CZECH TECHNICAL
UNIVERSITY
IN PRAGUE

F3

Faculty of Electrical Engineering
Department of cybernetics

Bachelor's Thesis

Estimation of the Mutual Pose of Camera Stereo Pair Subject to Parasitic Dynamics

Tomáš Kratochvíl
Open informatics

May 2022
Supervisor: Ing. Viktor Walter

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Kratochvíl Tomáš** Personal ID number: **492283**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Open Informatics**

Specialisation: **Artificial Intelligence and Computer Science**

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Estimation of the Mutual Pose of Camera Stereo Pair Subject to Parasitic Dynamics

Bachelor’s thesis title in Czech:

Estimace vzájemné polohy kamerového stereopáru pod vlivem parazitní dynamiky

Guidelines:

Review existing solutions aplicable for extrinsic calibration of a stereo pair of fisheye cameras onboard of a flying unmanned aerial vehicle (UAV). These cameras are a component of a relative localization system UVDAR, which provides estimates of relative poses of marked nearby UAVs [3]. While the initial mutual pose of the cameras can be estimated, the precise mutual pose of these cameras changes over time. This is due to the combined effects of their far-apart layout, attachment materials, environmental heat and vibrations of the UAV, often even leading to persistant warpage of their connecting holder. This effect leads to degraded precision of the estimated relative positions of surrounding UAVs in cases when a given target is observed by both cameras if their mutual relative pose is assumed to be static.
Based on the review, select an appropriate algorithm for the task and implement a program capable of dynamically obtaining the mutual pose of the camera pair in flight. This program has to be compatible with UAV system used by the MRS group and be able to operate during flight.
Optionally, the system may be tested in real-world deployment of UAVs where the properties of a leader-follower flight can be compared between cases when the mutual camera pose is based on a static prior estimate and when it is estimated dynamically.

Bibliography / sources:

[1] Scaramuzza, Davide, et. al. - ”A Flexible Technique for Accurate Omnidirectional Camera Calibration and Structure from Motion” - New York, 2006.
[2] Hartley, Richard a Zisserman, Andrew - „Multiple view geometry in computer vision“ - USA, 2000.
[3] Walter, Viktor a Vrba, Matouš a Saska, Martin - „On training datasets for machine learning-based visual relative localization of micro-scale UAVs“ – Paris, 2020.

Name and workplace of bachelor’s thesis supervisor:

Ing. Viktor Walter Multi-robot Systems FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **21.01.2022** Deadline for bachelor thesis submission: **20.05.2022**

Assignment valid until: **30.09.2023**

Ing. Viktor Walter	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

. _____ Date of assignment receipt

_____ Student's signature

Acknowledgement / Declaration

I would like to thank my supervisor Viktor Walter for his help and advises during the making of this thesis.

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, 20 May 2022

.....

Abstrakt / Abstract

Tato práce rozšiřuje již existující metodu vzájemné sebelokace bezpilotních helikoptér (dále jen UAV - Unmanned Aerial Vehicles). Systém UVDAR je založen na ultrafialových blikajících LED, které detekuje speciální UV kamera typu rybí oko. Na základě získaného rozložení LED na pozorovaném objektu systém dopočítává jeho 3D relativní polohu. Výhodou tohoto systému je, že nepotřebuje vzájemnou komunikaci a je spolehlivý v reálných podmínkach. Použité kamery jsou na UAV skupiny Multirobotických systémů ČVUT neseny v páru, který je umístěn pod určitým úhlem na dvou ramenech na palubě UAV. Použité rozložení těchto kamer vede k překryvu jejich zorného pole, čímž se přímo nabízí možnost zpřesnění lokalizace s využitím stereo efektu. Toto rozložení, které vyžaduje značnou vzájemnou vzdálenost kamer, v kombinaci s dynamikou UAV v letu a mechanickým namáháním, zavádí do vzájemných poloh kamer nejistoty. V práci se autor zabývá návrhem systému, který umožní v reálném čase dopočítávat danou odchylku od původního odhadu polohy kamer a zlepšit tak celý systém vzájemné relativní lokalizace.

Klíčová slova: UAV, UVDAR, epipolární geometrie, extrinsická kalibrace

Překlad titulu: Estimace vzájemné polohy kamerového stereopáru pod vlivem parazitní dynamiky

This thesis presents en extension of pre-existing mutual relative localization system for UAVs (Unmanned Aerial Vehicles) called UVDAR. This system is based on blinking ultraviolet LEDs detected by specialized UV-sensitive camera with fisheye lens. Using the known layout of the LEDs on the observed target the system estimates its relative 3D position. The primary advantage of this system is that it doesn't require direct communication and its reliability in real-world conditions. The cameras used are carried onboard UAVs of the Multi-Robot group in CTU, in pairs attached at on angle on two protrusions. The layout of these cameras allows for an overlap in their visual fields, directly suggesting improvement of the output estimates by exploiting their stereo-effect. However this layout-requiring significant mutual distance of the cameras, coupled with the flight dynamics and mechanical strain-introduces uncertainty into the known relative poses of the cameras. The author of this thesis proposes a system that can calculate these offsets in real time, and thus improve the entire estimate of relative pose of a target.

Keywords: UAV, UVDAR, epipolar geometry, extrinsic calibration

/ Contents

/ Figures

Chapter 1
Introduction

Today, unmanned aerial vehicles (UAV) are integrated as a part of many practical applications. They are used, for example, to explore areas that are too dangerous or inaccessible for people [1]. Popular applications include private purposes such as filming or delivery.

In certain situations, the limited load capacity of UAVs represents a bottleneck. Single UAV is not able to carry all the required load. Another risk factor is that a UAV may be damaged during the mission.

For this kind of situations, there is a concept of a team of UAVs (swarms and formations) which communicates and cooperates in order to complete the mission. Within swarms or even simple leader-follower systems, UAVs typically maintain their distance from their neighbours within a specified range. The ability of UAVs to do so is predicated on their ability to perform mutual relative localization. Relative localization is fairly simple to achieve in an open environment with GPS and RTK or indoors under motion-capture setups. However, obtaining relative localization of neighbours purely based on onboard sensors is challenging. [2]

This thesis focuses on said relative localization with help of the UVDAR system. Each UAV within the system has a blinking ultraviolet LED on each arm that can be identified by an UV sensitive camera. To be able to cover wide area around a UAV, a pair of UV cameras is used. The used cameras are equipped with fisheye lens, which supports very wide fields of view (FoV), over 180°.

The stereo effect, which is the primary subject of this work, is a side effect in the UVDAR system. We can use it to our advantage, assuming known rotation between the two cameras.

However, the camera mount is unstable due to vibrations and other external factors. Even 1° error in rotation results in inaccurate measurements. Therefore, it is necessary to create a system that will calculate the orientation error.

This thesis presents a solution as an optimization algorithm, that is able to quickly estimate this error in transformation between the cameras in stereo pair.

Chapter 2
State of the art

The usage of stereo vision has grown significantly in the recent past. Camera stereo pairs have replaced many other sensors for reconstructing local scenes. However, with the new approach come new problems to solve. One of them is a real-time calibration of a stereo pair. The obvious workaround is to use strong camera holders which would handle even the most challenging conditions. This is not a feasible idea in our specific case because of many reasons, such as weight, price of the prototyping, and impact absorption.

