Vision system prototype for UAV positioning and sparse obstacle detection

Piotr Jaron
Mateusz Kucharczyk

Department of Electrical Engineering
Blekinge Institute of Technology
Karlskrona 2012

Table of Contents

Acknowledgements:

The authors would like to express their gratitude to Torkel Danielson and the company Intuitive Aerial AB for providing hardware, materials but most importantly constant support and advice necessary to complete this thesis.

Abstract

For the last few years computer vision due to its low exploitation cost and great capabilities has been experiencing rapid growth. One of the research fields that benefits from it the most is the aircrafts positioning and collision avoidance. Light cameras with low energy consumption are an ideal solution for UAVs (Unmanned Aerial Vehicles) navigation systems. With the new Swedish law – unique to Europe, that allows for civil usage of UAVs that fly on altitudes up to 120 meters, the need for reliable and cheap positioning systems became even more dire. In this thesis two possible solutions for positioning problem and one for collision avoidance were proposed and analyzed. Possibility of tracking the vehicles position both from ground and from air was exploited. Camera setup for successful positioning and collision avoidance systems was defined and preliminary results for of the systems performance were presented.

List of Symbols

Matrix

D	Distortion Matrix
M	Camera Matrix
M^l	Camera Matrix of left camera
M^r	Camera Matrix of right camera
H	Homography Matrix
Q	Reprojection Matrix
H_2	Harris Matrix
T_k	Transformation that relates two camera poses
R_k	Rotation matrix at instant k
t_k	translation vector at instant k
C_k	Camera Pose at instant k

Object Coordinates

$P = [X_{real}\ Y_{real}\ Z_{real}\ 1]$	object's/feature's real world coordinates
$p^l = [x^l\ y^l\ 1]$	object's/feature's coordinates projected onto imager of left camera
$p^r = [x^r\ y^r\ 1]$	object's/feature's coordinates projected onto imager of right camera
w	scale factor
S	the size of the object
s	the object's image on the imaging plane
Z	the distance from the camera to the object/feature

d disparity

Camera Parameters

B baseline

τ pixel size

$f^{[mm]}$ focal length expressed in minimeters [mm]

$f^{[px]}$ focal length expressed in pixels [px] (square pixels)

$f_x^{[px]}$ focal length expressed in pixels [px] (rectangular pixels)

$f_y^{[px]}$ focal length expressed in pixels [px] (rectangular pixels)

C_x^l displacement along x-axis between optic axis and center of imager for the left camera

C_y^l displacement along y-axis between optic axis and center of imager for the left camera

C_x^r displacement along x-axis between optic axis and center of imager for the right camera

C_y^r displacement along y-axis between optic axis and center of imager for the right camera

$\Delta c = C_x^l - C_x^r$ difference between center of imagers

$\widetilde{B}$ estimated baseline (obtained through calibration)

$\widetilde{f_{px}}$ estimated focal length (obtained through calibration)

$\widetilde{\Delta c}$	estimated difference between center of imagers (obtained through calibration)
δ_B	baseline calibration error
δ_f	focal length calibration error
δ_c	center of imagers calibration error
e_B	baseline error
e_f	focal length error
e_c	imager center error

Obstacle Detection

λ	Harris corner measure
$[x_{center}\ y_{center}]$	Clouds Center
Z_μ	Clouds Depth
ρ_{cloud}	Cloud Density

1. Introduction

1.1 Purpose of the thesis

The purpose of this thesis is to design two vision system prototypes. First of them should allow for tracking specified marker within field of view of stationary cameras. An accurate estimate of the markers position in 3D coordinate system related to the cameras can them be achieved. Markers could be mounted on a UAV (Unmanned Aerial Vehicle) in a way that makes them visible independent of the vehicles orientation. Tracking and estimating the position of the marker is then equivalent to tracking and estimating the position of the UAV. This system has the advantage of calculating the position of the marker relative to the same stationary object for every image frame. Thus the error of estimation does not depend on the number of frames taken, and is practically only dependent on the camera parameters, and distance from the marker to the center of the coordinate system. The disadvantage of this system is that the cameras are stationary so the UAV can only be tracked in their initial field of view, also, tracking is impossible in case of occlusions – when non transparent objects appear between the cameras and markers. First system is described in section 3 of this paper.

Second system should allow for tracking of the position of an UAV and sparse obstacle detection, but with cameras mounted on the vehicle. No clearly defined marker can be used as the environment is constantly changing. Positioning should work for a wide range of altitudes as the maximum height of flight allowed for UAV-s in Sweden is 120m, without any lower boundary. For collision detection purposes it is assumed that environment has rather low obstacle density, which is largely the case, unless the aircraft is flying on a small altitude in densely populated areas. The greatest advantage of this system is that it works on board the UAV, so no stationary cameras on the ground are needed. Therefore the positioning can continue no matter how far the vehicle is from its takeoff position. Also, no specifically designed markers are needed, only naturally occurring elements of the environment are used for estimating the position. Another advantage is the lack of possibility that UAV will disappear from the sight of cameras due to occlusion, as the cameras are mounted directly on the object which position they have to track. However, this comes with a price, longer range and greater mobility are achieved with resignation from absolute,

stationary reference frame which was the cameras position on the ground. Thus the position estimation accuracy will suffer from error increasing with each image frame. Also, UAV-s tend to have problems with stabilizing their position, so position drift is another factor that has to be taken into account in the second system. Those problems combined tend to cause serious errors in position estimation during longer flights. Second system is described in section 4 of this paper.

It has to be noted that this paper deals with designing specific parts of the two systems mentioned above. It does not describe how the cameras can be mounted on UAV, which is vehicle-specific feature. It also does not specify what cameras exactly should be used, it does however contains information about desired parameters and set up of those devices. Transfer of data from cameras to image processing unit is another issue that is not addressed extensively in this paper, neither is the location of processing unit – whether should it be on board or on the ground – although both of this options are discussed in chapter 4. The thesis contains description of parts of the system that are related to vision and image processing which are: algorithms that are used for positioning and obstacle detection, extracting camera parameters to enable depth estimation and efficient image frames processing, specifications of implemented markers and ways to detect their exact position on the image as well as means that can negate or at least lessen problems that commonly occur in vision systems development.

1.2 Motivation

Over the last decade, vision systems in robotics have experienced rapid development. The main reason being their price, availability, and weight compared to other measurement devices. Additionally, methods of image processing got much easier to implement with increasingly faster and more efficient computer software and hardware. Another advantage of vision system in comparison to more standard sensors or sensor systems is the ability to distinguish between different features that can be found in the environment based on their appearance. Those are the reasons that made vision systems so popular with UAV-s even more than with other robots, specially ground moving. Flying vehicles have additional restrictions that makes standard sensor system hardly applicable or not applicable at all in their case. When we consider a ground moving vehicle, laser or ultrasound sensors will handle the task of collision detection perfectly, because it can be almost a 2D plane obstacle detection problem, while for flying vehicles that is not the case. It has to be able to spot obstacle in a much larger, 3D field of view with the added height dimension. So installing sensors mentioned above would result in insufficient area coverage. Another problem with UAV is that it cannot be as heavily encumbered as ground moving robot, which have to lead to either designing inefficient machines that are too large for their purposes, or switching from standard sensor systems to installing one or few light cameras on the aircraft. Then, after the images are processed, results can be comparable to those achieved with standard sensors although less money is spend and the design is more ergonomic. Moreover, image from camera is created instantaneously, whereas for any type of scanning measurement device it takes considerable amount of time. This can play a huge role in real time systems. All above considered, the motivation of the thesis was to propose a prototype of a system that could solve the problems of positioning and collision detection for UAV-s using only cameras. This could be very useful and play a large role in situations where GPS (Global Positioning System) is either non-installed on the vehicle or is jammed, which happens very often. Also the accuracy of GPS is sometimes not sufficient for specific task that the UAV has to accomplish. Many solutions has been proposed to the problem of positioning by combining GPS and vision systems, but using only the later is a task that yet has to be fully exploited.

1.3 Achieved results

For the first system described in section 1.1, a marker has been designed. Also, a mean of determining the markers position in varying environment conditions was developed using histogram modification and back-projection techniques. Depth estimation and 2D to 3D projection were exploited to allow for calculation of real world coordinates of the marker. Camera calibration methods were analyzed and implemented.

For the second system an algorithm for 3D and 2D positioning was realized based on Visual Odometry. Improved feature detector techniques using Shi-Tomasi and ORB feature detectors were implemented as means of making the system much more efficient and accurate. Method of sparse obstacle detection was proposed and realized as a collision detection part of the system. Simple visualization of systems performance was developed. Error causes were analyzed.

Note, that due to the fact that in this report varied problems are tackled, the preliminary results are shown in chapters describing issues which they concern.

2. Research and requirements

2.1 Related works

Throughout last decade many approaches were made to solve UAV positioning or/and obstacle detection problem. In [1] a structural landmark navigation system was proposed for successful navigation. This approach relied on a previously defined class of objects that are geometrically characteristic and appear often in industrial areas. Examples of such objects are bridges, churches or skyscrapers. System possessed a database which defined the characteristics of those landmark structures. The idea proved to be reliable in tests performed on Google Earth maps, as it was able to correctly identify a given landmark. It was stated that this approach would lead to successful positioning in most industrial environments as long as an appropriate database containing object descriptions would be provided to the system. It seems that main advantage of this idea is accurate positioning of an UAV without drift, as we possess an absolute reference system. However, this approach will work only in industrial areas, and on very high altitudes as the structure of whole landmark has to be clearly observed. Additionally, for the system to perform effectively in a large number of industrial environments, the database would have to be extremely large and hard to create.

When considering collision avoidance problem, [2] presents extreme example of a very popular method of depth perception. This method called structure from motion is described more extensively in subchapter 2.2. Idea behind [2] is to first use one camera mounted on UAV to detect an obstacle by means of feature detection which are described in subchapter 4.1 of this paper. Then the UAV would fly around the obstacle without closing in on it or getting farther away while capturing image frames with the camera. It would allow for using structure from motion to not only obtain the distance to obstacles but also a low accuracy model of it. Advantage of such an idea is that only one camera has to be mounted on the UAV. However it possesses a number of disadvantages which make it hardly applicable in any real life situation. Firstly, the obstacle detection system should not determine the way in which the UAV moves, on the contrary, it should adopt to it. Secondly, it would be very hard to realize such steady motion around the obstacle. Additionally, the UAV would have to

circle around every feature point or feature point group it detects, which would take all the flight time and much more.

In [3] another approach to collision detection for UAV-s was taken, foundation of which was organic insect vision system. Insect use a technique called optic flow to safely maneuver around obstacles. It is based on estimating how fast certain points in the image are moving. Assuming constant speed of an insect, closer points will have greater optic flow than those located further away. So the task of navigation between obstacles is performed by attempting to balance the optic flow on both sides of the insect, or by flying away from that side on which the optic flow vectors are too big. Novelty of the approach presented in [3] is based on using three color image channels to generate optic flow field – a matrix containing optic flow vectors instead of just using grayscale image for that purpose. The optic flow method of collision avoidance is quite popular and works well in navigating through corridors, canyons, or between buildings, but cannot cover every aspect of collision detection by itself. For example it does not handle very well situations in which the obstacle is at the front of an UAV, and it is pretty hard to use when there are only few small obstacles in the field of view of the vehicle.

Paper [4] proposes a vision system for aiding INU (Inertial Navigation Unit) in the task of positioning of aerial vehicle. System consists of a high quality camera and an algorithm that determines translation and rotation of camera between consequent image frames. This is achieved by first detecting features – characteristic elements in the image by using Harris corner detector. Feature detectors are described more extensively in subchapter 4.1 of this thesis. Then feature points from two consequent frames are matched and the movement of the camera can be calculated using homography. In the article, two methods are proposed to ensure better matching accuracy, RANSAC (Random Sample Consensus), and another one, developed by authors. The approach proves to have potential, but results are not conclusive whether it can be used in real life situations as the authors state that more complicated algorithms are required.

Great example of very effective positioning technique for UAV is given in [5]. The approach is called visual odometry and is spoken of in length in subchapter 4.2 of this thesis. Article describes a vision system that can aid GPS positioning in case of failure of the latter in any

part of the flight. It uses KLT (Kanada-Lucas-Tomasi) feature detector to extract features from consequent image frames. It is assumed that the aircraft moves on a very high altitude so that the ground can be considered a 2D plane regardless of the terrain shape and diversity. The only camera on UAV is facing down, so the altitude of the aircraft is extracted from INU rather than from vision system. The translation and rotation of UAV are calculated by matching features from one frame to the ones in another, and though a noticeable drift from real position is observed after long GPS-less flight, the system accomplishes its task in those specific conditions very well. The article proves that vision systems has a great potential in UAV navigation and presents successful usage of one in a real life situation. This approach and set up will not work for low altitude aircrafts or the ones flying in largely dynamic and industrial areas but in the role that it was intended to it works very well.

In [6], a very interesting idea for collision detection system can be found. It is complete in the way that it allows for detection of obstacles both on the sides of the vehicle and in front of it. The idea is to use cameras facing to the sides to estimate the optic flow, and two front facing cameras. Authors also suggest that only a pair of cameras can be used, positioned at the front, but they would have to have a very wide field of view – feature available by installing fish-eye lenses. The cameras in the front of the vehicle are arranged in a stereo rig – a requirement for successfully implementing stereo vision, a technique described more extensively in subchapter 4.1. It allows for creation of a depth map – grayscale visualization of distance from the stereo rig to the elements of the environment captured on an image frame. This way, it can be determined if any objects are too close to the UAV thus prompting an evasive maneuver. This idea has proven successful in handling any obstacles that appear in front of the Vehicle. As for the optic flow determined on both sides of the aircraft, it works well when the object flies through canyons, whether they are naturally occurring or human made urban canyons. The reason is that by estimating the optic flow field only, the absolute distance from an obstacle cannot be determined, but when the aircraft is flying through a canyon, the task is just to balance the optic flow on both sides. Then it can be assured that the vehicle is approximately equidistant from both sides of the canyon. The approach presented in the article is both interesting and sensible. It offers cheap system that will fulfill its role successfully when guiding an UAV through the urban or natural canyons. The only

disadvantage that can also be describing almost any UAV vision system is that it is not universal in terms of the environment.

2.2 Camera calibration, models and depth perception

2.2.1 Pinhole camera model

In this thesis for all calculations regarding distance, position and size estimation, a pinhole camera model will be used. There some operations needed to adjust this extremely simplified model for real world values calculation. They are described in length in further parts of the paper. In pinhole camera model we assume that there exist an opaque wall with one small aperture in the middle that allows only one ray of light at a time to pass. The ray is then projected onto image plane/projection plane which is at a distance equal to the focal length of the camera from the aperture wall. There are two advantages that result from this approach. The image is always in focus of the camera, and there is only one parameter that has impact on the image height – focal length. This allows for great simplification of calculation. The basic idea of pinhole camera model is presented in figure 2.2.1. If we assume that H is the real height of some object and h is its height on the projection plane, a simple mathematical equation (eq. 2.2.1.) can be written originating from a similar triangle rule.

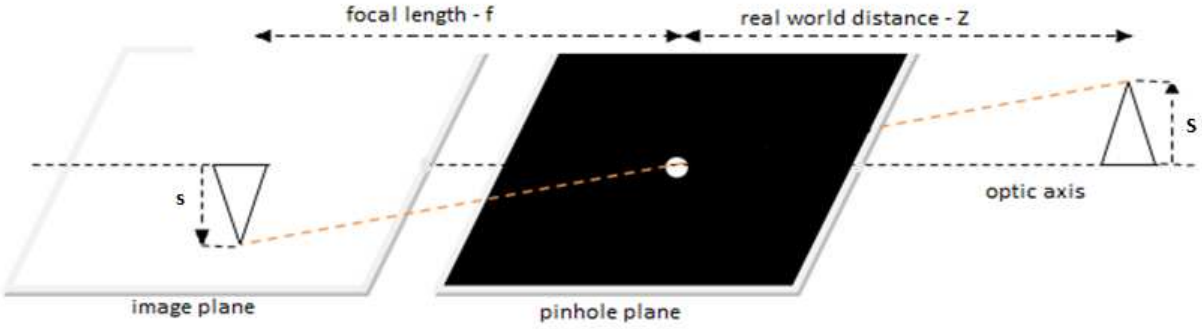

Figure 2.2.1. Visualization of the camera pinhole model. The projected ray of light travels from point with height H through the aperture and creates an image h on the image plane

$$\frac{S}{Z} = \frac{-s}{f} \qquad (2.2.1.)$$

Figure 2.2.1. can be rearranged as in [7] to be even simpler and more intuitive. Figure 2.2.2. expresses practically the same mathematical relation, but the image plane is moved in front of the pinhole plane so that the new similar triangles equation (eq. 2.2.2) does not possess a negative expression. The pinhole aperture on the pinhole plane now becomes center of projection through which all light rays have to pass, but after creating an image on the image plane.

