UKRAINIAN CATHOLIC UNIVERSITY

BACHELOR THESIS

Multi-camera visual obstacle avoidance for micro aerial vehicles

Author:
Mykola MORHUNENKO

Supervisor:
Ing. Matouš VRBA

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science

in the

Faculty of Applied Sciences
Department of Computer Sciences

Lviv 2022

Declaration of Authorship

I, Mykola MORHUNENKO, declare that this thesis titled “Multi-camera visual obstacle avoidance for micro aerial vehicles” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"Science, my lad, is made up of mistakes, but they are mistakes which it is useful to make, because they lead little by little to the truth."

Jules Verne

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences
Department of Computer Sciences

Bachelor of Science

Multi-camera visual obstacle avoidance for micro aerial vehicles

by Mykola MORHUNENKO

Abstract

The 21st century is a time of innovation and exploration in the fields of applied science such as physics, medicine, biology, programming and robotics, as well as their intersections and fusions. One of the research topics, that have recently gained much popularity, are Micro unmanned Aerial Vehicles (MAVs). MAVs became smaller, cheaper and more readily available. However, due to the rising popularity and utility of MAVs, some of their problems and limitations are highlighted. MAVs often rely on the Global Navigation Satellite System (GNSS), but due to GNSS inaccuracy in closed environments, the MAV requires an obstacle avoidance system that is compact and reliable.

In this thesis, a compact and reliable visual multi-camera obstacle avoidance system for MAVs is developed. Calibration of a non-planar stereo camera setup and extraction of obstacle positions in an unknown environment from pairs of 2D images are tackled in this work.

The proposed solution is designed to run onboard an MAV with a limited computational power considering size, weight and payload limitations. Performance of a prototype of the proposed solution was measured in laboratory experiments. The result proved that the system is ready for on-drone deployment and real-life tests.

Acknowledgements

I would like to thank my supervisor Ing. Matouš Vrba and all CTU MRS Group members for the opportunity to have an internship there for the last one and a half year, all of them were kind, they helped me a lot with advice's and shared their experience, CTU FEE for interesting and useful for this thesis courses. I want to admire that nothing during my studies would take place without my small family - Ukrainian Catholic University, especially Applied Science Faculty, all teachers and deanery for saving all students from all the worst sides of Ukrainian education system and providing only the best quality education without corruption and plagiarism.

Especially, I want to thank all defenders of my Motherland, The Armed Forces of Ukraine, who protect the whole Europe during the russo-Ukraian war at the cost of their own lives to make it possible for all of us to live in peace and for me to write this thesis.

Finally, I would like to thank my family who supported me during my whole life, my mother Svitlana and my father Roman.

Contents

List of Figures

List of Tables

List of Abbreviations

UAV	**U**nmanned **A**erial **V**ehicle
MAV	**M**icro Unmanned **A**erial **V**ehicle
ROS	**R**obot **O**perating **S**ystem
DoF	**D**egree **o**f **F**reedom
SLAM	**S**imultaneous **L**ocalization **A**nd **M**apping
SfM	**S**tructure **f**rom **M**otion
PnP	**P**erspective **n** **P**oints
FOV	**F**ield **O**f **V**iew
SVD	**S**ingular **V**alue **D**ecomposition
MRS	**M**ulti **R**obot **S**ystems Group
fps	**f**rames **p**er **s**econd
LiDAR	**L**ight **D**etection and **R**anging
RMS	**R**oot **M**ean **S**quared

List of Symbols

$\vec{t}$	a column vector that reresents a point or a vector
A	a matrix
$\mathsf{A}^\top$	transpose of a matrix
R	a rotation matrix 3x3, $\det(\mathsf{R}) = 1, \mathsf{R}^\top = \mathsf{R}^{-1}$
I	the identity matrix
l	a line
f	focal length
$\vec{x} \times \vec{y}$	cross product of $\vec{x}$ and $\vec{y}$
$[\vec{x}]_\times$	such matrix that $[\vec{x}]_\times \vec{y} = \vec{x} \times \vec{y}$
λ	any non-zero scalar
$\|\vec{a}\|$	the norm of the vector $\vec{a}$

Chapter 1

Introduction

Micro unmanned Aerial Vehicles (MAVs, also called drones or Unmanned Aerial Vehicles, UAVs) recently saw a rise in usage across various fields. Drones are already widely used in cinematography [1], advertising [2] and agriculture [3]. City emergency departments also use MAVs - firefighters can see and evaluate the situation from the sky, localize the source of fire and put it out [4]. Another field where MAVs can be used in the nearest future is transportation. Fast parcel delivery [5] and content transportastions [6, 7] are quite promissing fields of MAV application together with smart city concepts evolving [8]. Nowadays, even collaborative transportation systems are becoming realistic - multi-robot swarming algorithms are better developed, and that allows their usage for the transportation of large objects [9] that one drone alone can not lift. MAVs are also widely used in the military industry.

As drones are used so widely, there is a significant demand for an increase in drone-related safety. Many commercially available drones are expensive and quite heavy, so that accidents can be costly and dangerous to property and human health. Even if a drone is controlled remotely, obstacles can be hard to spot and avoid when the pilot flies far away. Similar limitations apply to flying in forests or cities (in situations when it is legal, for example - emergency departments and moviemakers can do it), where it may be hard to maintain the line of sight for the pilot. First-person view glasses may be used in such situations to mitigate these limitations to some degree, but still, only provide limited information about the surrounding environment. Most of these systems only provide a field of view (FOV) smaller or comparable to the FOV of a human eye (which is approx. a third of the full sphere FOV [10]). In this sense, autonomous robots can sense and avoid obstacles much better, but only if they have a well-designed system running onboard and enough sensors to cover the area around the robot.

Automatic obstacle avoidance systems become especially important in closed environments with many obstacles, such as forests, cities, or indoor environments. To ensure complete coverage of the surroundings of the MAV for the collision avoidance system, it can be equipped with multiple sensors pointing in all directions.

Considering this context, a compact obstacle avoidance system is a perspective field for research. Even though the idea is not new, there is no complete, publicly available visual obstacle avoidance system for MAVs that can cover the whole space around them with few sensors. One of the best is perhaps the Skydio system[1] which has six 4k 200° navigation cameras, and neural networks are used for the mapping, obstacle avoidance and path planning [11]. It is available only for the US military. The DJI Mavic 3 drone[2] seems to be a better solution with eight cameras and an infrared sensor at the bottom. Unfortunately, neither of these companies provide

[1]Skydio autonomy: https://www.skydio.com/skydio-autonomy
[2]DJI Mavic 3: https://www.dji.com/cz/mavic-3

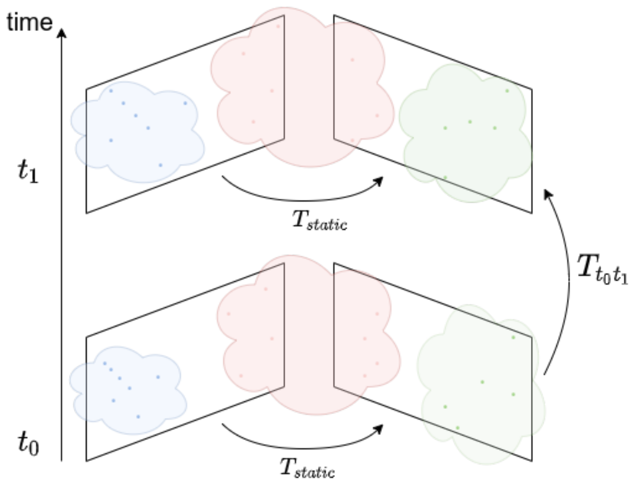


FIGURE 1.1: A general scheme of the multi-camera obstacle detection problem.

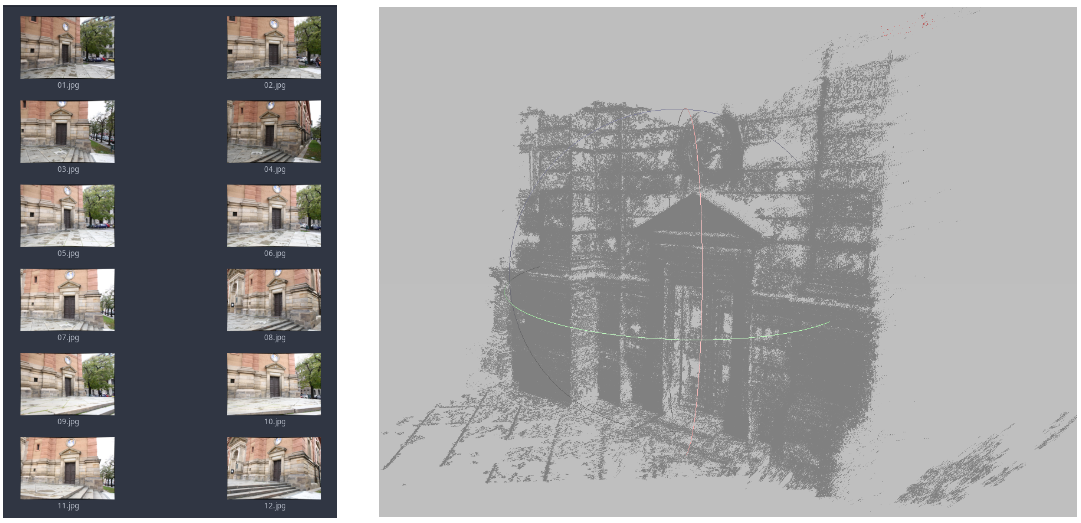
any scientific publications, system specifications or implementation details, so we can only conclude from the limited available information.

Figure 1.1 presents a scheme of the proposed system. In the figure, T_{static} is the transformation between two cameras mounted onboard the MAV. T_{static} is obtained with stereo pair calibration. At time t_0, a new pair of images is captured by the cameras. A feature detector extracts points of interest from the images. Points that lie in the part of the images corresponding to a section of the cameras' field of view that overlaps are then selected (the red point cloud in Figure 1.1), and correspondence matches between these points from the two images are found based on their feature descriptors. A calibrated projection model of the cameras and the transformation T_{static} are used to estimate 3D points from the matched point pairs. Then at time t_1, when a new pair of images is received, the same process is repeated. In parallel, an incremental Structure from Motion (SfM) algorithm processes sequences of images from each camera separately to estimate 3D points corresponding to nearby objects in the environment. However, monocular SfM algorithms can generally only estimate the environment up to an unknown scaling factor [12]. To address this problem, the red point cloud is used to find the scale of the points obtained using the SfM algorithm from images captured at t_0 and t_1 (the blue and green point clouds in Figure 1.1).

1.1 Related Works

There are many approaches to tackling MAV obstacle avoidance in the published literature using various sensors. In most articles, a stereo pair of two parallel cameras looking in the same direction (classical stereo pair) [13, 14, 15, 16] are used for that, but there are also approaches relying on monocular vision [17, 18, 19] or 2D or 3D Light Detection and Ranging (LiDAR) sensors [20]. Distance sensors relying on ultrasonic sound waves and time of flight sensors have low accuracy and are sensitive to noise, which is why they are typically not used alone but in combinations with other sensors [21, 22]. There are also convolutional neural network-based approaches for depth estimation from monocular cameras [18, 23, 24] and from stereo cameras [16].

These sensors have various advantages and shortcomings, making them suitable for different applications [25, 26]. 3D LiDARs are relatively heavy and expensive but can provide good precision and a 3D coverage of the environment. 2D LiDARs, which are typically more lightweight and significantly cheaper, have successfully been deployed on ground vehicles. However, they are not as suitable for onboard deployment on MAVs because, unlike ground vehicles, MAVs can have three degrees of translational movement, so a 2D LiDAR cannot cover all possible directions of movement, making it unsuitable for robust collision avoidance. Stereo camera sensors are generally more expensive and complex than monocular cameras but can provide precise depth images. Industrial stereo cameras typically come with a software development toolkit (SDK) and programming libraries, they are already calibrated from the factory and ready to be deployed and used, which makes them more user-friendly than a custom made stereo pair. However, it is infeasible to alter the hardware specifications in most cameras, for example, changing FOV or camera lenses. Such hardware limitations make it challenging to use in some research applications. As already mentioned, ultrasonic and infrared sensors have a low precision, relatively short range, and are sensitive to noise. However, they are often used because they are relatively small, cheap and light.



