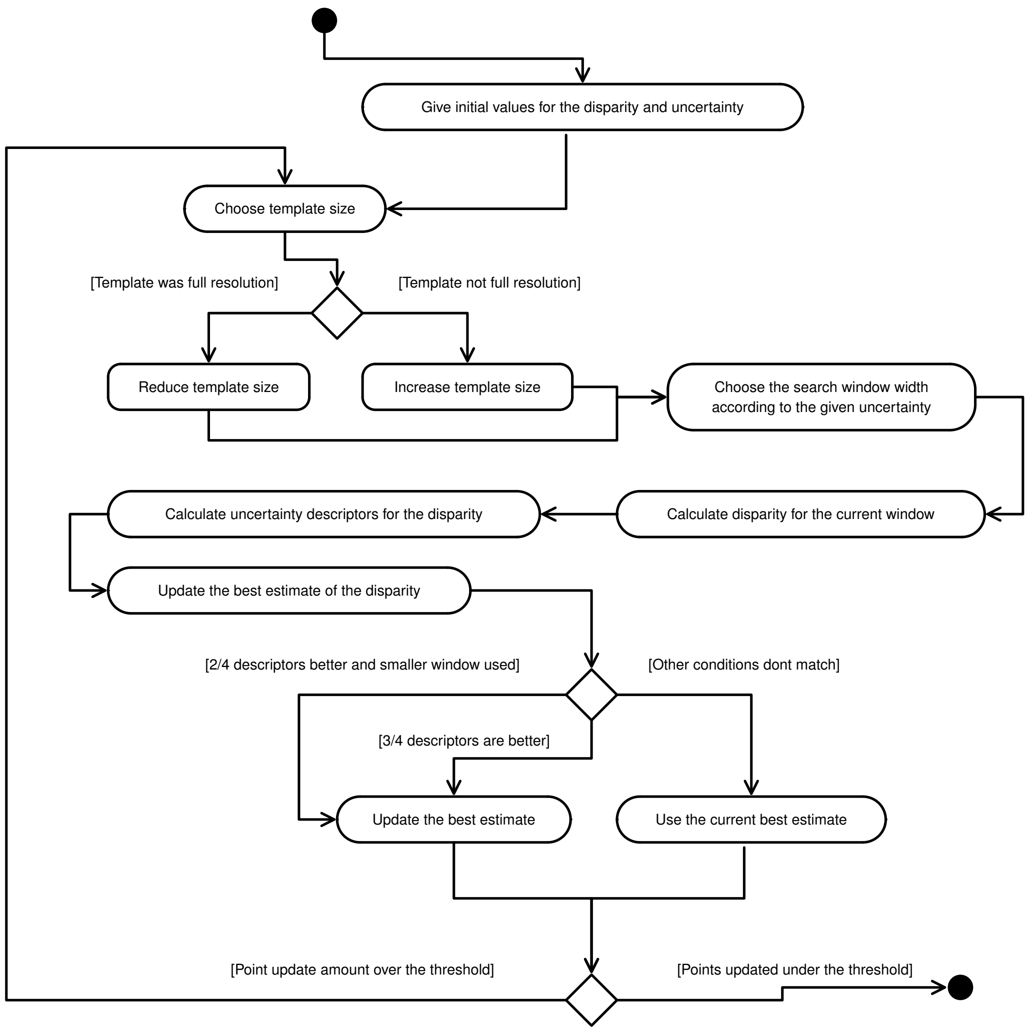

Give initial values for the disparity and uncertainty
Choose template size
[Template was full resolution]
[Template not full resolution]
Reduce template size
Increase template size
Choose the search window width according to the given uncertainty
Calculate uncertainty descriptors for the disparity
Calculate disparity for the current window
Update the best estimate of the disparity
[2/4 descriptors better and smaller window used]
[Other conditions dont match]
[3/4 descriptors are better]
Update the best estimate
Use the current best estimate
[Point update amount over the threshold]
[Points updated under the threshold]

Figure 3.26: This activity diagram is describing how the disparity search algorithm works. The algorithm runs iteratively as long as its needed for the end criterion to be reached. End criterion in this case is when lower than 30% of the points are updated. The threshold can be chosen freely but 30% was noted to be a good compromise between accuracy and run time.

image with one low resolution pass and then constrain the next calculation at the full resolution with the results from the low resolution stage. In this thesis, this kind of method was noted to be prone to errors on low texture surfaces. Therefore, the results from smaller window sizes are preferred if they are indicated to be of better accuracy compared with the result from the larger window. This means that if the algorithm manages to find a match at the smallest window size, it will be used as this results on the best resolution for the depth image. However, in case the match for the smaller window is not good enough, the preliminary results from the larger window will be used. Different window sizes used at different levels were presented on Figure 3.24.

3.5.1 Descriptors for match quality

Descriptors or cost functions quantify how good the current correlation match is. There are 5 descriptors. First of the descriptors is the maximal correlation, which describes how well the template matches the found area. This is useful when there is no repetitive texture and there is enough variance within the template so that one clear maximal value can be found. However, these conditions are not always met. This is why more descriptors are needed. Correlation does not have units as its describes the similarity of the areas. Equation 3.12 is used to convert correlation c to pixels based descriptor c_{px} by scaling it with s_c and image width i_w. The values of s_c was chosen empirically.

$$c_{px} = s_c(\frac{i_w}{c} - i_w) \tag{3.12}$$

The next descriptor is the local STD of disparity. Minimizing local STD ensures that smooth surfaces are preferred over noisy surfaces. This kind of descriptor was used in the work by Hirchmuller et al. at [9]. Local STD has units in pixels, so the uncertainty can be converted to distance uncertainty in 3D.

Descriptor for repetitive textures is called multi-peak uncertainty in this thesis and has the units of pixels. Multi-peak uncertainty describes the longest distance between the correlation peaks. The purpose of this descriptor is to make sure that search window does not get too small if there is a good match for repetitive texture from other descriptors, as for example windows in the building provide very good correlations with other similar templates, but it is hard to know straight away which of these is the right window as the correlation can be as high with the correct window as it is with all the other windows.

The last pixel based uncertainty is called downsampling uncertainty, which is caused by the reduction of resolution. This is used to ensure that the samples that have higher resolution are preferred over samples that have had their resolution reduced. Downsampling uncertainty is calculated by finding how many pixels the scaled version covers in the original image. This is based on assumption that the disparity can be calculated with accuracy of one pixel.