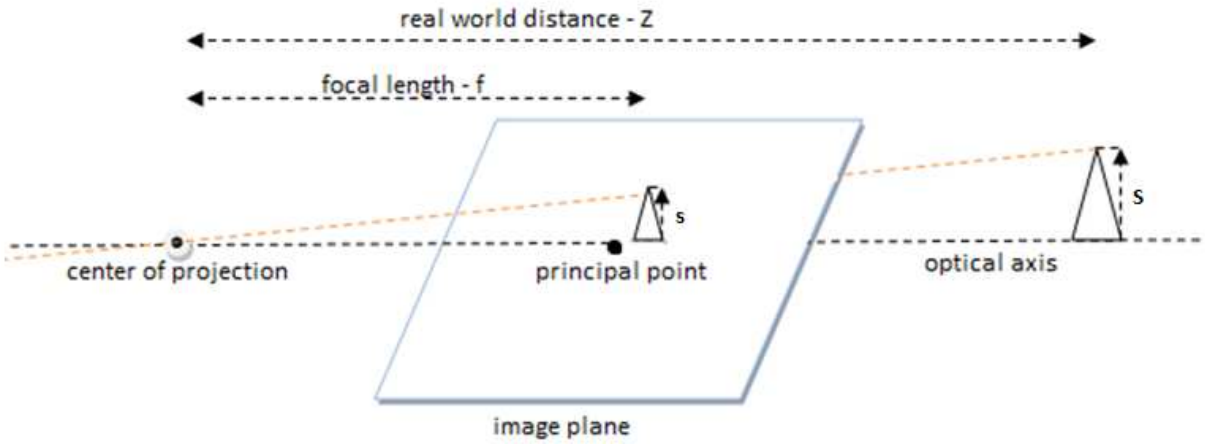

Figure 2.2.2. Simplified approach to pinhole camera mode. Image plane is now located between the object and the center of projection thus rotating the image with height h to an upright position.

$$\frac{S}{Z} = \frac{s}{f} \qquad (2.2.2.)$$

The point in Fig. 2.2.2. when optical axis crosses the image plane is called principal point. In ideal camera it is placed exactly in the center of camera imager chip, which is never true for reality. Thus, in order to be able to calculate real word dimensions of an object or real world coordinates of a point an adjustment has to be made. Two variables can be assumed that represent the shift between the center of camera imager and the principal point in x and y axis: C_x and C_y. Equations 2.2.3. and 2.2.4. present how the real world coordinates *(X,Y)* of a point with image coordinates (x, y) can be calculated given that Z and f are known. It has to be noted that in those equations two different focal lengths are used – f_x and f_y. It is due to the fact that pixels in most cameras are not exactly of square size, they are rather rectangular, thus the focal length will be different for *x* and *y* direction.

$$X_{real} = \frac{x - C_x}{f_x} Z \quad (2.2.3.)$$

$$Y_{real} = \frac{y - C_y}{f_y} Z \quad (2.2.4.)$$

2.2.2 Camera Calibration

To be able to use a camera in any project that includes relating image coordinates to real world coordinates it has to be calibrated. The purpose of this process is to calculate camera parameters crucial to transform the ideal pinhole model to more real mathematical description of how the device works. First of all the parameters that need to be extracted are focal lengths in both x and y direction, and the position of the center of coordinates on the image plane given by C_x and C_y. Those four parameters are called camera intrinsic parameters and are put together in camera matrix M presented in equation 2.2.5. The focal lengths are always calculated in pixels, as are the image center coordinates.

$$M = \begin{bmatrix} f_x & 0 & C_x \\ 0 & f_y & C_y \\ 0 & 0 & 1 \end{bmatrix} \quad (2.2.5.)$$

Another set of parameters that have to be extracted in the process of camera calibration are related with the existence of a lens in the device. Theoretically, manufacturing an almost perfect lens may be possible but in reality this never happens. Thus every lens introduces two types of distortions: radial and tangential. The former is caused by inappropriate shape of the lens and the latter depends on the accuracy of aligning the lens and camera imager chip. Disregarding any one of those distortions will most likely results in large calculation errors unless the distortions are negligible. In the calibration strategy that is assumed in this thesis, five distortion parameters will be extracted to a matrix D shown in equation 2.2.6. The ones that are marked with a letter k correspond to radial distortions whereas those marked with letter p correspond to tangential distortions. Parameter k_3 is placed at the end of the matrix because it is optional and nonzero only if a fish – eye lens is used.

$$D = [k_1 \quad k_2 \quad p_1 \quad p_2 \quad k_3] \tag{2.2.6.}$$

To start the process of camera calibration, a calibration object is needed. Ideally it would be some plane with distinctive and repeatable pattern, that would be easy to locate and trace on the image. In this thesis the most popular calibration object - checkerboard is used. It has a repeatable pattern of black and white squares and corners - places when those squares meet are one of the best features that are very easily locatable by using feature extraction algorithms. Calibration based on 3D object is also possible but much more complicated – it is much harder to created a good 3D calibration object and the results are not better than in 2D case. The calibration is done by taking pictures of the calibration object in different poses – either by moving the object itself or by moving the camera that is trained in the object. Then the homography is calculated to relate (map) the chessboard pattern on the image to its real world counterpart. Assuming that two points are considered, point $p = [x\ y\ 1]^T$ on the imager, and point $P = [X\ Y\ Z\ 1]^T$ on the chessboard plane the homography can be expressed by equation 2.2.7. Matrix W contains translation and rotation matrices that transform the chessboard on the imager to the chessboard on a 2D real world plane. The rotation can occur in three axis, and the translation can be made in three directions which altogether gives six values that have to be calculated with each homography. Matrix M was mentioned before and it contains four values of intrinsic camera parameters that has to be calculated. However it has to be noted that those values are constant for every view of the calibration object whereas matrix W is different every time. It means that for every position in which the chessboard is set, there are 10 unknown values that has to be found. [7] states that the mathematical description of each homography composes of eight equations. This will allow for finding values of 8 unknowns, two of which (intrinsic parameters) there is no need to solve for again, as they are the same for each view. The above means that at least two different views of the chessboard are needed in order to calibrate the camera. However, to get more diversified data, and to lessen errors, much more view are usually taken – between 8 and 25.

$$p = sHP, \tag{2.2.7.}$$

$$where\ H = MW\ and\ s\ is\ an\ arbitrary\ scale\ factor$$

Figure 2.2.3. shows the process of data collection for calibration done during the course of this thesis, visualized in Matlab calibration toolbox [8]. Camera is moved and then trained at stationary chessboard between images, twenty different poses of the calibration object are recorded. Positions of the chessboard corners are then extracted as in Figure 2.2.4. and used for calculating the cameras intrinsic parameters. Extrinsic parameters which are visualized in Figure 2.2.3. are translations and rotations of the chessboard recorded by the calibration software.

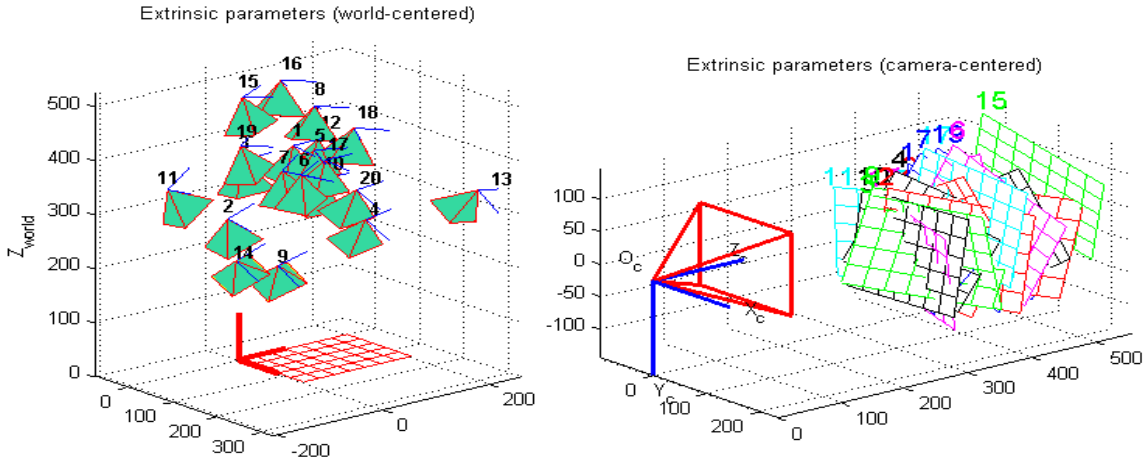

Figure 2.2.3. Visualization of the data collection process, that allows later to extract chessboard corner positions and perform calibration.

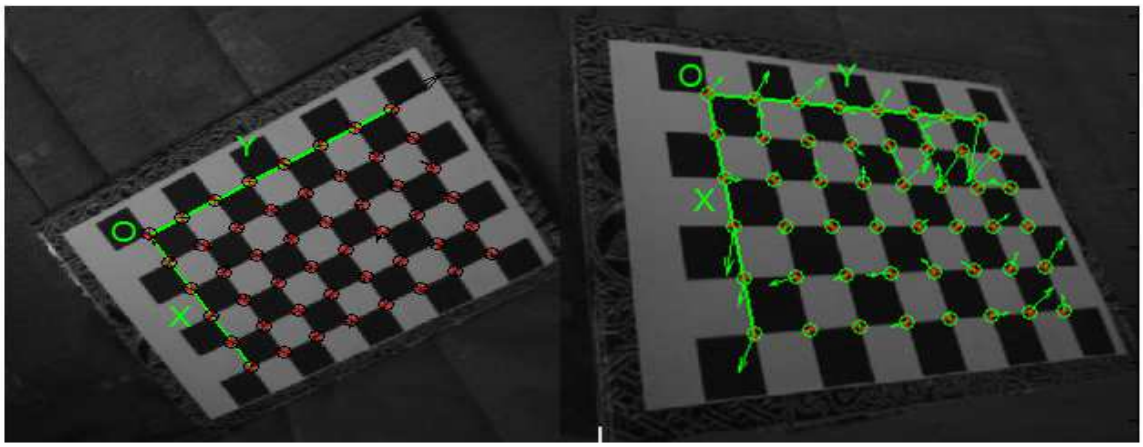

Figure 2.2.4. Marking the extracted corners positions from two different calibration object poses

An example of calibration results is shown in Figure 2.2.5. Four intrinsic and five distortion parameters were calculated. This will allow for removing tangential and radial distortion from every image that is taken by this particular camera. Additionally, those

results can be used to determine real world coordinates of an object which size is known according to equations 2.2.2., 2.2.3. and 2.2.4. However, the known size constraint is a considerable drawback that eliminates the possibility of using just one camera for the purpose of this thesis. A system is needed that can determine the real world coordinates of any point found on the camera imager not only objects that were previously measured.

```
Intrinsic parameters of left camera:

Focal Length:          fc_left = [ 543.24438   544.65995 ] ± [ 2.07149   2.05961 ]
Principal point:       cc_left = [ 312.76672   242.74611 ] ± [ 2.75955   2.09498 ]
Skew:             alpha_c_left = [ 0.00000 ] ± [ 0.00000  ]   => angle of pixel axes = 90.00000 ± 0.00000 degrees
Distortion:            kc_left = [ -0.11862   0.16042   -0.00176   0.00136  0.00000 ] ± [ 0.01104   0.03150   0.00090
```

Figure 2.2.5. Exemplary calibration results for PS3eye camera

2.2.3 Stereo calibration and stereovision

Stereopsis is one of the mechanisms used by animals having a pair of eyes facing the front, to judge distance. It is also widely used in computer vision as a simplest method of depth perception. According to [9] it can be defined as "*Awareness of the relative distances of objects from the observer, by means of binocular vision only and based on retinal disparity*". Disparity in computer vision can be thought of as the difference in angle under which given feature is seen on images taken from two point of views. Stereoscopy using animals (like humans) have their eyes facing the same direction and in parallel setup – they are located on one line that is perpendicular to the direction in which the head is facing. That is the only setup where stereopsis works and that is why cameras positions have to be arranged similarly.

Stereovision uses stereopsis and then triangulation to allow depth estimation with two cameras in a setup called stereo rig - two devices mounted firmly beside each other facing the same direction thus mimicking human vision system. Figure 2.2.6. explains the main idea behind stereovision. The variables C_x^l and C_x^r denote the centers of left and right camera imagers. B is the distance between the midpoint of lenses of both cameras called baseline while the position of the view of object P on the imagers is represented by xl and xr. As previously, f denotes the focal length. In reality no two cameras have the exact same focal length but in the process of stereo calibration described later in this

subchapter the whole stereo rig is adjusted new, common value of this parameter. Equation 2.2.8. shows how the depth (Z) can be calculated from triangulation. Disparity in this formula is calculated in pixels and equals the difference between positions of the image of object P on the left and right imager.

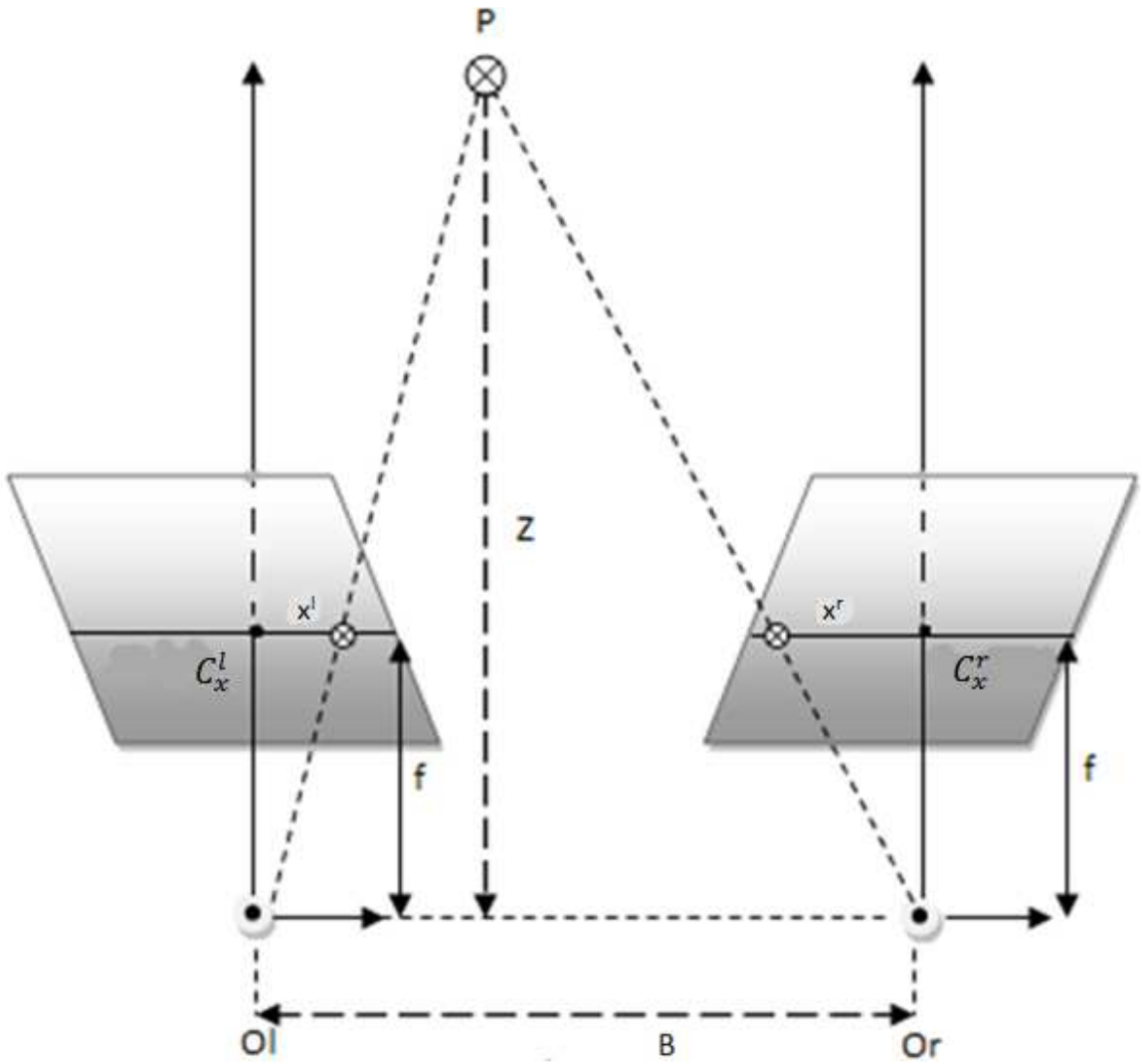

Figure 2.2.6. Illustration of the stereovision principle, by knowing parameters B and f, depth Z can be calculated using similar triangles rule

For the triangle equation to be used, some requirements considering both physical positioning of the cameras in the stereo rig and adjustments of the images taken by the devices have to be made. First of all, the cameras should be as close to horizontal alignment as possible, also the baseline between them should be short so that the cameras would have widest possible common field of view. Figure 2.2.7. shows one of the stereo

rigs that were used for this thesis. As for the requirements for images, there are few steps that have to be completed before the image frames can be effectively processed. After the cameras have been calibrated, the images are undistorted and rectified. Cropping is the last step of this image processing sequence as shown in the diagram from [4] in Figure 2.2.8. The purpose of undistortion is pretty straightforward – the image data would be inaccurate without it. Rectification is the process of aligning the image frames from the left and right camera. It is an extension of the alignment that was performed during the stereo rig setup, although this time it is done mathematically rather than physically. Rectification is needed for the values of x^l and x^r from equation 2.2.8. to be located on the same horizontal line on combined image frames as can be seen on Figure 2.2.9. Only that way the depth can be accurately estimated, otherwise the value of the disparity would always be greater than it should be. Rectification also is important for making sure that both cameras share the same plane, at least virtually – effect that is very hard to achieve by manually adjusting the cameras positions. Cropping is optional but helps in visualizing the common field of view of both cameras.