We, therefore, have to compensate for unstable holders with software. Many publications (e.g. [3]) suggest calibration by calculating the so-called essential matrix. This is a great way to calibrate cameras purely analytically, but it is only possible for pinhole camera models, where only slight deviations from this model can be corrected-for in the image. The same thing applies to most of the methods found since wide-angle lenses are not in wide use. [4] [5] With our wide-angle lens, the analytical approaches become more complicated. Another publication [6] states it is possible to find essential matrix even for wide-angle lenses using the RANSAC algorithm. This only works if five or more corresponding points are given. The results of this method claim the root-mean-squared re-projection error to be slightly over 0.5 pixels. In our stereo pair model, the cameras have a very small common field of view, hence it would be unlikely to have at least 5 common points from which to calculate the essential matrix. Due to the lack of points, the method is not suitable for our scenario. The camera only sees Ultra Violet blinking lights on the arms of a UAV, and therefore the number of points is determined by the number of lights the UV camera sees. This is unlike classic RGB cameras where it is easy to find certain points used for calibration. In RGB cameras, anywhere there is a higher contrast of colors, the point can be used as a corresponding point to essential matrix calculation. This disadvantage is greatly overweighed by all the advantages the UVDAR system brings to the table.

An option that can be used to address the issue is to use epipolar lines. These lines are explained in detail in chapter 5, but the main idea is to create a corresponding epipolar line for each point and then calculate an error function, which can be used in a gradient-based optimization. The difficulty of this optimization is finding a local gradient since the function is not analytically differentiable. Derivative-free optimization has been discussed many times. The obvious and the easiest way to implement the optimization is to use RANSAC, or rather slightly adjusted RANSAC. This method calculates a number of inliers depending on a certain threshold and then finds a random sample with the most inliers [7]. Since we have only a few points to calculate the error from, the inliers would cause us to find many solutions close to optimum. Therefore the algorithm calculates the error function for each random sample and selects the minimum. However, this method is not deterministic.

Chapter 3
Technologies

There are several technologies to use regarding 3D world reconstruction and distance measurement. The author took into account the two most popular technologies LiDAR and stereo vision, which are used particularly in automotive industry, as well as in robotics.

3.1 LiDAR

"LiDAR (acronym for Light Detection And Ranging) uses electromagnetic (EM) waves in the optical and infrared wavelengths. It is an active sensor, meaning that it sends out an EM wave and receives the reflected signal back." [8] The time between EM wave sent and received corresponds to the distance of reflected surface. The LiDAR which is considered is the rotating type. This LiDAR spins around its vertical axis and emits EM waves. Some of the LiDARs also have vertical spread, so it detects some objects even lower or higher than LiDAR itself. Among the biggest advatages of LiDAR are high precision and output rate [9]. Since LiDAR directly outputs points relative to its position, we can easily recreate local scene with almost no additional computation. Even if the holder caused small miscalibration, the final 3D reconstruction would be easily recalculated. Other computations would be necessary if the unit found itself in foggy, dusty or rainy environment. The reflective parts of these elements could cause issues, because they may count as objects, so this has to be filtered out.

LiDAR can be also used in relative localization, since it can fairly quickly reconstruct the scene a find its dynamic points. Some LiDARs have smaller density, meaning less amount of points detected on given volume. This could cause missing UAVs further away if they are in-between the rays.

3.2 Stereo vision

Stereo vision uses two cameras (in our case with fisheye lenses). Using these two cameras, we can obtain two different views of same environment, from which the 3D reconstruction is easily calculated.

Stereo vision on its own has disadvantages over LiDAR, since it may encounter similar obstacles such as rain, fog and it has singnificant computational complexity. This is not an obstacle in our case because of UVDAR implemetation. UVDAR offers UV sensitive cameras which detect blinking UV markers mounted on each arm of UAV in swarm. This significantly increases robustness to challenges of outdoor environments regardless of the time of day, and second, its use of active markers allows for retrieval of orientation or identity of a target [2].

Figure 3.1. Comparison between visible and UV camera footage from UVDAR. The UV image is significantly easier to process to retrieve UAV information [2].

Chapter 4
Fisheye lens

Fisheye (ultra wide angle) lenses are widely used commercially as well as in science. Their usage has the benefit of capturing much wider field of view than rectilinear optics. The problem with these lenses is the highly non-linear projection. It is crucial to have some sort of function which transforms each pixel into a direction vector of which span is a linear space of points that might be represented by a given pixel. The same thing applies to real-world vector transformation into a pixel. These transformations can be approximated, for example, by polynomials of certain degree.

4.1 Calibration

In order to achieve accurate calibration, we will use a tool called `OCamCalib`[1]. Calibration starts with a set of captured distorted grids. Each grid consists of MxN 2D points. The grid itself can be located manually with mouse selecting all the corners of given checkerboard. In the tool, there is an algorithm which does that automatically.

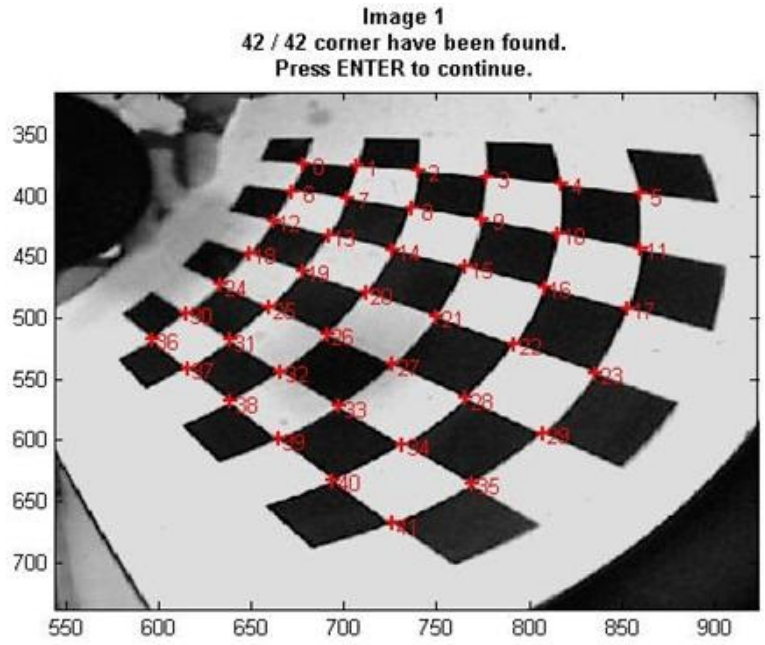

Figure 4.1. Extracted corners of given checkerboard [10].

The tool calculates a set of linear equations using the least-squares method [11]. This gives us intrinsic and extrinsic parameters which represent polynomials later used in methods *cam2world* and *world2cam*.

[1] `https://sites.google.com/site/scarabotix/ocamcalib-omnidirectional-camera-calibration-toolbox-for-matlab`

Chapter 5
Geometry

5.1 MRS geometry

Basic geometry functions used here are implemented in the `MRS library`[1]. The function which is used most often in the implementation is rigid transformation. It takes vector relative to some coordinate frame and transforms it into a vector representing the same point relative to other frame. It does not take into account the length of the vector, so the function presumes the vector represents point in infinity. Suppose pixel x_1 from left camera and x_2 from right represent a point in the real world which is so far away from the stereo pair that vectors represented by pixels on both images are almost parallel. This gives us exact boundary for pixel transformation from one camera to another. The goal of the thesis is to calibrate this function in real time for a stereo effect to operate correctly under unfavorable conditions.

5.2 Epipolar geometry

"The epipolar geometry is the intrinsic projective geometry between two views. It is independent of scene structure, and only depends on the cameras' internal parameters and relative pose." [12] The relative pose is what defines the relation between the two cameras. It gives us certain information on what we could say about points viewed in one camera relative to the other and vice versa. Let us take pixel x_1 from the left camera and also pixel x_2 seen by the right camera. These two points are projections of real world point X, which is visible from both cameras. The centers of the two cameras, together with the real world point X define a plane. This epipolar plane intersects the image of each camera and creates epipolar line, which is crucial in chosen method for this optimization. Both pixels x_1 and x_2 lie on their respective epipolar lines.