In addition to correlation, there is another nonpixel based descriptor that is calculated together with correlation. This descriptor is called SNR descriptor, as it is calculated from the height of the peak compared with the standard deviation of the correlation. The Equation that scales this is shown at 3.13 where n_{px} represents the SNR descriptor in pixel units and σ_c is the standard deviation computed from the whole correlation vector. The term c represents the peak correlation value. Scale s_n was defined empirically. This descriptor is needed because areas with very little detail might give high correlation values as the areas are similar, but these values will be high throughout the whole search window. This descriptor gives less weight to such results where the correlation peak is not clearly defined. On the other hand, a case where the correlation peak does not have a high value, but is clearly defined, will be trusted more. SNR uncertainty is combined with the correlation descriptor in the end to form only one descriptor.

$$n_{px} = s_n \frac{\sigma_c}{c} i_w \tag{3.13}$$

These descriptors are saved as a 4 channel image where the 1st channel is the multi peak uncertainty, 2nd channel is $\sqrt{n_{px}^2 + c_{px}^2}$. 3 rd image channel is the descriptor for standard deviation with the neighboring pixels. 4th channel is the downsampling uncertainty. A simple pessimistic estimate can be obtained by summing up all the errors together if quick comparison is needed between different results. The heuristics used in this thesis for deciding what is a better match is described in Figure 3.26, actual decision heuristics is at the bottom part of the figure.

3.5.2 Disparity to point cloud

The conversion of disparity to 3D coordinates was presented in Chapter 2.3. The disparity image is on a local coordinate system where Z-axis points away from the camera. The resolution of the disparity image is relatively low. One image pair results in roughly $300x300x3$ sized matrix of 3D coordinates, then converted to a point cloud with the size of $90000x3$. These coordinates were rotated with the IMU orientation to global orientation. The last step was to translate the points based on the GPS position to global coordinates.

The global reference frame is required to compare image pairs with each other. Visualization was mainly done with the Point Cloud Library as MATLAB's plotting functions were found to be inefficient when plotting large amounts of 3D-points.

3.5.3 Parallel computing

The algorithm is well parallelizable. Submitting work to slave clusters is done by submitting the data over the network. Parallelization in the slave machines was done using MATLABs "parfor" feature which divides a for loop into parallel threads.

Execution of the program gets divided to different slaves on the disparity calculation part ("Calculate disparity for the current window" in Figure 3.26). The cluster is constructed by serializing inputs to disparity calculation and sending them through TCP/IP. Each slave will get its own subset of rows to calculate from the master machine. When all the machines are finished with their calculations, they serialize their results and send them back to the master process, which combines all the rows to construct the disparity image. The calculation cluster used in this thesis composed of three slaves, where two were running on the same PC. Two of the slaves used CPU calculation and one slave utilized the GPU.

GPU was NVIDIAs GTX680 and CPU was Intel i7 3700k. Only calculations taking more than minute were sent to the cluster. For shorter ones the overheads on data transmission rendered the cluster inefficient. This resulted in roughly twice the calculation speed compared with using only CPU. However, an exact figure about the calculation times cannot be shown because the algorithm stops the calculation when the results are considered good enough. This means that parameters used for the calculation and the quality of the data affect the calculation time heavily. On average, one image pair took roughly $10 - 15min$ to process.

4. RESULTS

The goal of this thesis was to demonstrate the usefulness of stereo vision for aiding path planning of other robots. Path planning and navigation mesh generation are out of the scope of this thesis so the results can only be validated visually. The actual accuracy of the results can not be verified as there are no reference measurements of the terrain available.

The data has been collected from several types of scenes. Most of the data was collected on board the MAV while flying over area which contained dirt field and road. The field had piles of different types of construction materials, ranging from small boulders to fine sand. The field also contained buildings and cargo containers. The area was surrounded by a dense forest. Thus the data sets contain diverse targets, some with advantageous features for correlation matching and other parts, such as roofs of the buildings and containers contained regular texture which is challenging for the algorithm to match. The dirt field was relatively featureless, and thus served as a good test for the high resolution correlation matching. Hand held stereo images tested the system's performance in absence of vibrations and camera movement. The flexibility of the algorithm was tested with indoor scenes where assumptions about continuous surfaces is not valid.

Point clouds are rotated to the global orientation with the IMU data. Axes that are present in the images are representation of the north east down (NED) coordinate frame. The size of the axes vary between the images; the size is indicated in Figure caption. Point clouds are filtered to discard information that is considered too uncertain. Missing areas in the point clouds are caused by this filtering. Images are cropped to discard the edges of the disparity image where no real data is calculated. The color axis in the disparity image is scaled so that small noise pixels will be saturated.

This Chapter is presents results by data sets. Each Section contains a data set illustrated as figures which contain the image from one of the cameras, the disparity image and the point cloud. All the images are not shown here as one data set typically consisted of roughly 100-150 image pairs, where many of the images were too blurred by the movement. Some data sets are also very similar to one another and were left out. The results presented here illustrate the most typical problems in the system and the best possible performance attained. The last section illustrates the results from combining several point clouds using the IMU and GPS information.

4.1 Data set 10

Data set 10 was captured while flying above the area in the summer. There was no movement checking implemented for the camera triggering at the time of this data set. This resulted in errors with several stereo pairs.

Figures 4.1 to 4.3 illustrate how movement of the MAV combined with the different triggering times affect the disparity image and the resulting point cloud. There is a clear trend in the disparity image. This trend cannot be corrected by rotating the point cloud according to the IMU data; point cloud in Figure 4.3 has been corrected with the IMU data but the ground is not level. Problems on the depth estimate are also indicated in Figure 4.2 which presents the uncertainty for the depth estimate calculated by the algorithm.

