ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

V4RL VISION FOR ROBOTICS LAB
Vision for Robotics Lab
Prof. Margarita Chli

Semester Thesis

Calibration and set-up of two independent cameras mounted on servos for UAVs relative pose estimation

Spring Term 2017

Supervised by:
Lucas Teixeira
Fabiola Maffra

Author:
Bartolomei Luca

Declaration of Originality

I hereby declare that the written work I have submitted entitled

Calibration and set-up of two independent cameras mounted on servos for UAVs relative pose estimation

is original work which I alone have authored and which is written in my own words.[1]

Author(s)

Luca	Bartolomei

Student supervisor(s)

Lucas	Teixeira
Fabiola	Maffra

Supervising lecturer

Margarita	Chli

With the signature I declare that I have been informed regarding normal academic citation rules and that I have read and understood the information on 'Citation etiquette' (`https://www.ethz.ch/content/dam/ethz/main/education/rechtliches-abschluesse/leistungskontrollen/plagiarism-citationetiquette.pdf`). The citation conventions usual to the discipline in question here have been respected.

The above written work may be tested electronically for plagiarism.

Place and date	Signature

[1] Co-authored work: The signatures of all authors are required. Each signature attests to the originality of the entire piece of written work in its final form.

Contents

Abstract

Unmanned Aerial Vehicles (UAVs) are agile platforms that are used in robotics for a wide range of tasks. However, due to their ability to perform fast flights, they are challenging to track. In literature there exists a broad number of tracking algorithms that allow to estimate the pose of an UAV by means of an external camera. Generally these algorithms make use of a fixed camera, and therefore they loose track of the UAV if it moves out of the field of view. A possible solution to this issue is to allow the presence of a moving camera.

In this semester project, a system composed by a camera and a servo is considered. In particular, a toolbox for the calibration of the camera-servo set-up is proposed and analysed. First, the formalization of the problem is introduced. Then, the pipeline of the overall calibration is explained and evaluated.

The proposed toolbox shows satisfactory performances, even when used in combination with algorithms for UAV pose estimation and tracking.

Symbols

Symbols

θ	Servo rotation angle
$offset$	Distance between the servo and the camera
T_{AB}	Homogeneous transformation matrix mapping $B \rightarrow A$

Reference systems indices

W	World reference system
S	Servo reference system
C	Camera reference system
b	BlueFox2 Camera reference system
M	Marker reference system

Acronyms and Abbreviations

UAV	Unmanned Aerial Vehicle
SLAM	Simultaneous Localization and Mapping
FPS	Frames Per Second
PnP	Perspective-n-Point problem

Chapter 1

Introduction

In the recent years, the use of Unmanned Aerial Vehicles (UAV) has increased exponentially. Thanks to their capacity of adapting to different settings and situations, they have been largely employed in tasks such as safe-and-rescue missions or exploration of unsafe environments.

However, the full autonomous flight of these robotic platforms has not been reached yet. In order to increase the navigation capabilities, the research community has focused on the development of algorithms and strategies that could make spacial perception of the robots more robust. In literature there exists a huge amount of works on the problem of Simultaneous Localization and Mapping (SLAM) and, even tough the solutions that have been proposed through the years are various and valid, a lack of robustness still affects the state of the art.

Furthermore, UAV limited computational power, in addition to low payload capacities, creates significant issues and challenges. The number and the types of sensors, combined with the algorithms for on-line SLAM, become a fundamental choice that has direct consequences on the performances in the navigation task.
So far, the SLAM approaches with on-board sensing and computation have obtained good results in controlled laboratory environments, but the performances in real scenarios are worse. In fact, UAVs are agile platforms that are capable of fast movements. Therefore, due to the high speeds, the control and the tracking of a flying UAV are challenging, and the existing systems are not ready to operate fully autonomously in the real environments.

In this semester project, the focus is on the development of a set-up that could allow better tracking performances when the pose of an UAV is obtained by means of an external camera. Actually, even if various pose tracking algorithms have been developed and successfully proved even in outdoor applications, the problem of a continuous tracking of a flying UAV still persists.
More precisely, the ultimate goal of the project is to create a set-up that allows to track an UAV from another UAV. To solve this problem, a system composed by a servo and a camera is adopted, as shown in figure 1.1.
With the proper choice of the servo and the correct implementation of visual servoing algorithms, the camera could be able to follow the UAV to track during its flight. However, before reaching this step, a proper calibration of the Pan Camera, composed by the servo and the single camera, must be performed.

This report focuses on the calibration and the set-up of the camera-servo system and shows the impact of the proposed calibration method on an already existing

UAV pose estimation pipeline.

First of all, the related works and the mathematical model are presented. Then, the pipeline of the toolbox for the calibration is explained, presenting the results coming from both the simulation of the process and the calibration of the real set-up. Finally, the toolbox is evaluated and then the performances in a real tracking scenario are analysed.

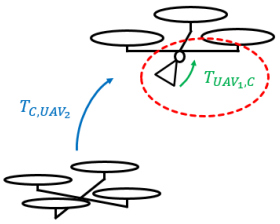

Figure 1.1: The tracking problem: here the focus is on the system servo-camera

Chapter 2

Related works

2.1 The Servo-Camera model

The system composed by a camera and a servo can be modelled as a Pan Camera. In literature, the problem of calibrating cameras that are able to rotate has been widely studied taking in consideration also the zooming property (*Pan-Tilt-Zoom Camera* - [1] [2]). Generally, Pan-Tilt-Zoom Cameras are used as components in wide-area surveillance systems. However, this project focuses on the applications in the robotic field, in particular for UAV pose tracking. The movement is limited to only one degree of freedom, without considering the presence of the zoom.

Many of the existing models of Pan Cameras assume a rather simple geometric motion, in which the axis of rotation is aligned with the camera imaging optics [3] [4]. Moreover, the assumption that the centre of projection is coincident with the centre of rotation holds. However, this approach will lead to extremely bad results when cheap hardware is used due to the lack of careful engineering.
More complex models have been proposed, but in this report the focus will be on a more general approach with an intermediate level of complexity [5].

Another important aspect of the existing camera calibration algorithms is the employment of relatively small calibration patterns. While small targets are appropriate for stationary cameras, they produce poor results when applied to cameras that can perform rotations. As in [6], the target is build in a way that covers an area as wide as possible. To obtain a high quality calibration, the pattern fills the working volume spanned by the camera as it is moved by the servo. More precisely, here the calibration target is built using AprilTags (Appendix A - [7] [8]).

2.2 The UAV pose estimation algorithm

In literature there exists a wide range of different solutions for the UAV pose estimation problem. With the increasing need for accurate pose tracking, many commercial motion tracking systems have been successfully developed. *Vicon* and *OptiTrack* are the most widely used, since they are able to estimate the pose with high accuracy and at high frame rate.

More simplistic approaches that make use of passive planar marker also exists, such as AprilTags [7], while a more versatile solution makes use of a constellation of infra-red LEDs [9]. The advantage of the second solution is that these markers are detectable under different illumination conditions. The pose of the UAV can be

obtained by correlating the prior knowledge of the marker disposition and their detection in the image. This problem is generally solved using the Perspective-n-Point formulation (PnP) [10].

The tracking algorithm that has been employed in this project is described in [11], developed by Marco Moos at Vision For Robotics Lab *V4RL*, ETH Zurich.
This solution presents numerous advantages. It makes use of a monocular external camera, enabling multi-hypothesis tracking using P3P formulation and a Particle Filter. This method achieves very fast and accurate pose estimation and an increased robustness against false detections and occlusions with respect to other existing algorithms.
From now on, when a reference about the tracking algorithm is made, it implicitly indicates Marco Moos' pipeline. Moreover, the superscript *Moos* in the transformations computations is used for more clarity.

Chapter 3

Methodology

3.1 Hardware set-up

The system to calibrate is composed by a servo *HiTech HS-485HB* (figure 3.1), capable of rotations up to 180 degree with a resolution of 1 degree, and a monochrome *BlueFox2 Camera* from MatrixVision[1]. To obtain the intrinsics of the camera, a calibration with `Kalibr` [12] [13] [14] has been performed.

Using a monochrome camera and a commercial servo keeps the hardware set-up simple and cheap. Servos are small rotary actuators and generally they are used in radio controlled models and in small-scale robotic applications; therefore generally their price is low. The typical servo mechanism consists of small electric motor driving a train of reduction gears.
In our experiments, the servo was controlled with an *Arduino UNO* board. The circuit is composed by two buttons, which can rotate the servo clockwise or counter clockwise. The connections are sketched in figure 3.2, while figure 3.3 shows the complete set-up camera and servo.

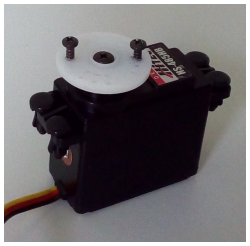

Figure 3.1: The servo used in this project

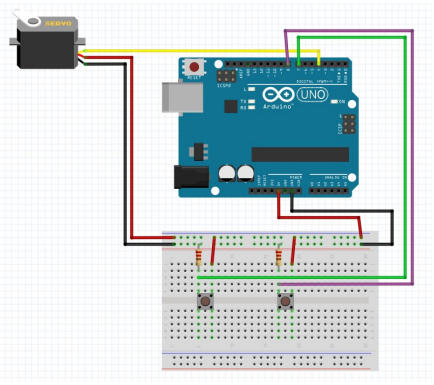

Figure 3.2: Arduino UNO sketch

Figure 3.3: The complete system camera-servo

[1] `www.matrix-vision.com`

3.2 Problem formulation

The ultimate goal of the overall project consists of the calibration of the system composed by the servo and the camera of figure 3.3. To fulfil the task, the pose of the servo T_{WS} and an accurate transformation T_{SC} between the camera and the servo have to be retrieved for different servo rotation angles. The rotation occurs about the axis of the servo and it is indicated with θ.
Moreover, this report concentrates only on the pan movement of the camera, without considering the tilt.