5.3 Epipolar line

In every stereo pair with common part of image exists a transformation between point x from one camera image to epipolar line in the other. This gives us accurate approximation of where the pixel corresponding to the same object in the world might be seen on the other camera image. Every epipolar line can be shortened to a line segment which starts from the inner edge of camera and ends in the representation of pixel in infinity.

[1] `https://github.com/ctu-mrs/uav_core`

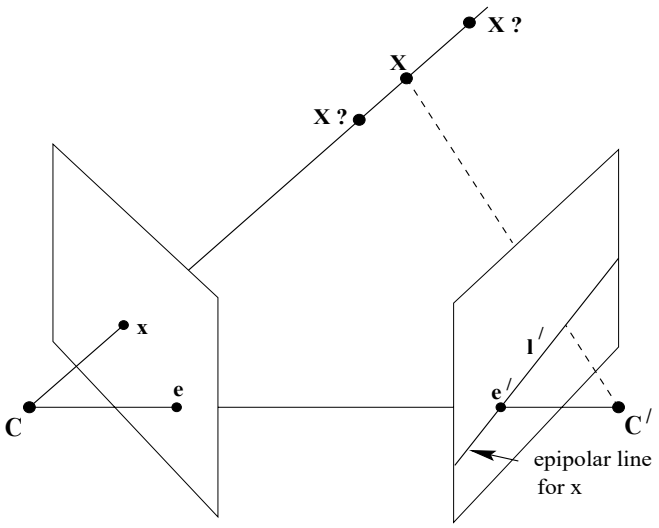

Figure 5.1. An image point x corresponds to a ray in 3-space defined by the first camera centre, C and x, as well as by the camera model. This ray is imaged as a epipolar line l' in the second view. [12]

5.3.1 Calculating epipolar line

Calculating an epipolar line is fairly simple using rectilinear lens in cameras. Suppose we have pixel x on left image. This pixel represents vector in which direction is the corresponding point in the real world. Then we take the point and centers of both cameras. These three points define plane, which intersects the view point of right camera. This intersection is epipolar line. Since we use wide angle lenses, the epipolar lines are rather epipolar curves, because the further from the camera center the point is the more the image is distorted. These curves do not have a general analytical solution, so the proposed solution is to take the point in infinity and shorten the previously infinitely long vector into a set of points along the optimal ray. With this method, the epipolar curve is a representation of points in different distances from the left camera.

5.4 Epipolar curve

Every epipolar curve in the proposed solution is approximated by polynomial of second degree. It was verified to describe the set of points created in the process mentioned before sufficiently well. In addition, calculating distance between the polynomial and a point can be done analytically. The distance is important for the error function applied later. For higher degrees of polynomial, the distance method lacks analytical solution. [13] On the other hand, lower degree of polynomial, which is a line, does not represent the reprojection accurately. Representing the curve with second degree polynomial has a great advantage, since for each pixel, we have to remember only three parameters.

Chapter 6
Materials and methods

There are many methods of re-calibration of stereo pair. However, the objective of the re-calibration discussed in this thesis is to approximate real-world relative pose between the two cameras of a stereo pair in real-time. We can assume the distance is immutable because of the type of attachment mechanism.

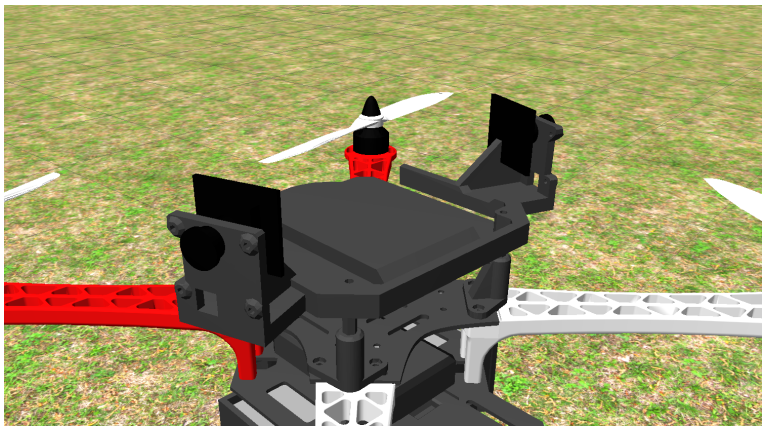

Figure 6.1. Camera holder.

The problem is the holder, which due to external disturbances causes the camera to rotate around the handling point with respect to the assumed orientation. There are two main values which are going to be crucial during the calculations. It is the actual rotation of each camera and the expected rotation. Even if the cameras were to wobble by the same angle (in each of the rotation axis), it still would be necessary to calibrate. The ultimate goal is to find the exact real world relative orientation between the two cameras, which can be also formulated as finding minimum of some error function. This error function is different for each method used.

6.1 Camera calibration

This part of relative localization is crucial for all the methods. As mentioned in the introduction, cameras in UVDAR system have wide angle lenses so the image is considerably distorted. It is very difficult to retrieve some relevant information with no additional calculation as is the case with a rectilinear lens. Therefore, the camera needs to be calibrated first. To calibrate such camera, the `OCamCalib`[1] tool is used. On input the program accepts images of non-square checkerboard and exports calibration file after several steps. The file allows us to use the *cam2world* and *world2cam* methods of a C language library. These functions represent relationship between given pixel and real world vector on linear span of which the given point occurs. However, this tool

[1] `https://sites.google.com/site/scarabotix/ocamcalib-omnidirectional-camera-calibration-toolbox-for-matlab`

needs an extension since it does not consider UV cameras. The extension is already available by `UVDAR`[2], which does not take as an input a checkerboard, but rather UV blinking points arranged in a grid.

Figure 6.2. Sample calibration images. These images are not be used in the UVDAR framework, but they demonstrate the curvature of the calibration pattern in a wide-angle lens well. [10]

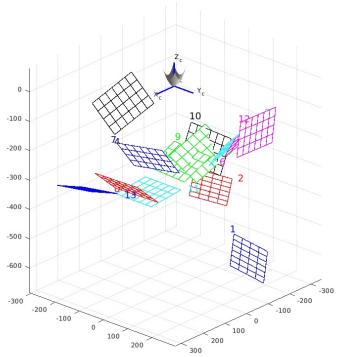

Figure 6.3. Extrinsic display of calibration images from UVDAR calibration.

6.2 Filter of useful area

Useful area is described as an intersection of the two separate views from both cameras. Useful points are then those which are in the useful area, and are seen by both cameras. To effectively filter said points in real time, the author chose to pre-calculate two matrices of the dimensions of the input camera images. For the sake of simplicity, the procedure is explained by creating only the right matrix. All matrix pixels are initially set to 0. The matrix has a value of 255 at the coordinates corresponding to world directions that are also observed from by second camera. We iteratively traverse the pixels over all the edges of the left camera. This way we get boundaries of the useful area. For the sake of speed, every n-th pixel can be used. One such pixel is then converted by the *cam2world* function into a unit vector denoting the corresponding pixel's optical ray. The vector is transformed using `TFROS`[3] into a view from the right camera using its initially assumed relative pose, which is then projected using *world2cam* function into a pixel on the right camera. After all transformations complete, there are several points projected to the right matrix, which determine the boundary of the field of view intersection. These boundaries can easily be filled with the value 255 using the *OpenCV*[4] library to get the desired space of pixels that are visible from the left camera. Points in the area with the value of 0 are ignored in later computations.

[2] `https://mrs.felk.cvut.cz/gitlab/waltevik/ocamcalib-uvdar`
[3] `http://wiki.ros.org/tf`
[4] `https://opencv.org/`

6.3 Error minimization

Now that we have all the necessary functions to perform error minimization there are multiple methods to choose from.