Figure 2.2.7. An example of a stereo rig setup, right and left camera are aligned horizontally and situated very close to each other.

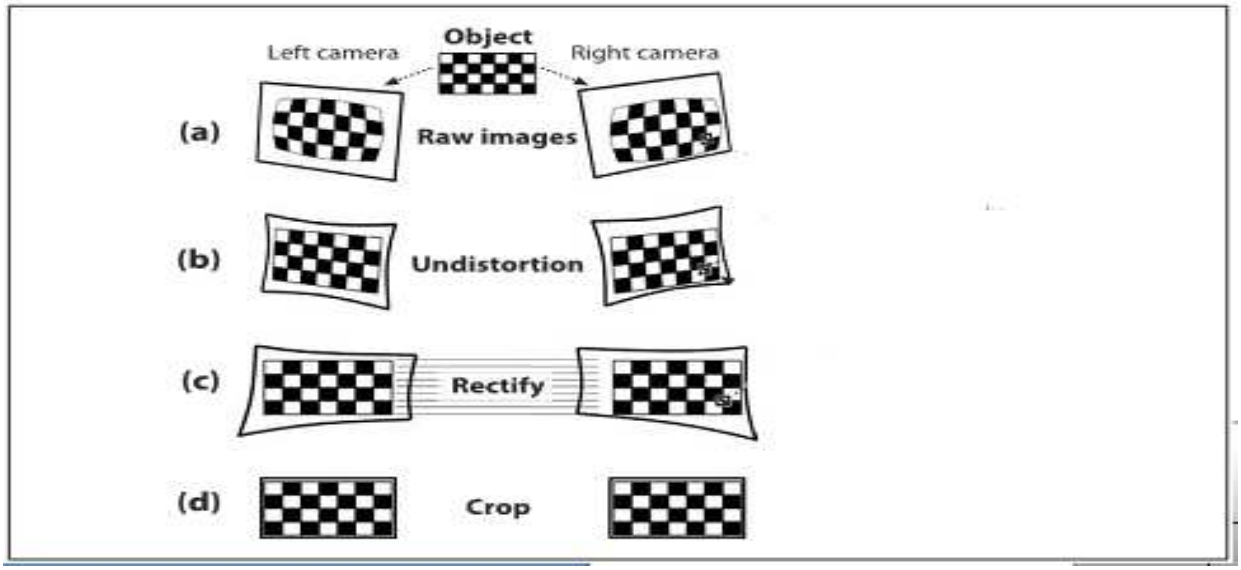

Figure 2.2.8. The frame processing required to use the images from left and right camera for depth and position estimation

Stereo calibration is a procedure quite similar to one camera calibration described previously in this paper. The unknown parameters are the same with only one addition – the distance between the centers of camera lenses *B* also has to be calculated. It is possible of course to measure it with a ruler, but the result will be very inaccurate. Stereo calibration can be done either "from scratch" – meaning that it is performed on cameras that weren't previously calibrated, or it can use the calibration matrices that were calculated during single camera calibrations for left and right camera as a starting point in calculations. During the course of this thesis it was concluded that the former approach yields extremely poor results and only the later approach was implemented. Process of stereo calibration is the same in the physical context as process of one camera calibration, only instead of showing the calibration object to one camera, it is shown to a stereo rig. Position of chessboard corners is marked and extracted by the calibration software only if the whole chessboard is visible for both left and right. The process of data collection for stereo calibration is visualized in Figure 2.2.9. One important thing to note is that when using stereovision both cameras have to take pictures at the same time so that they capture the exact same scene. If this condition is not fulfilled, then the results of calculations will be most likely fallacious. In this thesis this requirement is satisfied by employing multithread programming.

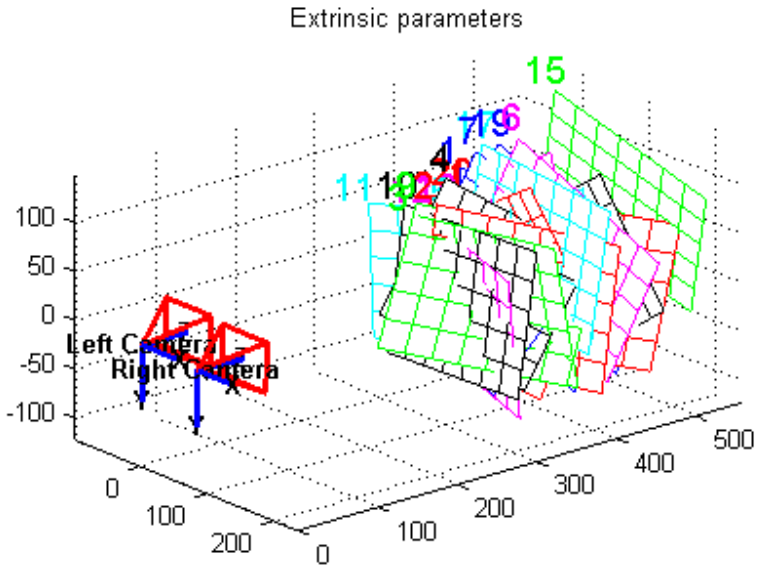

Figure 2.2.9. The process of data collection for stereo calibration visualized in Matlab

After performing all operations from diagram in Figure 2.2.8. the following results are obtained. New camera matrices *Ml* and *Mr* that contain both cameras intrinsic parameters. They generally have similar values to those from one camera calibration but focal length are now set to the same number for both devices – instead of thinking of stereo rig as two separate cameras it can be thought of as one that is capable of judging depth. Additionally, rectification produces four matrices that are later used for every image taken by the stereo rig to rectify them. Lastly but most importantly a reprojection matrix Q is given as in equation 2.2.9. This is the final result that allows to reproject any point that can be found in the stereo rigs field of view from 2D (*x*, *y*) to 3D real world (*X*,*Y*,*Z*) coordinates in a reference frame when cameras are at (0,0,0).

$$Q = \begin{bmatrix} 1 & 0 & 0 & -C_x^l \\ 0 & 1 & 0 & -C_y^l \\ 0 & 0 & 0 & f\,[px] \\ 0 & 0 & -\frac{1}{B} & \frac{(C_x^l - C_x^r)}{B} \end{bmatrix} \qquad (2.2.9.)$$

Now, assuming that a point is found on a frame taken by the left camera from the stereo rig with coordinates *x*, *y* and associated disparity d then equations 2.2.10. and 2.2.11. can be used to find real world coordinates X_{real}, Y_{real}, Z_{real}, of that point. Those formulas

were employed throughout this thesis whenever calculating real world coordinates of features found in the image is needed.

$$Q\begin{bmatrix} x \\ y \\ d \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} \quad (2.2.10.)$$

$$X_{real} = \frac{X}{W}, Y_{real} = \frac{Y}{W}, Z_{real} = \frac{Z}{W} \quad (2.2.11.)$$

It has to be noted that another technique exist called structure from motion that allows for depth calculation without the need of two cameras. In nature it is used by animals that have their eyes on the sides of the heads like squirrels or most birds to estimate distance. The basic principle behind this approach is the same as in the stereovision, including the math. The difference is that instead of having two cameras only one camera is used. After a picture of a scene is taken, the camera is moved horizontally and takes another picture. Those two frames are then used as in stereovision to judge depth by means of disparity. The main disadvantage of structure from motion is that camera has to move very fast or the observed scene has to be stationary for the two frames to be comparable.

The accuracy of depth or position estimation is the last issue to be addressed in this subchapter and more extensively in “Errors and Accuracy” section of this chapter. As was presented in equation 2.2.8. depth estimation is dependent on disparity which is a difference of position of an object in two image frames showing the same scene. Thus according to the formula, greater disparity relates to smaller depth and smaller disparity relates to greater depth as seen on the chart in Figure 2.2.10. This imposes a very serious limitations for depth estimation accuracy. For short distances the estimation is quite accurate, because when disparity is large a mistake of a few pixels does not influences the result as much, for greater distances however, one pixel may actually be the value of disparity so even the smallest error causes the depth to be judged poorly. Another problem of similar origin is low resolution when the distance gets greater which makes long range measurements unreliable – the actual effective range depends on the cameras resolution. For example, on the graph it can be seen that the disparity range from 10 to 40

covers distance range from about 111cm to 444cm whereas disparity range from 5 to 0 covers distance range from about 900cm to infinity.

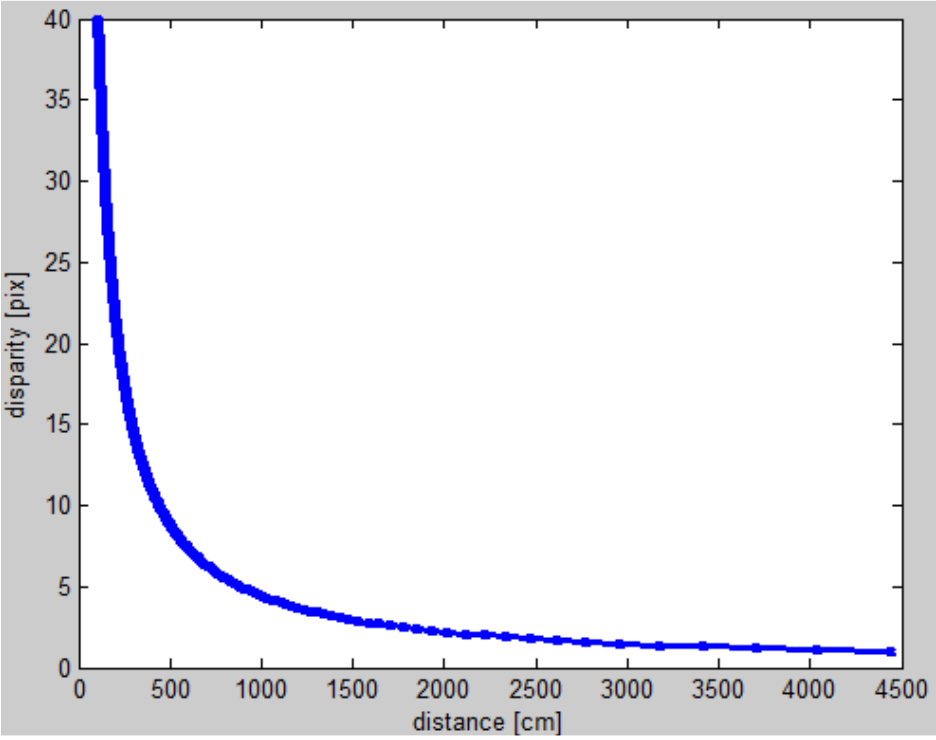

Figure 2.2.10. Disparity versus distance graph for PS3 eye cameras stereo rig

2.3 Hardware specification

Table 1 Camera Comparison

Model	*Price*	*FPS*	*Pixel Size H [μm]*	*Pixel Size V [μm]*	*Res.H [px]*	*Res.V [px]*	*B [cm]*	*f [mm]*	*f [px]*	*minDist [cm]*	*maxDist [m]*	*HFOV [deg]*
PS3 Eye[1]	$30,00	30	-	-	640	480	8,9	-	550	7,65	12,23	60,38
Surveyor SVN	$500,00	15	3,18	3,18	1280	1024	10,8	3,6	1132	9,51	30,42	58,96
Capella	$1 499,00	60	6,00	6,00	752	480	10	3,6	600	7,98	15,00	64,15
BB2-03S2C/M-60	$1 895,00	48	7,40	7,40	648	488	12	6,0	811	15,02	24,32	43,56
BB2-08S2C/M-60	$2 395,00	20	4,65	4,65	1032	776	12	6,0	1290	15,00	38,70	43,59
MEGA-DCS	$1 600,00	7,5	5,00	5,00	1280	960	9	12,0	2400	16,88	54,00	29,86
DeepSea G3 EVS	-	60	6,00	6,00	752	480	14	6,2	1034	19,24	36,17	39,98
PCI nDepth	$3 995,00	30	6,38	7,50	752	480	6	4,3	1034	8,25	15,50	39,98
FL2-14S3M-C[2]	-	15	-	-	1344	391	57,1	-	645	27,40	92,05	92,33

[1] Custom baseline length. Camera parameters obtained through calibration.
[2] Camera calibration parameters provided alongside test stereo video sequence *2010_03_09_drive_0019* by Andreas Geiger [18]

Camera parameters given by a manufacturer usually are: focal length f [mm], imager resolution [resH x resV], pixel size τ [μm], fps and whether sensor captures color or monochrome images.

Because in order to compute distance accurate camera parameters must be known, no auto-focus is allowed since change in focal length introduce the necessity of new calibration. Auto-white balance is not recommended too because features detection and matching is required between images from two cameras and between subsequent frames.

Due to required accuracy, cameras calibration is necessary even when all above parameters are given by a manufacturer. Nevertheless, they allow evaluation of cameras performances before purchasing. Parameters for several cameras are presented in Table 1.

Those parameters with conjunction with baseline (distance between two stereo cameras) are used to compute essential stereo vision parameters: focal length measured in pixels f_{px} [px], field of view (FOV) , minimum and maximum distance (Z_{min} and Z_{max}) using equation 2.3.1, 2.3.2 and 2.3.3 respectively. Z_{min} is a minimal distance at which object will be visible on both cameras, and thus its distance can be triangulated. Z_{max} is defined as distance for a disparity d equal four. Below this disparity level quantification error is too high to provide reliable distance measurement even when using sub-pixel accuracy what is illustrated on figure 2.4.4. Errors are described in more details in section 2.4.

$$f^{[px]} = \frac{f^{[mm]}}{\tau} = \frac{[mm]}{[\frac{mm}{px}]} \quad (2.3.1.)$$

$$HFOV = 2\tan^{-1}\frac{0.5\,resH}{f^{[px]}} = \frac{[px]}{[px]} \quad (2.3.2.)$$

$$Z_{min}[m] = \frac{Bf^{[px]}}{resH} = \frac{[m][px]}{[px]} \quad (2.3.3.)$$

$$Z_{max} \equiv Z(d = 4[px]) = \frac{Bf^{[px]}}{4} = \frac{[m][px]}{[px]} \quad (2.3.4.)$$

Color vs. Monochrome

Digital color cameras generally use a color filter array (CFA) in front of a monochrome sensor e.g. a Bayer filter (four pixels square pattern: one red, one blue, two green) is most common. Majority of green component is due to the fact that a human eye is more sensitive to green than either red or blue thus resulting image looks more natural. Because color imagers receive less light due to filters in front of the sensor, they have less sensitivity than the monochrome ones e.g. a Bayer filter absorbs about 2/3 of the light falling on each pixel.

Moreover, color resolution is lower than the luminance resolution because, in monochrome cameras, luminance information is collected at every pixel whilst, in color camera, only one of three color components is collected by each pixel. Thus direct output image would have much lower resolution than camera sensor. In order to reconstruct an output image of equal resolution to the original sensor resolution, missing color values for each pixel are interpolated using neighboring pixels. As the result, the output image has the original camera resolution with RGB values at each pixel and , in practice, the difference in spatial resolution is not high enough to be a important factor. Mostly it will show on the edges, where color aliasing will occur [10]. Thus only light sensitivity is concerned, and when low lighting levels are expected and color information are not essential, monochrome cameras are recommended.