The calibration pipeline decomposes the camera poses T_{WC} found at different servo angular positions into two components. The first one is the pose of the servo T_{WS}, while the second one is the transformation T_{SC} that links the camera to the servo reference system as a function of θ and the $offset$ between the camera and the servo:

$$T_{SC} = T_{SC}(\theta, offset) \tag{3.1}$$

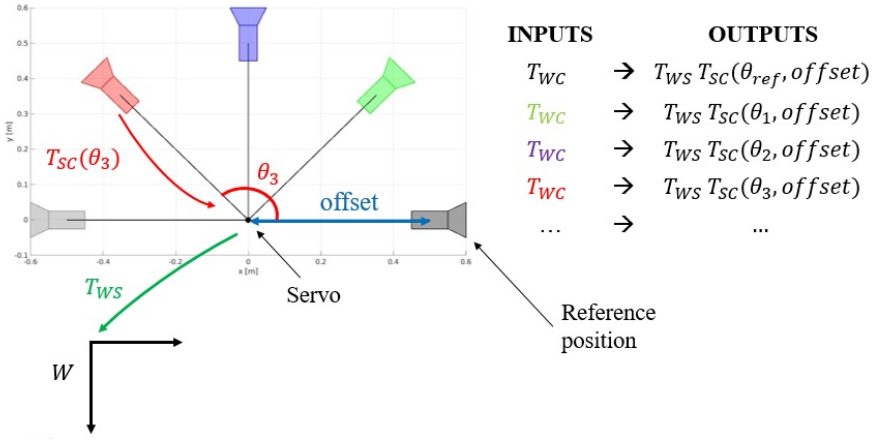

Figure 3.4: The scheme of the system servo-camera and the input/output of the calibration process

A simple scheme of the system can be found in Figure 3.4. The different colours of the camera indicates the different positions that it assumes for different angular positions of the servo. In the same figure, the overall input/output flow of the calibration process is represented.

3.3 The analytical model

To describe the transformation T_{SC} as a function $T_{SC}(\theta,\ offset)$, a proper mathematical description of the Pan Camera is needed.
Previous researches have used a rather simplified model, where pan movements are modelled as ideal rotations around the origin, followed by a perspective camera transformation. The overall process can be easily described as a sequence of matrix multiplications:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \boldsymbol{C}\ \boldsymbol{R_{pan}} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3.2}$$

where $\begin{bmatrix} x & y & z & 1 \end{bmatrix}^T$ is the 3D point in homogeneous world coordinates. The rotations are expressed trough the homogeneous matrix $\boldsymbol{R_{pan}} \in \mathbb{R}^{4\times4}$, while $\boldsymbol{C} \in \mathbb{R}^{3\times4}$ is the camera projection matrix. The computation returns the image plane coordinates $\begin{bmatrix} u/w & v/w \end{bmatrix}^T$ where the point is observed.
This model works fine for ideal Pan Cameras where the centre of projection coincides with the centre of rotation, but generally this assumption does not hold in reality. When a Pan Camera is built, it is very hard to check that the axis of rotation intersects the optical centre. Therefore, an improved Pan Camera model is needed.

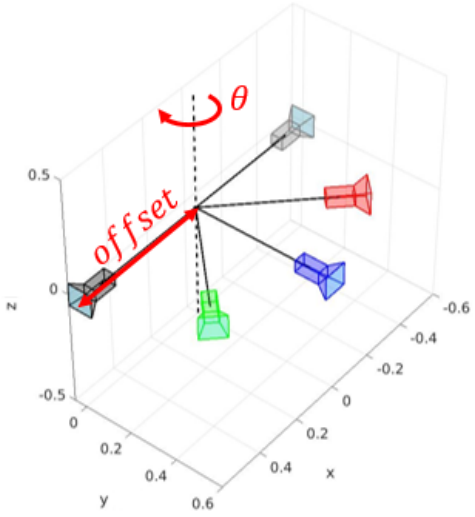

Figure 3.5: The rotation of the camera about the pan axis

In order to properly describe the system, a new parameter must be included in the model to get a better description of the camera geometry. As figure 3.5 shows, pan rotations can happen about any arbitrary axis in the space; the image plane and the optics rotate rigidly about this axis. The model can be formulated as:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \boldsymbol{C}\ \boldsymbol{T_{pan}}(offset)\ \boldsymbol{R_{pan}}(\theta)\ \boldsymbol{T_{pan}^{-1}}(offset) \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3.3}$$

where $\boldsymbol{R_{pan}} \in \mathbb{R}^{4\times4}$ is the homogeneous rotation matrix about the pan axis, while $\boldsymbol{T_{pan}} \in \mathbb{R}^{4\times4}$ is the homogeneous translation matrix that accounts for the offset between the origin and the rotation axis.

This model can be applied in order to describe T_{SC} as a function of the angle of rotation θ and of the $offset$, as suggested in equation 3.1. Therefore, to calibrate the system servo-camera, the transformation mapping from the camera to the servo reference frame can be written as follows:

$$T_{SC}(\theta,\ offset) = \boldsymbol{T_{pan}}(offset)\ \boldsymbol{R_{pan}}(\theta)\ \boldsymbol{T_{pan}^{-1}}(offset) \tag{3.4}$$

Equation 3.4 describes exactly a rotation θ that is performed at a distance $offset$ from the pan rotation axis.

3.4 Similarity to the Hand-eye calibration

The calibration idea that has been introduced presents many similarities to the Hand-Eye calibration problem. In its classical formulation, the goal is to recover the transformations linking the gripper of the robotic arm to the camera and the robot base reference to the world coordinate system. These mappings are indicated with X and Z in figure 3.6 respectively.

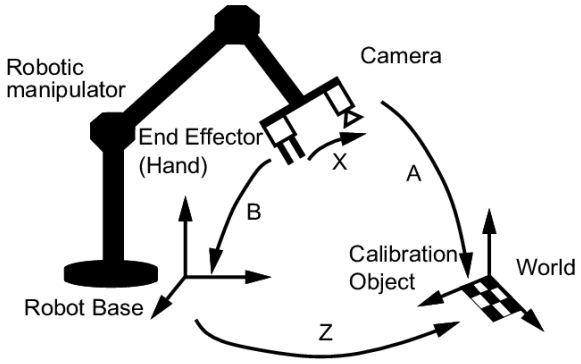

Figure 3.6: The Hand-Eye classical formulation - image taken from [15]

The calibration of the Pan Camera presents the same issues, but it focuses just on one of the two target transformations. In particular, the gripper is substituted by the servo and the overall process aims at finding the transformation between the servo and the camera.

Chapter 4

Toolbox pipeline

4.1 Introduction

In the previous section the calibration problem to solve has been introduced. Now the focus is on the pipeline of the complete calibration process. The proposed solution is schematically shown in figure 4.1. The camera is assumed intrinsically calibrated.

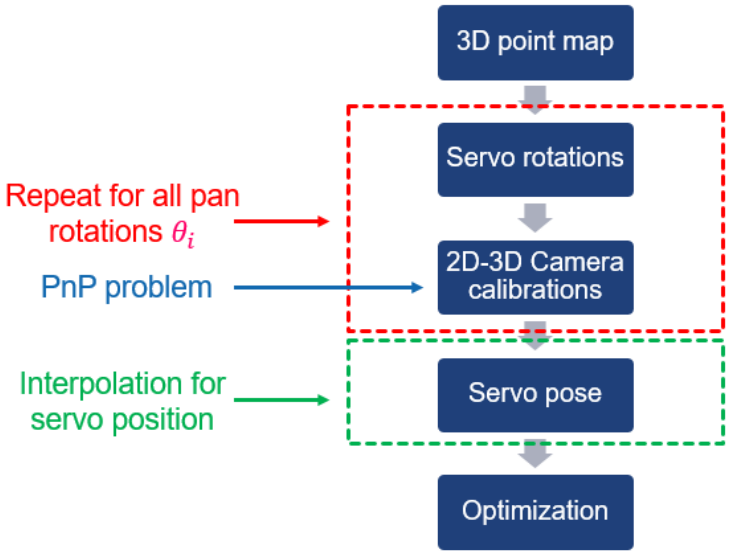

Figure 4.1: The pipeline of the calibration

The basic concept of the process is fairly simple.
The first step consists of the creation of a 3D point map, which represents the target needed to perform the camera 2D-3D calibration at each rotation step.
Once the position of the 3D points is known, the servo starts rotating and moves to the angular position θ_i for the i^{th} rotation. The camera pose is obtained from the 2D-3D points correspondences for each rotation step by solving a PnP problem as described in [10]. This process has to be repeated for all the possible servo rotations. Ideally, the range of rotation should be as wide as possible, in order to increase the accuracy of the method. As a reference, for our experiments the servo rotated for its full range of 180 degree. Furthermore, it must be considered that the amount of

allowed rotation is directly affected by the calibration target dimension and by the relative distance camera-target.
The servo position can be found by interpolating the different camera positions found for each rotation step, while the initial orientation of its reference system is arbitrary, since there are no constraints.
The last step of the pipeline consists of the optimization of the servo pose, of the angles of rotation θ_i for $i = 1, ..., n$ and of the $offset$ between the servo and the camera.

This calibration procedure has been tested both in simulation and in real world. The process is the same for both situations, with the exception of the 3D point map creation. In the following each step is explained in more details, while at the end of the chapter a comparison between the results from the simulation and the real system calibration is presented.

4.2 The toolbox pipeline step-by-step

4.2.1 3D point map

The creation of the 3D point map is crucial for a successful calibration. In a simulated environment the position of the 3D points is arbitrary and a dense map can be easily created. However, when the calibration is performed in a real scenario, this step introduces several difficulties.