6.3.1 Points pairing

The first method achieves optimization by finding certain number of points which both cameras see simultaneously. To begin with, the points need to be filtered. Author chose iterative method in which we choose n points along the right side of left image. All the steps are described for calculating the left filter, right filter is only mirrored. These points are mapped to point from the view of the right camera and create a boundary of field of view. During the filter process function *cam2world* is used, which from camera view point creates unit vector, this vector is then transformed to the view from the right camera and function *world2cam* displays transformed vector to the right camera image. From these points a polygon is filled. The polygon represents part of the right view also seen by the left camera. Then there is an easy way to filter those common points.

One of the ways to pair the points is to create a graph where every node represents a point and edges of the graph are angles between theirs respective vectors. With this built graph it is possible to explore every permutation of point pairs and find the one with the smallest error to the right side.

$$g* = \min_{g \in G} \sum_{i=1}^{n} \sum_{j=i}^{n} (A(g_i, g_j) - A(gr_i, gr_j))^2$$

where A is the angle of given vectors.

Solution of this optimization task is a graph, which maps each point of left camera to points of right camera. However, this is a very naive method, since its complexity is factorial and therefore the method runs adequately fast only for smaller number of points.

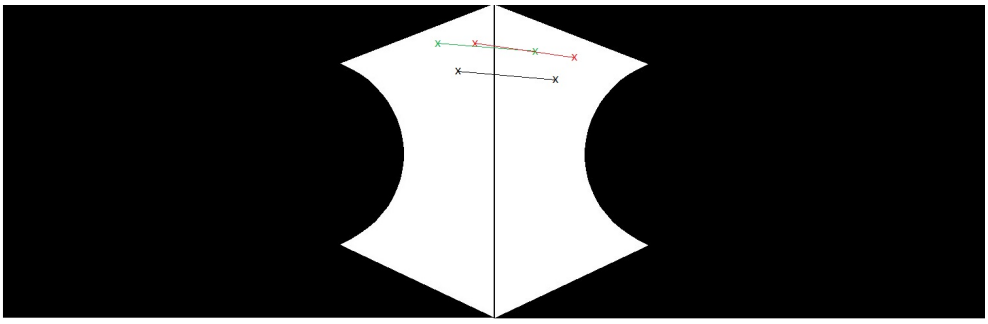

Figure 6.4. Example of paired UV lights to each other.

6.3.2 Error optimization for pairs of points

The idea of this method is to calculate a vector respective to the right camera based on a direction vector corresponding to a point seen in the left camera image and on the angle between the cameras. The right vector thus obtained can then be compared with a vector calculated by the *cam2world* function from the corresponding point in the right camera. It is possible to compare these vectors or their image. The images are compared by the square of the distances of the individual points. Vectors are compared with square of the angle between the vectors. The optimization function for

the angle between the cameras, which looks for a local minimum, therefore takes the sum of all errors calculated in this way. Theoretically, this function only needs to be differentiated according to the angle, and a descent along the error gradient could be performed iteratively. [14] This algorithm requires some initial estimate at the input (in our case it would be the expected relative orientation between the cameras), calculates the gradient of the function at this point and thy multiple of this gradient is subtracted from the current estimate. This would be done until the difference between the current and previous result was negligible.

However, the gradient of angular changes is difficult to calculate analytically. Assume that a relative change in the error function caused by a slight camera rotation step is calculated on a small scale - this is possible provided the changes are little. During large changes, there are problematic phenomena and it would be necessary to use quaternions or rotation matrices.

6.4 Epipolar lines approach

Epipolar lines are a geometric transformation of each camera point. In every stereopair, points from one camera can be visualized as a line in a second camera. The line represents every possible occurrence of a given point in the other camera, if the distance of the object is unknown. Since UVDAR uses wide angle lenses, the line is distorted into a curve. Chosen approach is to calculate epipolar curve for each point and in real time calculate its error.

6.4.1 Calculation of epipolar curves

To begin with, we need to calculate the curves. Calculation in stereo pair, where there are wide angle lenses, is very difficult analytically. However, we can obtain it simply by iterating over different lengths of the vector represented by given point. UVDAR does not provide transformation function with vector length. We take the point in infinity and transform that one. We take the transformed unit vector and iteratively add its scaled version to another vector representing relative position of the second camera. Then the resulting vectors are projected to the corresponding points in the second camera.

```
createPolynomialForPoint(point):
    worldP = cam2world(point)
    vecTransformed = transformer.transform(tfLeft, worldP)
    xAr, yAr = []
    for i in range(2, 0.1):
        vec = leftCamFromRight + vecTransformed * i
        point = world2cam(vec)
        xAr.append(point.x)
        yAr.append(point.y)
    return polyfit_boost(x, y, polynomialDegree);
```

Figure 6.5. Pseudocode of calculation epipolar curve for certain point.

Based on the observed curvature traced by the reprojected points, the author chose to represent for each pixel its epipolar line by a polynomial.

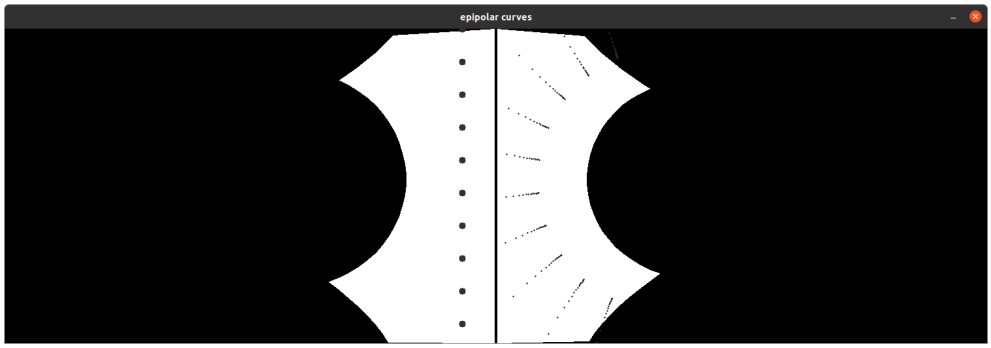

Figure 6.6. Example of a random set of points to which points representing different lengths of projected vector are calculated. The picture does not show the curvature well, but it is present and to represent those points by a line would result in great error.

To achieve this we use the `PolyFit`[5] library which has implemented functions for polynomial fitting on set of points. The returned parameters are then stored in 2D array on the corresponding pixel coordinates.

6.4.2 Distance measure

Calculating distance between a point and a polynomial or any other function is a topic which has been discussed extensively. The simplest approach is to derive distance function and find local minimum. Universal formula for distance from any function is:

$$d = \sqrt{(x - x_0)^2 + (f(x) - y_0)^2}$$

We aim to make the distance from function to be the smallest possible. Therefore we need to minimize said distance function. Since the square root does not alter the minimal x value (only y value) we can edit the function to be more easily differentiable.

$$(x - x_0)^2 + (f(x) - y_0)^2$$

We have to choose which degree of polynomial to use as $f(x)$. Using polynomial of degree 3 fits almost perfectly given points, but cannot be simply calculated using the derivative approach. The derivation step creates polynomial of higher degree of which the roots are harder to find. Therefore polynomial of degree 2 was chosen. The parabola fits the points as well as the polynomial of higher degrees, however the derivation step increases the degree to only 3 so using tools such as `WolframAlpha`[6] we can analytically calculate the roots:

$$((x - x_0)^2 + (ax^2 + bx + c - y_0)^2)' = 2((2ax + b)(ax^2 + bx + c - y_0) + x - x_0)$$

which can be re-arranged to:

$$2(2a^2x^3 + 3abx^2 - 2axy_0 + x + b^2x + 2acx - x_0 - by_0 + bc)$$

As we can see, the differentiated function is a polynomial of third degree, to which there exists a solution. A solution to

$$2(2a^2x^3 + 3abx^2 - 2axy_0 + x + b^2x + 2acx - x_0 - by_0 + bc) = 0$$

[5] `https://github.com/patLoeber/Polyfit`
[6] `https://www.wolframalpha.com/`

is

$$x = \frac{1}{6\sqrt[3]{2}a^2}\left((108a^4 + x_0 + 54a^3 + b + s)^{\frac{1}{3}}\right) -$$

$$-\frac{12a^3c - 12a^3y_0 - 3a^2b^2 + 6a^2}{3 \cdot 2^{\frac{2}{3}}\, a^2(108a^4x_0 + 54a^3b + s)^{\frac{1}{3}}} - \frac{b}{2a}$$

where

$$s = \sqrt{(108a^4x_0 + 54a^3b)^2 + 4(12a^3c - 12a^3y_0 - 3a^2b^2 + 6a^2)^3}$$

Since the distance between a point and a parabola is not ambiguous, there exists only one solution.