2.4 Errors and Accuracy

Two groups of distance estimation errors can be distinguish. In first group there are errors caused by inaccurately calibrated camera parameters. There are three such parameters: baseline, focal length and center of imager. We will refer to distance estimation errors caused by this parameters as baseline error $e_B(\delta_B)$, focal length error $e_f(\delta_f)$ and center error $e_c(\delta_c)$. δ_B, δ_f and δ_c expresses calibration accuracy, this is difference between respective parameter's calibration result and its true value as denoted by equation 2.4.1, 2.4.2 and 2.4.3

$$\delta_B = \widetilde{B} - B \tag{2.4.1.}$$

$$\delta_f = \tilde{f} - f \tag{2.4.2.}$$

$$\delta_c = \widetilde{\Delta c} - \Delta c \tag{2.4.3.}$$

Baseline error e_B is proportional to baseline calibration error δ_B while it decreases with baseline value itself as equation 2.4.4 states. Because baseline value *B* can be easily measured with accuracy of 1 mm, baseline error is less significant when baseline value *B* is large. This is shown on figures 2.4.1, 2.4.2 and 2.4.3 on which estimation errors can be compared.

$$e_B(\delta_B, Z) = \frac{\delta_B}{B} Z \tag{2.4.4.}$$

Focal length error e_f is expressed by equation 2.2.5, similar to that of baseline, but focal length value cannot be measured directly and so its accuracy is not so assured as in baseline case.

$$e_f(\delta_f, Z) = \frac{\delta_f}{f} Z \tag{2.4.5}$$

Center error e_c is an error caused by inaccurately calibrated coordinates of centers of left and right imager. Equation 2.4.6 shows exact formula for this error but if distance values are not very large $\delta_c Z \ll Bf$ then it can be approximated as $e_c(\delta_c, Z) \sim \frac{\delta_c}{Bf} Z^2$. High Bf coefficient causes this error to be very small for small distance values but it increases

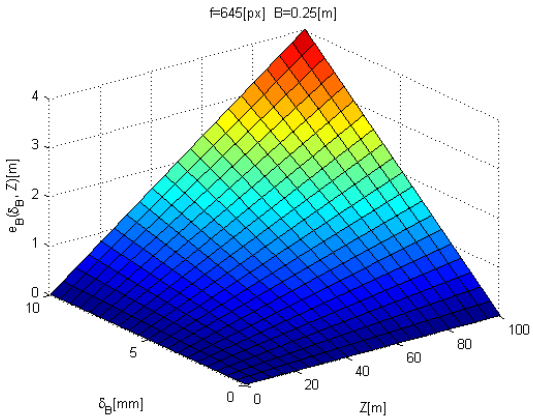

Figure 2.4.1 Baseline Error

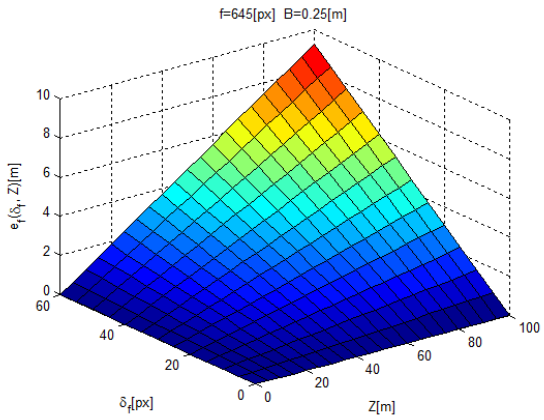

Figure 2.4.2 Focal Length Error

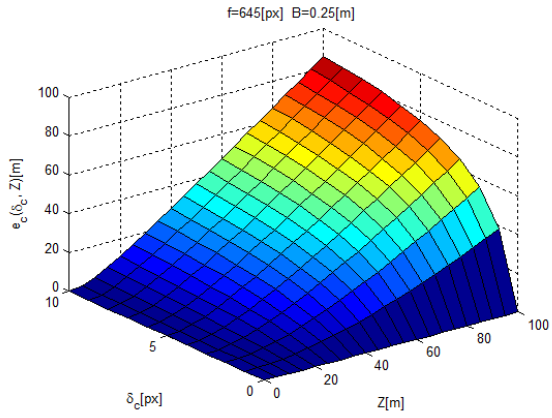

Figure 2.4.3 Center Error

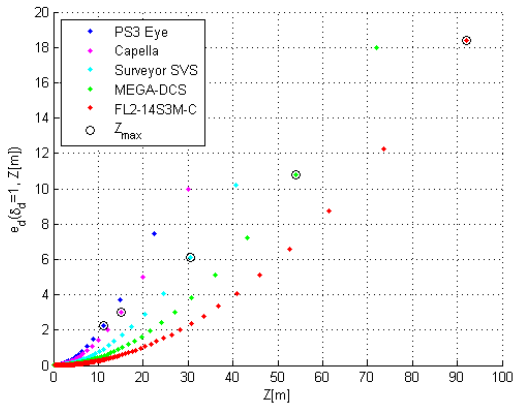

Figure 2.4.4 Quantization Error Comparison

rapidly as distance grows due to being proportional to squared distance. Center error impact is significant, especially when distance for far features is evaluated. Thus calibration of imagers centers must be very accurate to obtain reliable distance estimation.

$$e_c(\delta_c, Z) = \frac{\delta_c Z^2}{Bf + \delta_c Z} \qquad (2.4.6.)$$

Second group of errors is composed of errors caused by inaccurate disparity measurement. We will call it a disparity error $e_d(\delta_d)$ According to equation 2.4.7 it has similar form as center error e_c and thus can be approximated as $e_d(\delta_d, Z) \sim \frac{\delta_d}{Bf} Z^2$ if $\delta_d Z \ll Bf$. As mentioned above, error expressed by such equation increases rapidly as distance grows. Thus accurate feature detector must be used. Moreover, if feature

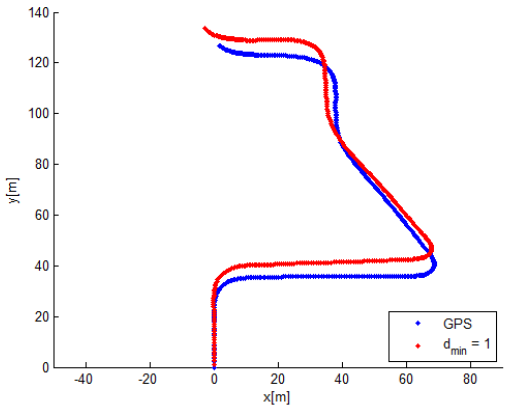

Figure 2.4.5 Position Estimation - all points used

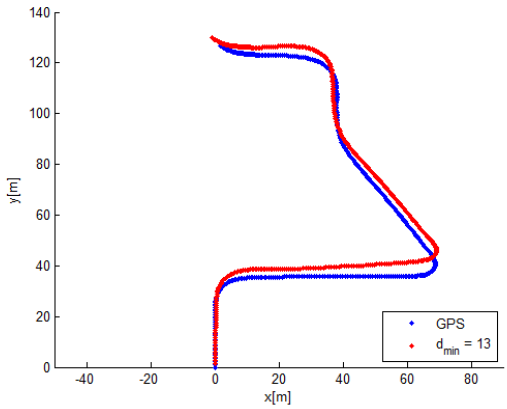

Figure 2.4.6 Position Estimation with accuracy constrain

detector do not implement sub-pixel accuracy, then disparity accuracy is limited to $\delta_d = 1\,pixel$. We will call disparity error for such a case a quantization error $e_c(Z)$ because it is caused by discrete nature of digital image. Quantization error is expressed by equation 2.4.8 and if $Z \ll Bf$ then it can be approximated as $e_q(Z) \sim \frac{Z^2}{Bf}$. Quantization error poses a serious limitation for distance estimation for distant features as is shown on figure 2.4.4 where quantization errors for several cameras from Table 1 are presented. If pixel-accurate feature detector is used then better results can be achieved if distant features will be discarded. figures 2.4.5 and 2.4.6 presents estimated vehicle trajectory compared with GPS data, figure 2.4.5 shows a case when all features are used to estimate camera pose whilst figure 2.4.6 shows the same trajectory estimated using only features that meet a constrain d > 13 pixels. Minimum disparity of 13 pixels was chosen basing on figure 2.4.4 and requirement that quantization error cannot be greater than 2 meters.

$$e_d(\delta_d, Z) = \frac{\delta_d Z^2}{Bf + \delta_d Z} \tag{2.4.7}$$

$$e_q(Z) = \frac{Z^2}{Bf + Z} \tag{2.4.8}$$

3. Ground Based Positioning System

Ground Based Positioning System provides absolute position of UAV in reference to system's placement and impose no restriction for size and weight of the system. Moreover, system can be connected directly to PC and use its computation power which is far greater than one available on UAV. Unfortunately, Ground Based Positioning System have very limited range of operation because accuracy drops very rapidly with the distance. Detailed information about errors are presented in section 2.4. Thus its application is restricted for short range operation, when absolute position information is essential, e.g. landing maneuver, and for such task system described here was designed.

UAV's appearance can vary greatly with distance and viewing angle thus designed system is detecting markers mounted on UAV instead vehicle itself.

Because ball has the same appearance from every angle and its center can be easily evaluated, this shape was chosen for a marker. Moreover, round marker detector can be easily designed to be robust to partial occlusion of a marker. Designed marker emits its own light in order to assure robust operation in case of poor lightning conditions. Color of the marker can be adjusted depending on environment. For outdoor operation in natural environment red markers are recommended due to a large amount of green and blue color in background caused by plants and sky. For indoor operation red color is discouraged due to a usually large amount of red component in artificial lighting.

Exposure time and gain of cameras for ground system is adjusted to achieve robust marker detection. Because designed marker emits its own light its essential to avoid saturation of image sensor, thus gain is set to minimal value. Exposure time is set at moderate level to assure clear marker registration whilst avoiding image blur caused by motion of a marker. Figure 3.1 and Figure 3.2 show marker detection in case of poor lightning condition. As a result of very low gain, background is hardly visible in this case. As shown on Figure 3.2c, proposed solution is robust to partial occlusion of a marker.

Marker is detected using color models of marker and background. YUV color space was chosen because its luma (Y) component is already separated and can be discarded, making model less sensitive for changing lighting conditions [11].

Figure 3.1a Registered image of a marker

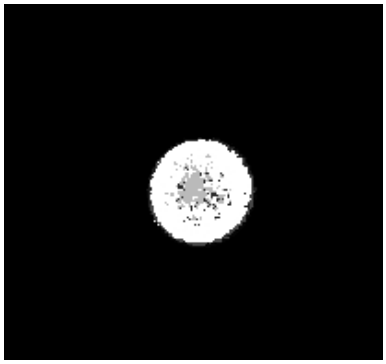

Figure 3.1b Back Projection for marker $\mathrm{P(m|v = V)}$

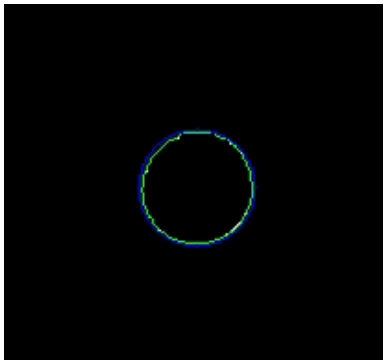

Figure 3.1c Detected position of a marker

Figure 3.2a Registered image of occluded marker

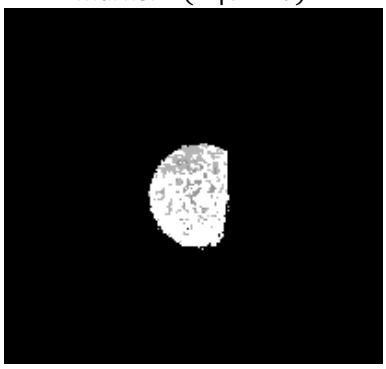

Figure 3.2b Back Projection for occluded marker $\mathrm{P(m|v = V)}$

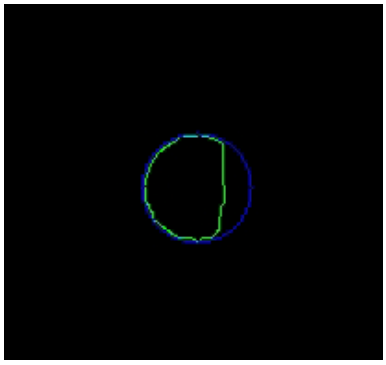

Figure 3.2c Detected position of occluded marker

Before starting normal operation, system learns marker and background probability models. We will denote $pixel\ is\ marker \equiv m$, $pixel\ is\ backgound \equiv b$, $pixel\ value \equiv v$. In order to eliminate camera noise, several frames are collected and pixels values are averaged. Those averaged pixels values populate histogram. Next, Probability Mass Functions $P(v = V|m)$ and $P(v = V|b)$ are inferred from histograms. Using Bayes' theorem Probability Mass Functions $P(m|v = V)$, i.e. probability that pixel with value V belongs to the marker, can be evaluated (equations 3.1 and 3.2). Probability $P(m)$ is assumed to be constant because marker can be placed on any part of the image with equal probability. Back Projection (BP) of the background image and Probability Mass Function $P(m|v = V)$ produce Threshold Map (TM), each pixel has assigned similarity level to marker that must be exceed in order to count this pixel as belonging to the marker. Figure 3.1 presents algorithm that uses pmf $P(m|v = V)$ and TM to find markers position in subsequent frames.

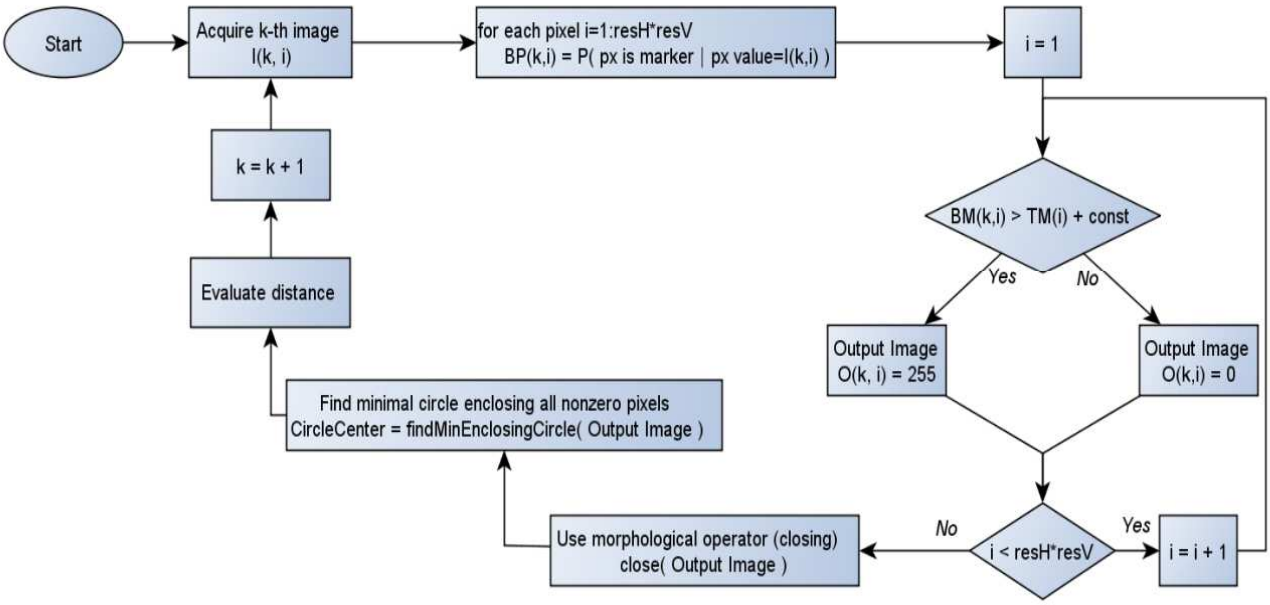

Figure 3.3 Ground Based Positioning System Algorithm

$$P(m|v = V) = P(v = V|m)\frac{P(m)}{P(v=V)} \tag{3.1.}$$

$$P(v = V|m) + P(v = V|b) = P(v = V) \tag{3.2.}$$

$$P(m) = const \tag{3.3.}$$

Knowing the marker's center, its distance can be computed in two ways. Using single camera and knowledge about size of the marker (equation 2.2.2) or using two cameras and triangulate distance (equation 2.2.7). Second method was chosen because it provides far better accuracy because possible baseline length is far greater then size of the marker. Detailed information about errors are presented in section 2.4.

4. Second system – air tracking

4.1 Feature detectors and matching.

Feature in the context of this paper can be defined as a small, distinguishable element of a scene that camera has recorded. They are used for various purposes in computer vision, including mainly position estimation and collision detection. As for the former, if three features are tracked from one frame to another, then the displacement of the camera between those two frames can be easily calculated as it is described in subchapter 4.2. of the thesis. In context of collision detection, high quality features are desirable because they increase the probability of an obstacle being detected. The quality of feature is a measure of how distinctive and easy to locate is it, but also how similarly it looks from different points of views. A large amount of feature detector algorithms exist an each of them is design to locate specific kind of features. In the course of this thesis two feature types were used: corners and blobs. Corner can be described as a point where two edges are intersecting in the image or more accurately, a point where a large gradient of intensity change can be observed in two directions. Corner has this advantage over an edge (gradient of intensity changes only in one direction) that it is much more distinctive and can be tracked from one image to another, whereas an edge can produce many identical feature points which cannot be distinguished from each other. The other popular type of feature as mentioned previously is a blob. This point of interest can be defined as an element of the image that is significantly different from its environment, either in intensity or in brightness.