The first approach made use of chessboards, in a typical camera calibration fashion. As it is shown in figure 4.2, two different types of chessboards have been used. However, due to the small size of the target and the limited distance between the camera and the target, the calibration results were terribly poor. Plus, since the implementation used OpenCV functions in order to detect the grids, both of the targets needed to be in full sight of the camera, limiting the servo rotation strongly. In order to avoid such constraint, the two chessboards were substituted by AprilTags chessboards (Appendix A - figure 4.3), which allowed to have partial detections in the image. Even though the amount of allowed rotation of the servo increased with respect to the previous approach, the targets were still too small and the returned numerical results were again not satisfactory.

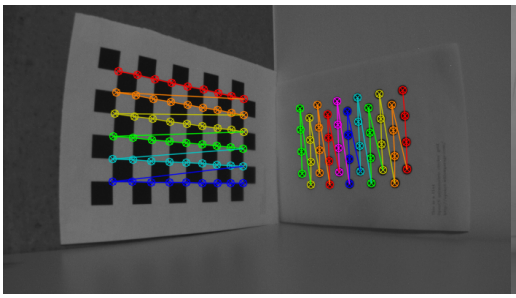

Figure 4.2: Chessboard and asymmetric circles grid with detections

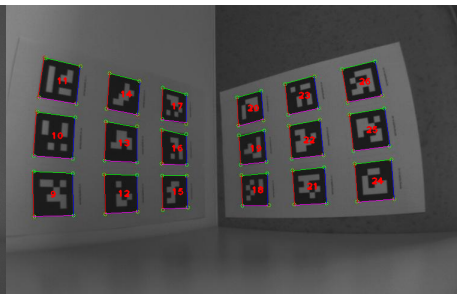

Figure 4.3: AprilTag chessboards with detections

Therefore, a solution that could allow larger servo rotations with a wider 3D point map was needed. The adopted solution involves again the use of AprilTags, but the overall process is completely different from the previous approaches.

To create the point map, a complete family of tags is attached to the walls (figure 4.4); the 3D points that constitute the calibration target are the four corners of each tag. Thus, to calibrate the system properly, it is necessary that the AprilTags cover a working volume as wide as possible. An example of tag detections in one frame is shown in figure 4.5.

Figure 4.4: Set-up of the tags for point map creation

Figure 4.5: The detections of the AprilTags for one frame

However, each tag has its own coordinate system; therefore, to express the points in a common reference frame, an algorithm able to find the transformations mapping from tag to world coordinates is needed. The basic concept for the concatenation of the transformations is shown in figure 4.6. The idea is to capture different images of the environment with a hand-held camera moving in the space and then concatenating the transformations T_{C,Tag_i} and $T_{W,C}$ to find T_{W,Tag_i}.

First of all, the world reference frame must be located. The easiest choice is to make it coincide with the reference system of one of the tags (called *reference tag*). In particular, the first image of the dataset must contain the reference tag to have a proper initialization. For each frame that has been captured, the transformation $T_{Tag_i,C}$ between the camera and the i^{th} tag in the image can be found by solving a PnP problem.

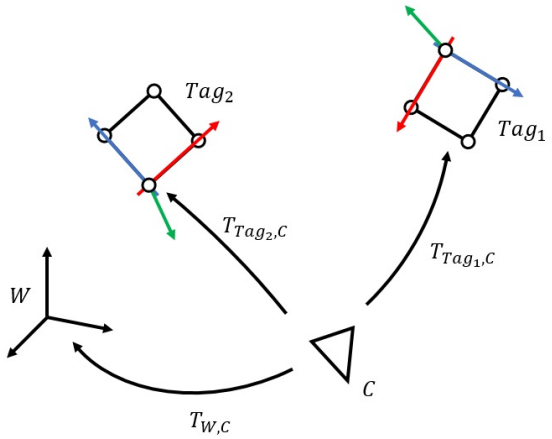

Figure 4.6: Transformation concatenation for the creation of the point map. C represents the camera, while W is the world reference frame

Then, to express the four corners of the i^{th} tag in world coordinates, the transformation linking the i^{th} reference system to the world must be found. There are two possible situation for this concatenation, depending on whether the reference tag is present in the image or not.

If the reference tag is visible, as in figure 4.7, the transformation between the reference and all the other tags can be obtained directly:

$$T_{W,Tag_i} = T_{W,C}\ T^{-1}_{Tag_i,C} \tag{4.1}$$

where W is the world reference frame and C is the camera.

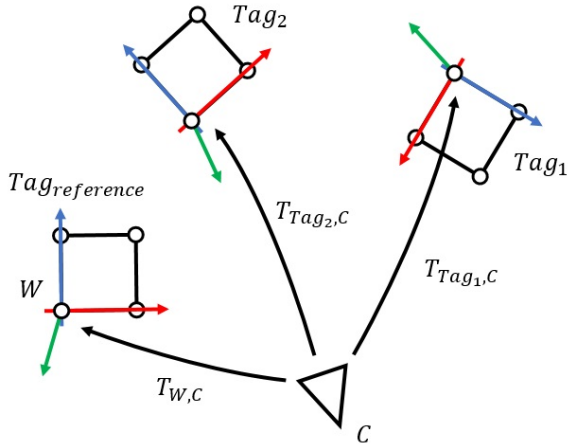

Figure 4.7: Situation 1 - the reference tag is present in the image

If the reference is missing, then to obtain T_{W,Tag_i} the transformation $T_{Tag_i,C}$ must be multiplied by T_{Tag_j,Tag_i}, where Tag_j is another tag in the image whose transformation T_{W,Tag_j} is known. This situation is depicted in figure 4.8. In order to make the procedure work, small movements of the camera between consecutive frames are assumed. The concatenation of the transformations is as follows:

$$T_{W,Tag_i} = T_{W,Tag_j}\ T_{Tag_j,C}\ T^{-1}_{Tag_i,C} \tag{4.2}$$

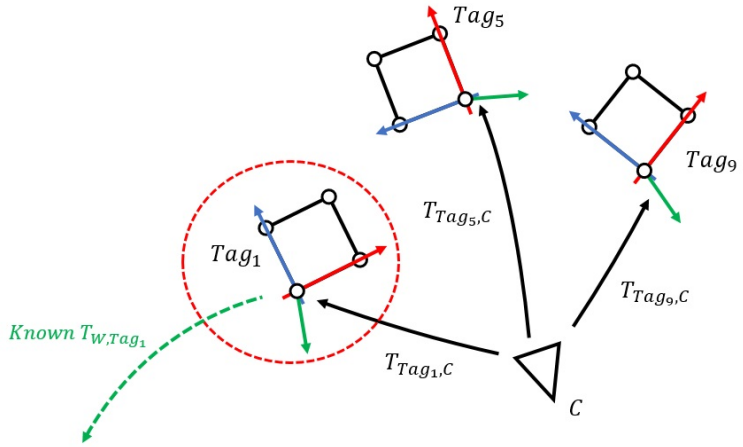

Figure 4.8: Situation 2 - the reference tag is missing in the image. Here, tag number 1 has already been localized in the world reference system

Once all the mappings between all the tags and the reference are known by using equation 4.1 and equation 4.2, the 3D point map can be built. Since the positions of the corners of the tag in the local reference frame of each tag are known, their positions in world coordinates can be easily retrieved:

$$ {}_{W}P_j = T_{W,Tag_i} \ {}_{Tag_i}\tilde{P}_j \ for \ j = 1,2,3,4 \tag{4.3} $$

where ${}_{Tag_i}\tilde{P}_j$ is the j^{th} corner of the tag in the local reference frame expressed in homogeneous coordinates.

The overall procedure is summarized in Algorithm 1. To increase the robustness of the process, only frames with at least four tags detections are processed in order to make the concatenation process more reliable. Moreover, the transformations are concatenated as described in equations 4.1 and 4.2 only if the average re-projection error for the computation of $T_{Tag_i,C}$ in the current frame is below a threshold τ defined by the user. In our experiments, τ was set to 2 pixel.

The procedure that has been described so far returns a first guess of the point map, which is then refined by bundle adjustment. The goal is the minimization of the re-projection error by solving a non-linear least squares optimization problem using Levenberg–Marquardt algorithm.

$$ (P^*_{1,\dots,n}, T^*_{WC_{1,\dots,N_f}}) = \underset{P_{1,\dots,n},\ T_{WC_{1,\dots,N_f}}}{\arg\min} \sum_{i=0}^{N_f} \sum_{j=0}^{N^i_{obs}} ||C(T_{WC_i},\ P_{1,\dots,n}) - m^i_{j,2D}||^2 \tag{4.4} $$

The minimization problem can be formulated as in equation 4.4, where:

- n is the number of 3D points;
- N_f is the number of frames;
- $C(T_{W,C_i},\ P_{1,\dots,n})$ is the reprojection model;
- T_{W,C_i} the camera pose in the i^{th} frame;
- N^i_{obs} the number of 2D observations $m^i_{j,2D}$ in the i^{th} frame

The optimized point map is shown in figure 4.9.