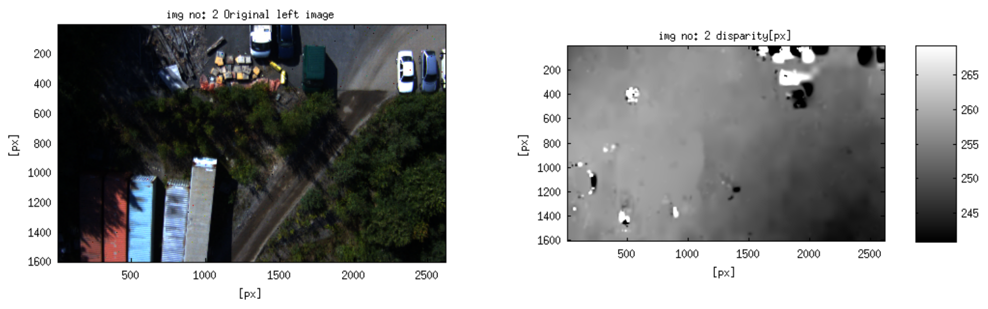


Figure 4.1: Original image and the disparity extracted from a stereo image pair. There is a trend toward the lower right corner.

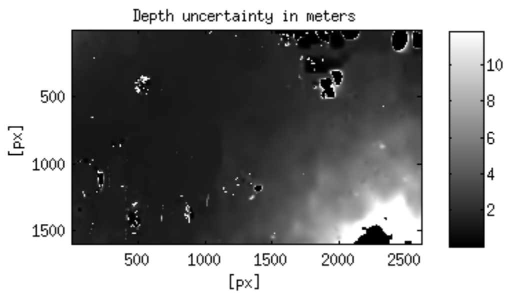


Figure 4.2: Accuracy for the depth estimate. Completely black areas in the middle of the bright areas are caused by data being discarded on areas considered to have highly uncertain depth estimate.

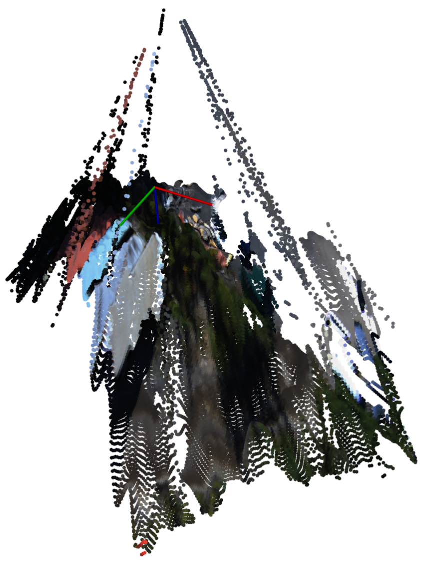


Figure 4.3: This is the point cloud extracted from disparity image presented in Figure 4.1. This image is an example of camera movement between images, resulting in incorrect calibration. Axes are scaled to $1m$

Figures 4.4 and 4.5 illustrate the performance of the algorithm on a thick forest during summer. Most of the treetops can be clearly distinguished from the point cloud image. However, the ground cannot be seen through the tree branches. Right-bottom corner of the disparity image and thus the point cloud falls strongly, contrary to the real terrain. The point cloud generation algorithm determined that the values for this point cloud were initially too large or too small. The baseline was artificially modified based on the altitude of the MAV to force the median depth of the depth image to correspond to MAV's altitude.

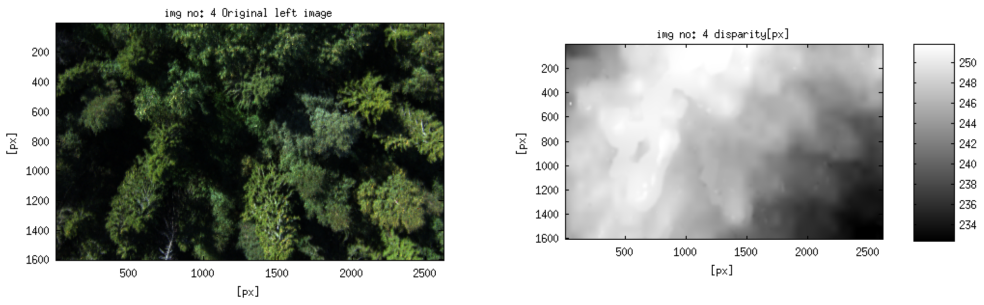


Figure 4.4: Original image and the disparity extracted from the stereo pair. The point cloud is presented in Figure 4.5.

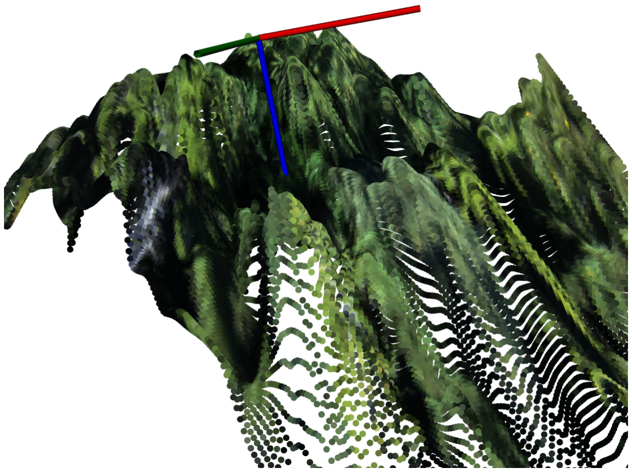


Figure 4.5: Axes are scaled to $10m$. This is the point cloud extracted from disparity image presented in Figure 4.4. This image is and good example of thick forest. Some of the trees are clearly visible on the point cloud.

Slightly different forest image is presented at Figures 4.6 and 4.7. The forest is less dense and some variation in the forest density can be seen on the point cloud and disparity images. The lower density of the forest manifests itself as lower height and smaller disparity. There is a curved slope in the images, not present on the actual terrain. Like the point cloud in Figure 4.3, This point cloud was also forced to use different baseline because the distance with the original calculation was out of range. Altitude of the MAV was used to find new estimate for the baseline.

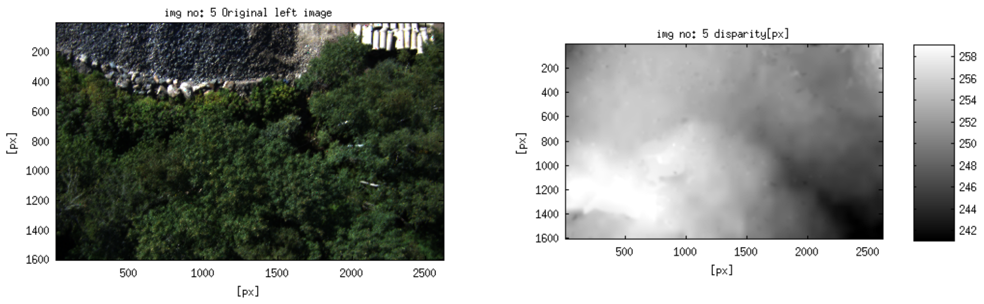


Figure 4.6: Original image and the disparity extracted from the stereo pair. This image is from the edge of the forest. The disparity image indicates that there is less dense forest on the right side of the image. This is not clearly visible on the camera image.