The choice whether to detect corners or blobs is determined by the environment the UAV is flying in and its predicted average altitude. If the aircraft is flying so high for most of the time, that the terrain can be considered a 2D plane then both types of feature detection techniques are valid and the choice is made based on speed and computational complexity of the algorithm. However if a 3D positioning case is considered e.g. the UAV is flying on a very low altitudes or the environment is heavily industrialized like a big city with a lot of high buildings then the corner detection is the best choice. Two feature detector algorithms

that were used in this thesis are ORB (Oriented FAST and Rotated BRIEF) feature detector [12] and Shi - Tomasi feature detector [13] . Both of them are corner detectors but are focused on other qualities.

ORB feature detector is a newly modified version of FAST (Features from Accelerated Segment Test) algorithm. The modifications allow to deal with main problems of the original detector which were: lack of orientation indicator of detected feature and realatively low quality of detected features. Using ORB feature detector can be described in four main steps described in [7] and summarized below. After those operations features are recorded and their characteristics stored in specific descriptor – class that describes parameters of interest points. Descriptors will be spoken of more extensively in the later parts of this subchapter.

- **step 1: Using FAST feature detector to recognize N features on the image.**

FAST is an algorithm used mainly in real time applications because of its low computational complexity which makes it run very quick on modern hardware. In this approach, pixel in the image considered a valid feature when intensity of surrounding pixels located on a circle with defined radius differ significantly from its own. The level of that difference is specified by the sole parameter of FAST – threshold. The higher the threshold, the smaller number of features will be detected. Their quality will of course be better than in the case with lower threshold so it is a decision of quality over quantity. In the FAST implementation that is used in ORB, the radius of a circle at center of which the point of interest is located equals 9 pixels. The principle of how FAST works is shown in Figure 4.1.1. and described below.

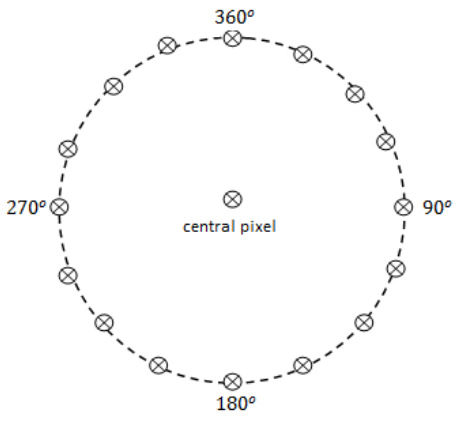

Figure 4.1.1. Pixels that are considered while determining whether given point of interest can be a valid feature

The algorithm will record the central pixel as a valid feature if an arc of length at least equal to the ¾ of circles circumference can be found, on which all pixels have greater or smaller intensity than the central pixel. It means that to discard pixel as a non-feature, only four pixels have to be checked for that quality: the ones located at 90, 180, 270 and 360 degrees. If at least two of them do not differ from the central pixels it cannot be recorder as a valid feature.

- **step 2: Sorting detected features according to Harris corner measure.**

Harris corner measure [14] is a mean of classifying a quality of a corner based on computing the eigenvalues of Harris matrix – a 2D structure tensor defined as H_2 in equation 4.1.1. In this formula I_x and I_y are partial derivatives of a given 2D grayscale image I while u and v are dimensions of a considered image patch that include the potential corner. The *w(u, v)* is a weighting function

$$H_2 = \sum_u \sum_v w(u,v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \langle I_x^2 \rangle & \langle I_x I_y \rangle \\ \langle I_x I_y \rangle & \langle I_y^2 \rangle \end{bmatrix} \quad (4.1.1.)$$

If we assume the eigenvalues of the Harris matrix as α_1 and α_2 then the Harris corner measure λ can be calculated as in equation 4.1.2. Variable w in this formula is a scale factor.

$$\lambda = \alpha_1 \alpha_2 - w(\alpha_1 + \alpha_2)^2 \quad (4.1.2.)$$

- **step 3: Choosing K best features where K<N.**

The desired number of features from sorted data set is chosen for further processing. This and the previous step are a guarantee that only best quality features from an image will be collected which is not always true for original FAST implementation.

- **step 4: Determining orientation component of detected features**

As mentioned previously, FAST features do not posses information about orientation. In ORB this information is acquired by calculating angle between center of an image patch and its *intensity centroid* [12]**.** If a moment of an image patch is defined by equation 4.1.3. [12], then the orientation of a corner - θ can be calculated as in equation 4.1.4. [12].

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y) \tag{4.1.3.}$$

$$\theta = atan2(m_{01}, m_{10}) \tag{4.1.4.}$$

Another feature detector used during the course of this thesis was developed by Shi-Tomasi. It is a modification of a Harris corner detector. According to Harris a corner is present when both eigenvalues of matrix H from equation 4.1.1. have high values - it means that intensity change is observed in both x and y directions. Shi-Tomasi has proven to provide at least as good results as its predecessor. In this approach a given corner is considered a good feature when smaller of the eigenvalues extracted from Harris matrix in equation 4.1.1. is greater than a specific quality threshold. This relationship can be expressed as in equation 4.1.5. where t is the threshold. Value of t is not constant and varies from image to image. It depends greatly on the quality of the best corner in the image – the one with the biggest $\min(\alpha_1, \alpha_2)$. This means that the Shi-Tomasi feature detector is somewhat adapting to the quality of corners in a given image. In the implementation of this algorithm used in the thesis, threshold value is set as a fraction of the best corners quality and can assume a value between zero and one. Figures 4.1.2. and 4.1.3. show the best 100 features found in the same aerial image by Orb and Shi-Tomasi feature detectors.

$$\min(\alpha_1, \alpha_2) > t \tag{4.1.5.}$$

Figure 4.1.2. One hundred best features detected in an aerial image by ORB feature detector

Figure 4.1.3. One hundred best features detected in an aerial image by Shi-Tomasi feature detector

There is an obvious difference between those figures – a number of different points has been selected by those two algorithms. This happens from the following reason: although both feature detectors use more or less the same type of estimation of corners quality based on Harris matrix, in ORB features are detected by FAST algorithm, and then evaluated while in Shi-Tomasi, the process of detection itself is based on Harris matrix eigenvalues

In UAV positioning problem feature detection is the first step to designing a successful algorithm. To be able to estimate the change in the UAV position, tracking of detected features from frame to frame has to be performed. In order to do this, a sufficient number of features from previous image frame have to be located in the next image frame. The process is called feature or keypoint matching. To describe it lets assume that two consecutive image frames are available taken at high frame per second rate by a camera on a moving UAV. After some features have been detected in the first image frame, their characteristics e.g. location, orientation, size and quality are obtained and stored by descriptor extractor algorithm which type varies depending on the type of feature detector used. In this thesis BRIEF [7] descriptor extractor is used for both ORB and Shi-Tomasi feature detector. It is a simple binary descriptor, that has proven to be relatively unaffected by changes in lighting, image blurring and perspective distortions. Additionally, it works very fast compared to other

similarly efficient algorithms. When characteristics of features from the first image frame are extracted, the same procedure is applied to the second image frame – keypoint detection and description extraction. Then, matcher algorithm is applied to two set of keypoints from two considered frames. It uses features characteristics to determine which of those from first image frame match those from second image frame. The output of a matcher algorithm is a point from one frame that have similar descriptions to specific point in the other. Matcher algorithm employed in the thesis is based on comparing vectors which contain characteristics of one feature each. Geometric sum of the elements of each such vector is compared and then the list of matching features is created. Figure 4.1.4. presents two consecutive aerial image frames with marked features. The lines in the picture connect each feature in one frame to its best match in the other frame. In ideal situation, when all matches would be correct, all lines would be parallel which is not the case. The reason is that the Figure 4.1.4. was to illustrate an extreme example of incorrect matching. That is why hard to process image frames were chosen which do not contain any distinct features – it is simply a field of grass. Additionally the matching process was carried out on color pictures, which yields much worse results than when dealing with grayscale images. However, bad matching is a great problem also in standard situations, where relatively feature-rich image frames are obtained during UAVs flight. Even a couple

Figure 4.1.4. Visualization of the output of a feature matching algorithm – unsatisfactionary results

of incorrectly matched points can cause a significant difficulty in correctly estimating the aircrafts position. Therefore an approach based on robust matcher algorithm proposed in [15] is used in the thesis to ensure correct matching even in the hardest situations. The procedure has three steps that are explained below:

- **step 1: Performing the ratio test**

Features are detected and matched as before, but the matcher algorithm chooses two points that best match the specific point in the other frame instead just one. Then it compares the geometric sums of vectors with those points characteristics. If the difference between them is lower that a specific threshold, it means that this is an ambiguous case and those features should be rejected as unreliable matching material. The thought process behind it is that there should be only one strong and clear match for each keypoint. If there are two good matches which quality does not differ significantly then granted that the matcher algorithm is itself very simple, there is no guarantee that the right match will be picked. Thus after this step, each feature that does not fulfill the one strong match requirement is rejected.

- **step 2: Symmetrical matching test**

In this very important step, the algorithm verifies whether when one keypoint is a best match for other keypoint, the other keypoint is also the best match for this keypoint. Thus, basically the matching between two features is valid and accepted only when those features are best matches for each other. If such situation does not occur, it clearly signalizes an incorrect match, because it is not unequivocal.

- **step 3: Epipolar constraint and RANSAC**

When two views of the same scene are considered, there exists a fundamental matrix F that relates screen position of any point from the first view to its screen position from the second view. To simplify, it defines where to look for a point from one image frame in the other image frame that depicts the same scene. Fundamental matrix is the essential part of the epipolar constraint presented in equation 4.4.6., where p is a 3x1 vector of screen coordinates $(x,y,1)$ of specific feature in one image frame and p' contains coordinates of the same feature in the other image frame. The main idea behind the epipolar constraint is as follows: given two images

presenting similar scene, if an image of a keypoint with coordinates p is located in one image frame, than the image of the same keypoint in the other image frame has to lie on a line Fp. The size of the fundamental matrix is 3x3 and the Fp, called the *epipolar line* is represented as a 3x1 vector $[l_1\ l_2\ l_3]$ such that $l_1x + l_2y + l_3 = 0$. Thus the task of F is mapping a point from one image into a line in the other image.

$$p'^{t}Fp = 0 \qquad (4.1.5.)$$

The usefulness of the epipolar constraint in matching process is enormous. It allows for very robust verification of the correctness of matching, and radically decreases the area in which the algorithm searches for a match, thus greatly increasing the speed of the whole process. This tool in connection with the previous two steps assures that only good matches will be accepted. However there exist a major difficulty when using the epipolar constraint in this case. The reason is that at least eight good matches are needed to calculate the fundamental matrix *F*, which is the problem because in this algorithm the last step is using the fundamental matrix to check whether the matches are good. The solution is the RANSAC (Random Sample Consensus) algorithm that allows for achieving correct results while basing on data that contains some incorrect samples. First, it is assumed that there are more correct samples than incorrect ones. This assumption can be clearly made in the case of this matching algorithm, because of the two previous filtering steps. Then, the fundamental matrix is calculated based on eight randomly chosen matches. Lastly, it is verified, how many of the overall matches fulfills the epipolar constraint imposed by this specific matrix. Then the whole process is repeated a number of times, and the fundamental matrix with greatest number of "supporting" matches is chosen. This matrix is then used to verify all the matches in the epipolar constraint test. Figure 4.1.5. presents the effect of applying robust matcher algorithm to two image aerial image frames from figure 4.1.4. White lines on the image connect correctly matched points, They are all close to being parallel and have approximately the same length which are characteristics for good feature matching. The feature detection algorithm was set to find 500 keypoints in both images only 40 of which survived all three steps of the matching process. Using three step filtration of image feature matches creates reliable set of data on which further processing can be carried out without having to worry about possible consequences of dealing with incorrect samples. It works well in even the worst case scenario – on image frames that do not possess a large number of distinctive features.

Figure 4.1.5. Visualization of the output of a robust feature matching algorithm – correct results

4.2 Onboard positioning system

Visual Odometry

Visual Odometry is the process of estimating the position and orientation of the vehicle using data acquired from camera images. Visual Odometry estimates vehicle's frame-to-frame motion and, through integration of this motion, obtain vehicle's position in reference to its starting location. Visual Odometry, like all kinds of odometry, is sensitive to integration error (motion-drift error) thus very careful camera calibration and accurate feature detection is required for Visual Odometry to be used effectively.

The motion estimation problem can be formulated as follows. Two subsequent camera positions are related by the transformation $T_k \in \mathbb{R}^{4x4}$ of the following form:

$$T_k = \begin{bmatrix} R_k & t_k \\ 0 & 1 \end{bmatrix} \tag{4.2.1}$$

where $R_k \in SO(3)$ is the rotation matrix, and $t_k \in \mathbb{R}^{3x1}$ is the translation vector. Camera pose is defined by means of its rotation and translation in reference to its position and orientation at starting point. Current camera pose C_k is a result of integrating subsequent rotations and translations, thus can be expressed in iterative form as:

$$C_k = C_{k-1} T_k \tag{4.2.2}$$

In order to compute transformation T_k, set of corresponding features between previous and current frame must be found. To achieve this goal, two different approaches can be used [16]:

- First one is features tracking. Features are found in one frame and then, using local search techniques (e.g. correlation), the same features are searched for in the next frame. This method is more suited for low-scale environment where frame-to-frame motion is small.
- Second one is to find the set of good features in each image separately and match them with each other. The matching is performed by comparison of similarity between features descriptors. This topic was described in details in section

4.1 Feature detectors and matching. In this approach, images taken from distant viewpoints can be used and so motion-drift error can be reduced. Thus this method is superior over first one for large-scale environment and was used in this thesis to design UAV's positioning system.

Low altitude subsystem – Front-Facing Stereo Camera

In order to determine transformation T_k stereo camera and 3D-to-2D correspondences algorithm was used. 3D-to-2D algorithm estimates motion from 3D points, triangulated using stereo images taken in instant *k-1*, and corresponding features in image taken in instant *k*. Equation 4.2.3 presents relation between transformation T_k and 3D points and their corresponding 2D points.

$$MT_k \begin{bmatrix} X_{k-1} \\ Y_{k-1} \\ Z_{k-1} \\ 1 \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ w_k \end{bmatrix} \tag{4.2.3}$$

Problem of determining position and orientation of the camera, given N 3D points P^i known and their 2D projections p^i onto the image, is called Perspective from n Points (PnP). In case of transformation T_k problem can be formulated as finding transformation T_k that minimizes the image reprojection error [17]:

$$\underset{T_k}{arg\ min} \sum_i^N \left\| p_k^i - \tilde{p}_{k-1}^i \right\| \tag{4.2.4}$$

where $\tilde{p}_{k-1}^i$ is reprojection of 3-D point P_{k-1}^i using the transformation T_k as follows:

$$MT_k P_{k-1}^i = \tilde{p}_{k-1}^i \tag{4.2.5}$$

At present day, there are many different solutions of PnP problem. Minimal case requires three 3D-to-2D correspondences and is called Perspective from three Points (P3P). In conjunction with RANSAC, Perspective from three Points is a standard method for 3D-to-2D algorithm in the presence of outliers.