Algorithm 1: the creation of the point map

Data: Images of the environment
Result: First guess 3D point map
initialize database transformations;
forall *images* **do**
  detect AprilTags;
  if *number of tags* ≥ 4 **then**
    for $i = 1$ *to number of detected tags* **do**
      find $T_{Tag_i,C}$;
      if *reprojection error* $\leq \tau$ **then**
        if *reference tag is present* **then**
          $T_{W,Tag_i} = T_{W,C}\ T^{-1}_{Tag_i,C}$;
          save T_{W,Tag_i} in the database;
        else
          search detections for tags already localized;
          if *found localized tag* Tag_j **then**
            $T_{W,Tag_i} = T_{W,Tag_j}\ T_{Tag_j,C}\ T^{-1}_{Tag_i,C}$;
            save T_{W,Tag_i} in the database;
          else
            discard image;
          end
        end
      else
        discard image;
      end
    end
  end
end

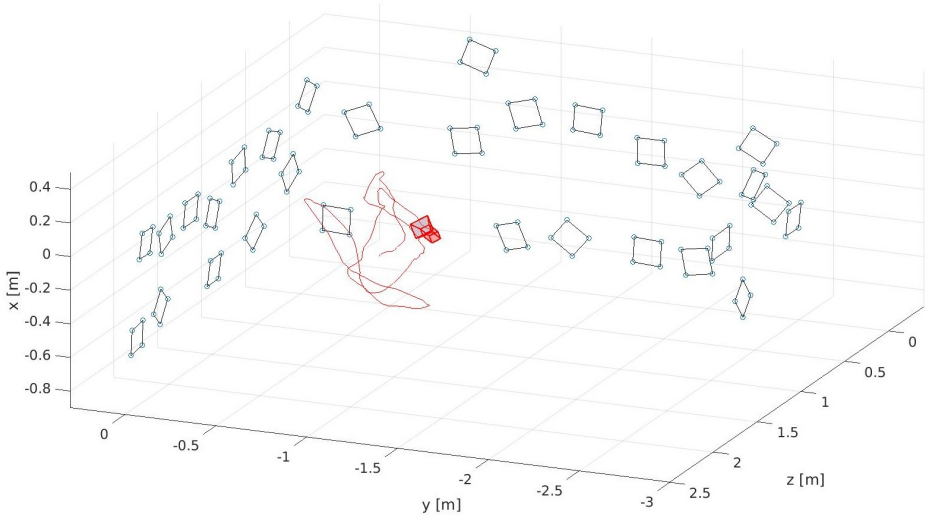

Figure 4.9: The point map after bundle adjustment with the camera trajectory

4.2.2 Servo rotation and camera calibration

After having computed a refined 3D point map, the system servo-camera is used. Ideally, the servo should be able to rotate for its full range in order to obtain more accurate results from the calibration.
This step is fairly simple. The servo starts rotating and it moves towards the new angular position. The rotation angle is indicated with θ, which is the absolute angle measured from the reference position set at 0 degree.

For each rotation step, the camera pose can be found by solving a PnP problem. Even though the 3D points are known with high accuracy, the solution found by solving the 2D-3D correspondences problem may be inaccurate due to external factors, such as illumination conditions or failure in the AprilTags detections. Therefore, only the camera poses that can be found with an average re-projection error in the current frame lower than an user-defined threshold τ_{rot} are saved. For our experiments, τ_{rot} was set to 4 pixel.
For the hardware and its set-up, refer to Section 3.1.

4.2.3 Servo pose

The servo position can be directly obtained from the camera positions. Ideally, the camera should move along a circumference, therefore the servo position should coincide with its centre. In order to find it, various step must be followed.

1. Find least squares fitting plane for the camera positions;
2. Project the camera positions on the plane;
3. Fit a circumference through the projected camera positions and find its centre.

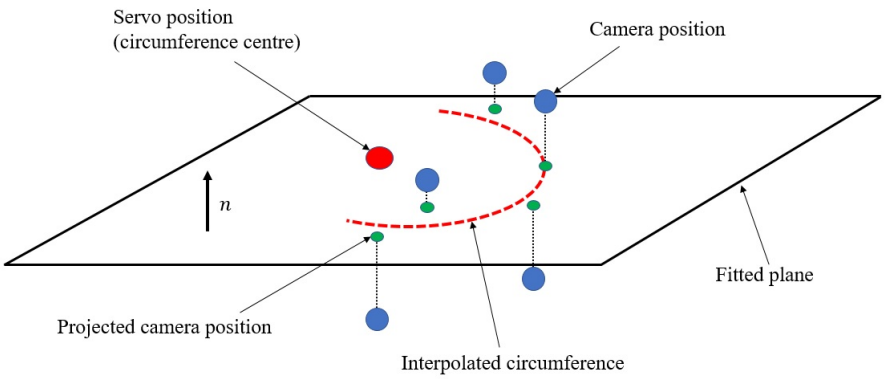

Figure 4.10: The interpolation process

The process is depicted in figure 4.10.
The different camera positions in world coordinates are indicated with $p_i \in \mathbb{R}^3$ $i = 1, ..., N$ and they can be expressed in the matrix form

$$P = \begin{bmatrix} p_1^T \\ p_2^T \\ \vdots \\ p_N^T \end{bmatrix} \in \mathbb{R}^{N \times 3}$$

The computation of the least squares plane is fairly simple. The first step consists of the computation of sample covariance matrix Σ:

$$\Sigma = \frac{1}{N-1}(P - \mu)^T (P - \mu)$$

where $\mu \in \mathbb{R}^3$ is the mean of the camera positions:

$$\mu = \frac{1}{N}\sum_{i=0}^{N} p_i$$

The normal to the plane n can be found by using the Singular Value Decomposition of Σ.

$$USV^T = \Sigma$$

The unit norm n corresponds to the third column of U. Since n is known, the plane can be written as

$$ax + by + cz = d$$

where a, b and c are the entries of n. Instead, d is the intercept of the plane that can found by computing the dot product $n^T\mu$, since the mean μ belongs to the fitted plane.
Once the plane is known, each 3D camera position $p_i = [x_c, y_c, z_c]^T$ can be projected onto it:

$$\begin{cases} x_c^{proj} = x_c - a\frac{ax_c+by_c+cz_c+d}{a^2+b^2+c^2} \\ y_c^{proj} = y_c - b\frac{ax_c+by_c+cz_c+d}{a^2+b^2+c^2} \\ z_c^{proj} = z_c - c\frac{ax_c+by_c+cz_c+d}{a^2+b^2+c^2} \end{cases}$$

The servo position can be found by fitting a circumference through the projected points and computing its centre $[x_{centre}\ y_{centre}\ z_{centre}]^T$.

Differently from the position, the initial servo orientation is not subjected to any constraints; therefore it is arbitrary. For the sake of simplicity, it is assumed that the initial orientation of the servo reference frame is aligned with the one of the camera in the reference position, with additional rotations of 180 degree about the z axis and of -90 degree about the x axis, as shown in figure 4.11.

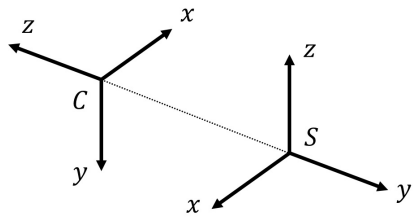

Figure 4.11: The initial relative orientation of the servo S and the camera C reference systems

4.2.4 Optimization

The last step of the toolbox pipeline is the optimization.

$$(T^*_{WS}, \theta^*_{1,\dots,n}, offset^*) = \underset{T_{WS},\theta_{1,\dots,n},offset}{\arg\min} \sum_{i=0}^{N_{frames}} \sum_{j=0}^{N^i_{obs}} ||C_{pan}(T_{WS}, \theta_{1,\dots,n}, offset) - m^i_{j,2D}||^2 \tag{4.5}$$

The problem to solve is again a minimization of the re-projection error and it can be formulated as in the equation 4.5 where:

- N_f is the number of frames, coincident with the number of rotation steps;

- $C_{pan}(T_{WS}, \theta_{1,\dots,n}, offset)$ is the re-projection model that takes into account the analytical model of the Pan Camera described in equation 3.4;
- N^i_{obs} the number of 2D observations $m^i_{j,2D}$ in the i^{th} frame.

By solving the minimization problem, the optimized servo pose T^*_{WS} can be recovered. Plus, from the optimized angles of rotations $\theta^*_{1,\dots,n}$ and the $offset^*$ between the camera and the servo it is possible to compute the transformations T^*_{SC} following the formulation described in Section 3.3:

$$T^*_{SC}(\theta^*, offset^*) = \boldsymbol{T_{pan}}(offset^*)\ \boldsymbol{R_{pan}}(\theta^*)\ \boldsymbol{T^{-1}_{pan}}(offset^*) \tag{4.6}$$

4.3 The simulation

Before being applied to real set-up, the pipeline has been tested in simulation. The process is shown in figure 4.1 with the exception of the point map creation. Since the simulated calibration is performed in an arbitrary environment, the 3D points can be easily located by the user. In our experiments, the point map was constituted by three walls as in figure 4.12. On each wall, 16 7-by-7 grids were placed. The colours of the walls do not have any particular meaning; they are used in order to make the 3D representation of the point map more clear to the reader.

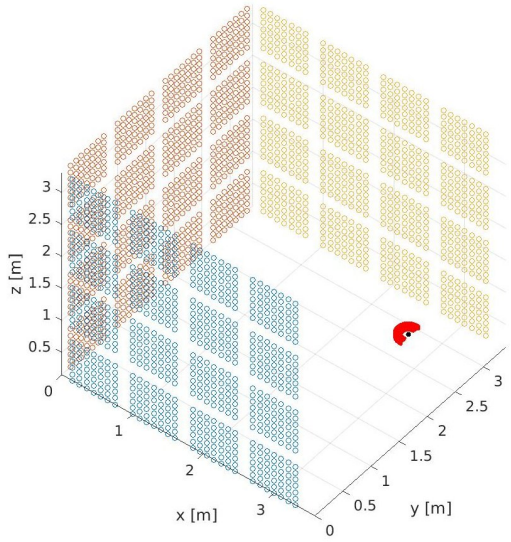

Figure 4.12: The environment of the simulation

The simulation was built to understand what the expected performances of the toolbox could be, depending on the servo resolution. Therefore, in order to create a simulation that could resemble reality as close as possible, the servo rotation was corrupted by a 0 mean Gaussian noise $\mathcal{N}(0, \sigma^2)$. In particular, the noise variance σ^2 was chosen depending on the servo resolution that was currently tested (table 4.1).