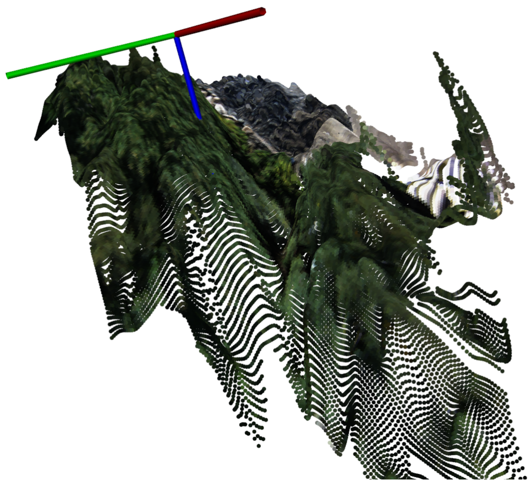


Figure 4.7: This is the point cloud extracted from disparity image presented in Figure 4.6. Lower forest density noted on the disparity image is visible on here as well. Axes are scaled to $10m$.

4.2 Data set 12

Data set 12 was recorded later in the fall when simple decision logic was implemented to prevent images being taken when the position estimate was not good enough or the angular velocity of the MAV was over the threshold. Algorithm for choosing the optimal time to trigger is presented more in detail at Figure 3.14. The data sets chosen here represent typical cases in which the images are not corrupted by movement.

Figures 4.8 to 4.10 present the performance of the algorithm in ideal conditions with distinct features and little movement between the images. Reference measurements would be needed to validate the point cloud estimate. Uncertainty of the depth estimate can be seen in Figure 4.9.

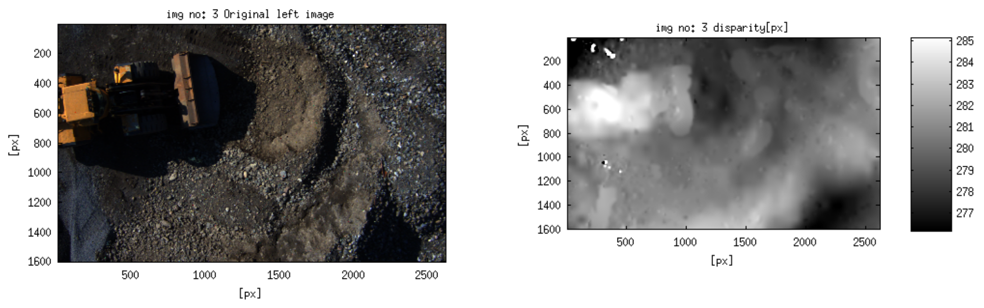


Figure 4.8: Original image and disparity extracted from the stereo pair. The form of the terrain is clearly visible in the disparity image.

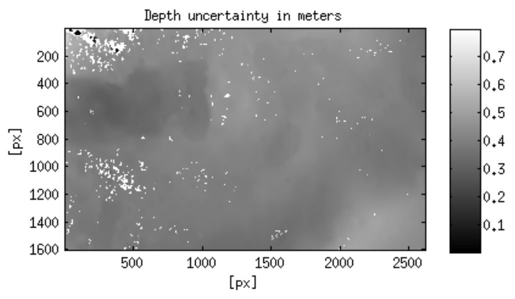


Figure 4.9: Estimated accuracy for the depth estimate. This represents a case where the surface estimate is good.

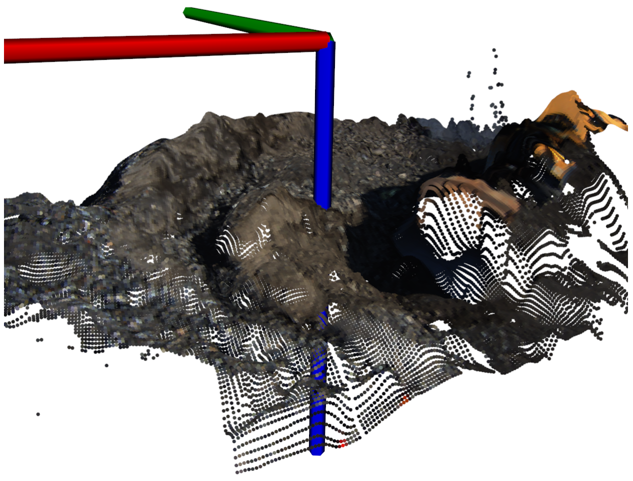


Figure 4.10: Point cloud extracted from disparity image in Figure 4.8. Some of the points on the lower left edge are missing due to estimate being too inaccurate. Small errors are also present on the upper right corner where the spike in height is not present in the real terrain. Axes are scaled to $10m$.

Example of ideal performance when taking images at lower altitudes (below $5m$) can be seen in Figures 4.11 and 4.12. It is possible to distinguish some of the larger rocks from the disparity image and the point cloud.

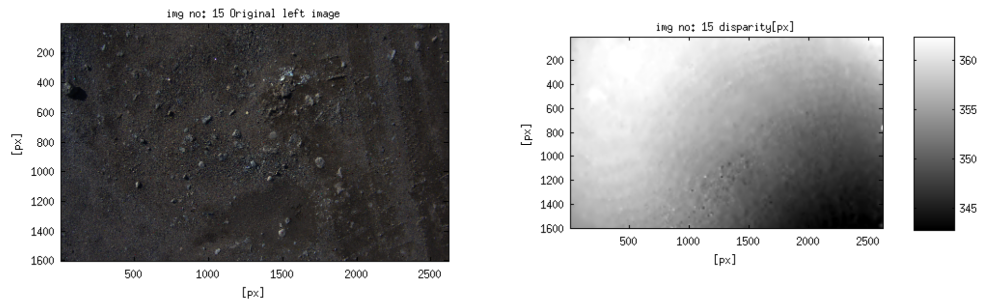


Figure 4.11: Original image and the disparity extracted from the stereo pair. Image taken below $5m$. Small details such as rocks are visible in the disparity image.