(A) Input images. (B) Output of an SfM algorithm.

FIGURE 1.2: Ilustration of Structure from Motion, source - `https://github.com/Myralllka/CTU_3d_computer_vision/`

Real-time Simultaneous Localization And Mapping (SLAM) systems can also be used for obstacle avoidance [27]. These problems are closely related. SLAM keeps track of the robot's position while constructing and updating a map of an unknown environment. At the same time, obstacle avoidance is a problem of detecting and avoiding the nearest obstacles in an unknown environment to keep the robot safe from harmful collisions. Both problems are related to making a 3D map of an unknown environment, but the precision of distance measurements to the nearest objects is much more critical for obstacle avoidance. Sometimes SLAM can be a considerable overhead because it usually includes saving, updating the map, and robot localization in the map, while obstacle avoidance only needs real-time information about the robot's surrounding.

Structure from Motion (SfM) is another family of algorithms that may be used to implement a system for avoiding obstacles. SfM is a method of 3D reconstruction from a continuous sequence of images as illustrated in Figure 1.2. Using this approach, a dense point cloud as in Figure 1.2b can be computed, and obstacles can be detected [28]. This algorithm alone has multiple problems. Firstly, it can not recover information about such parts of an image as the robot's shadow or any other object moving with the same velocity in the same direction as the camera. Secondly, it requires images from at least two different views to work. If there are moving objects, their correct position relative to a moving camera can not be obtained. These problems can be tackled by combining SfM with other algorithms.

1.2 Problem definition

This thesis aims to design a visual obstacle avoidance system for MAVs, create a physical device implementing the system and measure its performance. The proposed solution assumes an MAV with a limited size and lifting force that constrains the number, weight and size of its onboard sensory equipment. The MAV is thus equipped with two calibrated cameras with a known transformation between them. The fields of view of the cameras overlap enough to detect close obstacles. Framerate of the cameras is sufficient to operate in real-time, and the frames are synchronized in time. The MAV is also equipped with an onboard computer with enough computational power to process the images at a rate sufficient for obstacle avoidance. The flight environment is assumed to be well-lit and contains objects with well-distinguishable visual features that the cameras can observe. These features should be unique so that the same feature can be unambiguously matched in images from the two cameras. The expected output is a reconstruction of the 3D environment around the drone in the form of a set of points representing the nearest objects. Depending on the estimated distance, these are the possible obstacles that should be avoided. The system can be integrated with the MAV control system to provide obstacles, so the solution working rate on the MAV's hardware should be sufficient for agile manoeuvering and obstacle avoidance.

Chapter 2

Preliminaries

2.1 Homogenous coordinate systems

A homogenous coordinates system is a mathematical tool to simplify certain geometrical operations by adding an extra dimension. For example, the general scaling, rotation and translation are expressed in non-homogenous coordinates as:

$$\vec{m}_1 = \mathsf{SR}\vec{m}_0 + \vec{t}, \tag{2.1}$$

$$\mathsf{S} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}, \ \mathsf{R} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}, \ \vec{t} = \begin{bmatrix} t_x \\ t_y \end{bmatrix}, \tag{2.2}$$

where m_0 is the original point, m_1 is the transformed point, S is a scaling matrix, R is a rotation matrix and $\vec{t}$ is a translation vector.

The same operations can be expressed in a homogenous coordinate system using only matrix multiplication as

$$\begin{bmatrix} \vec{m}_1 \\ 1 \end{bmatrix} = \begin{bmatrix} \mathsf{SR} & \vec{t} \\ \vec{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \vec{m}_0 \\ 1 \end{bmatrix} = \mathsf{S}_H \mathsf{R}_H \mathsf{T}_H \begin{bmatrix} \vec{m}_0 \\ 1 \end{bmatrix}, \tag{2.3}$$

$$\mathsf{S}_H = \begin{bmatrix} s_x & 0 & 1 \\ 0 & s_y & 1 \\ 0 & 0 & 1 \end{bmatrix}, \ \mathsf{R}_H = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \ \mathsf{T}_H = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.4}$$

where $\mathsf{S}_H, \mathsf{R}_H$ and T_H are homogenous transformation matrices.

So in the homogenous coordinate system, these transformations can be combined and expressed as matrix multiplications.

2.2 Pinhole camera model

A pinhole camera, the canonical perspective camera model - is a model of a simple camera without any optics. The first example is the camera obscura - a dark room with a small hole through which the image from outside is projected on the opposite wall. This model can be used to express camera geometry with a field of view angles less than 180°.

2.2.1 Camera coordinate system

In the physical implementation of the camera obscura, the projective plane is on the opposite side from the projection center (or camera center $\vec{C}$ in the pinhole camera model), and the image is reversed and mirrored. However, in most computer vision literature, authors assume that it is on the same side as the object (see Figure 2.1).

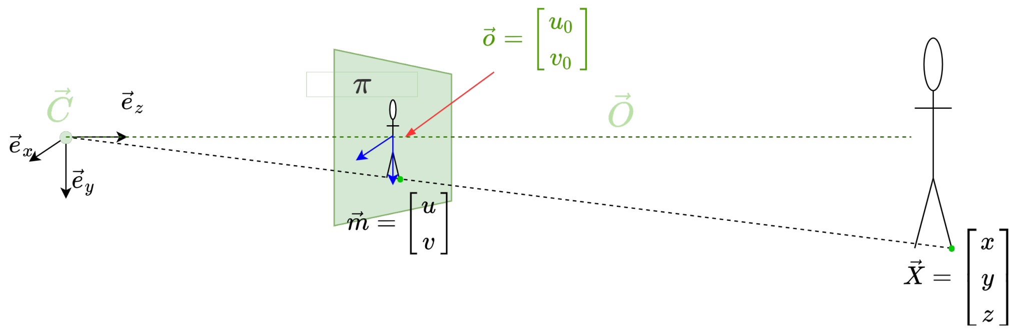

$\vec{o} = \begin{bmatrix} u_0 \\ v_0 \end{bmatrix}$
$\vec{C}$
$\vec{e}_z$
$\vec{e}_x$
$\vec{e}_y$
π
$\vec{O}$
$\vec{m} = \begin{bmatrix} u \\ v \end{bmatrix}$
$\vec{X} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$

FIGURE 2.1: The scheme of a pinhole camera model.

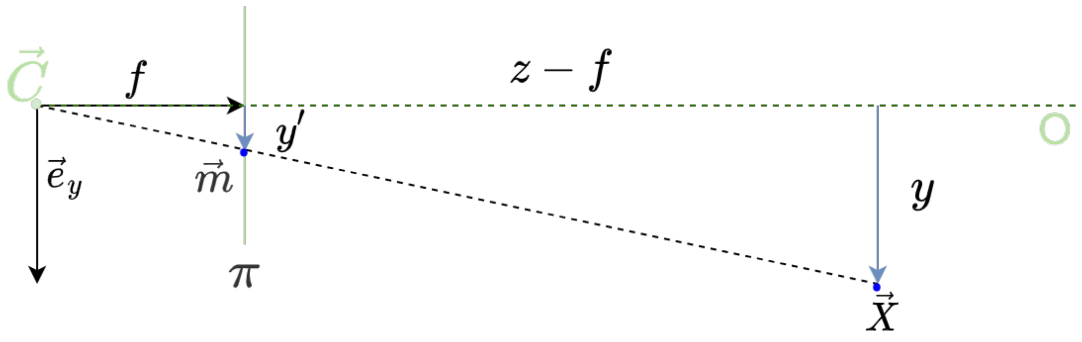


FIGURE 2.2: The pinhole camera model, y-z plane.

In Figure 2.1 a camera with camera center $\vec{C}$ in a coordinate system with origin at $\vec{C}$ and basis vectors $(\vec{e}_x, \vec{e}_y, \vec{e}_z)$ is observing a human. Each point $\vec{X} = (x, y, z)^\top$ in a world coordinate system has a projection $\vec{m} = (u, v)^\top$ on a plane π which is located at distance f from the camera center (refer to Figure 2.2). The optical axis $\vec{O}$ is a ray perpendicular to plane π, and on the image the point $\vec{O} \cap \pi = \vec{o}$ is the center of the image, see Figure 2.3.

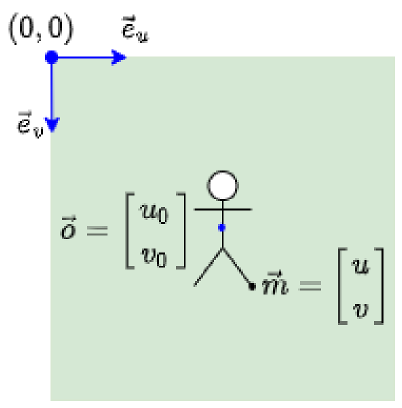
2.2.2 Camera matrix

The camera calibration matrix is a matrix that includes the camera's *intrinsic* parameters - focal length, pixel skew angle θ (illustrated in Figure 2.4) and principle point coordinates $\vec{o} = (u_0, v_0)$ (see Figure 2.3). The camera calibration matrix can be expressed as

$$\mathsf{K} = \begin{bmatrix} f_x & -f_x \cot(\theta) & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}, \text{ units: } [f_x] = \text{px}, \ [f_y] = \text{px}, \ [u_0] = \text{px}, \ [v_0] = \text{px}, \tag{2.5}$$

where $f_x = \frac{f}{\|\vec{e}_u\|}$ and $f_y = \frac{f}{\|\vec{e}_v\| \sin\theta}$ represents the focal length of a camera in the horizontal and vertical image units. $\vec{e}_u$ and $\vec{e}_v$ are the image basis vectors (refer to Figure 2.4) and f is a focal length, $[f] = m$.

Most modern digital cameras have no skew and square pixels, so in most cases, the camera matrix can be simplified to:



$(0,0)$
$\vec{e}_u$
$\vec{e}_v$
$\vec{o} = \begin{bmatrix} u_0 \\ v_0 \end{bmatrix}$
$\vec{m} = \begin{bmatrix} u \\ v \end{bmatrix}$

FIGURE 2.3: The pinhole camera model, x-y plane.