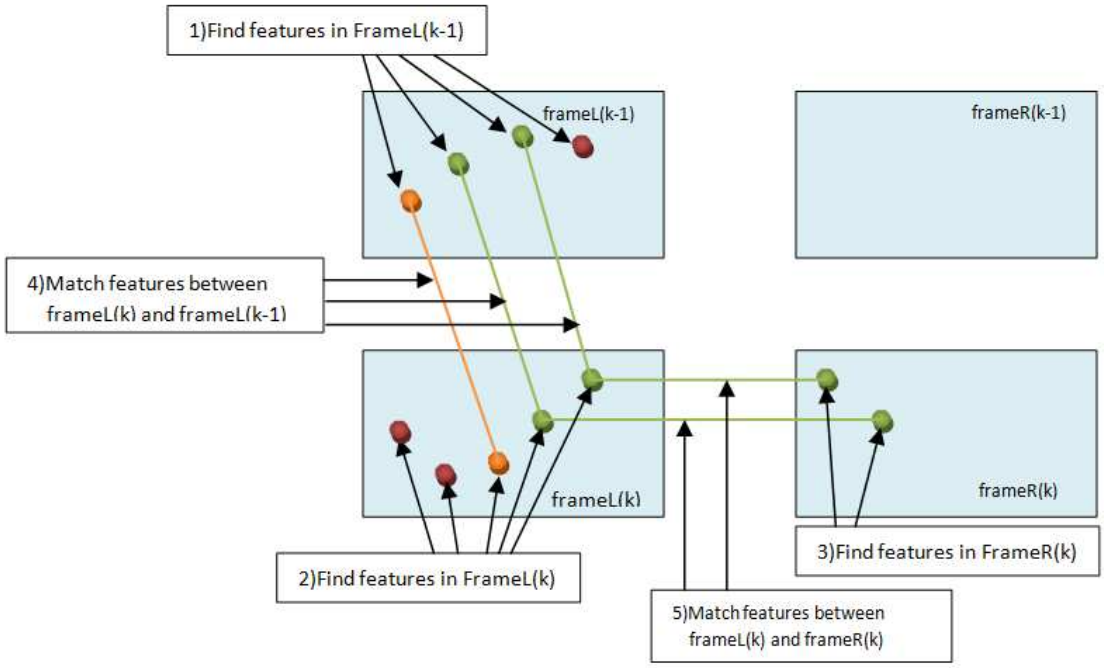

Figure 4.2.1 Perspective from three Points: 3D-to-2D correspondences

Algorithm for determining camera pose at instant k can be described as follows:

1. Find features in frameL(k-1)
   1.1. Detect Keypoints:
   keypointsL(k-1,:) = detectKeypoints(frameL(k-1))
   1.2. Extract descriptors:
   descriptorsL(k-1,:) = extractDescriptors(keypointsL(k-1, :))
2. Find features in frameL(k)
   2.1. Detect Keypoints:
   keypointsL(k,:) = detectKeypoints(frameL(k))
   2.2. Extract descriptors:
   descriptorsL(k,:) = extractDescriptors(keypointsL(k, :))
3. Find features in frameR(k)
   3.1. Detect Keypoints:
   keypointsR(k,:) = detectKeypoints(frameR(k))
   3.2. Extract descriptors:
   descriptorsR(k,:) = extractDescriptors(keypointsR(k, :))

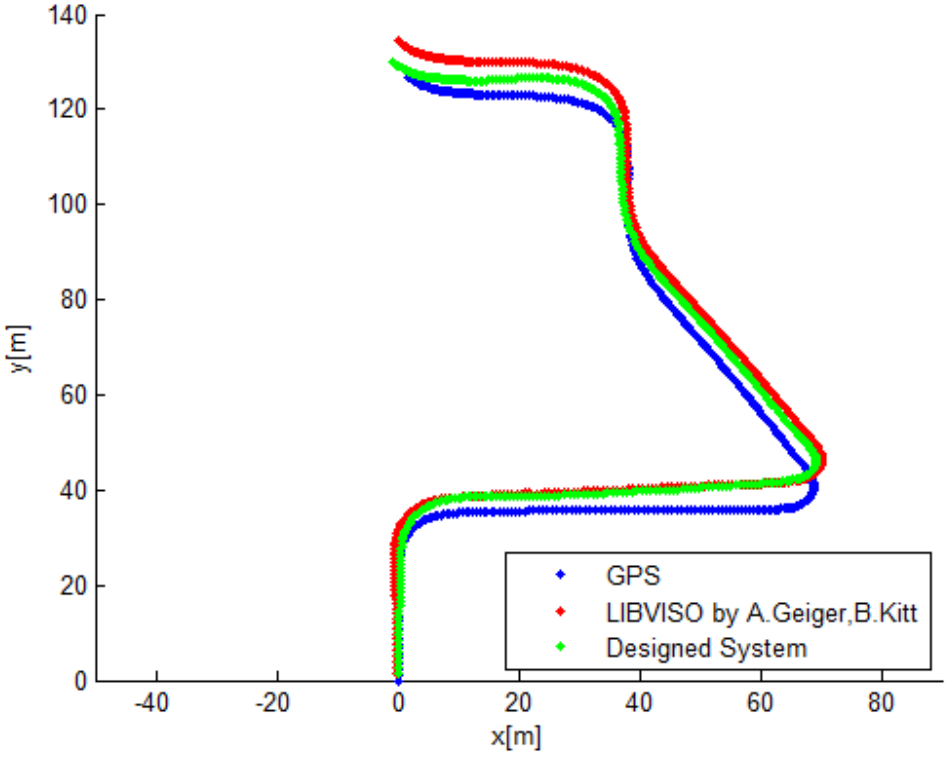

Figure 4.2.2 Position Estimation vs. GPS

4. Match features between frameL(k) and frameL(k-1)

   matchesDescriptorsCurrentLPreviousL = matchDescriptors(descriptorsL(k,:), descriptorsL(k-1,:))

   matchedDescriptorsCurrentL(k,:) = matchesCurrentLPreviousL(1,:)

   matchedDescriptorsPreviousL(k,:) = matchesCurrentLPreviousL(2,:)

5. Match features between frameL(k) and frameR(k)

   5.1. Take only features that have corresponding (matching) feature in frameL(k-1). On Figure 4.2.1 those are represented by green and orange dots.

   matchedDescriptors = matchedDescriptorsCurrentL(k,:)

   5.2. Search for matching features in frameR(k)

   matchesCurrentLCurrentR = matchDescriptors(matchedDescriptors, descriptorsR(k,:))

6. Triangulate 3D points for instant k from left-to-right correspondences

   points3D = triangulate(matchesCurrentLCurrentR)

7. Estimate Camera Pose from 3D-to-2D correspondence

Points that are used to obtained set of 3D points and corresponding 2D points are shown on Figure 4.2.1 as green dots. For more details concerning feature detection and feature matching see Chapter 4.1 Feature detectors and matching. Figure 4.2.3 presents implementation of 3D-to-2D algorithm. Figure 4.2.2 present estimated position of the vehicle compared with GPS data and results obtained using LIBVISO library[18]. In both cases test stereo video sequence 2010_03_09_drive_0019 [19] was used.

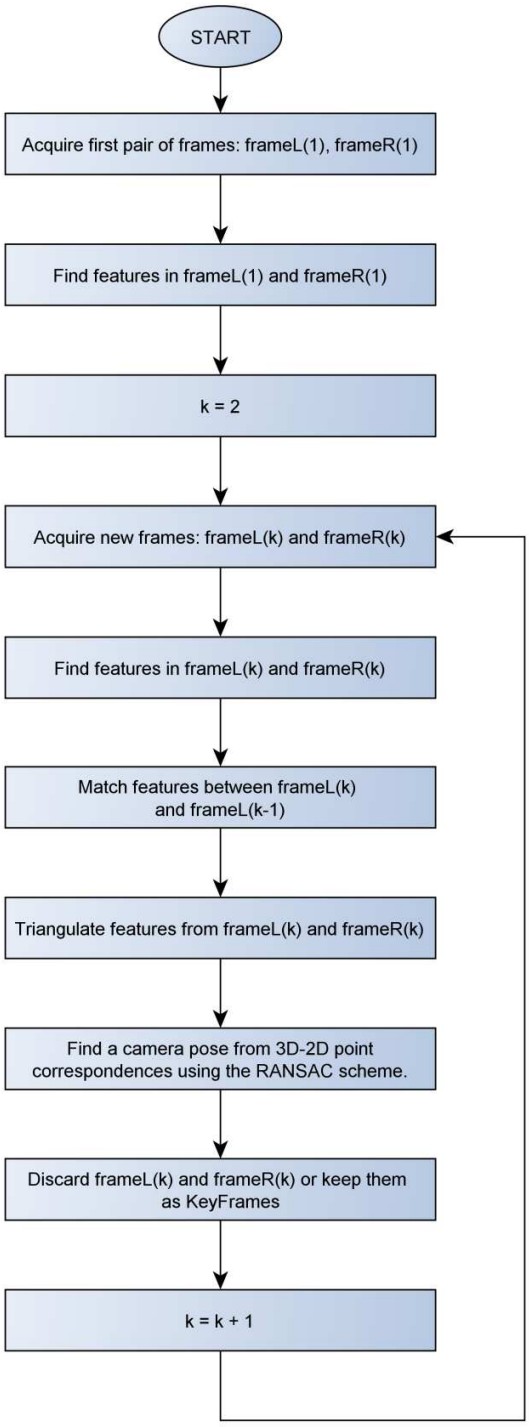

Figure 4.2.3 Low altitude subsystem algorithm

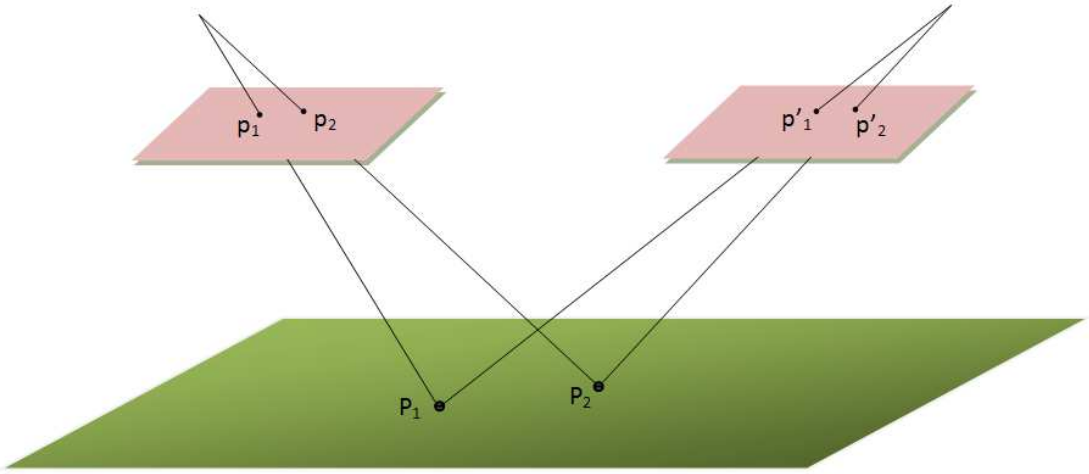

Figure 4.2.4 Homography

High altitude subsystem – Down-Facing Mono Camera

If UAV operates at high altitude then there is no enough features in FOV of front-facing stereo cameras and UAV's motion cannot be estimated. Second Visual Odometry subsystem was designed to estimate UAV's motion in such cases. It is composed of single down facing camera that takes pictures of the ground. If flight altitude is much higher than high of the surrounding objects then all features can be considered as laying on a single plane. This assumption allows homography between two frames to be computed and is illustrated on figure 4.2.4.

Whilst perspective transform maps 3D points in space to a two dimensional plane with preservation of perspective, homography maps 2D points in two-dimensional plane to corresponding 2D points in another two-dimensional plane with preservation of straight lines. In homogenous coordinates, corresponding points are related by 3x3 homography matrix as follows:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = H \begin{bmatrix} x \\ y \\ w \end{bmatrix} \qquad (4.2.6.)$$

Although matrix H contains nine elements it has only eight degrees of freedom - multiplying H by any non-zero constant will not alter homography. Thus eight unknowns must be found and

at least four corresponding points are required to find a solution. The camera rotation and displacement $T_k = [R_k|t_k]$ can be computed from homography matrix decomposition:

$$\mathrm{H_k} = \mathrm{M}\left(\mathrm{R_k} + \mathrm{n}\frac{\mathrm{t_k^T}}{\mathrm{d}}\right)\mathrm{M}^{-1} \qquad (4.2.7.)$$

where *M* is the camera matrix, R_k is the rotation matrix, *t* is translation vector, *n* is the unit normal vector to the plane being observed and expressed in coordinates of camera at instant *k-1*, *d* is the distance of principal point of the camera at instant *k-1* to the plane. More detailed information about homography decomposition can be found in [19].

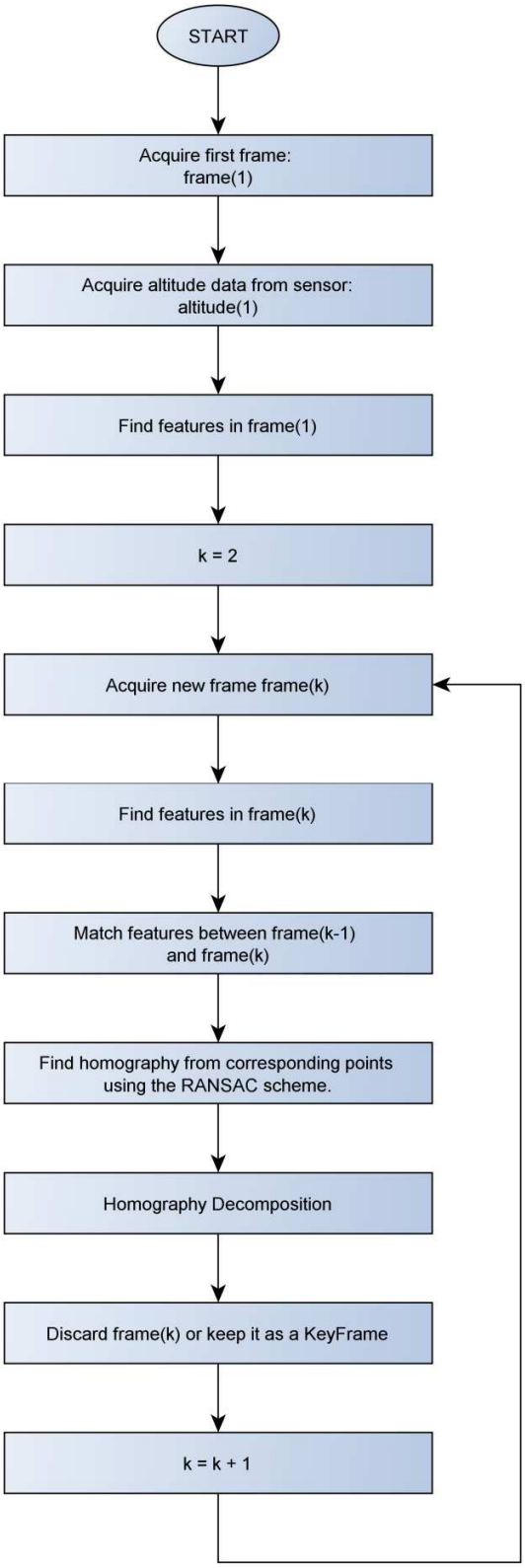

Figure 4.2.5 High Altitude System Algorithm

4.3 Sparse obstacle detection

Obstacle detection for UAV is commonly designed for a specific type of obstacles or clearly defined environmental conditions. The collision detection algorithm developed during the course of this thesis is designed for a relatively open space with limited number of obstacles. Thus it may not work as well in small, crowded rooms, or when the aircraft is flying at low altitudes in highly populated and industrialized areas. The basis of the algorithm relies on finding and grouping together distinct features of the environment that are within detection range of the UAVs stereo-rig. Figure 4.3.1. contains diagram presenting the image frame processing steps of the algorithm. Before the algorithm was implemented, stereo-rig calibration process has been carried out as described in subchapter 2.2.2. of this thesis. The obstacle detection algorithm proceeds as follows (numbers on the list below correspond to block numeration in figure 4.3.1.).

1. Images from left and right cameras in the stereo-rig are acquired. The frames are preprocessed which means that the calibration parameters of cameras has been extracted and images are themselves rectified.
2. If the stereo-rig has been used before and common field of view is defined for it, then step **6** can be already implemented, if not, additional processing is required as described in the following three points.
3. Homography matrix is found which relates points from left image frame to points in the right image frame. It defines which part of the left image frame is visible in right image frame and vice versa.
4. The common views of both cameras from the stereo rig are marked in the left and right image frames thus defining the common field of view for the whole rig.
5. Left and right image frames are cropped to display only the common field of view thus creating two new image frames – those will be used for further processing. The cropping parameters are remembered and used on every new acquired image frame. The example of whole operation presented in steps **3**,**4** and **5** done on one image frame can be observed in figure 4.3.2. where the view of the right camera is altered.