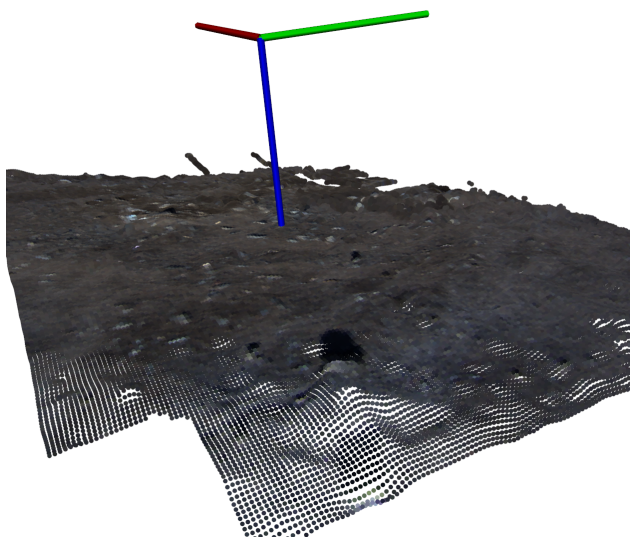


Figure 4.12: Point cloud extracted from disparity image in Figure 4.11. The ground estimate is detailed and only a small portion of the points on the lower left and upper right corners are discarded. Axes are scaled to $1m$.

Another forest example taken later in the fall can be seen in Figures 4.13 and 4.14. This stereo pair has been taken close to the edge of the forest. Many of the trees can be clearly distinguished from the disparity image and the point cloud but its not possible to distinguish ground level.

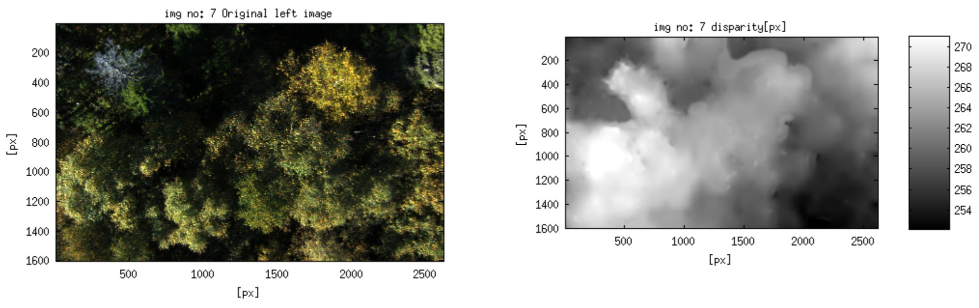


Figure 4.13: Original image and disparity extracted from the stereo pair. Trees are clearly visible in the disparity image. Shape of the trees can be distinguished.

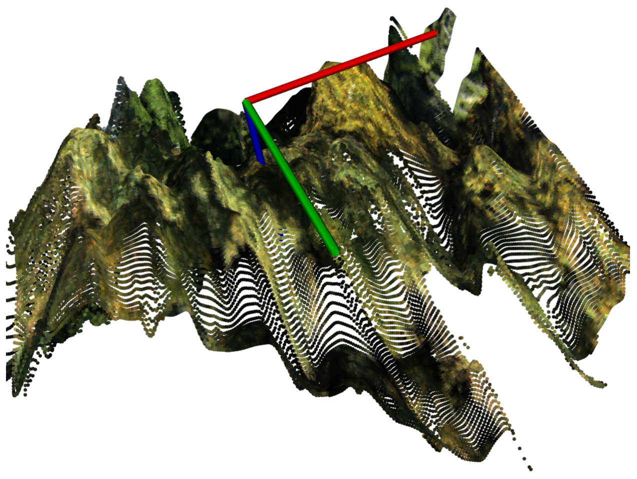


Figure 4.14: Point cloud extracted from disparity image in Figure 4.13. Overall curvature of the point cloud is smaller than in the forest images of the data set 10. Axes are scaled to $10m$.

Another example of flat ground can be seen in Figures 4.15 and 4.16. This stereo pair has been taken from higher altitude (roughly $8m$) compared to Figure 4.11. This results on features seen in the ground being smaller, resulting in more failed matches between the images.

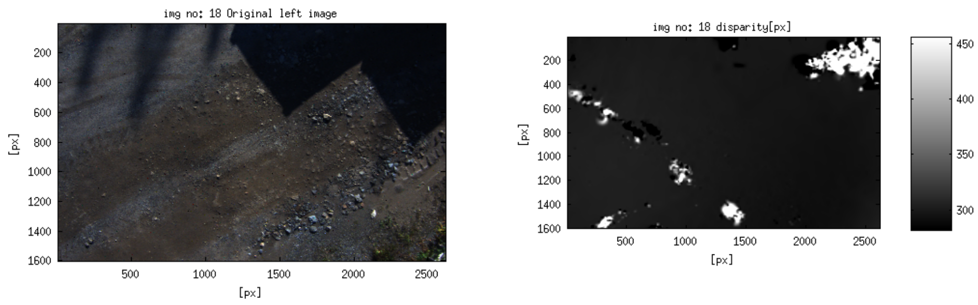


Figure 4.15: Original image and disparity extracted from the stereo pair. Noisy disparity image makes it harder to distinguish real features.

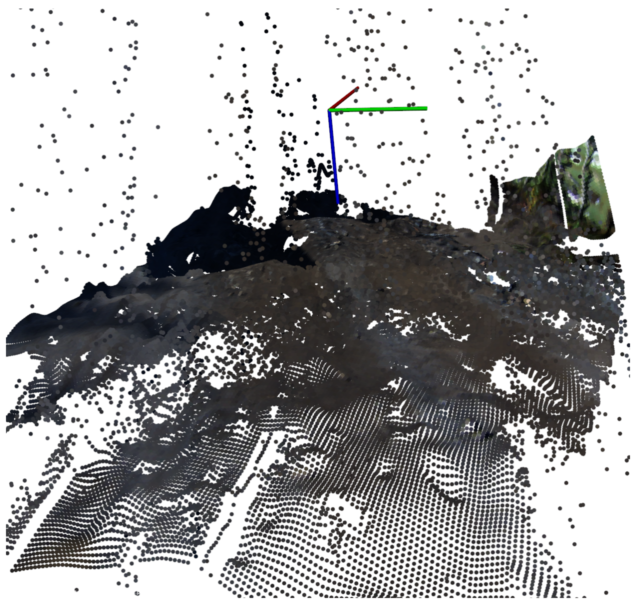


Figure 4.16: Point cloud extracted from disparity image presented in Figure 4.15. Ground and features are quite visible but point cloud is corrupted by a higher noise compared to point cloud in Figure 4.12. Axes are scaled to $1m$.