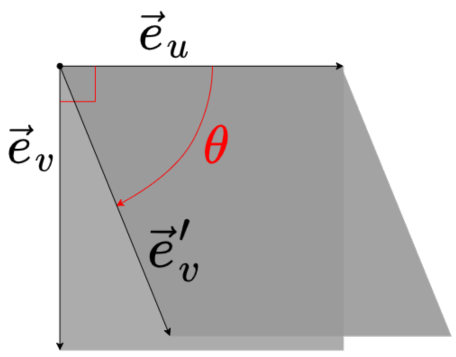


FIGURE 2.4: A scheme of a skewed pixel. $\vec{e}_u$ and $\vec{e}_v$ are the image basis vectors, where $\|\vec{e}_u\| = \frac{w_s}{w_{im}}$, $\|\vec{e}_v\| = \frac{h_s}{h_{im}}$, w_s and h_s are physical width and height of a camera sensor, w_{im} and h_{im} are the image width and height in pixels. $\vec{e}\,'_v$ is an edge of the skewed pixel and θ is the skew angle.

$$\mathsf{K} = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.6}$$

2.2.3 Projection matrix

The image projection matrix P is used to translate a point from a world coordinate frame to an image coordinate frame. The canonical projection matrix P_0 assumes that the camera is in the world coordinate center and that the calibration matrix $\mathsf{K} = \mathsf{I}$. P_0 is expressed as

$$\mathsf{P}_0 = \begin{bmatrix} \mathsf{I} & | & \vec{0} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{2.7}$$

However, this case is impractical. The canonical projection matrix is not used in practice because each real camera is different. Instead, the image projection matrix P is used with a camera matrix K to transform the canonical P_0 to the specific P:

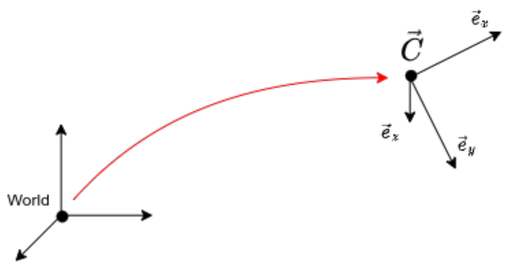


FIGURE 2.5: Transformation from world to camera coordinate frames.

$$\mathbf{P} = \mathbf{K}\left[\mathbf{I} \mid \vec{0}\right] = \begin{bmatrix} f_x & 0 & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{2.8}$$

The world coordinate center is usually not located at point $\vec{C}$ as illustrated in Figure 2.5. It may be rotated using a rotation matrix $\mathbf{R}$ and translated by a vector $\vec{t}$ where $\mathbf{R}$ is a $3x3$ matrix with $\det(\mathbf{R}) = 1$ and $\mathbf{R}^{-1} = \mathbf{R}^\top$. Therefore, in the general case the projection can be expressed as

$$\mathbf{P} = \mathbf{K}\left[\mathbf{R} \mid \vec{t}\right] = \mathbf{K}\left[\mathbf{R} \mid -\mathbf{R}\vec{C}\right], \tag{2.9}$$

where $\vec{C}$ is the camera's position in the world reference frame.

Image point $\vec{m} = (u, v)^\top$ can be obtained from a 3D point $\vec{X}$ using the projection matrix $\mathbf{P}$ as

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \tag{2.10}$$

$$\lambda \begin{bmatrix} \vec{m} \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} \vec{X} \\ 1 \end{bmatrix}, \tag{2.11}$$

where $\lambda > 0$ is a free scaling parameter.

2.2.4 Skew-symmetric 3x3 matrix

A skew-symmetric or antisymetric matrix of vector $\vec{b} = (b_1, b_2, b_3)^\top$ is defined as

$$[\vec{b}]_\times = \begin{bmatrix} 0 & -b_3 & b_2 \\ b_3 & 0 & -b_1 \\ -b_2 & b_1 & 0 \end{bmatrix}. \tag{2.12}$$

This matrix has several useful properties, but the most important in this thesis is that it generalizes a cross product as matrix multiplication:

$$\vec{a} \times \vec{b} = [\vec{a}]_\times \vec{b}. \tag{2.13}$$

The notation is taken from [29], p. 581.

2.3 Epipolar geometry

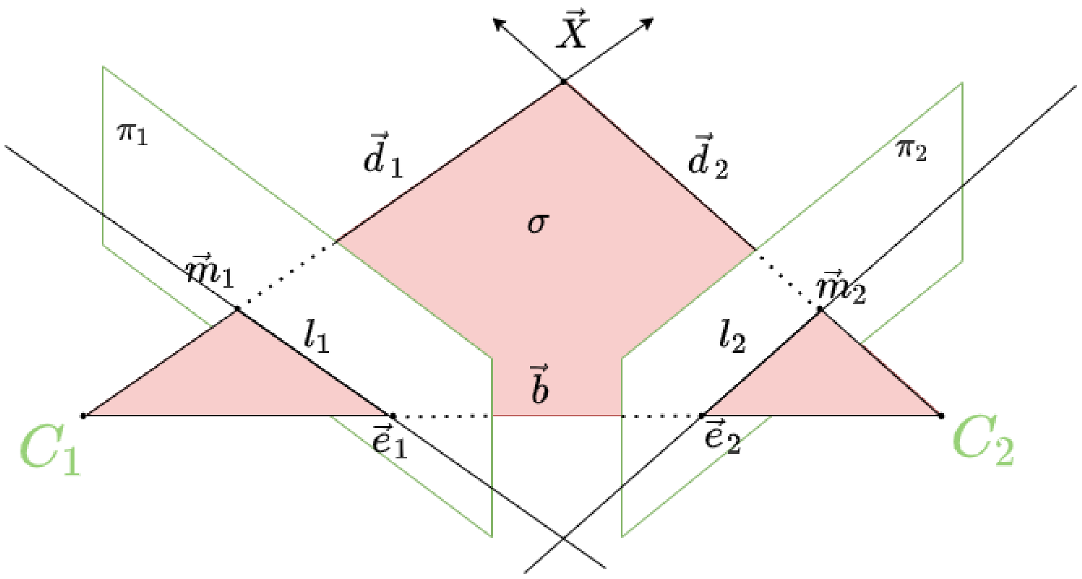


FIGURE 2.6: A scheme of epipolar geometry.

Figure 2.6 shows a scheme of two cameras with different optical centers $\vec{C}_1$ and $\vec{C}_2$ connected with a base vector $\vec{b} = \vec{C}_2 - \vec{C}_1$. Both of the cameras observe the same 3D point $\vec{X}$. Projections of this point to π_1 and π_2 are $\vec{m}_1$ and $\vec{m}_2$, respectively. Points $\vec{C}_1$, $\vec{C}_2$ and $\vec{X}$ form an *epipolar plane* σ. The lines $\sigma \cap \pi_1 = l_1$ and $\sigma \cap \pi_2 = l_2$ are *epipolar lines*. The epipolar line l_1 passes through an *epipole* $\vec{e}_1$ and through point $\vec{m}_1$, where $\lambda[\vec{e}_1|1]^\top = \mathbf{P}_1\vec{C}_2$. Similarly, line l_2 passes through an *epipole* $\vec{e}_2$ and through point $\vec{m}_2$, where $\lambda[\vec{e}_2|1]^\top = \mathbf{P}_2\vec{C}_1$.

2.3.1 The epipolar constraint

Having a set of two cameras, the relationship between them and the constraints on them can be expressed by two matrices: the *essential matrix* $\mathbf{E} \in \mathbb{R}^{3x3}$, $(\text{rank}(\mathbf{E}) = 2)$, and the *fundamental matrix* $\mathbf{F} \in \mathbb{R}^{3x3}$, $(\text{rank}(\mathbf{F}) = 2)$. Matrix $\mathbf{E}$ is obtained as

$$\mathbf{E} = \mathbf{R}_2[\vec{C}_2 - \vec{C}_1]_\times \mathbf{R}_1^\top = [-\vec{t}_{21}]_\times \mathbf{R}_{21} = [\vec{b}]_\times \mathbf{R}_{21}, \tag{2.14}$$

and matrix $\mathbf{F}$ as

$$\mathbf{F} = \mathbf{K}_2^{-T}\mathbf{R}_2[\vec{C}_2 - \vec{C}_1]_\times \mathbf{R}_1^\top \mathbf{K}_1^{-1} = \mathbf{K}_2^{-T}[-\vec{t}_{21}]_\times \mathbf{R}_{21}\mathbf{K}_1^{-1} = \mathbf{K}_2^{-T}\mathbf{E}\mathbf{K}_1^{-1}, \tag{2.15}$$

where $\mathbf{R}_i$ and $\vec{t}_i$ are rotation and translation of the i-th camera in the worls coordinate frame, $i \in \{1,2\}$, $\mathbf{R}_{21} = \mathbf{R}_2\mathbf{R}_1^\top$ is a relative camera rotation and $\vec{t}_{21} = -\mathbf{R}_2\vec{b} = \vec{t}_2 - \mathbf{R}_{21}\vec{t}_1$ is a relative camera translation.

An algebraic expression of the important properties of matrix $\mathbf{F}$ using notation from section 2.3 are:

$$l_1 = \mathbf{F}^\top \begin{bmatrix} \vec{m}_2 \\ 1 \end{bmatrix}, \tag{2.16}$$

$$l_2 = \mathbf{F} \begin{bmatrix} \vec{m}_1 \\ 1 \end{bmatrix}, \tag{2.17}$$

$$\mathbf{F}\begin{bmatrix}\vec{e}_1\\1\end{bmatrix} = \mathbf{F}^\top\begin{bmatrix}\vec{e}_2\\1\end{bmatrix} = 0, \tag{2.18}$$

$$\begin{bmatrix}\vec{m}_2 & | & 1\end{bmatrix}\mathbf{F}\begin{bmatrix}\vec{m}_1\\1\end{bmatrix} = 0. \tag{2.19}$$

Matrix $\mathbf{F}$ maps points from π_1 to epipolar lines on π_2 and vice versa (eqs. (2.16) and (2.17)); epipoles $\vec{e}_1$ and $\vec{e}_2$ are basis vectors of the right and left nullspaces of $\mathbf{F}$, respectivly (eq. (2.18)). Epipolar constraint described by eq. (2.19) means that a point and its correspondent line are on the same plane (see Figure 2.6).

2.4 Stereo vision

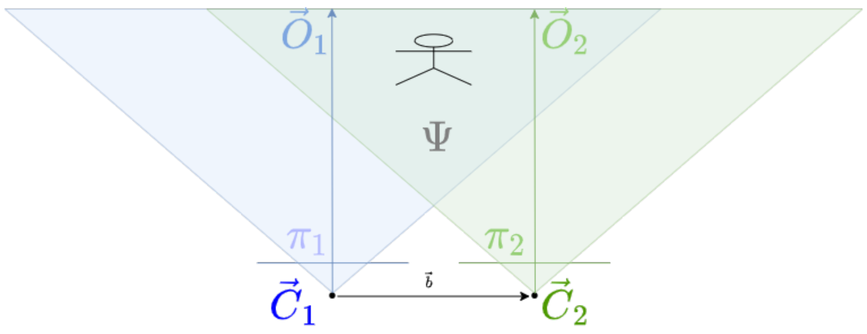


FIGURE 2.7: Illustration of a typical stereocamera setup. Two cameras have optical centers at $\vec{C}_1$ and $\vec{C}_2$ and projection planes π_1 and π_2. The base vector between the camera centers is shown as a vector $\vec{b} = \vec{C}_2 - \vec{C}_1$. Both cameras have parallel optical axes $\vec{O}_1$ and $\vec{O}_2$. The common area visible by both cameras is marked as Ψ.

FIGURE 2.8: An example of an industrial stereocamera Intel Realsense D455.

A typical digital image represents 2D information about a scene projected onto the image plane as in Figure 2.1. Computer stereo vision is a process of extracting a depth image from a pair of images of the same scene. Such depth images are often used in SLAM, obstacle avoidance, or detailed reconstruction of the environment in robotics. In chapter 1, the most common methods of stereo vision are described.

Among the scene depth estimation approaches, there are the computationally expensive neural network-based monocular approaches and costly LiDAR sensors. A typical stereo camera is a tradeoff between price and processing complexity.

Usually, a stereo camera has a pair of cameras located at a distance $\|\vec{b}\|$ pointing in the same direction, similarly to the human eye (a scheme is in Figure 2.7). Sometimes, a fusion of multiple cameras and other sensors is used, such as in the case of the Intel Realsense shown in Figure 2.8 that has two IR cameras, a color sensor, and an infrared projector to project some features onto the environment in there are insufficient features already present (for example when observing a white wall).

2.5 Reprojection error

In an ideal case, when the calibration matrix K has no error and the estimated transformation between the two cameras of the stereopair is accurate, rays $\vec{d}_1$ and $\vec{d}_2$ corresponding to the 3D projections of $\vec{m}_1$ and $\vec{m}_2$ (refer to Figure 2.6) intersect in the point $\vec{X}$. However, calibration only estimates the parameters up to some precision because of noise, measurement errors, the limited resolution of the cameras etc. The reprojection error is used to measure this precision.