The results of the simulation are shown in figure 4.13, where the average re-projection error is reported as a function of the servo angular position (the x-axis is discretized with a step of 10 degree for a better representation). The results shown in the plot are the mean values of 50 runs of the simulation.
As expected, as the servo resolution deteriorates (i.e. higher rotation step), the

error increases. Plus, the minimum error is achieved at the reference position at 0 degree. As the servo moves, the error tends to increase. However, the re-projection error remains in an acceptable range for smaller servo resolution.

Servo resolution [deg]	σ^2 [deg^2]
0.5	0.1
1.0	0.2
2.0	0.4
5.0	1.0
10.0	2.0

Table 4.1: The noise variance σ^2 depending on the servo resolution

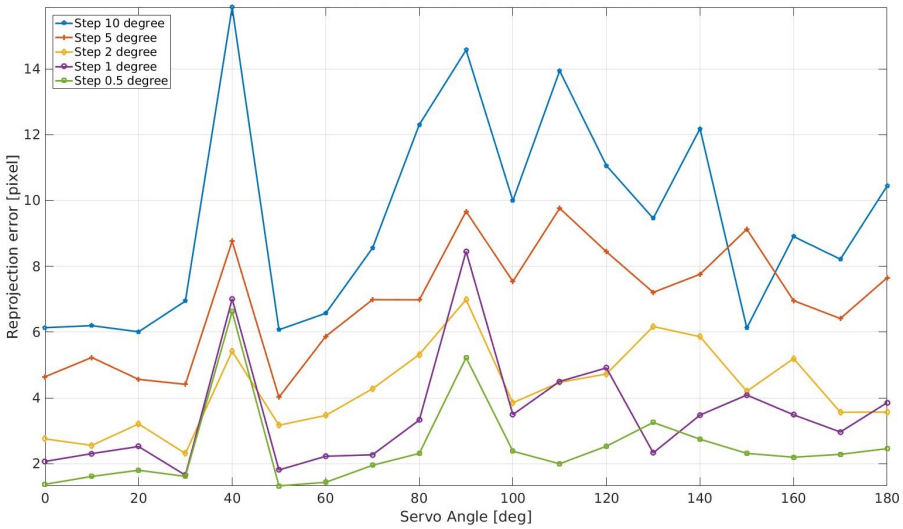

Figure 4.13: The error of the simulation for different servo resolutions in different servo positions

4.4 The real set-up calibration

In order to asses the performances of the calibration toolbox on the real set-up, the average re-projection error per frame has been plotted against the servo rotation angle in figure 4.14. Once again, the error is reported only for a discretization step of 10 degree for a clearer representation.

Once again, as the servo moves away from the reference position at 0 degree, the error increases. Plus, differently from the simulation, the creation of the calibration target is not ideal, therefore a new source of error is introduced.
However, the impact of the 3D point map evaluation should be limited on the overall performances. The process of capturing the images for the creation of the point map is simple and can be completed efficiently even by a non-expert user.

4.5 Comparison of the results

The results of the simulation and of the real case are compared to get a better feeling of the performances of the toolbox. As figure 4.15 shows, the performance of the toolbox in the real world is worse than the one of the simulation. The real set-up

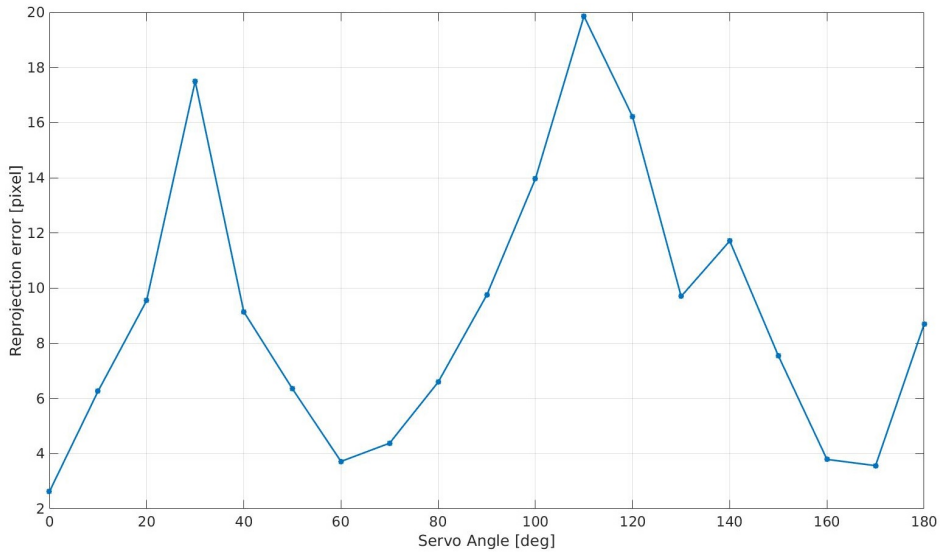

Figure 4.14: The error of the calibration in the real scenario

behaviour is indicated by the black line. In particular, a qualitative comparison reveals that the real servo (with nominal resolution 1 degree) has a behaviour close to the simulated servo with resolution 10 degree.
The degradation of the results is due to various effects that were not taken into account in the simulation. For instance, the link between the camera and servo is not ideal and therefore there are several unmodelled relative movements between the servo and the camera. Plus, the servo is not perfectly attached to the ground, which causes some backslash during the rotations.
Another important effect is the non-alignment of the camera with respect to the centre of rotation. Due to inaccuracies when the set-up was built, the offset between the camera and the servo is not perfectly radial.

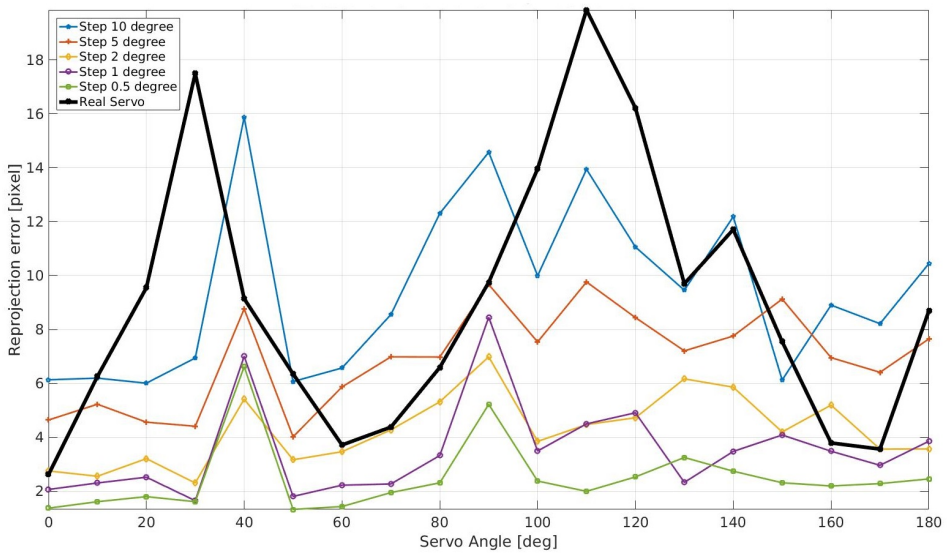

Figure 4.15: Comparison of the simulation error against the real scenario performance

The comparison between the simulation and the real world scenario can give a rough insight of the toolbox performances. In the next section, the toolbox is evaluated more rigorously. Moreover, its performances when it is used in combination with an UAV pose tracking framework are analysed.

Chapter 5

Toolbox evaluation

In this section, the evaluation of the toolbox is presented. In particular, two different evaluations have been performed:

1. Toolbox versus `Kalibr`
2. Toolbox in a maker tracking framework

In the following, the evaluation set-up is first described; then the procedure and the results of the evaluations are shown.

5.1 Evaluation set-up

The hardware set-up used for the evaluation of the toolbox consists of two cameras, one fixed and one able to move. In the experiments, the fixed camera was camera 1 of the VI-Sensor, while to second one is the BlueFox2 Camera mounted on the servo (figure 5.1).

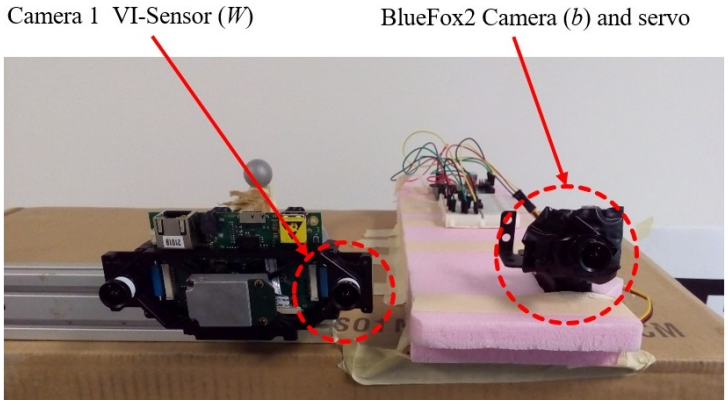

Figure 5.1: The hardware set-up for the evaluation

The world reference frame is assumed to be coincident with the coordinate system of the VI-Sensor. Therefore, the reference system of the fixed camera is indicated with W, while the BlueFox2 is represented with b. Moreover, the system servo S - BlueFox2 Camera b is assumed calibrated with the toolbox before the start of the evaluation process.
Finally, a total of 5 datasets have been collected for both the evaluations. Each dataset is composed by 15 different servo rotations in the range [75 - 117 degree] with a step rotation of 3 degree.

5.2 Toolbox versus `Kalibr`

5.2.1 Procedure

The first evaluation consists of the comparison between the results obtained from the toolbox and the ones returned by `Kalibr`.