6.4.3 Optimization

Now we have a method which from every point creates epipolar curve and an error function. The function which is being optimized is some sort of black box in which calculation of epipolar curves and of error takes place. Therefore, analytical solution can not be found. In the first step there is in a general case no other way than to start with random angle between the two cameras. However, in our case the initial angle does not have to be completely random. We can expect the angle to be `almost` as it was from the beginning. Therefore the angle is chosen from starting point plus minus range for Euler's angles, which will by denoted by the aviation terms roll, pitch and yaw (see Figure 6.7).

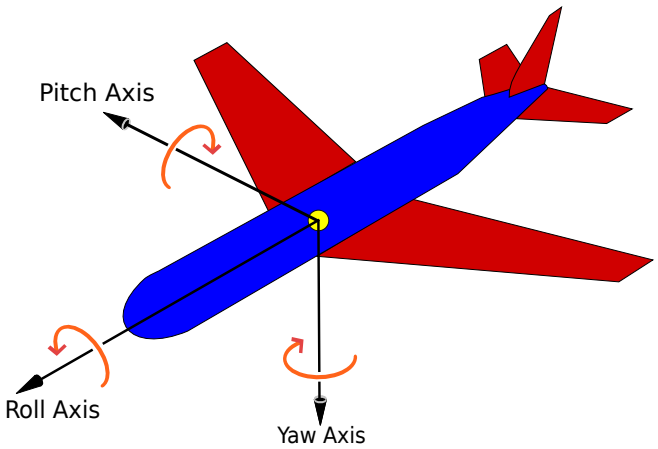

Figure 6.7. Roll, Pitch, Yaw (RPY) euler's angles. The airplane is a better representation of said terms, since UAV might confuse the roll and pitch terms. *(Jrvz. Courtesy of wikipedia.org)*

Once the optimization is done, the resulting RPY is then used as a starting point. But the question of how to choose the range remains. For the first iteration the range is set to be large enough to most likely capture the real angle difference. With that large of a range, the iterations do not find accurate solution quickly enough. However, they give us better understanding of how the stereo pair is differentiated. With the next iteration comes new starting point and the range is halved. With this approach the error function converges to minimum.

```
calibrateCameras(steps, range):
    left_size = left_queue.size()
    right_size = right_queue.size()
    min_queue_len = left_size if left_size < right_size else right_size
    left = getNHistory(left_queue, min_queue_len)
    right = getNHistory(right_queue, min_queue_len)
    min_error = INT_MAX
    best_quat = null
    for i in range(steps):
        quat = generateRandomQuaternion(range)
        for j in range(min_queue_len):
            left_points = left[j]
            right_points = right[j]
            for p in left_points + right_points:
                createPolynomialForPoint(p);
            er += epipolarError(points, polynomials)
        if er < min_error:
            min_error = er
            best_quat = quat
        if i = steps / 2 and min_er > threshold:
            range *= n
    applyQuaternion(quat)
```

Figure 6.8. Pseudocode of the optimization algorithm.

The algorithm can reach a sub-optimal local minimum. This should be avoided by expanding the set of points from which the calculations are done. The proposed solution takes into account points from history. Therefore the ambiguity should considerably decrease. Another problem occurs when some iteration returns new starting point and the given range does not include the desired solution. This way the solution might get closer each step, but it never gets to the optimal solution. To avoid this problem, if the error is large enough after certain number of steps, the range increases and a wider range of solutions is tested.

```
...
        if er < min_error:
            min_error = er
            best_quat = quat
        if i = steps / 2 and min_er > threshold:
            range *= n
...
```

Figure 6.9. Edited part the optimization algorithm, where the problems mentioned above are addressed.

Chapter 7
Results

In this chapter, calibration methods based on the epipolar approach are presented. The test results of the first method used are not shown since its robustness is highly limited by a number of UV lights seen in the common view. As mentioned earlier, the pre-existing methods are suited for a larger amount of points seen. However, the first method 6.3.1 is applicable only to a smaller number of points. Therefore none of these methods is suitable for our problem. The epipolar curves-based approach shows to be a fast and reliable method for calibration of stereo pair with wide-angle lenses with a prior close estimate of relative pose.

7.1 Filter of useful area

Figure 7.1. Area in the right camera image that is also visible in the left camera image.

7.2 Error function

The calculation of an error from section 6.4.2 is crucial for stereo pair calibration. In this section, various counts of UV lights seen are measured. The measuring is done on both sides of the stereo pair. In theory, it could be done only on one side since the rotation is reversed from the second camera. However, this increases the potential error after optimization, since unilateral calibration is more ambiguous than bilateral, and we would be discarding information on the specific pixel positions of the points from one of the cameras.

First, we will show the error of reprojections of one UAV with the same height as the stereo pair.

As shown in the above figure, the greatest error is caused by roll rotation, followed by pitch. The yaw rotation itself causes, as expected, almost no error since the height of the observer and UAV observed are the same. The epipolar curve for points in the center of y axis is a horizontal line and by rotating the yaw angle, the points move on

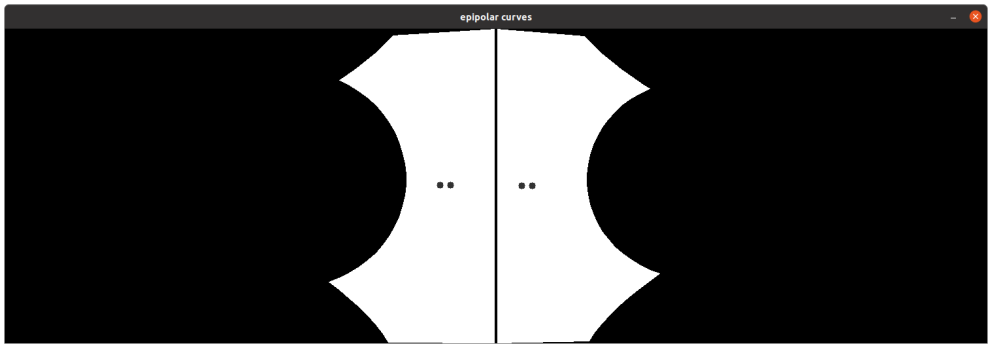

Figure 7.2. One UAV in common view on the same height facing the camera directly.

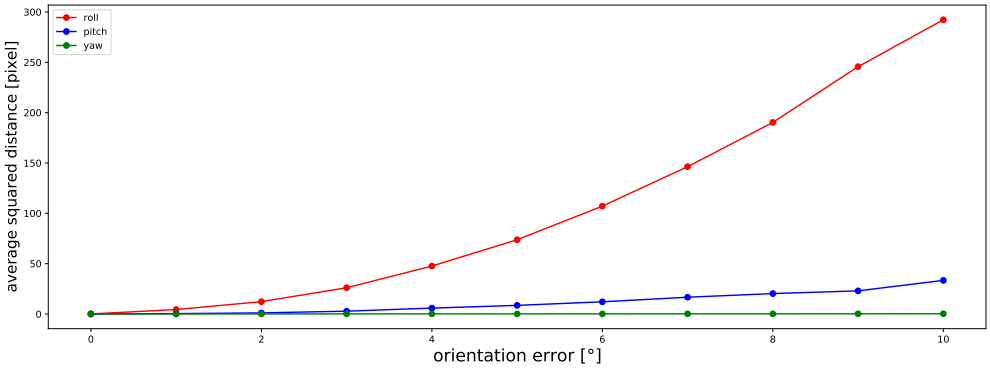

Figure 7.3. Average errors for roll, pitch and yaw rotation.

the same epipolar line so the error does not increase. There is a minor change in the error as we approach larger angles. This is caused by inexact world conditions. The height is not exactly the same, therefore the rotation does move points slightly away from the curve.