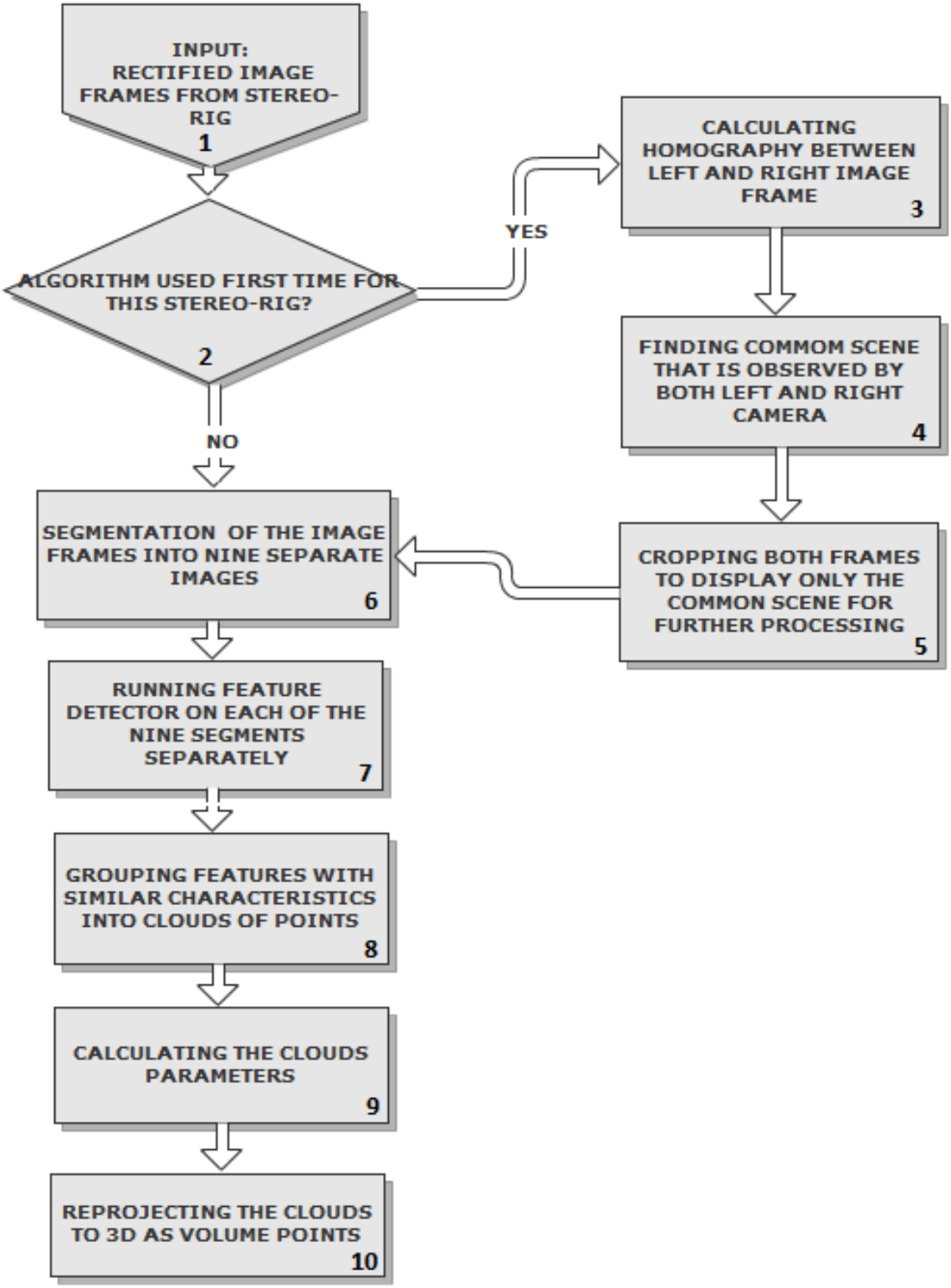

Figure 4.3.1. Graph depicting all steps of image frames processing carried out by obstacle detection algorithm

First, two rectified image frames from left and right cameras are shown . No further processing has been applied to them. Then, after homography has been calculated the result is presented on the right image frame (figure 4.3.2.c). Black area depicts what is seen by the left camera in the right image frame. Lastly, the common field of view for both cameras is marked on the right image frame (figure 4.3.2.d). It has to be noted that the common field of view depends on the cameras position in relation to each other and has nothing to the with current image seen by the them. As the cameras are mounted in the stereo-rig set up, their position in relation to each other does not change. This means that once the cropping area for left and right image frames has been calculated, the results can be applied to all other image frames acquired by using the same stereo-rig. I can be observed that the last picture in the figure 4.3.2.d suggests that the common field of view for both cameras is an ideal rectangle which is not the case. But in order to simplify further processing and ensure greater accuracy for the next steps the algorithm uses common rectangular field of view. This approach will prove extremely helpful in the next step.

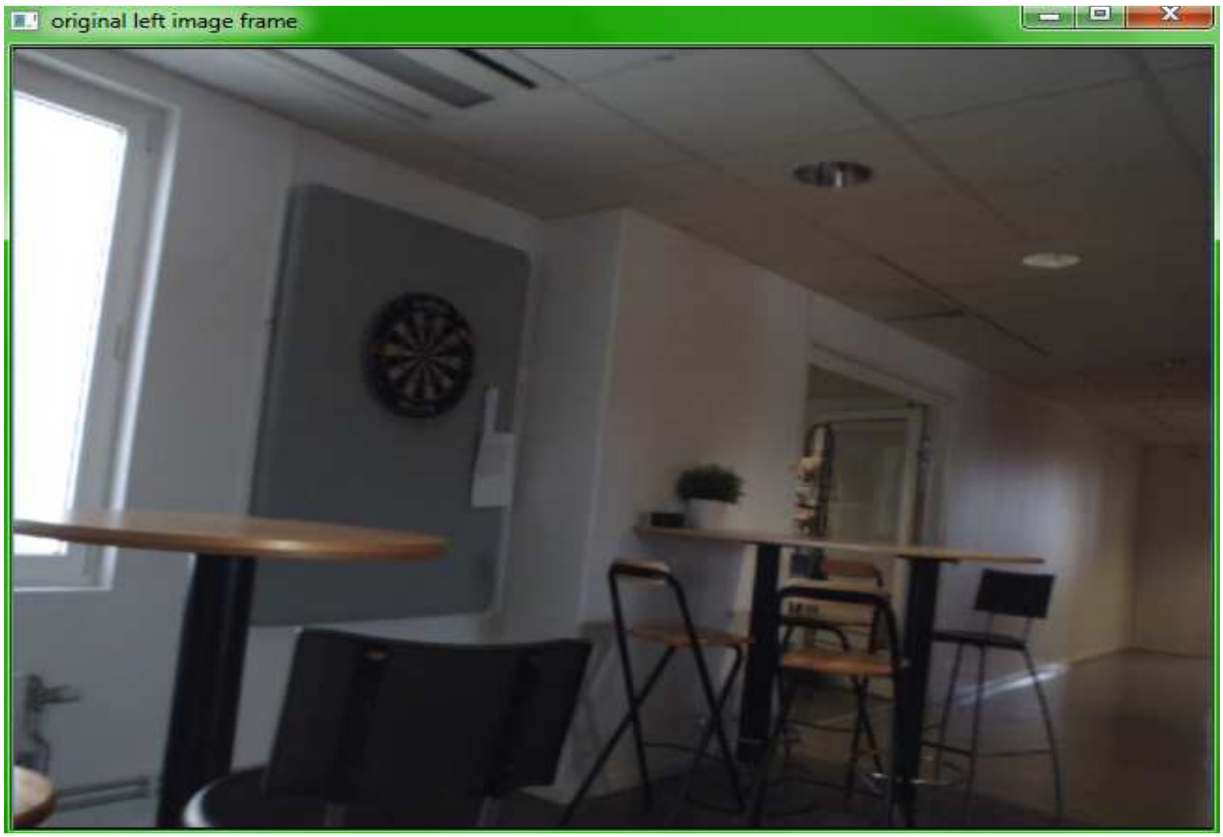

Figure 4.3.2.a Rectified but otherwise unprocessed image frame from the left camera

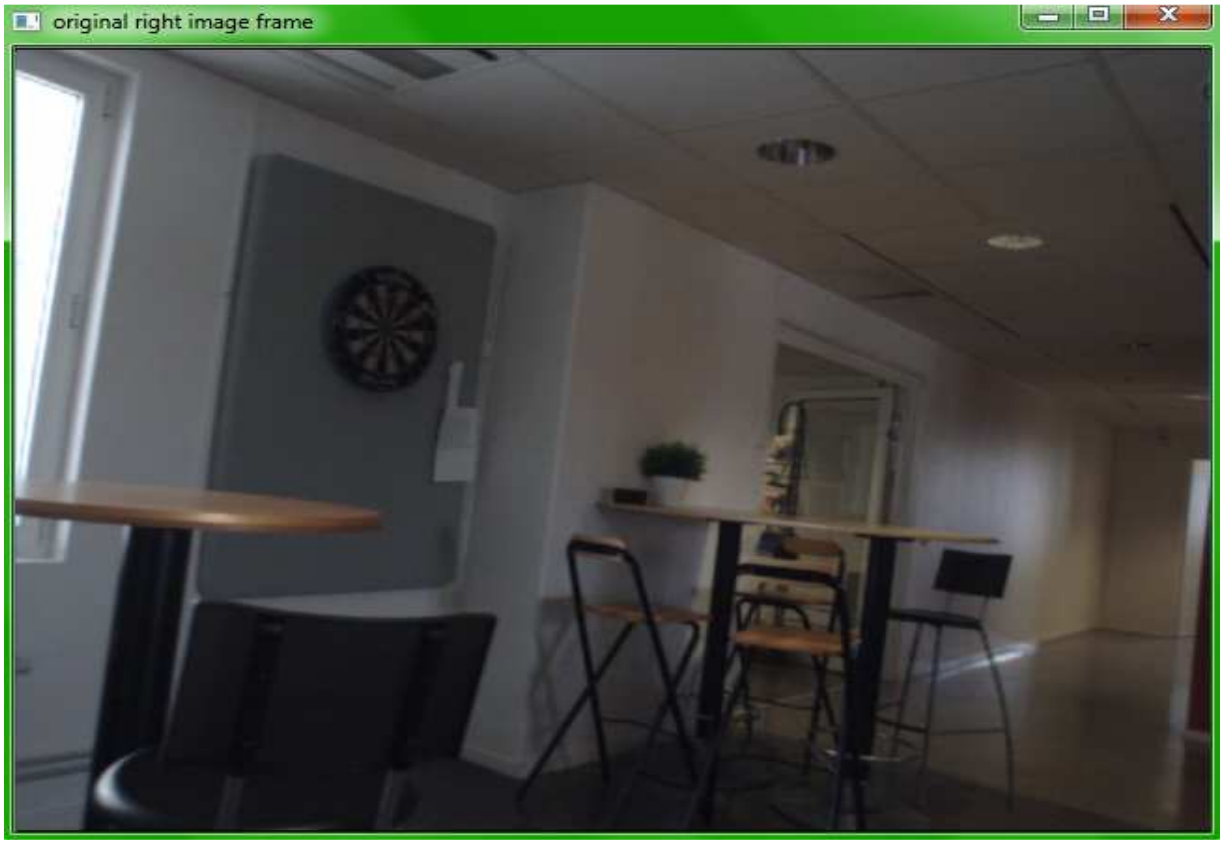

Figure 4.3.2.b Rectified but otherwise unprocessed image frame from the right camera

Figure 4.3.2.c The view of the left camera in the right image frame –how much fields of views of both cameras overlap

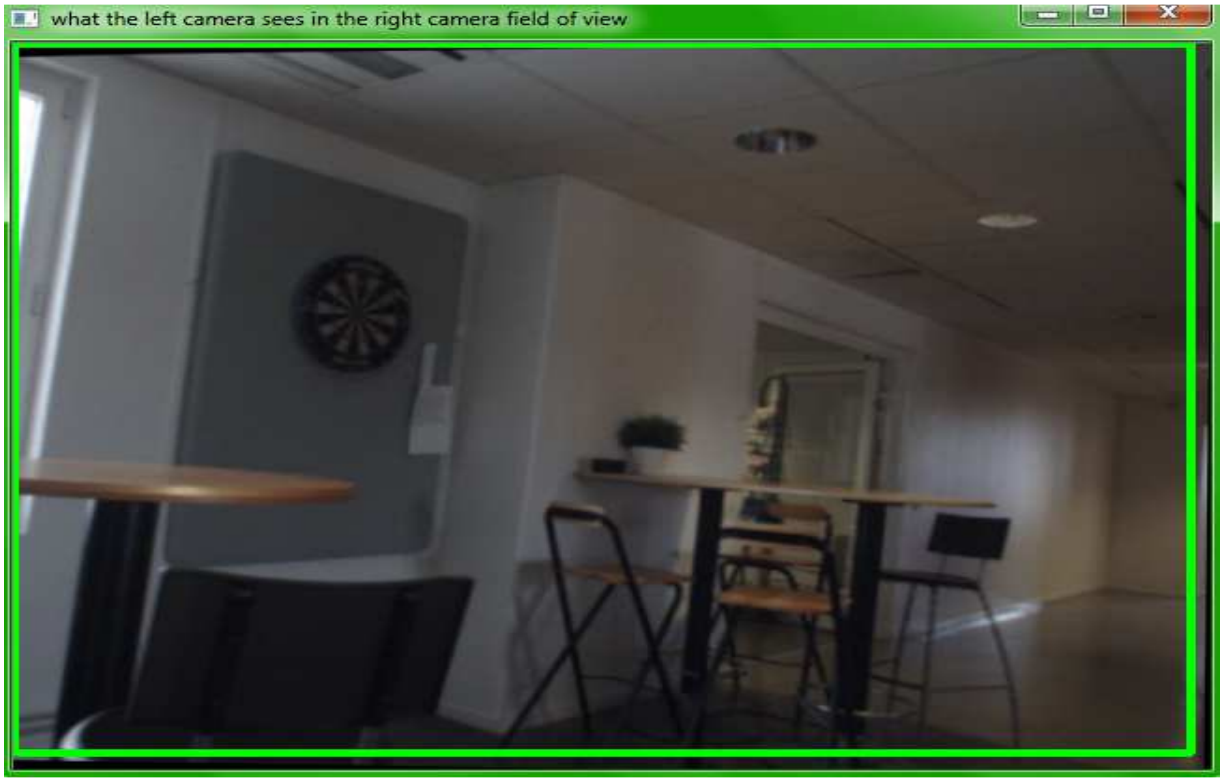

Figure 4.3.2.d The view of the left camera in the right image frame –how much fields of views of both cameras overlap

6. All feature detectors work in a way that results in finding the best features in the image. Usually there is no mechanism that assures even distribution of detected keypoints in the image, neither there is need for one. This works good enough in most cases and is sufficient for calculating camera movement between image frames, as the position of features is not as important as their number and robustness. However, if obstacle detection is considered, the uneven distribution of features becomes a significant issue. If an image frame with one extremely feature-region then most feature detectors will acquire keypoints only from that region thus ignoring all other potential obstacles that may be present on the observed scene. So in this step, a way to prevent such situations has been implemented. Each image frame is divided into nine equal ROI (Regions Of Interest), thus creating eighteen small images instead of two full sized ones. The visualization of the process can be observed in figure 4.3.3. At this moment the importance of previous steps can be appreciated. If the image frames would be segmented without determining the common field of view

beforehand, then each ROI from the left image frame would present a slightly different scene than its counterpart from the right image frame. That in turn means that some regions of the common field of view could not be used by the algorithm. Additionally, the rectangular common field of view allows for easier and more efficient image segmentation. Thus the previous processing enables the algorithm to exploit provided data as much as possible.