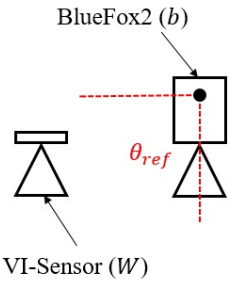

Figure 5.2: The reference position of the two cameras

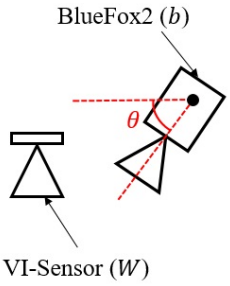

Figure 5.3: Rotated position of the two cameras

The first step of the evaluation is the choice of the reference position of the BlueFox2 Camera with respect to the VI-Sensor. For both the evaluations, it is chosen to be at the servo angular position θ_{ref} equal to 90 degree, as shown in figure 5.2. In this position, a stereo calibration of the two cameras is performed using `Kalibr`. With this approach, the transformation mapping from the reference system of the moving camera to the fixed one can be retrieved easily. Such transformation is indicated with $T_{Wb}^{Kalibr}(\theta_{ref})$.

The idea that stands behind the evaluation is fairly simple. Once $T_{Wb}^{Kalibr}(\theta_{ref})$ is known, it is possible to compare the transformations $T_{Wb}^{Kalibr}(\theta)$ and $T_{Wb}^{Toolbox}(\theta)$ mapping from the BlueFox2 Camera to the VI-Sensor coordinate system for a given angular position θ of the servo (figure 5.3).

$T_{Wb}^{Kalibr}(\theta)$ can be obtained directly from the stereo calibration of the system, while $T_{Wb}^{Toolbox}(\theta)$ can be computed by concatenating the transformation returned by the toolbox. The concatenation proceeds as follows:

$$T_{Wb}^{Toolbox}(\theta) = T_{Wb}^{Kalibr}(\theta_{ref})\ (T_{Sb}^{Toolbox}(\theta_{ref}))^{-1}\ T_{Sb}^{Toolbox}(\theta)$$

where $T_{Sb}^{Toolbox}(\theta_{ref})$ and $T_{Sb}^{Toolbox}(\theta)$ can be obtained from the calibration of the system servo-camera.

The numerical evaluation is performed by computing a homogeneous error matrix $E(\theta)$:

$$E(\theta) = T_{Wb}^{Kalibr}(\theta)\ (T_{Wb}^{Toolbox}(\theta))^{-1}$$

Ideally, $E(\theta)$ is an identity matrix. Obviously, due to the numerous sources of error that have already been described, it is completely different from the ideal case.
The basic concept of this evaluation is the comparison of the position and orientation of the BlueFox2 Camera with respect to the VI-Sensor as they are computed with two different tools. Thus, $E(\theta)$ contains the information about the amount of error in both orientation and position calculation of the proposed toolbox with respect to `Kalibr`.

Figure 5.4: The data for the first evaluation from the Bluefox2 camera (left) and the camera 1 of the VI-Sensor (right)

The data collection

To perform the stereo calibration with `Kalibr`, a series of data such as the ones in figure 5.4 has been collected using a chessboard. The image on the left is the view from the BlueFox2 camera, while the one on the right is the image captured by camera 1 of the VI-Sensor.
Since the two cameras were not synchronized initially, the recorded streams of images had to be preprocessed before being used for the comparison computation. In order to synchronize the images, the BlueFox2 Camera recorded at 65 FPS, while the VI-Sensor used a much lower frame rate, nominally 2 FPS. Once the first asynchronous data have been collected, the synchronization of the streams could be easily completed by comparing the timestamps of all the images. The process was automated by using a simple `Python` script.
The process was repeated for each servo angular position θ in the specified range.

5.2.2 Results

The homogeneous error matrix $E(\theta)$ was calculated for each servo rotation for all the dataset. Then, for each $E(\theta)$ computation, the error could be extracted. In particular, the absolute error in both position and orientation is analysed.

Absolute error in orientation

The amount of error in orientation can be obtained by extracting the error rotation matrix $R(\theta) \in \mathbb{R}^{3\times3}$ from $E(\theta)$ and then computing the three rotation angles *roll*, *pan* and *tilt*. It is possible to find the absolute error on roll, pan and tilt angles in the servo position θ as follows:

$$\begin{cases} |error_{roll}(\theta)| = |atan2(R_{23}(\theta), R_{13}(\theta))| \\ |error_{pan}(\theta)| = |atan2(\sqrt{R_{13}(\theta)^2 + R_{23}(\theta)^2}, R_{33}(\theta))| \\ |error_{tilt}(\theta)| = |atan2(R_{32}(\theta), -R_{31}(\theta))| \end{cases}$$

This process is repeated for each servo angular position θ and for each of the 5 datasets. The absolute error in all the components can be averaged over the datasets and then represented as a function of θ. The results are shown in figure 5.5.

As expected, as the servo moves from the reference position, the error increases in roll, pan and tilt. Moreover, the largest amount of error is present on the pan, since it is directly affected by the servo rotation. The smallest values are on the tilt, while the roll presents intermediate errors. The misalignment of the two cameras plays a

crucial role. Since the BlueFox2 Camera and the VI-Sensor are not perfectly aligned, the VI-Sensor sees every rotation of the system servo-camera as the composition of roll and pan. However, the errors remain in an acceptable range. Slightly worse results are obtained by the pan for servo positions in the range [75-90] degree, where the maximum peak (∼3 degree) is reached.

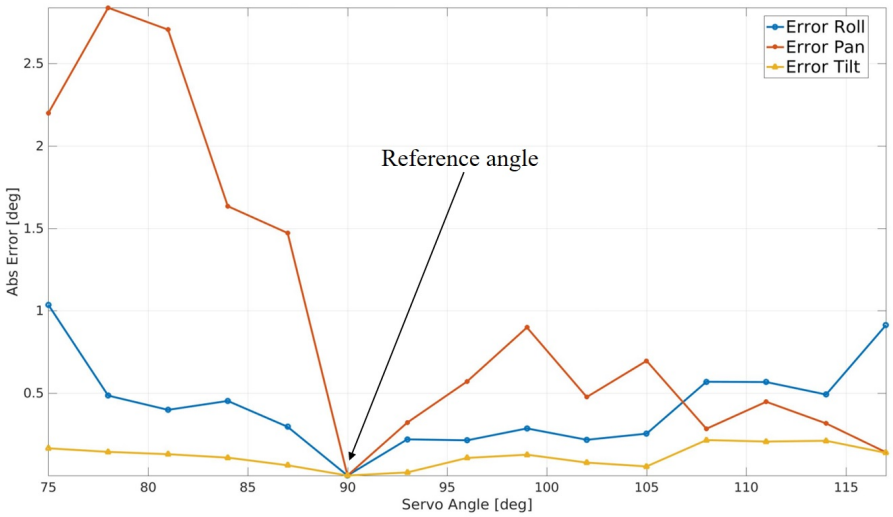

Figure 5.5: The absolute error in orientation

Absolute error in position

The same kind of evaluation can be done for the position error. It is possible to extract the translational error vector $t(\theta)$ from $E(\theta)$ and average the absolute values of the errors on x, y and z over the datasets. The results are shown in Figure 5.6, where the averaged error is presented as a function of the angular position of the servo. Once again, as the servo moves away from the reference position, the error increases. In this case, the error has similar values among the three components x,

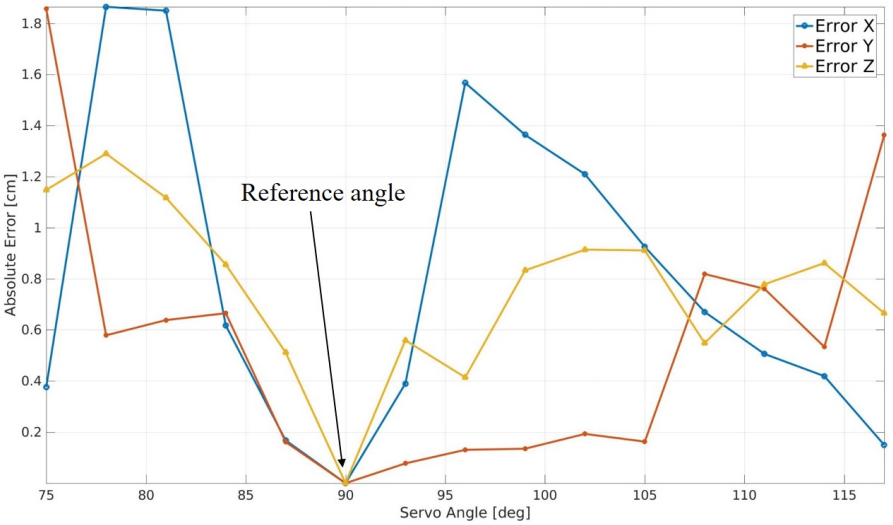

Figure 5.6: The absolute error in position

y and z. As in the case of the orientation, the reason of such a decomposition in the three directions is due to the misalignement of the two cameras.
To get a better feeling of the results, the average distance of between the two cameras over the datasets is about 13 cm. Here the error reaches a peak of 1.85cm on the x and y, but it remains constrained in an acceptable range for all the rotation angles θ.

5.3 Toolbox in a maker tracking framework

5.3.1 Procedure

The second evaluation is based on the performances of the toolbox when it is employed in combination with a UAV pose tracking algorithm. As explained in Section 2.2, the algorithm for pose tracking that has been utilized is the one developed by Marco Moos. Thus, every transformation that is obtained by this pipeline is indicated with $Moos$ as superscript.