After the 3D position of a point is computed based on its projections to the two cameras, to measure its reprojection error, it is projected back to an image, and a distance of this reprojection from the original projection was used to obtain the 3D position estimate is computed. Consider a general projection matrix P_i as

$$\mathsf{P}_i = \begin{bmatrix} (\vec{p}_{i,1})^\top \\ (\vec{p}_{i,2})^\top \\ (\vec{p}_{i,3})^\top \end{bmatrix}. \tag{2.20}$$

The reprojection error is defined as

$$e^2(\underline{X}) = \sum_{c=1}^{2} \left[\left(u_c - \frac{(\vec{p}_{c,1})^\top \underline{X}}{(\vec{p}_{c,3})^\top \underline{X}} \right)^2 + \left(v_c - \frac{(\vec{p}_{c,2})^\top \underline{X}}{(\vec{p}_{c,3})^\top \underline{X}} \right)^2 \right], \tag{2.21}$$

where $m_c = \begin{bmatrix} u_c \\ v_c \end{bmatrix}$, $\underline{X} = \begin{bmatrix} \vec{X} \\ 1 \end{bmatrix}$ and c corresponds to the camera index.

Chapter 3

Methodology

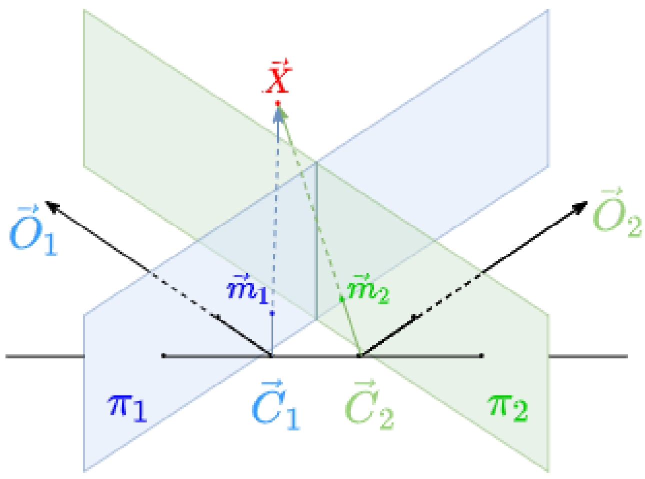


FIGURE 3.1: Ilustration of the proposed approach. $\vec{C}_1$, $\vec{C}_2$ are optical centers of two cameras with projection planes π_1 and π_2. $\vec{O}_1$ and $\vec{O}_2$ are the corresponding optical axes. The 3D point $\vec{X}$ is in the cameras' overlapping field of view, $\vec{m}_1$ and $\vec{m}_2$ are its projections on corresponding image planes.

The main task of this thesis is to create a compact obstacle avoidance system such that it can be mounted on MAVs with size and weight restrictions. The proposed method is related to SfM or optical flow algorithms as well as standard stereo matching algorithms.

Firstly, synchronized images from both cameras are captured. Features are detected using the ORB feature extractor [30] in the areas of these images that correspond to the overlapping part of their fields of view. These features are then matched using the brute-force matcher, described in section 3.4. The 3D position of each of these features relative to the cameras is estimated using a calibrated projection model and known relative transformation between the cameras. Finally, the obtained 3D positions are the output of this algorithm.

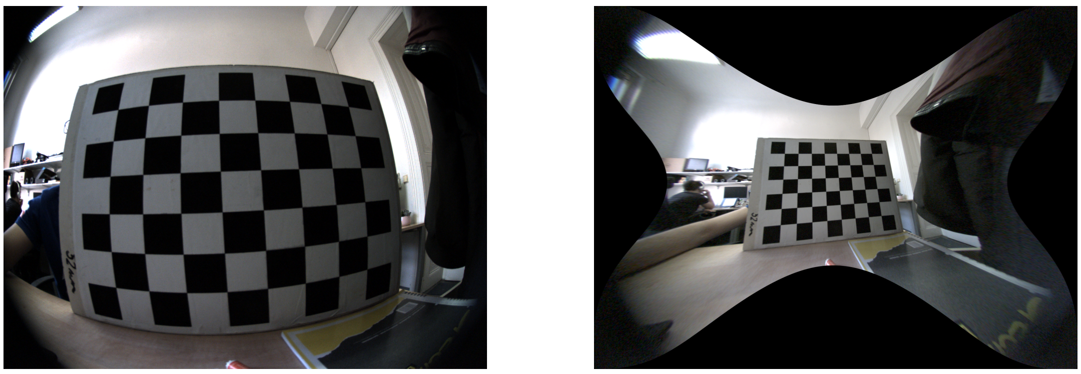
The quality of the 3D point estimation depends on the cameras' relative transformation, the calibration of their projection model, the keypoint (feature) extractor, and the matcher. The obtained 3D points can be used to estimate the scale in an SfM algorithm that runs parallel to the described method (refer to Figure 1.1). The estimated distance to nearby objects can be used in a feedback loop inside the MAV's control system to correct path planning, considering the obstacles found.

This chapter describes the mathematical model of the problem and all algorithms used in the process: calibration of the projection model, calibration of the relative transformation between the cameras, feature extraction, feature matching, and estimation of the 3D position of the matched features.

3.1 Description of the optical setup

The optical setup assumed by the proposed method is shown in Figure 3.1. There are two cameras with optical centers at $\vec{C}_1$ and $\vec{C}_2$ with a known static translation $\vec{t}_{21}$ and rotation $\mathbf{R}_{21}$ between coordinate frames of the cameras. The rotation represented by $\mathbf{R}_{21}$ is assumed to be close to a 90° rotation in the epipolar plane (marked as σ in Figure 2.6). As described in section 1.2, parameters of the mathematical projection model of the two cameras are assumed to be known. In practice, these are obtained using a calibration process described in the next section. Furthermore, it is assumed that the images coming from the cameras are synchronized and that the FOVs of both cameras have an intersecting zone. Let us denote images from the two cameras as I_1 and I_2, a point in the environment $\vec{X}$ and its images in the two cameras $\vec{m}_1$ in image I_1 and $\vec{m}_2$ in image I_2.

3.2 Projection model of a camera and its calibration



(A) Original image with radial distortion. Objects with straight edges appear curved in the image due to the distortion.

(B) Image with no distortion. Edges that are straight in 3D are straight in the image.

FIGURE 3.2: An image from the camera before and after applying undistortion.

Camera calibration is the process of empirically estimating the camera calibration matrix $\mathbf{K}$ (refer to eq. (2.6)) and distortion parameters of the camera's optical system for the pinhole camera model described in section 2.2. Usually, this is done with some pattern with predefined parameters like a chessboard or more advanced markers (ChArUco and ArUco [31] etc).

The camera calibration is necessary for geometrical image correction, distortion elimination, obtaining metric information, and further distance estimation (see Figure 3.2).

In the real world, lenses have distortion (see Figure 3.2a). To compensate that distortion, a polynomial model is often used with coefficients $k_1, ..., k_6$ for radial distortion and p_1, p_2 for tangential distortion.

3.2.1 The minimal problem for camera calibration

It is the problem of obtaining a projection matrix P given $n = 6$ correspondences of 3D scene points and 2D image points $\{(\vec{X}_i, \vec{m}_i)\}_{i=1}^{n}$. Let the projection matrix P be

$$\mathsf{P} = \begin{bmatrix} \vec{q}_1^\top & q_{14} \\ \vec{q}_2^\top & q_{24} \\ \vec{q}_3^\top & q_{34} \end{bmatrix}. \tag{3.1}$$

The equation (2.11) can be expanded to

$$\lambda_i u_i = \vec{q}_1^\top \vec{X}_i + q_{14}, \quad \lambda_i v_i = \vec{q}_2^\top \vec{X}_i + q_{24}, \quad \lambda_i = \vec{q}_3^\top \vec{X}_i + q_{34}, \tag{3.2}$$

where $\vec{m}_i = \begin{bmatrix} u_i \\ v_i \end{bmatrix}$ is an image point, $\lambda \in \mathbb{R}^+$, $i \in \{1, 2, ..., 6\}$. After elimination of λ_i, we obtain

$$(\vec{q}_3^\top \vec{X}_i + q_{34}) u_i = \vec{q}_1^\top \vec{X}_i + q_{14}, \tag{3.3}$$

$$(\vec{q}_3^\top \vec{X}_i + q_{34}) v_i = \vec{q}_2^\top \vec{X}_i + q_{24}. \tag{3.4}$$

$$\mathsf{A}\vec{q} = \begin{bmatrix} \vec{X}_1^\top & 1 & \vec{0}^\top & 0 & -u_1 \vec{X}_1^\top & -u_1 \\ \vec{0}^\top & 0 & \vec{X}_1^\top & 1 & -v_1 \vec{X}_1^\top & -v_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vec{X}_k^\top & 1 & \vec{0}^\top & 0 & -u_k \vec{X}_k^\top & -u_k \\ \vec{0}^\top & 0 & \vec{X}_k^\top & 1 & -v_k \vec{X}_k^\top & -v_k \end{bmatrix} \begin{bmatrix} \vec{q}_1 \\ q_{14} \\ \vec{q}_2 \\ q_{24} \\ \vec{q}_3 \\ q_{34} \end{bmatrix} = \vec{0}, \tag{3.5}$$

so for $k = 6$, $\mathsf{A} \in \mathbb{R}^{12 \times 12}$, $\vec{q} \in \mathbb{R}^{12}$. If A has rank 12, there is no non-trivial null space for A.

Equation (3.5) can be solved by the so-called *Jack-Knife estimation*. Let us denote a matrix A with the i-th row removed as A_i. The *Jack-Knife estimation* iterates through $i \in 1, \ldots, 12$. In each iteration, if the right null-space of A_i is not empty, the matrix P_i can be decomposed to K_i R_i and $\vec{t}_i$. Minimisation of the reprojection error from section 2.5 for $i \in \{1, \ldots, 12\}$ then can be used to find the best estimate of P.

3.2.2 Distortion correction

After the matrix P is obtained, parameters of the distortion can be found as well. The equation (2.10) can be rewritten using the relation from equation (2.9) as

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathsf{K}[\mathsf{R}|\vec{t}] \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{3.6}$$

Let us now redefine this equation to consider the distortion. A 3D point in the camera frame can be expressed as

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = [\mathsf{R}|\vec{t}] \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{3.7}$$

The distortion model is then defined as

$$x'' = \frac{x_c}{z_c}\frac{1+k_1r^2+k_2r^4+k_3r^6}{1+k_4r^2+k_5r^4+k_6r^6} + p_1(r+2x') + 2p_2\frac{x_cy_c}{z_c^2}, \tag{3.8}$$

$$y'' = \frac{y_c}{z_c}\frac{1+k_1r^2+k_2r^4+k_3r^6}{1+k_4r^2+k_5r^4+k_6r^6} + 2p_1(\frac{x_cy_c}{z_c^2}) + p_2(r+2y'), \tag{3.9}$$

where $x' = (\frac{x_c}{z_c})^2$, $y' = (\frac{y_c}{z_c})^2$, $r = x' + y'$. Then, the corresponding undistorted point will be

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} x'' \\ y'' \\ 1 \end{bmatrix}. \tag{3.10}$$

Considering eq. (2.6), the image of a point $\vec{X}$ seen through the calibrated camera with a projection matrix $\mathbf{P}$ is obtained using eqs. (3.6) to (3.10). Firstly, the point is projected to an abstract projection plane (eq. (3.7)). After that, distortion compensation is applied using the model described by equations (3.8) and (3.9). Finally, the point is transformed from the metric system of the abstract projection plane to the image coordinate system (eq. (3.10)).

3.3 General multicamera pose calibration

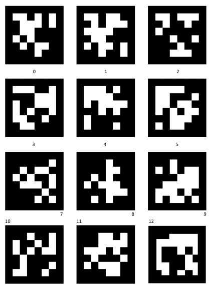


FIGURE 3.3: The calibration pattern using AprilTags that was used for the stereopair calibration.