This error measurement was not really showing true error since the UAVs were static and on the same height. The plot in figure 7.5 shows error induced by offset in orientation about each axis for more UAVs in different parts of the common view. It is still static, but the yaw rotation should increase the error.

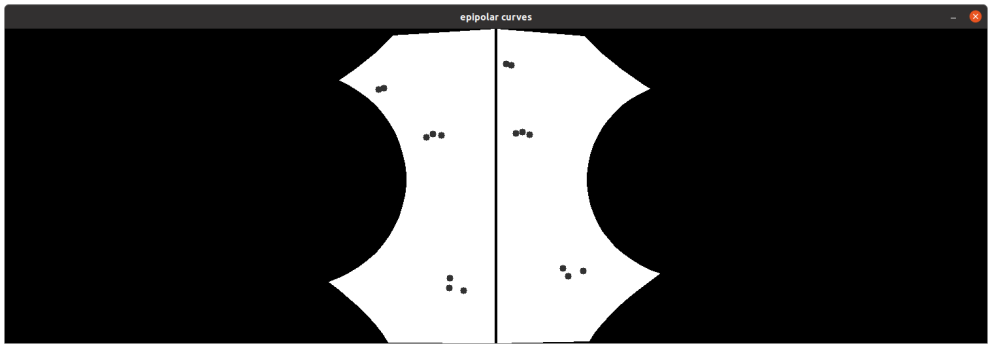

Figure 7.4. Three UAVs in different corners of common view.

The error in scenario 7.4 is significantly larger than before. This is due to the number of points seen. Since the error adds up the distances for each point and its corresponding epipolar curve. This does not cause issues when optimizing only from one set of points

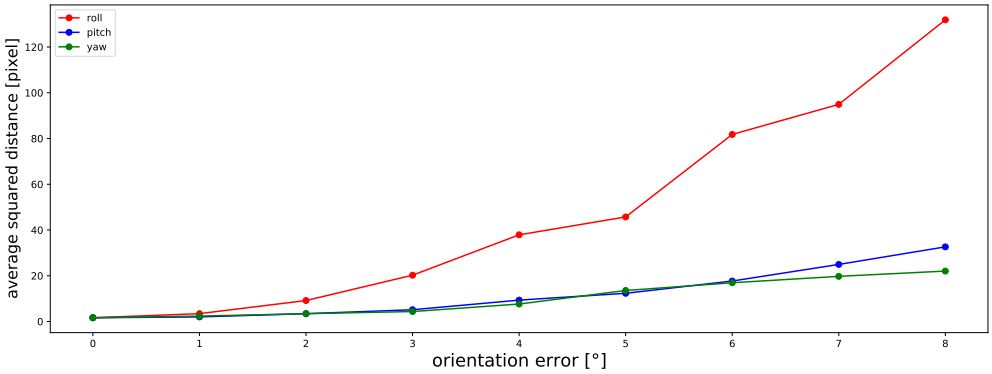

Figure 7.5. Average errors of scenario 7.4 for roll, pitch and yaw.

as the points on the right side and left side must have the same cardinality. However, the problem might occur when looking into history. There might be stereo pair images, where the number of points differs from others in history. Therefore the error function need to be normalized by the number of points seen.

As we can see, the growing trend of error is still present in each rotation even by as small an angle as 1°. The greatest error is once again caused by the roll rotations. Pitch and yaw show less significant differences in smaller angles, but the error induced by a shift in pitch grows much more rapidly as the points move further away from the horizontal line as the image is more distorted there.

The tests indicate, that the chosen error function can be used effectively to minimize the error in estimated relative camera pose. That is a significant step, since the existing methods only applied for a greater number of points detected. The proposed error function works for as few as 2 points and for as many as the core can handle.

However, situations, when the observed UAV is close to the cameras, is challenging for the proposed method.

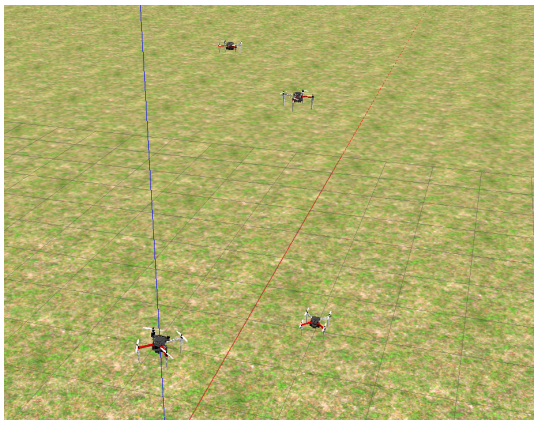

Figure 7.6. The scene from which the data from 7.5 were obtained. The image is shot in `Gazebo simulator` [1].

As can be seen, the epipolar curves generated for closer points are not as accurate as the ones for the further UAVs. The closest UAV is 3 meters away from the observing UAV and the middle UAV is 5 meters. The assumption is that the closer the points are, the less accurate the epipolar curves become. We will test this hypothesis. The

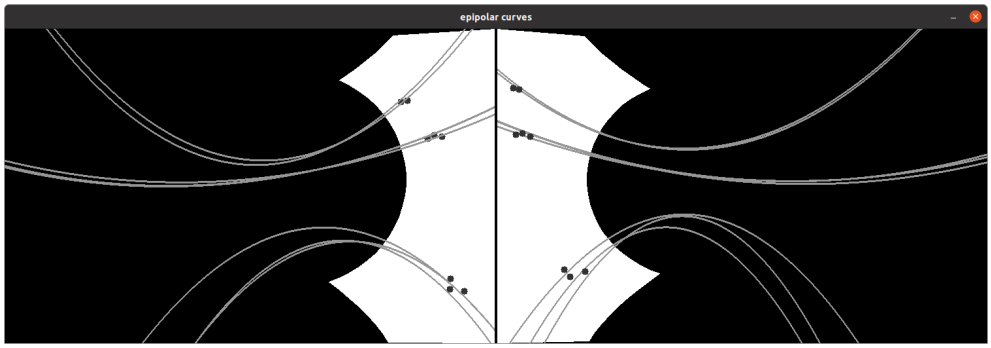

Figure 7.7. Scene from the view of a stereo pair with generated epipolar curves.

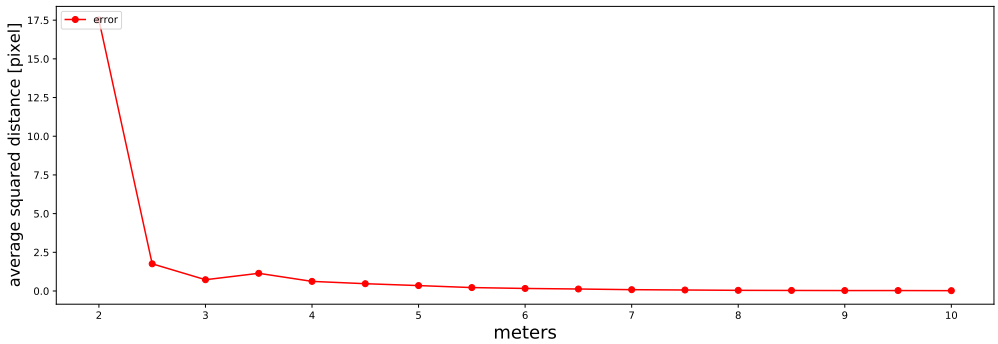

Figure 7.8. Error as the observer UAV furthers away from the scene.

scene is created by two UAVs, one of which is the observer and is static at the height of one meter. The second one is on the ground placed at a range of distances in front of the observer.