4.3 Other tests

Several other data sets were captured on varying conditions. The Purpose of these tests was to eliminate errors caused by the MAV and to explore how the system would perform on use cases that it was not initially designed for, like indoor scenes or cameras pointing parallel to the ground. The stereo camera was held in hand, so that errors caused by the movement are not as pronounced as in the other tests. Key results from these data sets are presented on this section.

Stereo pair from an indoor data set can be seen in Figures 4.17 and 4.18. This image is a good example of the best performance achievable indoors. There are good unique features in the curtains but the rest of the image is lacking distinctive features. The disparity image shows that a good estimate is found for the curtains, and parts of the shelves are also visible. The point cloud in Figure 4.18 shows that the curtains are rendered with very little noise, but there are heavy deformations on walls and the shelves. The computer on the table at the lower left corner of the image, however, is at the correct location.

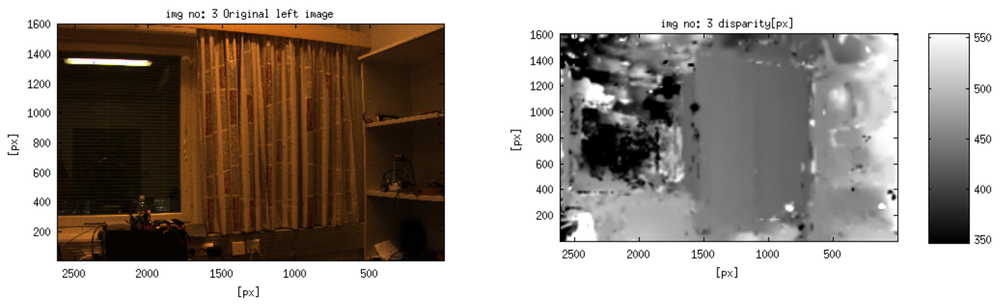


Figure 4.17: Original image and disparity extracted from the stereo pair. Curtains are the only part of the scene with enough features to result in a good estimate. Window is interesting because some of the features are visible on the surface of the window and some are reflected from the other side of the room, resulting in noisy surface estimate for the window.

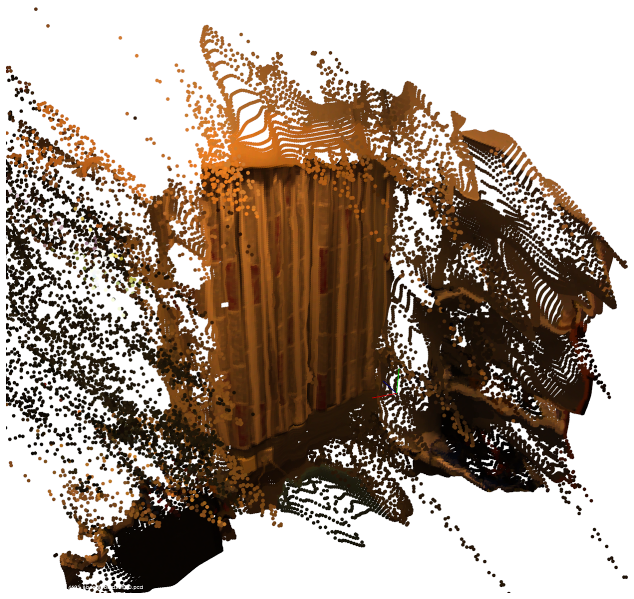


Figure 4.18: Point cloud extracted from disparity image in Figure 4.17. Point cloud has a good estimate at the curtains but other parts of the room such as the shelves are deformed or noisy. Axes are scaled to $10cm$.

Simple outdoor scenes were also tested with the stereo rig being hand held. Such a case is presented in Figures 4.19 and 4.20. These figures present the performance with random textured surfaces like grass and highly repetitive texture seen on the walls of the building. Artifacts resulting from the camera movement are absent on these images. The area hidden by the tree on the right side of the image has many points which seem to form a wall, even though there is no such feature on the real scene.

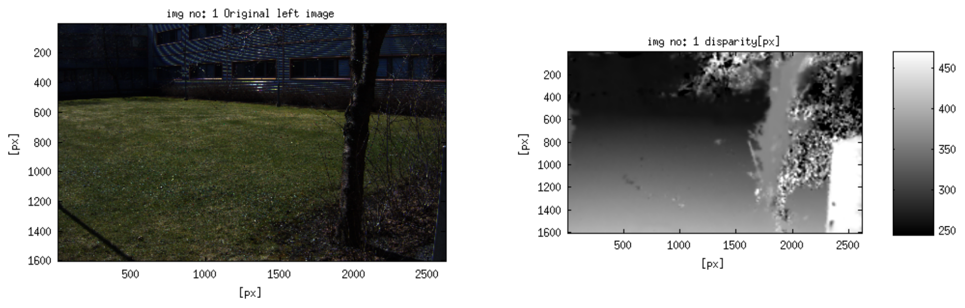


Figure 4.19: Original image and disparity extracted from the stereo pair. Image was taken by holding the stereo rig in hand, resulting in high quality images with precise calibration. This can be observed as a smooth disparity estimate.

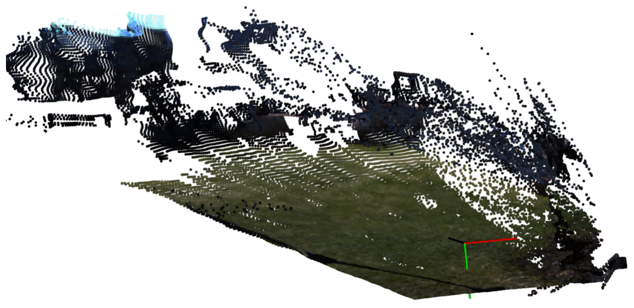


Figure 4.20: Point cloud extracted from disparity image in Figure 4.19. Problems caused by the tree branches can be seen clearly on the right side of the image. Axes are scaled to $1m$.