Stereo pair calibration is a process of estimating the essential matrix $\mathbf{E}$ (defined in section 2.3) for a camera pair, which also expresses the relative rotation matrix $\mathbf{R}_{21}$ and relative translation vector $\vec{t}_{21}$ of a camera pair.

There are multiple algorithms implementing stereo pair calibration. Usually, a calibration pattern is used as the one shown in Figure 3.2 for a standard stereo camera with parallel or converging optical axes. Most algorithms assume that the whole pattern is seen in both images. However, this is a disadvantage for cameras with a small overlapping zone and diverging optical axes. For this reason, it is better to use some other pattern, for example, a set of AprilTags [32] which can be detected separately. The pattern used in this work is shown in Figure 3.3.

3.3.1 Least-square estimation of transformation

The first approach assumes that an initial estimate of the relative pose of the cameras is obtained using a manual measurement. The only necessary step is to correct the pose of one camera with respect to the other. If there is no initial pose provided, the first camera pose is considered the world coordinate frame, and the relative transformation of the second camera is obtained. Firstly, corresponding parts of the calibration patterns are detected in I_1 and I_2, and their 3D positions are computed. The AprilTag markers [32] were used in this thesis, which also provides a unique identification of the markers and an estimate of their full relative pose. In this way, two sets of 3D poses are obtained, each corresponding to the detected markers in one image. Then, transformation parameters between the two point sets are estimated, and the correction transformation $\mathbf{T}_c$ is computed, where

$$\mathbf{T}_c = \begin{bmatrix} \mathbf{R}_c & \vec{t}_c \\ 0 & 1 \end{bmatrix}. \tag{3.11}$$

The algorithm is based on the analysis of the covariance matrix $\sum_{ab} \in \mathbb{R}^{3\times 3}$ of the input sets a and b. It estimates parameters $\mathbf{R}_c$ and $\vec{t}_c$ such that

$$\frac{1}{n}\sum_{i=1}^{n} \|b_i - (\mathbf{R}_c a_i + \vec{t}_c)\|^2 \tag{3.12}$$

is minimized, where n equals to size of a and b. All details with derivation are available in [33]. A correct pose of the second camera then can be obtained by applying $\mathbf{T}_c$ to $\mathbf{R}_{21}$ and $\vec{t}_{21}$.

3.3.2 PnP-based estimation of transformation

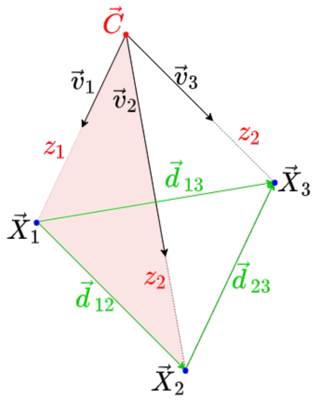


FIGURE 3.4: Visualization of the P3P problem. $\vec{C}$ is the camera center, vectors $\vec{v}_1$, $\vec{v}_2$ and $\vec{v}_3$ are vectors pointing to 3D points $\vec{X}_1$, $\vec{X}_2$, and $\vec{X}_3$, respectivly. The mutual position of the 3D points is known and expressed by vectors $\vec{d}_{12}$, $\vec{d}_{23}$, and $\vec{d}_{13}$. Scalars z_1, z_2, and z_3 are absolute distances from each 3D point to the camera center in world coordinate units (meters).

Another, the more general approach is based on solving a Perspective-n-Point (PnP) problem. PnP is the problem of estimating a camera pose (translation and rotation) given a known set of n 3D points and their respective 2D projections to an image of a calibrated camera. Mathematically, the situation can be expressed as

$$\lambda_i \begin{bmatrix} \vec{m}_i \\ 1 \end{bmatrix} = \mathbf{KR}(\vec{X}_i - \vec{C}), \quad i \in \{0..n\}. \tag{3.13}$$

No initial pose estimation is needed for this algorithm, but it can accelerate the solver if there is one.

P3P

The situation when $n = 3$ is the minimal amount of points to solve the PnP problem. This specific variant of PnP is called P3P. Firstly, let us define a vector $\vec{v}_i \in \mathbb{R}^3$ corresponding to a projection of a point in the image $\vec{m}_i$ as $\vec{v}_i = \mathsf{K}^{-1}\begin{bmatrix}\vec{m}_i \\ 1\end{bmatrix}$. From eq. (3.13), the following relation can be obtained:

$$\lambda_i \vec{v}_i = \mathsf{R}(\vec{X}_i - \vec{C}). \tag{3.14}$$

If there is no rotation, the situation will look like in Figure 3.4 where vectors $\vec{d}_i$ are known, so eq. (3.14) simplifies to a system of three equations with three unknowns (vector $\vec{C}$). If there is a non-zero rotation, it can be eliminated first. Let us define a helper variable z_i as the distance of a point $\vec{X}_i$ from $\vec{C}_i$:

$$|\lambda_i| \cdot \|\vec{v}_i\| = \|\vec{X}_i - \vec{C}\| = z_i. \tag{3.15}$$

Considering only angles between $\vec{v}_i$ and applying the cosine law per $\triangle \vec{C}\vec{X}_i\vec{X}_j$, for $i, j \in \{1, 2, 3\}, i \neq j$, the relation

$$\|\vec{d}_{ij}\| = z_i^2 + z_j^2 - 2z_i z_j c_{ij} \tag{3.16}$$

may be obtained, where $\|\vec{d}_{ij}\| = \|\vec{X}_j - \vec{X}_i\|$, $c_{ij} = \cos(\angle \vec{v}_i \vec{v}_j)$. After solving the system of three equations with three unknown z_i, there will be up to 4 solutions. Each solution should be either verified on additional points [34] or sorted by reprojection error (refer section 2.5). Having this, $\vec{C}$ can be found by trilateration (3 sphere intersection) from $\vec{X}_i$ and z_i; then λ_i from eq. (3.15) and R from eq. (3.14).

P3P + RANSAC

RANSAC stands for Random sample consensus. It is an iterative method of estimating the parameters of a model from a set of observed data that contains both inliers and outliers. The more general PnP task can be solved as a combination of P3P and RANSAC algorithms.

At each iteration, 3 points out of n are randomly sampled, and R and $\vec{t}$ are computed. Then, the obtained transformation is confirmed on all n points using the reprojection error described in section 2.5. In the next iteration, the same process is repeated. If the maximum number of iterations is reached or the change in relative error is insufficient, R and $\vec{t}$, which provided the smallest error, are considered to be the result.

In practice, more sophisticated methods are employed, as in [35, 36].

3.4 Feature extraction, matching and filtering

In computer vision, *features* typically refer to representations of unique pieces of information from the image scene, such as points, edges, and objects. A feature detector is an algorithm for extracting features from an image. There are many such algorithms, but the ORB feature extractor is used in this thesis. The authors in [30] claim that ORB has comparable accuracy as the state-of-the-art SIFT detector while being a few times faster, which is also supported by other research [37].

The next step of the proposed method is mutually associating features in images from the two cameras, corresponding to the same physical objects in the environment. This is done by a feature matcher, which compares descriptors of the features provided by the feature detection algorithm. Some feature matching algorithms perform a brute-force comparison of all combinations of features, others use nearest neighbors approximations or even neural networks [38].

A brute-force matcher is one of the simplest matchers, and it is used in the proposed solution. It takes one descriptor from one set, computes its similarity to all descriptors from the second set, and matches it with the most similar feature. This process is repeated to maximize the overall similarity between the two sets.

Even after this process, there are typically some outliers. Distance from features to the corresponding epipolar lines can be used to filter them. This distance can be computed using the equation (2.19). If the distance of the matched feature from the corresponding epipolar line in the other image is larger than a specified threshold, the match is dismissed as an outlier.

3.5 Feature position estimation

Positions of the observed 3D points in the scene can be computed from pairs of correspondent points taken by calibrated cameras with a known relative pose. This process is called triangulation. Correspondences for triangulation are received as a result of feature detection, matching, and filtering.

3.5.1 Shortest distance triangulation

One triangulation method is based on finding the shortest distance between two rays. According to the epipolar geometry properties described in section 2.3 and visualized in Figure 2.6, vectors $\vec{d_1}$ and $\vec{d_2}$ intersects at the 3D point $\vec{X}$. However, in the real world, the rays can be at some distance from each other due to imperfect stereo pair calibration, so $\vec{X}$ is estimated as the point closest to both lines.

It is possible to compute $\vec{d_1}$ and $\vec{d_2}$ in a common coordinate frame from $\vec{m}_1$ and $\vec{m}_2$, since the relative pose of the cameras is known. Then, let us define two lines d_1 and d_2 in a vector form:

$$d_1 : \vec{p}_1 = \vec{C}_1 + t\vec{d}_1, \tag{3.17}$$

$$d_2 : \vec{p}_2 = \vec{C}_2 + s\vec{d}_2, \tag{3.18}$$

where $\vec{d}_1$ and $\vec{d}_2$ are directional vectors, $\vec{C}_1$ and $\vec{C}_2$ are 3D points located on the respective lines (camera centers in this case), $s \in \mathbb{R}$ and $t \in \mathbb{R}$ are free parameters that uniquely define points $\vec{p}_1$ and $\vec{p}_2$. Distance between the two lines is minimal at the point where the vector $\vec{l} = \vec{p}_2 - \vec{p}_1$ is orthogonal to d_1 and d_2, which can be expressed using a dot product as

$$\vec{l} \cdot \vec{d}_1 = 0, \tag{3.19}$$

$$\vec{l} \cdot \vec{d}_2 = 0. \tag{3.20}$$

After substituting eqs. (3.17) and (3.18) to these equations, a system of two equations with two unknowns s and t is obtained. This system has a unique solution unless the rays are parallel. Using s and t, the points $\vec{p}_1$ and $\vec{p}_2$ can be obtained and then $\vec{X}$ is calculated as $\vec{X} = \frac{\vec{p}_1+\vec{p}_2}{2}$.

3.5.2 SVD triangulation

Triangulation using Singular Value Decomposition (SVD) is another method that is more precise and widely used. It computes 3D points from 2D correspondences using the camera matrices $\mathbf{P}_1$, $\mathbf{P}_2$. This method is derived and described in detail in [29], p.312. Here, a short summary is provided.

The projection equation (2.10) can be rewritten as

$$\lambda_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \mathbf{P}_i \begin{bmatrix} \vec{X} \\ 1 \end{bmatrix}, \tag{3.21}$$

where $\mathbf{P}_i$ decomposes as in eq. (2.20), and $\lambda_i \neq 0, i \in \{1,2\}$. After eliminating λ_1, λ_2 we obtain the following set of equations:

$$\mathbf{D} \begin{bmatrix} \vec{X} \\ 1 \end{bmatrix} = \vec{0}, \quad \mathbf{D} = \begin{bmatrix} u_1(\vec{p}_{1,3})^\top - (\vec{p}_{1,1})^\top \\ v_1(\vec{p}_{1,3})^\top - (\vec{p}_{1,2})^\top \\ u_2(\vec{p}_{2,3})^\top - (\vec{p}_{2,1})^\top \\ v_2(\vec{p}_{2,3})^\top - (\vec{p}_{2,2})^\top \end{bmatrix}, \quad \mathbf{D} \in \mathbb{R}^{4\times 4}. \tag{3.22}$$

The result of the triangulation is the eigenvector corresponding to the smallest eigenvalue. The eigenvectors and eigenvalues are obtained using SVD decomposition of $\mathbf{D}^\top\mathbf{D}$, which is a solution of an equation

$$\mathbf{USV}^\top = \mathbf{D}^\top\mathbf{D}, \tag{3.23}$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices and $\mathbf{S}$ is a non-negative diagonal matrix.