The graph in Figure 7.8 shows a large value of the proposed error function when the UAV is close to the observer. This can be also caused by an inaccurate initial orientation. Let us consider the same scenario, apply the calibration function and show the epipolar curves when the observing UAV increases its distance from the observed UAVs.

As we can see in Figure 7.9, when optimizing from a closer distance, the solution is relatively accurate.

The optimization seems to have found a suitable solution even from a closer view in Figure 7.10. There are visibly greater errors than in the previous scenario. However, the improvement from the initial values is still significant.

Distant UAVs have a very small error. However, due to the limited camera resolution, distant objects might appear fit the epipolar curves correctly. We can fairly easily estimate the accuracy of the proposed error function by optimizing from the same scenario several times and calculating the covariance matrix. This can be done thanks to the algorithm using a random seed. First, we need an average of quaternions, a solution to this problem is well known [15].

7.3 Optimization

With the error function, it is simple to find the minimum. The following experiment focuses on testing speed and accuracy depending on the number of steps and chosen

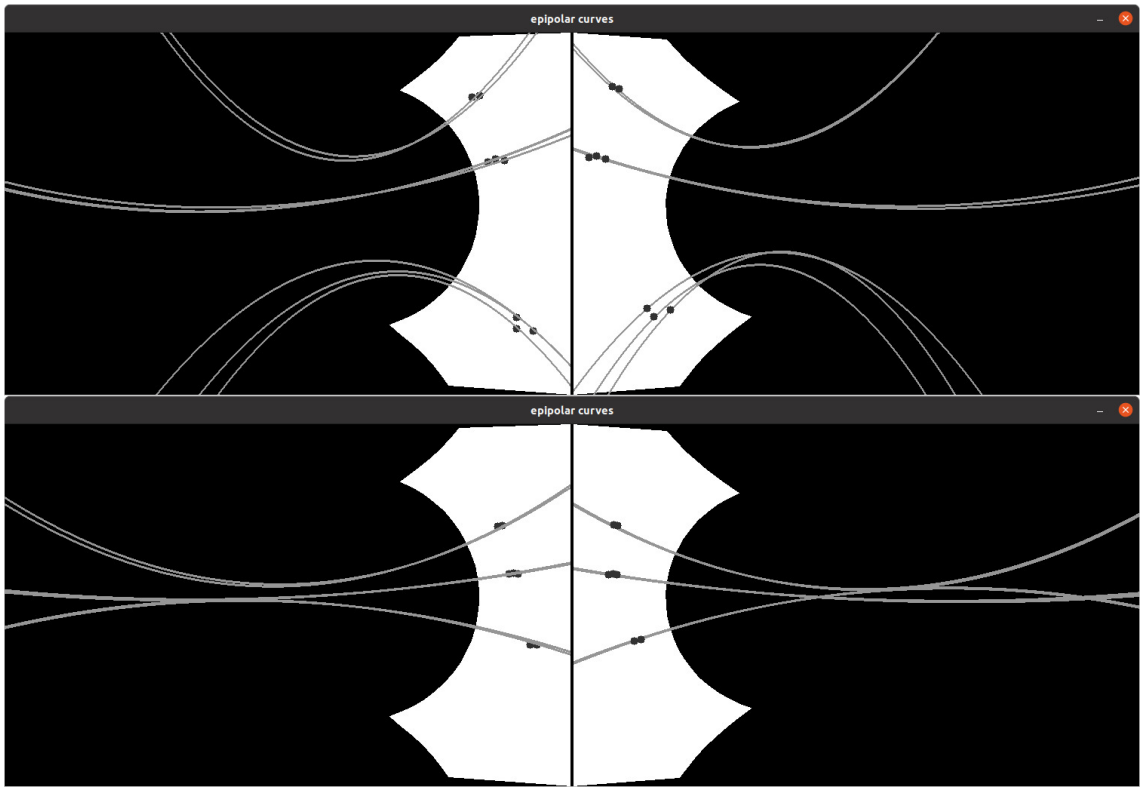

Figure 7.9. Calibration done when the scenario was set as before, meaning the observing UAV is close to the observed UAVs.

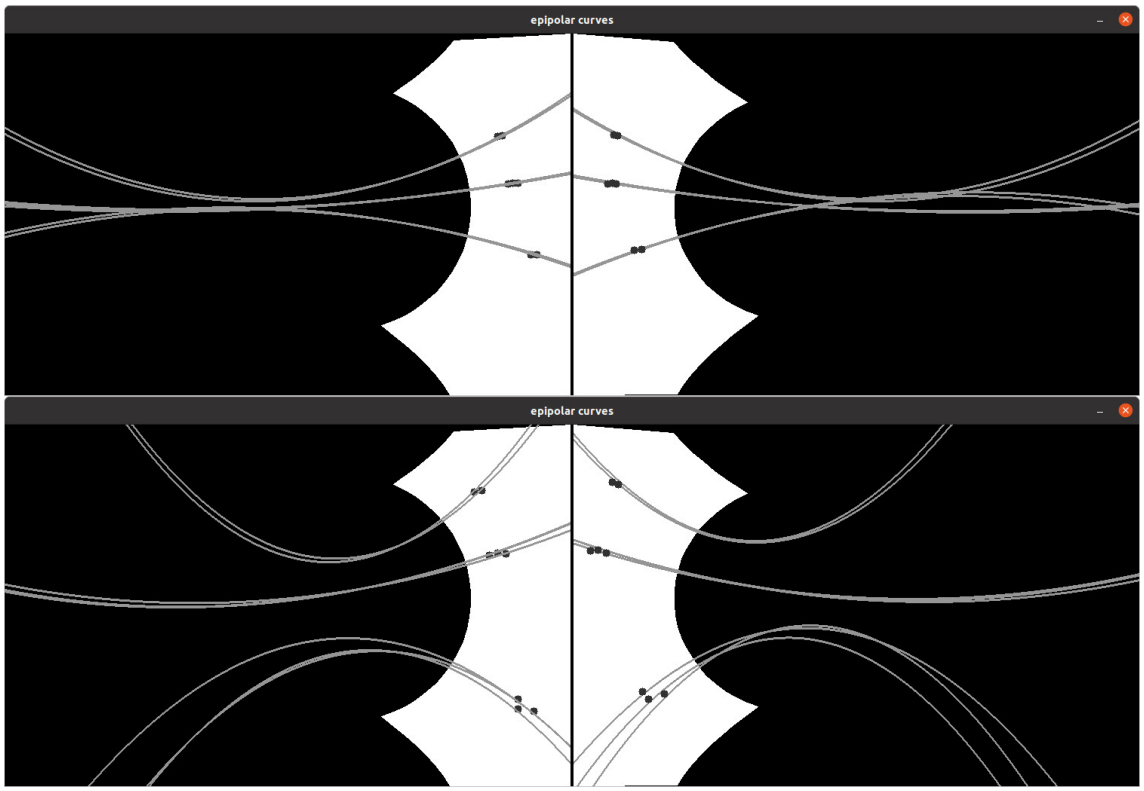

Figure 7.10. Calibration done when the observing UAV is at least 8 metres away from the closest UAV.

range from which Euler's angles are generated. The speed also depends on how far back in time the iteration looks and the number of points in set. However, that value is not changed in the speed experiment.