Figure 5.7: The data for the second evaluation - tracking of the marker. The image on the left is recorded by the Bluefox2 Camera, while the one on the right is obtained from camera 1 of the VI-Sensor

The goal of the second evaluation is to understand the impact of the error introduced by the toolbox in the pose tracking process. To asses the performances, data such as the one in figure 5.7 have been collected. Moreover, instead of tracking directly an UAV, the object that has been utilized in the experiment is the marker shown in figure 5.8.
The marker has been tracked at the same time both from camera 1 of the VI-Sensor and from the BlueFox2 Camera, with an average distance cameras-marker of ∼2.5 meter. After the marker has been detected for different servo rotations, the data could be processed. As for the first evaluation, the reference position θ_{ref} of the servo is set at 90 degree in order to evaluate $T_{Wb}^{Kalibr}(\theta_{ref})$.

Similarly to the previous evaluation, the idea is to compare two transformations, T_{WM}^{Moos} and $T_{WM}^{Toolbox}(\theta)$, where M represents the reference frame of the marker. As it is possible to see from figure 5.9, the transformation T_{WM}^{Moos} linking the marker to the world reference system (assumed coincident with the one of the VI-Sensor) can be obtained directly from the tracking algorithm. Instead, to obtain the transformation $T_{WM}^{Toolbox}(\theta)$, the results from the tracking must be concatenated

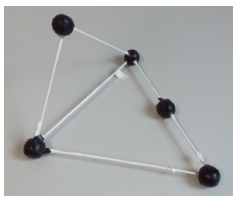

Figure 5.8: The marker for the tracking

with the output of the calibration toolbox:

$$T_{WM}^{Toolbox}(\theta) = T_{Wb}^{Toolbox}(\theta)\ T_{bM}^{Moos}$$

where

$$T_{Wb}^{Toolbox}(\theta) = T_{Wb}^{Kalibr}(\theta_{ref})\ (T_{Sb}^{Toolbox}(\theta_{ref}))^{-1}\ T_{Sb}^{Toolbox}(\theta)$$

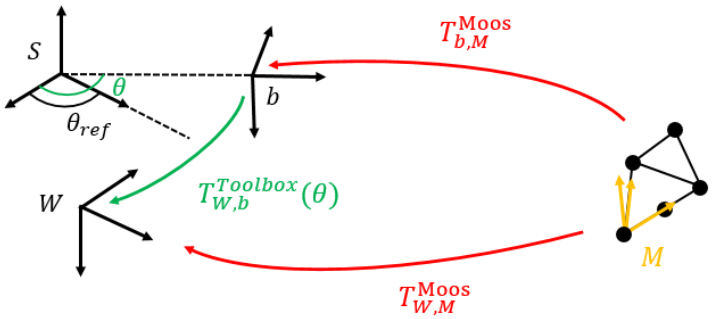

Figure 5.9: The transformation concatenation for the second evaluation

The error at each servo position θ is then obtained by

$$E(\theta) = T_{WM}^{Moos}\ (T_{WM}^{Toolbox}(\theta))^{-1}$$

The meaning of $E(\theta)$ is well defined. The basic idea of the evaluation is to obtain the pose of the marker in the world coordinate system by using two different approaches and then compare the results.

In the first approach, the transformation T_{WM}^{Moos} is directly obtained from the tracking algorithm without any further processing. Therefore, even if it encodes the intrinsic error of the tracking pipeline (the pose of the marker is different from the real one - for more details see [11]), this transformation is assumed to be the ground truth. Instead, the second transformation $T_{WM}^{Toolbox}(\theta)$ contains information about the error of both the tracking and the calibration toolbox.
The two transformations, T_{WM}^{Moos} and $T_{WM}^{Toolbox}(\theta)$, should compute the same pose of the marker. However, due to the errors that acts on each one of them, the results differ. This situation is shown in figure 5.10. The marker is seen in position A by the VI-Sensor when the pose is obtained directly, but when $T_{WM}^{Toolbox}(\theta)$ is utilized, the marker is seen in position B with a different orientation with respect to A. The error matrix $E(\theta)$ describes the difference in orientation and position between A and B. As for the first evaluation, the error can be divided into orientation and position error; in particular, the error is analysed in the absolute value.

Baseline error

To grab a better understanding of the results, an error baseline is introduced. To compute it, the same idea of concatenated transformations can be exploited, but using the two fixed cameras of the VI-Sensor.
The process is basically the same. First, the stereo camera is calibrated with `Kalibr` in order to obtain T_{WC}^{Kalibr}, where W represent camera 1 of the VI-Sensor, while C indicates the reference system of the second camera.
Then, as depicted in figure 5.11, the same procedure can be applied. The comparison is made between T_{WM}^{Moos} and $\tilde{T}_{WM} = T_{WC}^{Kalibr}\ T_{CM}^{Moos}$.

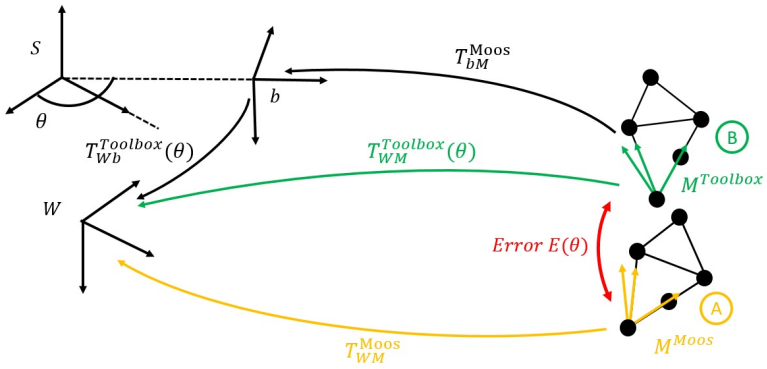

Figure 5.10: The meaning of the error $E(\theta)$: the marker is assumed in position A with a given orientation when the pose T_{WM}^{Moos} is used directly, while it is assumed in position B with a different orientation when $T_{WM}^{Toolbox}(\theta)$ is utilized. $E(\theta)$ encodes the difference in both rotation and orientation between A and B

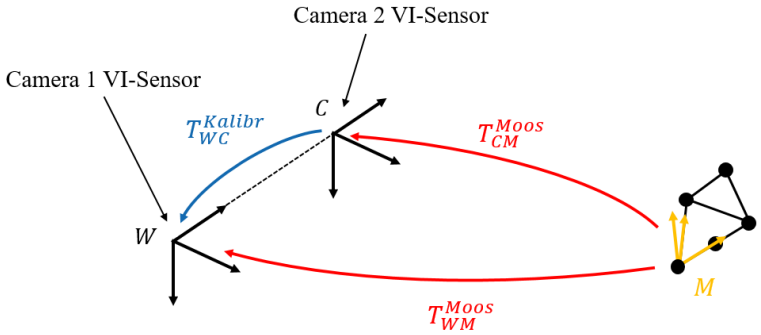

Figure 5.11: The transformation concatenation for the baseline computation

Assuming that the calibration of the VI-Sensor is performed with high precision, the error matrix $E_{baseline}$ encodes the error of the tracking pipeline alone. It can be decomposed into rotation and translation, giving an idea of the expected performances of the toolbox in combination with the tracking algorithm. The results of the baseline evaluation are shown in table 5.1.
Another possible approach for the error baseline computation could have been the use of an external tool for pose tracking of the marker, such as *Vicon* room.

Angle	Absolute error [deg]
Roll	1.41
Pan	1.48
Tilt	0.70
Position	Absolute error [m]
x	0.030
y	0.024
z	0.011

Table 5.1: Absolute error baseline

5.3.2 Results

Absolute error in orientation

In figure 5.12 the absolute errors in roll, pan and tilt are represented as a function of the servo angle of rotation θ. The baseline is represented with a red dotted line, while the blue line is the mean of all the absolute errors for all datasets; the vertical lines represent the variance.
The performances of the toolbox are satisfactory. The absolute error does not exceed the 6 degree in all the components, which have all a similar behaviour. The pan angle shows a slightly worse results than the roll and the tilt. This trend was expected, since it is directly affected by the servo movements.
The results for some of the servo rotation angles are reported in Table 5.2.

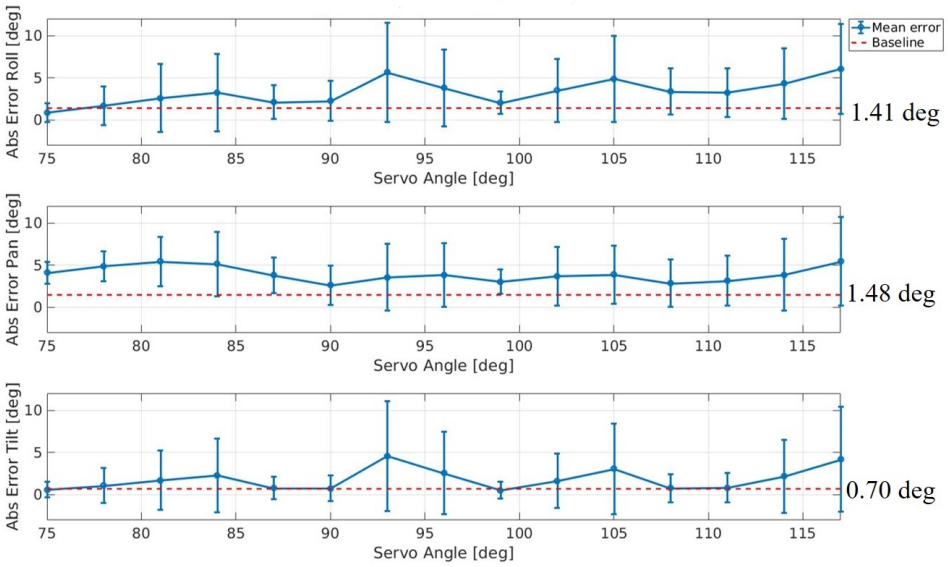

Figure 5.12: The absolute error in orientation

(a) Absolute error roll

Servo angle θ [deg]	Mean [deg]	Standard deviation [deg]
75	0.896	1.114
90	2.280	2.332
105	4.888	5.112
117	6.059	5.303

(b) Absolute error pan

Servo angle θ [deg]	Mean [deg]	Standard deviation [deg]
75	4.107	1.305
90	2.628	2.343
105	3.880	3.454
117	5.433	5.239

(c) Absolute error tilt

Servo angle θ [deg]	Mean [deg]	Standard deviation [deg]
75	0.605	0.903
90	0.760	1.507
105	3.079	5.232
117	4.173	6.205

Table 5.2: Absolute orientation error

Absolute error in position

Figure 5.13 shows the absolute errors in position along x, y and z directions. The error is always close to the baseline and presents satisfactory trends. The highest value is reached on the z when the servo is in position 93 degree. The error is high ($\sim$0.5 m on a distance marker-camera of 2.5 m), but it is due to a failure in the marker pose tracking process. Therefore, this peak should be considered an outlier. The results for some servo rotation angles are reported in Table 5.3.