Let the eigenvector corresponding to the smallest eigenvalue as $\vec{h} = (x', y', z', w')^\top$. The triangulated point then can be expressed as

$$\vec{X} = \begin{bmatrix} \frac{x'}{w'} \\ \frac{y'}{w'} \\ \frac{z'}{w'} \end{bmatrix}. \tag{3.24}$$

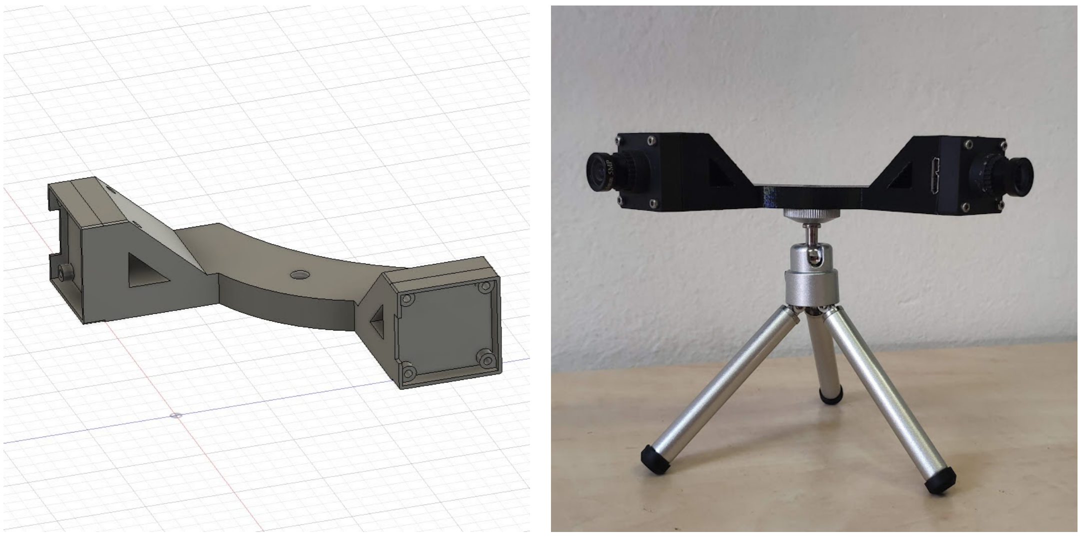
The improved version of this algorithm called "The Golden Standard Triangulation Method" is more widely used in practice. It combines SVD triangulation with Sampson correction (description is available in [29], p. 314).

Chapter 4

Implementation

This chapter describes the implementation of the solution. The used hardware is presented in section 4.1, the software tools and integration of all parts together in section 4.2. The code can be found on GitHub: stereo pair driver[1] and the main module[2].

4.1 Hardware



(A) The CAD model of the prototype. (B) The printed prototype

FIGURE 4.1: A prototype of the proposed solution.

A CAD model of a camera mount was created in the Fusion360 software considering the requirements of a 90° rotation between cameras and distance of the cameras close to the average MAV size. A picture of the design from the CAD software is in Figure 4.1a. The 3D-printed prototype with the cameras mounted is in Figure 4.1b

Basler daA1600-60um cameras were chosen for this project because they have a global shutter which is important when deployed onboard a fast-moving platform, good image quality, and high framerate, making them suitable for the proposed method based on the assumptions defined in section 1.2. More details regarding camera parameters are in Table 4.1.

[1]https://github.com/Myralllka/UAV_basler_stereopair_driver
[2]https://github.com/Myralllka/UAV_localisation_from_cameras

Parameter	Value
Lens mount type	S-mount
Data transfer protocol	USB 3.0
Max. frame rate	60 fps
Resolution (HxV)	1600 px x 1200 px
Resolution	2 MP
Price	289.00 EUR

TABLE 4.1: daA1600-60um specifications.

Lenses of the cameras were chosen so that the resulting horizontal FOV is approximately 120°, which provides a sufficient overlapping zone to detect features in Ψ (see Figure 2.7).

The solution was developed and tested on the Lenovo ThinkPad X280 with Intel(R) Core i7-8550U CPU and 16Gb RAM. Intel NUC with Intel(R) Core i7-10710U CPU and 16Gb RAM is used as the onboard computer for the MAV. No dedicated GPU is needed.

4.2 Software tools

The proposed solution uses the Robot operating system (ROS) [39] as a middleware. ROS is an open-source ecosystem with hundreds of already implemented algorithms and libraries to interact between different parts of a robot's system, such as sensor drivers, image processing algorithms, planners, controllers etc. The AprilTag detector and a driver for Basler cameras used in this thesis are based on publicly available modules from the ROS community.

The MRS UAV system [40] is used as a drone control environment. It is based on ROS, but it is a unique framework for MAVs to implement and test path planning, control, computer vision, object tracking, and many more MAV-related problems.

The OpenCV[3] open-source computer vision library and the Eigen[4] open-source library for efficient linear algebra were used for the implementation. PCL[5] which stands for Point Cloud Library is an open-source library used for point cloud processing. All of the software presented in this thesis was implemented in C++.

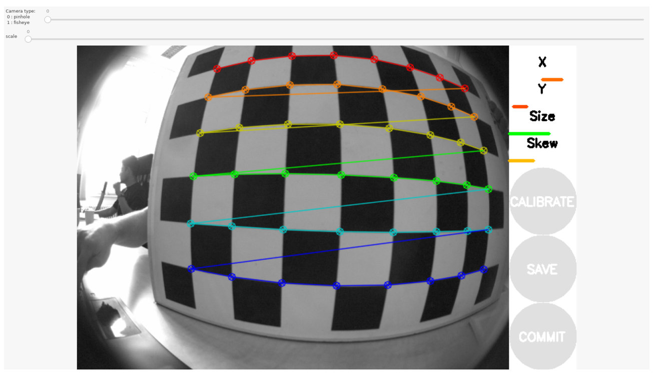


FIGURE 4.2: The single camera calibration process.

[3] https://opencv.org/
[4] http://eigen.tuxfamily.org
[5] https://pointclouds.org/

Calibration of parameters of the camera projection and distortion models (described in section 3.2) was performed using the ROS Camera Calibration package[6]. A chessboard pattern is used for the calibration, and its physical parameters must be specified to the calibration program. The calibration process itself is done interactively using a graphical user interface (shown in Figure 4.2). The interface displays the progress of the process to the user, and when a sufficient dataset is obtained, the user can trigger the parameter optimization. Results of the optimization are then saved to a file for later use.

The two calibration methods described in section 3.3 were both implemented to use an AprilTag calibration pattern, shown in Figure 3.3. The main reason why this pattern was chosen is that each tag can be detected individually, so there is no need to have all of them simultaneously in the overlapping zone. The first method, described in subsection 3.3.1, is implemented using the "Least-square estimation of transformation between two point sets" [33] implementation from Eigen. Measurements from the CAD model of the camera mount were used as the initial estimate of the cameras' relative poses. The second method, described in subsection 3.3.2, is implemented using the OpenCV PnP solver. Because the publicly available implementation of the AprilTag detector outputs only 3D poses of detected tags, the detector was modified to output also 2D coordinates of corners of the detected markers, which is needed for the PnP algorithm and during the debug session for computing the reprojection error.

After both cameras and the stereo pair are calibrated, the following steps are feature detection, matching, and filtering, described in section 3.4. The example from a real-world experiment is in Figure 4.3.

The synchronization is done on a stereo pair driver level. As described in section 3.4, the ORB features detector and a brute-force matcher implementations from the OpenCV were used for feature detection and matching. The next step is reconstructing the 3D scene from the obtained feature pairs. The method described in subsection 3.5.1 is implemented from scratch, and the implementation of triangulation described in subsection 3.5.2 is taken from OpenCV.

[6]`http://wiki.ros.org/camera_calibration`

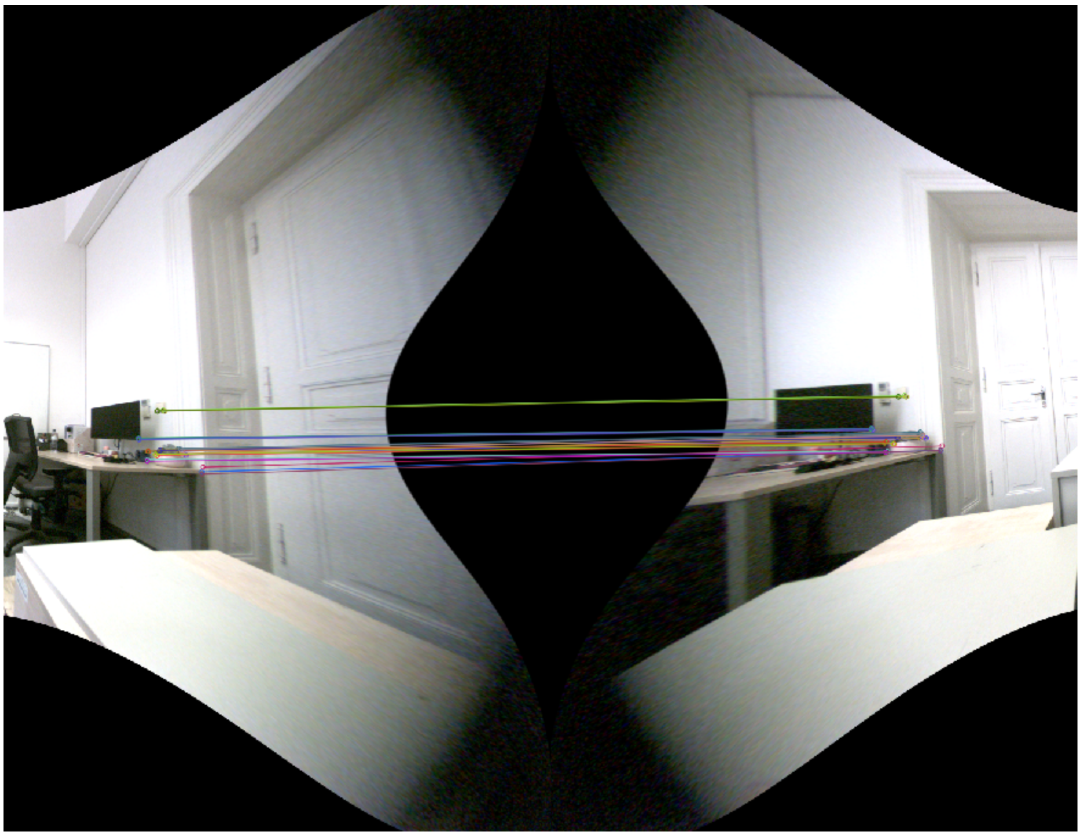


FIGURE 4.3: The result of feature detection, matching, and outliers filtering. There are cropped segments of images from the left camera (left half) and right camera (right half). Detected and filtered features are marked as colored circles, and matched correspondence pairs are connected with lines of the same color as their points.

Chapter 5

Evaluation

5.1 Calibration quality

Another tool from the same ROS Camera Calibration package is used to compute the camera calibration error. It outputs a reprojection RMS (Root Mean Squared) error for each frame, which is defined as

$$\mathsf{RMS}(x) = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \hat{x}_i)^2}{N}} \tag{5.1}$$

where x stands for detected poses of chessboard's corners in the frame, $\hat{x}$ is a ground-truth pose of chessboard corners in 3D coordinate system and N stands for the number of such corners. The pattern was moved in front of each camera to obtain RMS error for different positions and orientations. The mean and variance of 200 frames were taken to analyze the quality of the computed camera parameters. The reprojection error varies depending on the pose of the chessboard but, in general, is relatively stable, see Table 5.1.

		Left camera	Right camera
Mean of RMS reprojection error	[px]	0.06323214286	0.05357391304
Variance of RMS reprojection error	[px]	0.00001159968	0.00002222914

TABLE 5.1: RMS reprojection error.

5.2 Triangulation quality

Two experiments were conducted to measure the triangulation quality. The same pattern with visible feature points in a featureless environment was used. The process of the experiment is shown in Figure 5.1.