All the following measurements were processed using 5 images from history and 7 points representing 7 arms of 3 UAVs, with 20° of range. Each measurement is repeated 5 times in order to receive more statistically significant results.

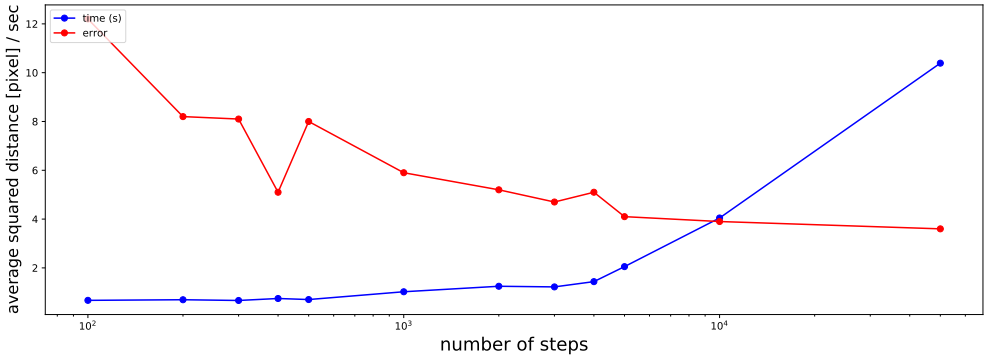

Figure 7.11. Time and error depending on number of iteration steps of optimization algorithm (x log scale).

The accuracy starts to decrease at around the 1000 iteration steps. The calibration is relatively fast, considering the naive optimization algorithm and 7 UV lights in the common view, which is much more than we could expect in the common view. The algorithm can do optimization with an error of 9.7 every second, which means the external influences can cause 20° difference in every axis of rotation and the function would still find its minimum. We will explore the accuracy of what it would look like with a range of only 10. This is a more realistic scenario. The time is not measured since the range does not affect the time spent on the calculation.

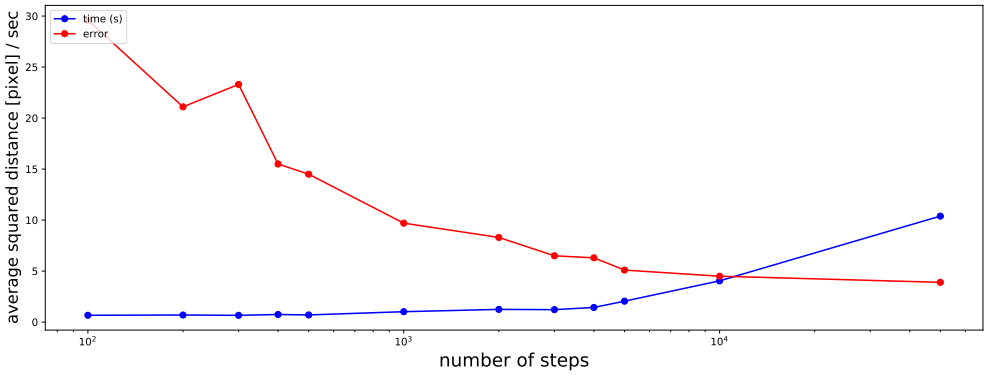

Figure 7.12. Error depending on number of iteration steps with 10° range (x log scale).

As can be seen in Figure 7.12, with a higher number of steps, the error does not significantly improve. However, splitting 1000 steps, the accuracy improve more than with 20° of range.

Chapter 8
Conclusion

Different methods of camera calibration with wide angle lens in the stereo pair were analyzed in terms of speed and functionality. Many prior solutions for the problem of extrinsic calibration of a stereo pair exist, but they were determined not to be suited for our current system. They are either meant for pin hole model cameras or there must be a minimum number of points from which to calibrate. None of the above is typically true in the UVDAR systems. The author presented methods that will substantially improve the error rate of mutual relative localization of cooperating UAVs using UVDAR system. The speed of this system might be improved, as naive random optimization method was implemented. However, the black box error function was shown to be adequate for the stereo pair calibration refinement we set out to achieve. Further research is needed to optimize the proposed method in terms of computational expense and test of a working prototype of the system in a real environment is planned.

It is possible to combine existing solutions with the proposed. The proposed solution is unique in that it does not need many points to achieve calibration. Existing methods work for at least 7 points. Therefore using proposed solution for smaller amount of points and existing solutions for larger amounts might be a good idea.

The aim of this project was to research existing methods of stereo pair calibration and to propose and implement one that is suitable for UVDAR system. The research showed very few methods suitable for our environment, with none of them fitting it perfectly. The author proposed new solution using epipolar curves and an error function to effectively find a good approximation of relative angle rotations of the two cameras in stereo pair.

References

[1] Tanvir Kazi, and Siddiqui Ahmed. *A Realistic Simulation for Swarm UAVs and Performance Metrics for Operator User Interfaces*. 2017.
https://scholarworks.unr.edu/bitstream/handle/11714/2095/AhmedSiddiqui_unr_0139M_12412.pdf?isAllowed=y&sequence=1.

[2] Walter Viktor, Staub Nicolas, Franchi Antonio, and Saska Martin. *UVDAR System for Visual Relative Localization with application to Leader-Follower Formations of Multirotor UAVs*. 2019.
https://homepages.laas.fr/afranchi/robotics/sites/default/files/2019g-WalStaFraSas-preprint.pdf.

[3] Li Jia, Duan Ping, and Wang Jinliang. *Binocular Stereo Vision Calibration Experiment Based on Essential Matrix*. 2015.
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7387576.

[4] Zhang Zhengyou. *Camera Calibration with One-Dimensional Objects*. 2004.
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1300559.

[5] Heikkila Janne. *Geometric camera calibration using circular control points*. 2000.
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=879788.

[6] Fu Qiang, Quan Quan, and Cai Kai-Yuan. *Calibration of Multiple Fish-Eye Cameras Using a Wand*. 2014.
https://www.researchgate.net/publication/263699526_Calibration_of_Multiple_Fish-Eye_Cameras_Using_a_Wand.

[7] Chum Ondrej, and Matas Jiri. *Optimal Randomized RANSAC*. 2008.
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4359381.

[8] McManamon Paul. *LiDAR Technologies and Systems*. 2019.
https://aleph.cvut.cz/F?func=direct&doc_number=000821821&local_base=DUPL&format=999.

[9] Broggi Alberto. *A closer look at LiDAR and stereovision*. 2020.
https://www.ambarella.com/blog/a-closer-look-at-lidar-and-stereovision/.

[10] Scaramuzza Davide. *Omnidirectional Camera Calibration Toolbox for Matlab*. 2009,2013.
https://sites.google.com/site/scarabotix/ocamcalib-omnidirectional-camera-calibration-toolbox-for-matlab.

[11] Scaramuzza Davide, Martinelli Agostino, and Roland Siegwart. *A Flexible Technique for Accurate Omnidirectional Camera Calibration and Structure from Motion*. 2006.
https://rpg.ifi.uzh.ch/docs/ICVS06_scaramuzza.pdf.

[12] Hartley Richard, and Zisserman Andrew. *Multiple View Geometry in Computer Vision*. 2004.
http://www.r-5.org/files/books/computers/algo-list/image-processin

g/vision/Richard_Hartley_Andrew_Zisserman-Multiple_View_Geometry_in_Computer_Vision-EN.pdf.

[13] *Multivariate Abel–Ruffini*. 2015.
https://link.springer.com/content/pdf/10.1007/s00208-015-1309-6.pdf.

[14] Yin Jianghua, Wang Lingzhi, and Jiang Xianzhen. *A modified PRP conjugate gradient method with Armijo line search for large-scale unconstrained optimization*. 2017.
https://ieeexplore.ieee.org/document/8027748.

[15] *Averaging Quaternions*.
http://www.acsu.buffalo.edu/~johnc/ave_quat07.pdf.