Table 4.1 summarizes the uncertainty for the depth images. The estimate was calculated by first estimating the accuracy of the correlation match in pixels. Pixel units were then converted to meters by displacing the estimated disparity with the pixel uncertainty. This resulted in a difference on metric scale. The mean and median presented in Table 4.1 are calculated from the points that were not discarded due to very high uncertainty. High mean values are result of small amount of points having very high uncertainty, this is why median is a better descriptor for the overall quality of the match. Examples of the original uncertainty estimates were presented in Figures 4.2 and 4.9.

Table 4.1: Mean and median values calculated from the uncertainty estimate for the depth image.

Figure	Median $[m]$	Mean $[m]$
4.19	0.0753	13.7
4.8	0.419	8.08
4.13	2.01	5.35
4.11	0.0418	1.83
4.15	0.174	7.65
4.1	2.03	7.12
4.4	2.38	5.33
4.6	1.53	4.32

4.4 Combining point clouds

Point clouds need to be combined to form a map about a larger area. This is not straightforward as there will be conflicting information from different point clouds. In this thesis, the point clouds were combined only based on the IMU and GPS data, with no consideration for the content of the point cloud. This means that point clouds are not matched to one another in any ways, which results in a non-continuous terrain with overlapping areas not matching.

All the point clouds calculated from data set 12 are shown in Figure 4.21. Positions of the individual point clouds are roughly where they should be, but as seen in Figures Figures 4.22 and 4.23, there are gaps between point clouds. Some point clouds fail completely to match and clusters of points do not form smooth and continuous surfaces.

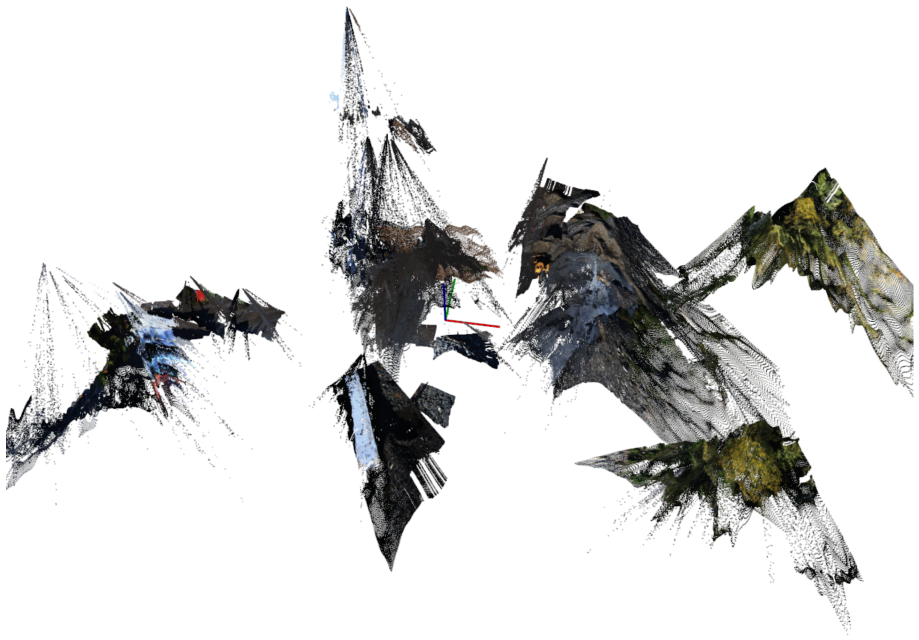


Figure 4.21: All the point clouds calculated from the data set 12 presented combined using GPS location and IMU attitude. Axes on the image are sized $10m$.

Figure 4.22 is a section of the Figure 4.21. It is showing how four point clouds are matching one another. Areas are overlapping but there is clearly offset between the point clouds. Similar situation can be seen in Figure 4.23 where the road should be continuous surface, but the point clouds are not connecting. In addition, both figures have residual points from point clouds that were either misplaced or used wrong calibration. These residual points are prominent on the left side of the Figure 4.23.

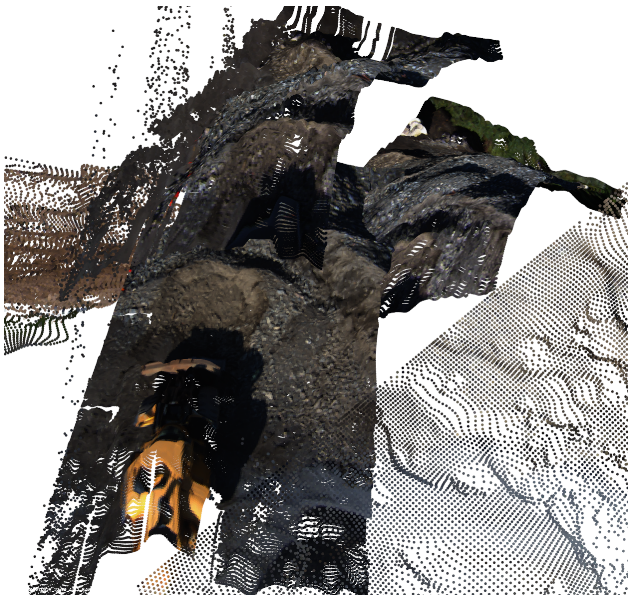


Figure 4.22: Smaller section of the Figure 4.21. Same area is presented also in Figure 4.10 but without all the other point clouds.

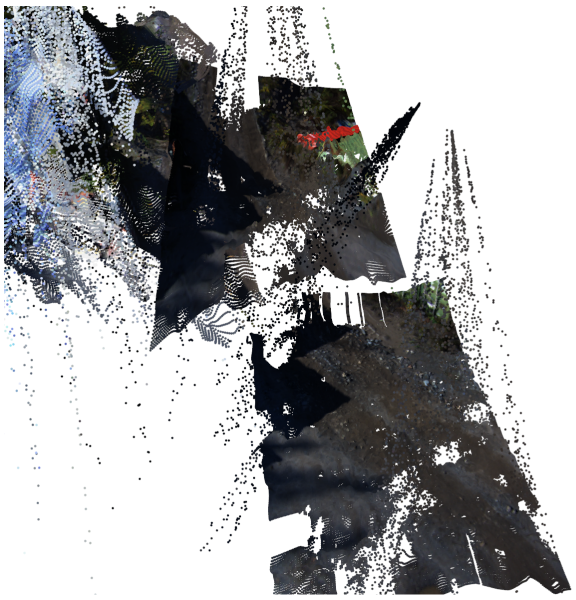


Figure 4.23: Smaller section of the point clouds presented in Figure 4.21. The road visible should be a continuous surface.