5.2.1 Experiment setup

The reprojection error is not a good metric to evaluate the triangulation performance. The disparity variance decreases with increasing the distance (see Figure 5.2b), which means that feature points are closer together in the image, and the reprojection error decreases (see Figure 5.2a). Another approach was chosen to measure the quality of distance estimation. A pattern with multiple features was attached to a white wall in a corridor within a featureless environment. The camera prototype on a tripod was placed at different distances from the wall and was set up so that its $Y - Z$ plane was

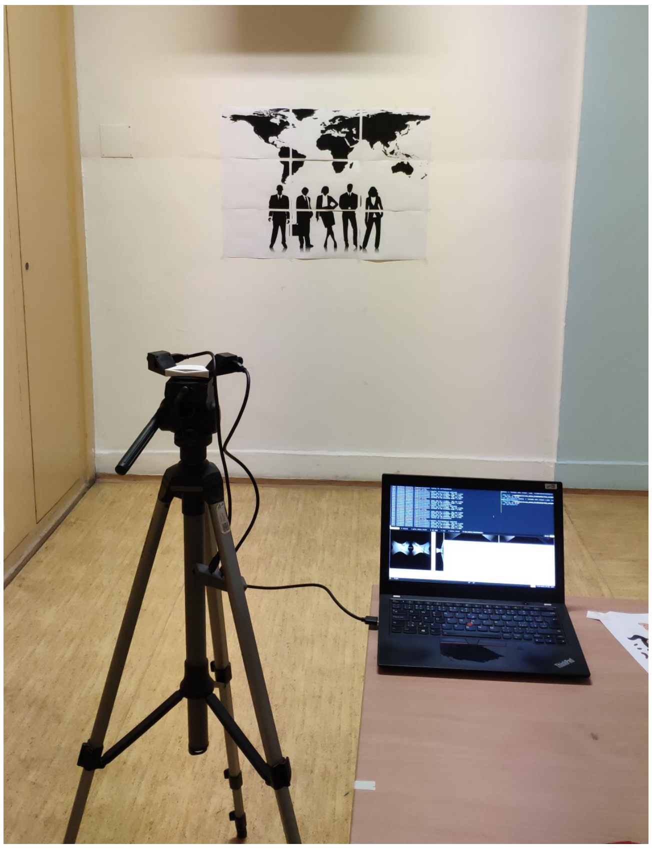


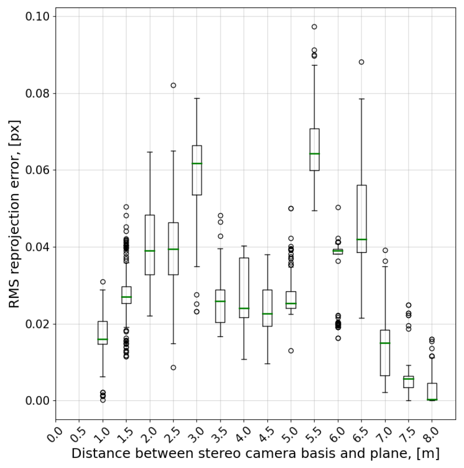
FIGURE 5.1: The setup for experiments. The designed prototype is located on a tripod, at distance d from the plane Ω with visual features. The with $Y - Z$ plane of the cameras' common coordinate frame is parallel with Ω. A set of 9 A4 papers with features was used to make Ω distinguishable.

parallel to the wall as in Figure 5.1. Based on this, we assume that the ground-truth position of all observed 3D points lies on a plane whose orientation and distance from the coordinate frame's origin are known. Then, several measurements were made, and the obtained data were collected and saved to post-process separately, analyze and obtain plots.

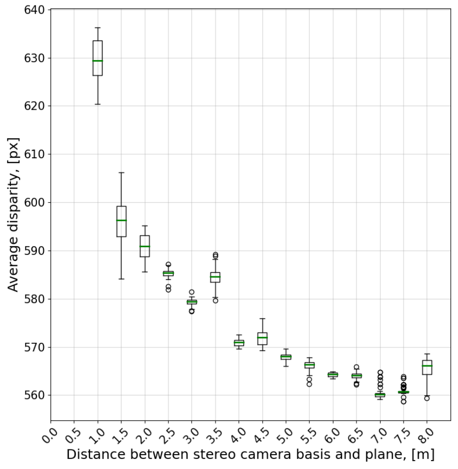
Let us define the wall plane as Ω, the distance from the origin of the stereo pair's common frame to the wall as d, an estimated plane as Ω', and an estimated distance to the plane as d'. The main goal of the experiments was to measure the precision of triangulation and compare it with the theoretical error, introduced in [41] as a function of the distance based on the geometrical and optical parameters of the setup. The theoretical error is defined as

$$e_Z = \frac{Z^2\delta}{bf}, \tag{5.2}$$

where Z is the real distance, δ is the matching error in pixels, f is the focal length and $b = \|\vec{b}\|$ (see Figure 2.7). To compute the theoretical error for the current prototype, the following values was used: $\delta = 1\,\text{px}$, $f = 38\,\text{mm}$ and $b = 14.5\,\text{mm}$.



(A) Stereo pair setup RMS reprojection error.



(B) Average disparity for all detected keypoints. Disparity is a difference between coordinates of two matched feature points in an image pair.

FIGURE 5.2: Green lines represent the median for n measurements at a distance d. A box marks an interval from the first quartile to the third quartile. The whiskers go from each quartile of the measured values to the lower and upper fences. Black points represent outliers.

5.2.2 Distance to an estimated plane

To find the distance d', Ω' should be estimated using the triangulated feature points. In this experiment, for each set of 3D points corresponding to a specific distance d, Ω' is estimated using RANSAC to filter outliers (an implementation from PCL was used). The error is defined as

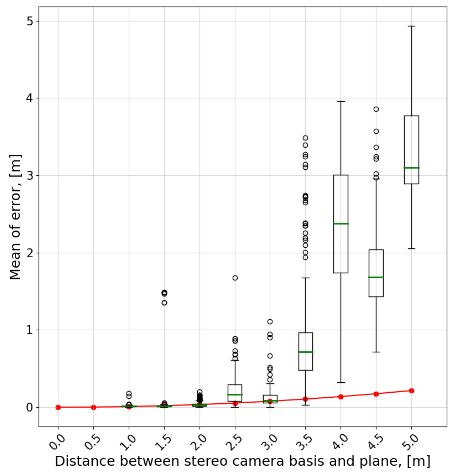
$$e = \frac{1}{k} \sum_{j=0}^{k} |d - d'_j|, \tag{5.3}$$

where d'_j is the estimated distance to Ω' for each measurement and k is the number of measurements.

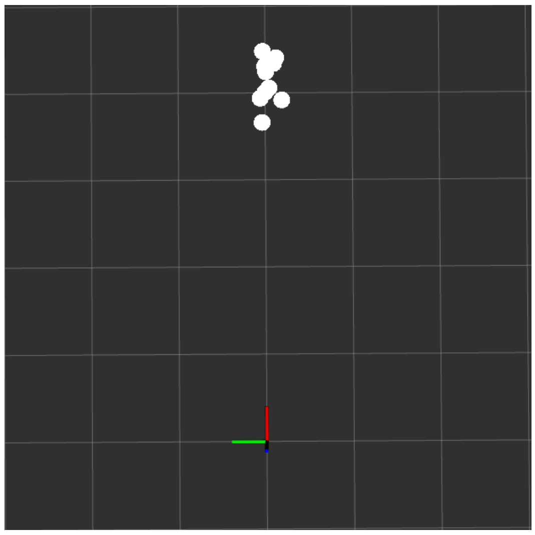
In this experiment, the absolute error e was measured for each distance d from 1 m to 5 m with a step of 0.5 m. The result of this experiment is shown in Figure 5.3a. According to the figure, the distance deviates significantly after 3 m, and the error grows much faster than the theoretical caused by the camera's limited resolution. In Figure 5.3b a top view of the experiment is shown for $d = 4$ m. It can be seen that the plane can not be estimated correctly at this distance using RANSAC anymore because of the too large error in distance estimation for each triangulated point.

5.2.3 Distance to a predefined plane

With increasing the distance d, feature points become less distinguishable due to the limited resolution of the camera. The estimated plane Ω' is inaccurate from a certain distance because of high noise in the X direction of the points and low spread of the points in the $Y - Z$ direction (see Figure 5.3b), but still, the points are close to the actual plane Ω. That is why the second experiment was constructed.



(A) The first experiment results. The red line represents the theoretical error for the given setup. Green lines represent the median for n measurements at a distance d. A box marks an interval from the first quartile to the third quartile. The whiskers go from each quartile of the measured values to the lower and upper fences. Black points represent outliers.



(B) Top view of the experiment. The grid's cell size is 1 m×1 m. The colored axes represent the stereo pair's coordinate system. Red color stands for X axis, green for Y and blue for Z. The point cloud (white dots) represents the 3D points reconstructed from feature points seen by both cameras.

FIGURE 5.3: Results from the first experiment: distance to the estimated plane.

The experiment was done using the same setup as described in subsection 5.2.1. The difference is that Ω' is not estimated using the least squares RANSAC method to fit the plane to the points, but as the plane parallel to Ω on a distance e'_i, that is defined as

$$e' = \frac{1}{k}\sum_{j=0}^{k} |p'_j|, \tag{5.4}$$

where k is the total number of detected points at each d and $|p'_j|$ is the absolute distance from point $\vec{p}'_j$ to Ω.

The ground-truth distance d for the current experiment was from 1 m to 8 m as the maximum distance at which key points can be detected (empirically found) with a 0.5 m step. The result of the second experiment is shown in Figure 5.4, where the measured error corresponds to the theoretical error up to $d = 4$ m with a slight deviation. After 4 m, the difference between the theoretical and measured error grows larger with a larger variance. One possible cause of this difference is an error in camera calibration and fixed focal length, so detected patterns on longer distances are blurred and less accurate.

5.3 Rate testing

To ensure that the obstacle detection is practical for real-world deployment, not only triangulation quality is important, but also the rate at which the MAV can receive the data from the obstacle avoidance module and the delay of the data. The data rate, maximal and minimal delays averaged over 5 min are shown in Table 5.2.

Parameter	Value
Minimal delay	0.042 s
Maximal delay	0.196 s
Average rate	10.607 Hz

TABLE 5.2: Measured data delay and rate of the proposed system.

5.4 Experiments summary

The quality of the distance estimation to obstacles was demonstrated in the two experiments. The first experiment, described in subsection 5.2.2, shows that the precision of the proposed method is relatively high for distances up to 3 m, and the second experiment, described in subsection 5.2.3 provided further insight into the statistics of the error over longer distances. The estimated distance error on 3 m is less than 3 %. The maximum distance at which 3D points could be reliably estimated is 8 m with an error of up to 1.5 m. On longer distances, feature points can not be reliably detected anymore. The proposed solution works with an average frequency of 10 Hz.