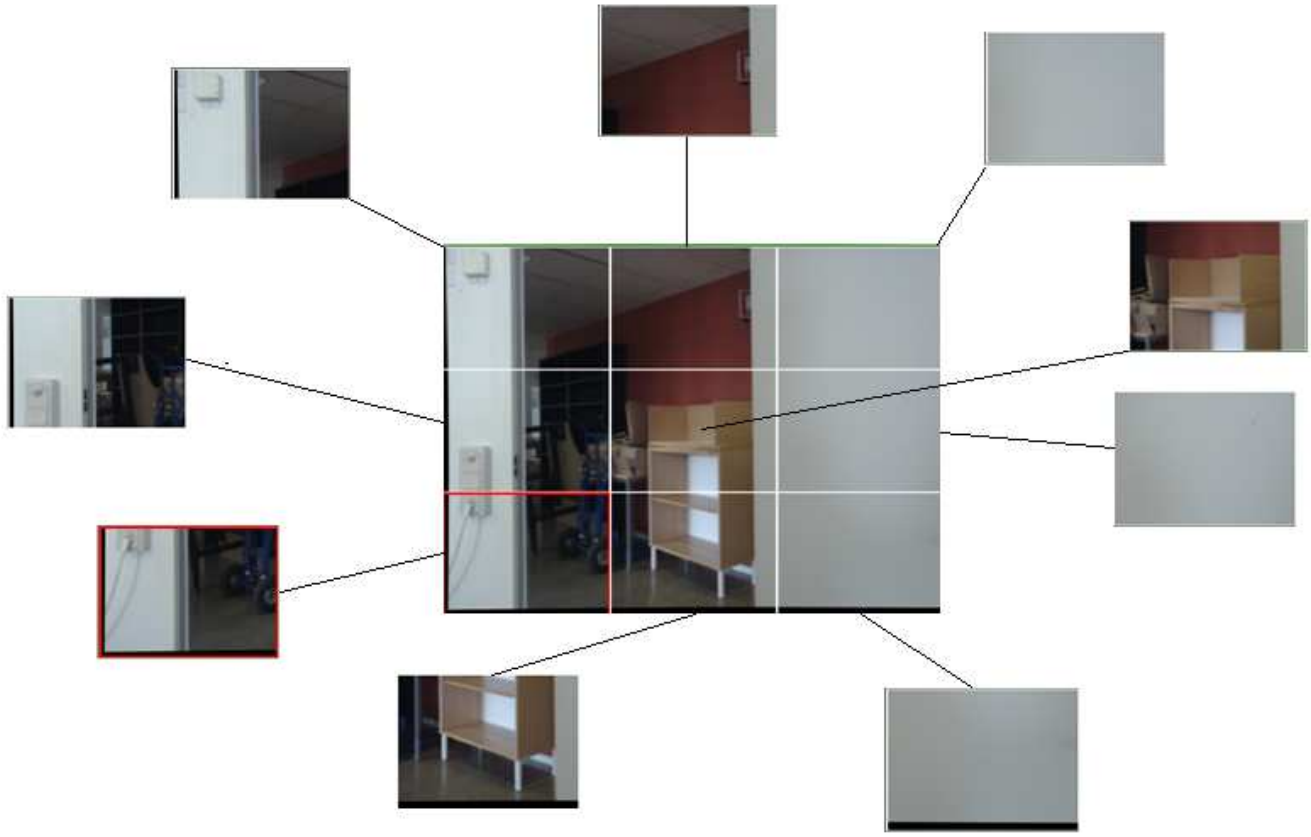

Figure 4.3.3. ROI segmentation shown on an example picture taken by the right camera from the stereo-rig

7. After the segmentation, feature detector algorithm processes each ROI as a separate picture, thus ensuring that keypoints will be detected in every part of the image. This procedure does not require more time or more computational power in comparison to searching the whole picture for features without considering the ROIs separately. The reason for that is that the surface in which the algorithm detects keypoints is

exactly the same in both cases. This step provides the algorithm with equally distributed features which can then be used for much more accurate scene analysis.

8. For the purpose of this step it is assumed that groups of keypoints that are close together and have similar distance from the stereo rig are a part of the same obstacle. Thus, parameters of the keypoints such as depth and distance from each other are calculated and features with similar characteristics are grouped together into clouds of points. The exact thresholds of when one cloud should end and another begin ought to be adjusted to the cameras accuracy and the distance on which the obstacles should be detected. Examples of groping points into clouds are shown in figure 4.3.4. In each of the two ROIs on the left two clouds have been created as features detected in them do not all belong to the same object or surface. In the two ROIs on the right large solitary clouds of points have been created since the features in them strongly resembled each other in the means of depth.

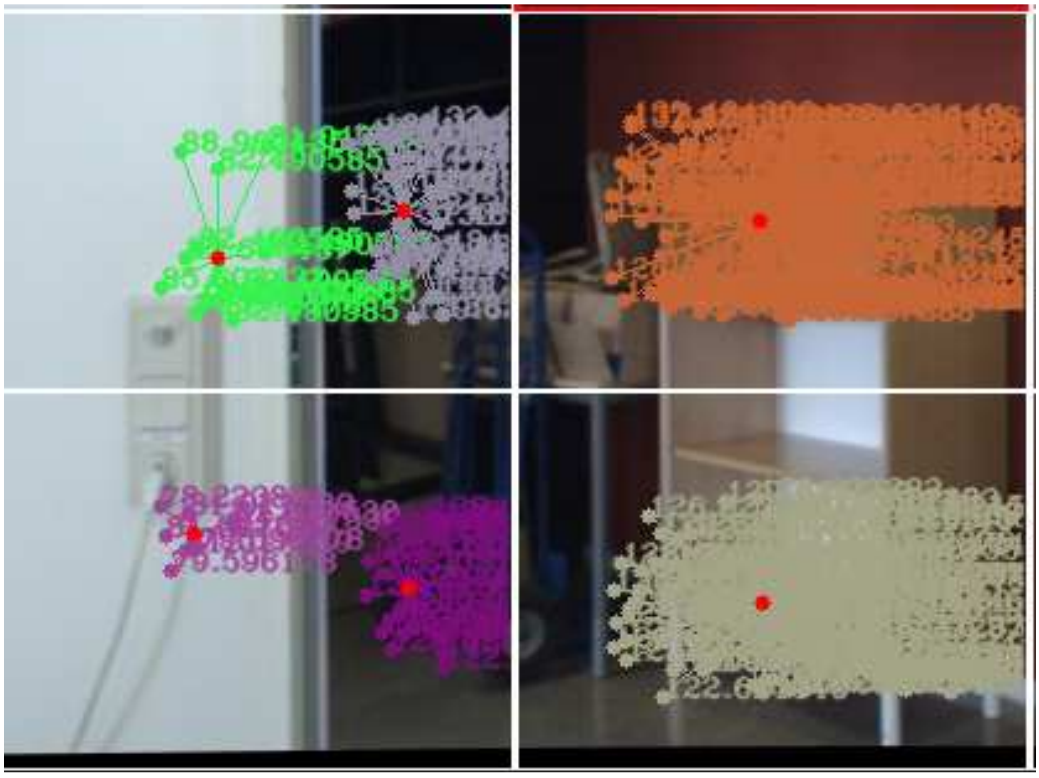

Figure 4.3.4. Example of grouping features into clouds of points with similar characteristics

9. Parameters of each cloud as a separate entity are calculated. This includes:

coordinates of the clouds center – the *x* and *y* coordinates of each clouds center are calculated as a mean of x and y coordinates of the clouds n keypoints. This is shown in equation 4.3.1. Clouds centers are marked in figure 4.3.4. with red dots.

$$x_{center} = \frac{1}{n}\sum_{i=0}^{n} x_i \ , y_{center} = = \frac{1}{n}\sum_{i=0}^{n} y_i \tag{4.3.1.}$$

cloud radius – it is the distance between the center of the cloud and the farthest point from it calculated in the real world coordinates.

average depth of the clouds keypoints – the mean of the depths of clouds keypoints calculated similarly as in equation 4.3.2. The depth of all keypoints is estimated using equation 2.2.7.

$$Z_\mu = = \frac{1}{n}\sum_{i=0}^{n} Z_i \tag{4.3.2}$$

standard deviation of the depth of clouds keypoints – contains statistical information about how the depth of the keypoints Z_i varies in the cloud, it is calculated as in equation 4.3.3.

$$\partial^2 = \sqrt{\frac{\sum_{i=0}^{n}(Z_\mu - Z_i)^2}{n}} \tag{4.3.3}$$

cloud density – measure of how densely features are packed in the cloud, it is calculated by dividing number of keypoints in the cloud by the clouds maximum volume – a sphere with cloud radius r as in equation 4.3.4.

$$\rho_{cloud} = \frac{3n}{4\pi r^3} \tag{4.3.4}$$

10. The most essential clouds parameters are reprojected into real world 3D coordinates as presented in equation 2.2.10. This includes the clouds center, and the clouds radius. Every cloud is treated from this point as a spherical obstacle. All features that belong to the particular cloud are either inside the sphere or at its surface, thus ensuring maximum level of safety. Visualization of the obstacle group detected by the algorithm is presented in figure 4.3.5 and figure 4.3.6. Green arrowhead marks the position of the stereo rig, its size is arbitrary but can be adjusted to the size of considered UAV. Gray spheres are the obstacles that were developed from clouds of features. Their size is regulated by the distance between the clouds center at its farthest point – radius of the sphere, while the position is defined by reprojecting the

clouds center to the real world 3D coordinate system. The center of the system is the stereo rig, and all obstacles are positioned according to this double camera set-up.

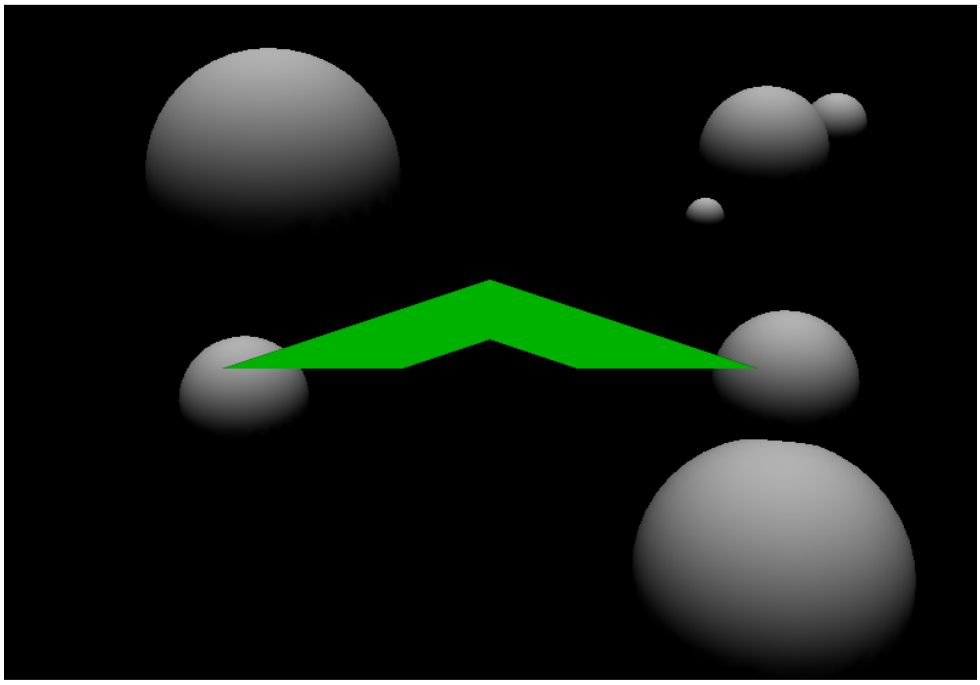

Figure 4.3.5. Visualization of an obstacle set created from reprojecting clouds of points into real world 3D coordinates

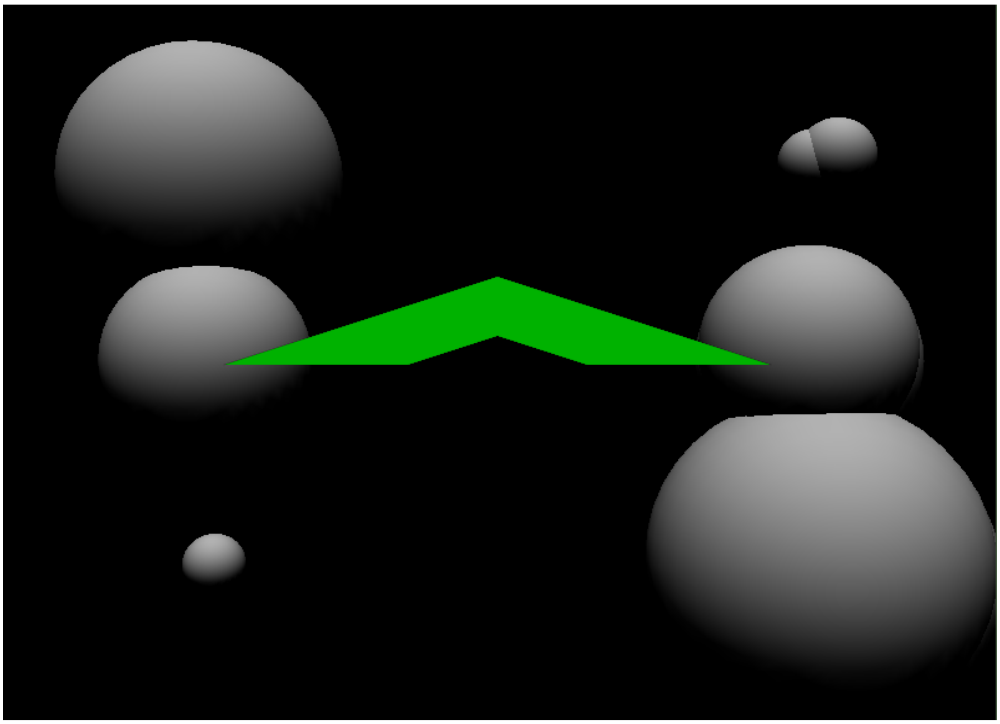

Figure 4.3.6. Visualization of an obstacle set created from reprojecting clouds of points into real world 3D coordinates

5. Conclusions and Further Works

In this thesis the problem of UAV positioning and sparse obstacle detection by using computer vision algorithms was tackled. Camera calibration problem was analyzed and the calibration and stereo-calibration were implemented to allow for accurate position estimation. The most common causes of errors were identified and their influence examined.

Two approaches were considered: ground tracking of the aircraft and onboard tracking and obstacle detection. For the former, a marker identification method was developed and its position was estimated using stereovision and 2D to 3D reprojection. For the later, the mechanism of Visual Odometry was employed to solve 2D and 3D positioning case. Additionally, method of sparse obstacle detection was proposed based on turning clouds of feature points into computationally less expensive spherical obstacles.

The topic is however far from being finished. In the future in can be further developed by testing the algorithms on high-quality hardware that will allow for accurate performance evaluation. Additionally the algorithms can be upgraded to work in different environmental conditions or for a longer periods of time. The way of integrating the algorithms with a camera system that will be mounted onboard an UAV can be invented. Further research on sub pixel accuracy can be performed to study its influence on the performance of positioning algorithms.

References:

[1] E. Michaelsen, K. Jäger, D. Roschkowski, L. Doktorski, and M. Arens , “Object-Oriented Landmark Recognition for UAV-Navigation”, Pattern Recognition and Image Analysis, 2011, vol. 21, no. 2, pp. 152–155

[2] Syed Irtiza Ali Shah, “Vision Based 3D Obstacle Detection Using Single Camera for Robots/UAVs”, MSc dissertation, Mechanical Engineering, Georgia Institute of Technology, 2009

[3] F.F.Khalil, “Optical flow techniques in biomimetic UAV vision”, International Workshop on Robotic Sensors: Robotic and Sensor Environments, 2005, pp.14-19

[4] J.Z.Sasiadek, “Feature matching for UAV navigation in urban environments”, 15th International Conference on Methods and Models in Automation and Robotics, 2010, pp. 164 – 169

[5] G.Conte, “An Integrated UAV Navigation System Based on Aerial Image Matching”, Aerospace Conference, 2008, pp.1-10

[6] S.Hrabar, “Combined optic-flow and stereo-based navigation of urban canyons for a UAV”, RSJ International Conference on Intelligent Robots and Systems, 2005, pp. 3309-3316

[7] G.Bradski and A.Kaehler, “Learning OpenCV”, 1st ed., Sebastopol, O’Reilly Media, 2008

[8] J.Y. Bouguet, “Camera Calibration Toolbox for Matlab” [Online]. Available: http://www.vision.caltech.edu/bouguetj/calib_doc/ [Accessed: 2012-04-02]

[9] The Free Dictionary by Farlex, “Medical Dictionary” [Online]. Available: http://medical-dictionary.thefreedictionary.com/stereopsis [Accessed: 2012-07-21]

[10] Fluid Imaging Technologies, “Color vs. Black & White Cameras” [Online]. Available: http://www.rainsoft.de/ [Accessed: 2012-07-01]

[11] C.Poynton, “Color FAQ” [Online]. [Accessed: 2012-05-13]

[12] E.Rublee, V.Rabaud, K.Konolige and Gary Bradski, “ORB: an efficient alternative to SIFT or SURF”, The Plessey Company pic. 1988

[13] J. Shi and C. Tomasi (June 1994). "Good Features to Track,". 9th IEEE Conference on Computer Vision and Pattern Recognition. Springer.

[14] C. Harris and M.J. Stephens, “A combined corner and edge detector”, Alvey Vision Conference, 1988, pp. 147–152

[15] R.Laganière, “OpenCV 2 Computer Vision Application Programming Cookbook”, 1st ed., Birmingham, UK, Packt Publishing Ltd., 2011

[16] D.Scaramuzza and F.Fraundorfer, ”Visual Odometry: Part II - Matching, Robustness, and Applications”, IEEE Robotics and Automation Magazine, vol. 19, issue 2, 2012.

[17] D.Scaramuzza and F.Fraundorfer,” Visual Odometry: Part I - The First 30 Years and Fundamentals”, IEEE Robotics and Automation Magazine, vol. 18, issue 4, 2011.

[18] Geiger, “Andreas Geiger's Homepage”, [Online]. Available: http://www.rainsoft.de/ [Online]. [Accessed: 2012-06-12]

[19] E.Malis and M. Vargas, “Deeper understanding of the homography decomposition for vision-based control”, INRIA, no. 6303 , September 2007