5. DISCUSSION

Chapter 4 presented the achievements of this study. This required solving several problems not foreseen when this study was commenced. Some of the problems remain unsolved and the following sections will discuss these more in detail. The final section will conclude how the initial objectives were fulfilled and what can be benefited from this study in the future.

5.1 Results

The objective of this thesis was to explore the possibilities of the high resolution, small base line to distance stereo vision. The Initial goal was to obtain matching point clouds and to construct a map of the terrain. This goal was not fully realized. On the other hand, the potential of the high resolution stereo vision is visible on the results.

Good performance was achieved for the data set 12. Figures 4.14 and 4.16 represent the best quality that was achieved on board the MAV. However, many of the stereo pairs had to be discarded even from data set 12 because of the camera movement or blurring of the images.

It is important to note that even though we get sub-pixel accuracy for the position of the maximum correlation, the pixel size is larger for the low resolution images used at the beginning, thus the estimates done with the large windows are not accurate to the sub-pixel level at full resolution. This is taken into account at the estimate for the accuracy of the match. Even at the highest resolution, the sub-pixel accuracy of the maximal correlation is still not a certain indication of a matching feature, especially with smaller template sizes. This is why heuristics were used in this thesis to calculate the uncertainty for the depth estimate.

GPS errors were tested by repeating static measurements, see Figure 3.11. The estimate of the GPS accuracy is also given by the RTK-lib, but this information was found to be rather unreliable. The accuracy of the attitude measurement is estimated within the Kalman Filter in real time during flight. Similar results were acquired by shaking the unit on a level table and seeing how much variation could be caused to the attitude.

The algorithms and software architecture developed in this thesis provide a good basis for moving towards real time applications. The distributed computation developed for this thesis can be used to offload computation from the MAV's processor to a processing cluster to reach near real time performance even with the current complex algorithm. This results from the fact that the computation time is linearly dependent on the number of machines in the cluster; as long as the amount of machines stays under the vertical resolution of the images.

5.2 Outstanding problems

All the problems could not be solved during this thesis. Movement of the camera between the images was apparent especially on the data set 10 where most of the depth images had a noticeable trend. A good example of this trend is on the point cloud presented on Figure 4.7 as much of the features can be distinguished from the background but the ground level is not horizontal.

The non linear nature of the disparity-to-metric conversion makes errors due to camera movement challenging as the error can often be larger than any actual feature present on the scene. This was the case with Figure 4.3 where the errors were so large that no features could be distinguished from the point cloud.

Some of the errors were removed by forcing the depth values to a sane range. One of the methods used was checking if the median of the distance image had reasonable values and forcing the median disparity to a safe value by applying a constant offset to the disparity. The limit used here was $60m$. Safe range was determined from the height given by the IMU. This means that the terrain was assumed to be a flat plane and the errors were considered only caused by translation. Large errors like these in Figure 4.7 were left on many of the point clouds, but this solved much of the issues that manifested from distance estimates close to infinity. However, forcing the baseline to a safe value only makes the point clouds look better, but does not fix the actual error.

Matching point clouds proved more challenging than initially thought, see Figures 4.22 and 4.23. This is caused by localization and IMU attitude estimate errors causing the point cloud to move to a different position than in reality. MAV localization errors are generally rather small, in order of a few meters, depending on conditions. Attitude errors cause offset that increases with distance from the cameras. Attitude errors were estimated to be roughly $+-1Deg$. Camera movement between the stereo images causes the most unpredictable error as it changes in non-linear manner with distance from the cameras, see Figures 3.5 and 4.12. A rough estimate for the magnitude of the error can be estimated, but more information sources are needed to compensate for these errors.

5.3 Conclusion

Many of the problems described in the section 5.2 can be mitigated with right hardware and algorithm choices. The largest errors caused by the triggering difference in cameras can be negated with machine vision cameras. Much of the point cloud matching errors can be solved by matching the stereo image pairs with other image pairs. This results in information about camera translation and rotation between the images. The same algorithm can be used to get an estimate for the camera movement between the images although the best solution to camera movement would be to use cameras with accurate triggering.

To improve the results further, hardware would be needed to compensate for the movement of the MAV as the benefits of the high resolution are radically reduced by the motion blur. Blurring can be removed by isolating the cameras from the MAV body and decoupling the movement of the stereo rig by using a gimbal to keep the camera orientation separate from the orientation of the MAV. Other possibility is to use cameras with small triggering delay so that they can be accurately triggered when the MAV is not moving.

In conclusion, this thesis proved important to highlight areas that are important for aerial stereo vision. The algorithms developed for disparity computation are also a good starting point for developing more real time oriented algorithms.

BIBLIOGRAPHY

[1] R.A. Boie and I.J. Cox. An analysis of camera noise. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(6):671–674, 1992. ISSN 0162-8828. doi: http://doi.ieeecomputersociety.org/10.1109/34.141557.

[2] Olivier Faugeras. *Three-dimensional computer vision : a geometric viewpoint*. MIT Press, Cambridge, Mass, 1993. ISBN 0-262-06158-9.

[3] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, Cambridge, UK New York, 2003. ISBN 0-521-54051-8.

[4] Peter I. Corke. *Robotics, Vision & Control: Fundamental Algorithms in Matlab*. Springer, 2011. ISBN 978-3-642-20143-1.

[5] J.Y. Bouquet. Camera calibration toolbox for matlab. Available online, 2008. URL `http://www.vision.caltech.edu/bouguetj/calib_doc/index.html`.

[6] Michael Warren and Ben Upcroft. High altitude stereo visual odometry. In *Proceedings of Robotics: Science and Systems*, Berlin, Germany, June 2013.

[7] Riku Hellevaara. RTK-GPS lentävässä kuvausalustassa ja koordinaattien määrittäminen kuvasta. Master's thesis, Tampere University of Technology, Finland, 2013.

[8] Angus P. Andrews Mohinder S. Grewal. *Kalman filtering : theory and practice using MATLAB*. Wiley, New York, 2001. ISBN 0-471-39254-5.

[9] H. Hirschmuller. Stereo processing by semiglobal matching and mutual information. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(2): 328–341, 2008. ISSN 0162-8828.