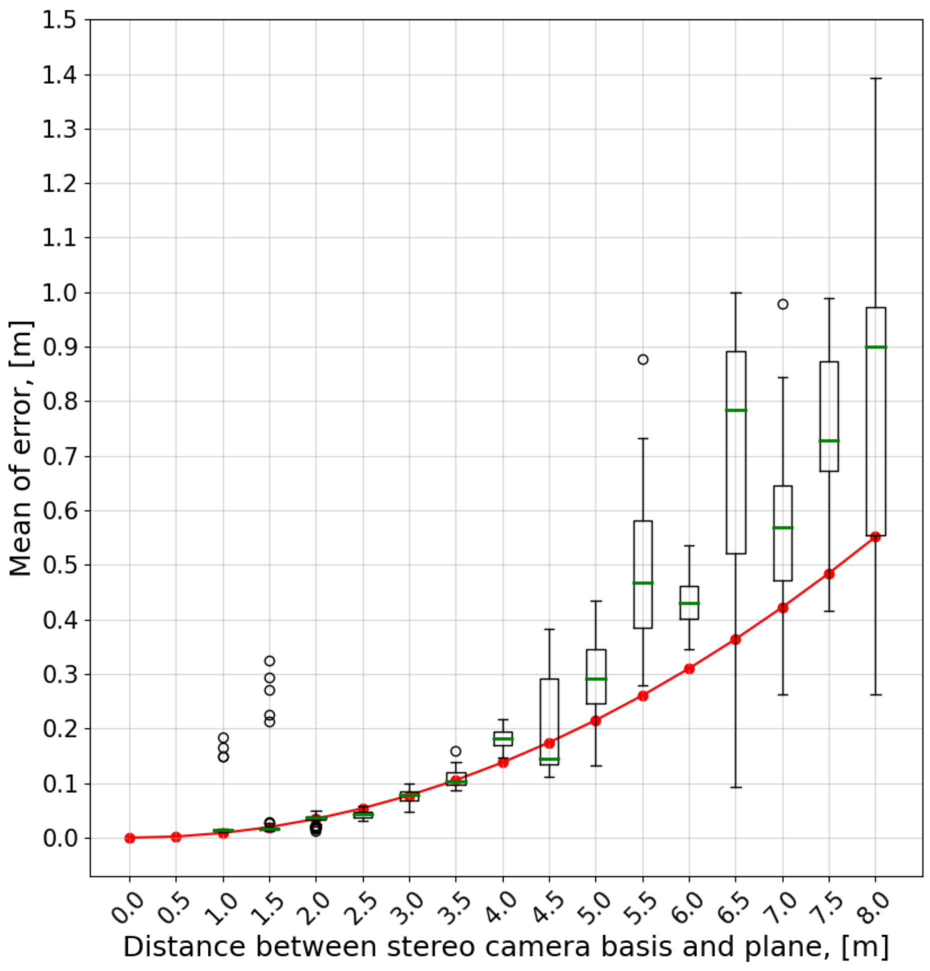


FIGURE 5.4: Results from the second experiment. The Red line represents the theoretical error for the given setup. Green lines represent the median for n measurements at a distance d. A box marks an interval from the first quartile to the third quartile. The whiskers go from each quartile of the measured values to the lower and upper fences. Black points represent outliers.

Chapter 6

Conclusion and future work

A multi-camera vision-based obstacle avoidance system for MAVs was presented in this thesis. Its main advantage is the reduced number of cameras to cover a bigger area and the usage of multiple monocular cameras, which can provide data for another algorithm running onboard the MAV. It needs only a CPU to process data at a fast rate; no GPU is required. The proposed solution consists of a working prototype and two ROS packages: the stereo pair driver and the package for obstacle detection. The obstacle detection package also has scripts to check the stereo pair calibration quality using epipolar error (used for data filtering after keypoints detections, described in section 3.4) for debugging, data collection and collected data analysis.

The proposed solution was evaluated in a laboratory experiment, described in chapter 5. The experiment demonstrates that the working distance for the proposed setup is up to 8 m, while recommended distance is up to 3.5 m. The working frequency is not high enough to fly at high speed, so it is one of desired directions for development. The possible way to solve this issue is to change the image synchronization part of a stereo camera driver to remove inefficiencies due to image copying.

The future steps in this project's development are to integrate the proposed solution with the MRS MAV control system, test it in a real-life experiment and extend the number of cameras to four to cover the whole area around the MAV.

To increase the number of detected key points (refer to Figure 1.1, red point cloud), a bigger overlapping field of view is needed. This can be achieved by selecting more appropriate lenses for the cameras to obtain a larger field of view. Instead of the pinhole camera model used in this approach (refer section 2.2), the fish-eye camera model and 180° lenses can be used in that case [42]. A 90° overlapping field of view could be achieved using this approach instead of 30°, so the 360° MAV's horizontal FOV would be covered with only four cameras.

Another possible approach is to combine the proposed method with SfM. Images from the cameras can be used separately to make 3D points using SfM, but using the output from the current solution, SfM results from different cameras can be combined, and the scale estimation can be simplified and improved.

This thesis aimed to design a visual multi-camera lightweight obstacle avoidance system for MAVs. This assignment was satisfied, and the developed prototype has shown good performance, as was demonstrated in real-world experiments. The method has a potential for future research and improvements and combination with other algorithms, onboard deployment and real-world experiments.

Bibliography

[1] I. Mademlis, N. Nikolaidis, A. Tefas, I. Pitas, T. Wagner, and A. Messina, "Autonomous UAV cinematography," *ACM Computing Surveys*, Sept. 2020.

[2] F. Ullah, F. Al-Turjman, S. Qayyum, H. Inam, and M. Imran, "Advertising through UAVs: Optimized path system for delivering smart real-estate advertisement materials," *International Journal of Intelligent Systems*, Mar. 2021.

[3] J. Kim, S. Kim, C. Ju, and H. I. Son, "Unmanned aerial vehicles in agriculture: A review of perspective of platform, control, and applications," *IEEE Access*, 2019.

[4] V. Pritzl, P. Stepan, and M. Saska, "Autonomous flying into buildings in a firefighting scenario," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, May 2021.

[5] R. She and Y. Ouyang, "Efficiency of UAV-based last-mile delivery under congestion in low-altitude air," *Transportation Research Part C: Emerging Technologies*, Jan. 2021.

[6] A. Gupta, T. Afrin, E. Scully, and N. Yodo, "Advances of UAVs toward future transportation: The state-of-the-art, challenges, and opportunities," *Future Transportation*, Sept. 2021.

[7] M. Aloqaily, O. Bouachir, I. A. Ridhawi, and A. Tzes, "An adaptive UAV positioning model for sustainable smart transportation," *Sustainable Cities and Society*, 2022.

[8] S. Ortiz, C. T. Calafate, J.-C. Cano, P. Manzoni, and C. K. Toh, "A UAV-based content delivery architecture for rural areas and future smart cities," *IEEE Internet Computing*, Jan. 2019.

[9] T. Bacelar, J. Madeiras, R. Melicio, C. Cardeira, and P. Oliveira, "On-board implementation and experimental validation of collaborative transportation of loads with multiple UAVs," *Aerospace Science and Technology*, Dec. 2020.

[10] M. F. Deering and S. Microsystems, "The limits of human vision," in *Sun Microsystems, 2nd International Immersive Projection Technology Workshop*, 1998.

[11] J. Jang, "Ai-powered autonomous drone for organic, unit-level isr." White Paper from Skydio. available at `http://www.blowinglotsofweirdstuffup.com/guide.htmlhttps://pages.skydio.com/rs/784-TUF-591/images/Skydio-x2d-ai-powered-autonomous-drone-defense-white-paper.pdf` (2020-05-25).

[12] M. Westoby, J. Brasington, N. Glasser, M. Hambrey, and J. Reynolds, “"structure-from-motion" photogrammetry: A low-cost, effective tool for geoscience applications," *Geomorphology*, 2012.

[13] H.-Y. Lin and X.-Z. Peng, "Autonomous quadrotor navigation with vision based obstacle avoidance and path planning," *IEEE Access*, 2021.

[14] B. Ruf, S. Monka, M. Kollmann, and M. Grinberg, "Real-time on-board obstacle avoidance for uavs based on embedded stereo vision," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Sept. 2018.

[15] H. Oleynikova, D. Honegger, and M. Pollefeys, "Reactive avoidance using embedded stereo vision for mav flight," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[16] S. Back, G. Cho, J. Oh, X.-T. Tran, and H. Oh, "Autonomous UAV trail navigation with obstacle avoidance using deep neural networks," *Journal of Intelligent & Robotic Systems*, Sept. 2020.

[17] L. Mejias, S. McNamara, J. Lai, and J. Ford, "Vision-based detection and tracking of aerial targets for UAV collision avoidance," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, Oct. 2010.

[18] Z. Zhang, M. Xiong, and H. Xiong, "Monocular depth estimation for UAV obstacle avoidance," in *2019 4th International Conference on Cloud Computing and Internet of Things (CCIOT)*, IEEE, Dec. 2019.

[19] C. Bills, J. Chen, and A. Saxena, "Autonomous mav flight in indoor environments using single image perspective cues," in *2011 IEEE International Conference on Robotics and Automation*, 2011.

[20] S. Ramasamy, R. Sabatini, A. Gardi, and J. Liu, "LIDAR obstacle warning and avoidance system for unmanned aerial vehicle sense-and-avoid," *Aerospace Science and Technology*, Aug. 2016.

[21] N. Gageik, P. Benz, and S. Montenegro, "Obstacle detection and collision avoidance for a uav with complementary low-cost sensors," *IEEE Access*, 2015.

[22] E. M. Nor, S. B. M. Noor, M. R. Bahiki, and S. Azrad, "Implementation of high-gain observer on low-cost fused IR-OS sensor embedded in UAV system," *IOP Conference Series: Materials Science and Engineering*, dec 2017.

[23] H. Yu, R. Hong, X. Huang, and Z. Wang, "Obstacle detection with deep convolutional neural network," in *2013 Sixth International Symposium on Computational Intelligence and Design*, 2013.

[24] B. Park and H. Oh, "Vision-based obstacle avoidance for UAVs via imitation learning with sequential neural networks," *International Journal of Aeronautical and Space Sciences*, Feb. 2020.

[25] S. Huang, R. S. H. Teo, and K. K. Tan, "Collision avoidance of multi unmanned aerial vehicles: A review," *Annual Reviews in Control*, 2019.

[26] W. G. Aguilar, V. P. Casaliglla, and J. L. Pólit, "Obstacle avoidance for low-cost uavs," in *2017 IEEE 11th International Conference on Semantic Computing (ICSC)*, 2017.

[27] M. A. Moreno-Armendariz and H. Calvo, "Visual SLAM and obstacle avoidance in real time for mobile robots navigation," in *2014 International Conference on Mechatronics, Electronics and Automotive Engineering*, IEEE, Nov. 2014.

[28] D.-J. Lee, P. Merrell, Z. Wei, and B. E. Nelson, "Two-frame structure from motion using optical flow probability distributions for unmanned air vehicle obstacle avoidance," *Machine Vision and Applications*, June 2008.

[29] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[30] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *2011 International Conference on Computer Vision*, IEEE, Nov. 2011.

[31] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, June 2014.

[32] D. Malyuta, C. Brommer, D. Hentzen, T. Stastny, R. Siegwart, and R. Brockers, "Long-duration fully autonomous operation of rotorcraft unmanned aerial systems for remote-sensing data acquisition," *Journal of Field Robotics*, Aug. 2019.

[33] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Apr. 1991.

[34] M. A. Fischler and R. C. Bolles, "Random sample consensus," *Communications of the ACM*, June 1981.

[35] V. Lepetit, F. Moreno-Noguer, and P. Fua, "EPnP: An accurate o(n) solution to the PnP problem," *International Journal of Computer Vision*, July 2008.

[36] J. A. Hesch and S. I. Roumeliotis, "A Direct Least-Squares (DLS) method for PnP," in *2011 International Conference on Computer Vision*, 2011.

[37] H. Sharif and M. Hölzel, "A comparison of prefilters in ORB-based object detection," *Pattern Recognition Letters*, July 2017.

[38] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superglue: Learning feature matching with graph neural networks," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[39] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source robot operating system," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics*, May 2009.

[40] T. Baca, M. Petrlik, M. Vrba, V. Spurny, R. Penicka, D. Hert, and M. Saska, "The MRS UAV system: Pushing the frontiers of reproducible research, real-world deployment, and education with autonomous unmanned aerial vehicles," *Journal of Intelligent & Robotic Systems*, Apr. 2021.

[41] C. Chang and S. Chatterjee, "Quantization error analysis in stereo vision," in *[1992] Conference Record of the Twenty-Sixth Asilomar Conference on Signals, Systems Computers*, 1992.

[42] K. Kanatani, "Calibration of ultrawide fisheye lens cameras by eigenvalue minimization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013.