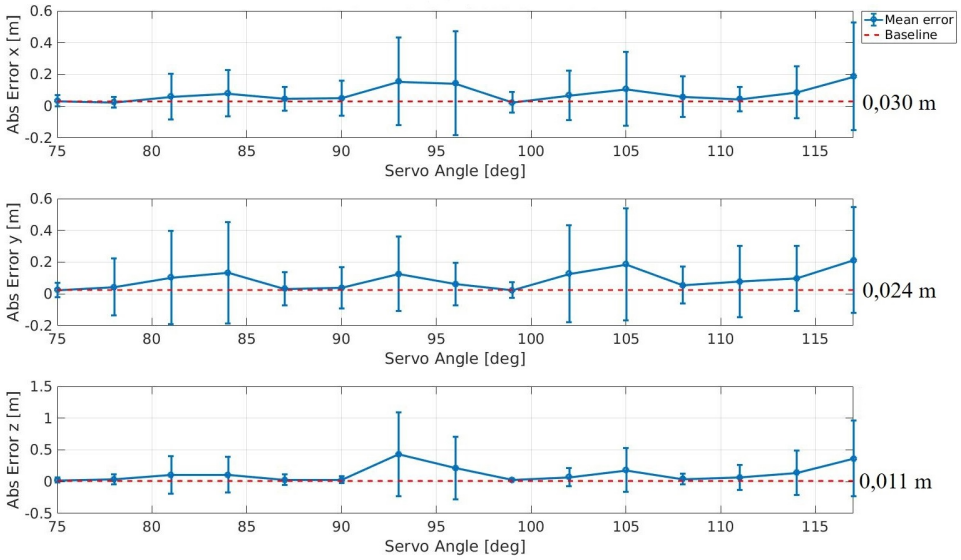

Figure 5.13: The absolute error in position

(a) Absolute error x

Servo angle θ [deg]	**Mean [m]**	**Standard deviation [m]**
75	0.033	0.034
90	0.052	0.110
105	0.110	0.231
117	0.187	0.338

(b) Absolute error y

Servo angle θ [deg]	**Mean [m]**	**Standard deviation [m]**
75	0.025	0.046
90	0.039	0.131
105	0.187	0.353
117	0.213	0.334

(c) Absolute error z

Servo angle θ [deg]	**Mean [m]**	**Standard deviation [m]**
75	0.025	0.045
90	0.030	0.052
105	0.183	0.346
117	0.366	0.598

Table 5.3: Absolute position error

Chapter 6

Conclusion and future work

6.1 Conclusions

In this report a novel toolbox for the calibration of Pan Cameras has been proposed. Moreover, the model that has been implemented is different from the one that is usually employed. The centre of rotation can be on an arbitrary axis in the space; therefore, the model takes into account not only the rotation of the camera, but also the offset that exists between the camera and the centre of rotation.

The calibration pipeline can be divided into four main steps. Initially, a 3D point map has to be created using an approach similar to SLAM in order to build the calibration target. Once the position of the points is known, the system servo-camera starts rotating. For each rotation step, the camera pose is computed by solving a 2D-3D correspondence problem. When the rotations are completed, the servo position can be easily retrieved by interpolating the different camera positions, while the initial orientation is arbitrary. The last step consists of the optimization of the servo pose, of the angles of rotation of the servo and of the offset between the camera and the centre of rotation.

Overall, the toolbox shows satisfactory performances, given the presence of many sources of error. A comparison between the simulation and the calibration performed with the real set-up indicates that many effects that are not take into account in the model influence the results; for instance, the non-ideal link between the camera and the servo and the backslash when the system rotates have not been included in the mathematical formulation of the problem.

Furthermore, even when the toolbox is used in combination with a marker tracking algorithm, the error it introduces on the marker pose estimation is limited, both in orientation and position. Therefore, the toolbox can be exploited for practical application such as UAV tracking, expecting good performances.

6.2 Future work

The toolbox presents a high number of possible expansions.

The most immediate upgrade would be the introduction of another degree of freedom, i.e. the tilt movement. The system becomes more complex, since two servo would be required, as shown in figure 6.1. By introducing anther rotation, the system could be used for application such as UAV tracking. In fact, the movements

of the system could cover a much larger field of view with respect to the proposed solution.

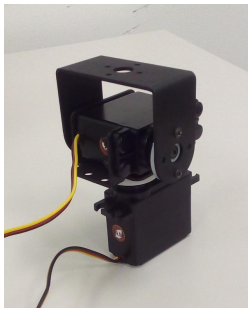

Figure 6.1: The system for pan and tilt rotations

Once the tilt movement is introduced, the last step could be the implementation of a visual servoing algorithm. With the introduction of a controller, the overall set-up would be complete and fully functional, ready for the challenges of real world scenarios.

Bibliography

[1] S. N. Sinha and M. Pollefeys, "Pan-tilt-zoom camera calibration and high-resolution mosaic generation," *Computer Vision and Image Understanding*, vol. 103, no. 3, pp. 170–183, 2006.

[2] S. Sinha, , S. N. Sinha, and M. Pollefeys, "Towards calibrating a pan-tilt-zoom camera network," in *In Workshop on Omnidirectional Vision and Camera Networks at ECCV*, 2004.

[3] J. Sanchez-Riera, J. Salvador, and J. R. Casas, "Indoor PTZ Camera Calibration with Concurrent PT Axes," in *4th International Conference on Computer Vision Theory and Applications (VISAPP '09)*, A. Ranchordas and H. Araújo, Eds., vol. 2. Lisbon, Portugal: INSTICC Press, 2009, pp. 10–15.

[4] J. Bano, A. Hostettler, A. Nicolau, C. Doignon, H. S. Wu, M. H. Huang, and L. a. Soler, *Advances in Visual Computing: 8th International Symposium, ISVC 2012, Rethymnon, Crete, Greece, July 16-18, 2012, Revised Selected Papers, Part I*, 1st ed., ser. Lecture Notes in Computer Science 7431. Springer-Verlag Berlin Heidelberg, 2012.

[5] C. Tomasi and J. Zhang, "How to rotate a camera," in *Proceedings of the 10th International Conference on Image Analysis and Processing*, ser. ICIAP '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 606–.

[6] J. Davis and X. Chen, "Calibrating pan-tilt cameras in widearea surveillance networks," in *In IEEE International Conference on Computer Vision*, 2003.

[7] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2011, pp. 3400–3407.

[8] R. Andrew, S. Johannes, and O. Edwin, "AprilCal: Assisted and repeatable camera calibration," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, November 2013.

[9] M. Faessler, E. Mueggler, K. Schwabe, and D. Scaramuzza, "A monocular pose estimation system based on infrared LEDs," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[10] L. Kneip, D. Scaramuzza, and R. Siegwart, "A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation," in *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, ser. CVPR '11. IEEE Computer Society, 2011, pp. 2969–2976.

[11] M. Moos, L. Teixeira, and M. Chli, "Tracking of multiple uavs for aerial manipulation," 2016.

[12] P. T. Furgale, T. D. Barfoot, and G. Sibley, "Continuous-time batch estimation using temporal basis functions," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 14-18 May 2012, pp. 2088–2095.

[13] P. Furgale, C. H. Tong, T. D. Barfoot, and G. Sibley, "Continuous-time batch trajectory estimation using temporal basis functions," *The International Journal of Robotics Research*, vol. 34, no. 14, pp. 1688–1710, 2015.

[14] J. Maye, H. Sommer, G. Agamennoni, R. Siegwart, and P. Furgale, "Online self-calibration for robotic systems," *The International Journal of Robotics Research*, vol. 35, no. 4, pp. 357–380, 2016.

[15] A. Tabb and K. Ahmad Yousef, "Parameterizations for reducing camera reprojection error for robot-world hand-eye calibration," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[16] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.

Appendix A

AprilTags: an introduction

Since AprilTags represent a fundamental part for the comprehension of this report, here a brief summary of what they are and how they work is presented. The text is based on [7].

A.1 Basic principles and ideas

AprilTags are a visual fiducial system that has been employed in a wide range of tasks, from camera calibration to augmented reality and robotics. For instance, the tags can be used as artificial fiducial features in order to create controllable experiments, simplifying the development of systems where the perception is not the main objective.

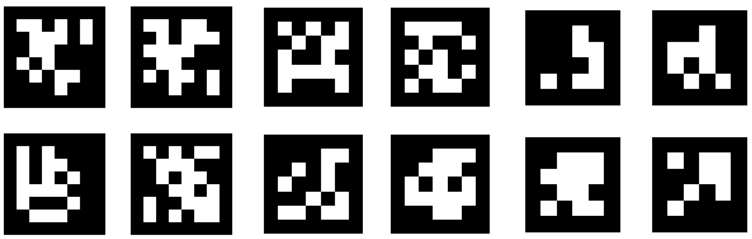

Figure A.1: Examples of AprilTags

AprilTags use a 2D bar code style, with the ultimate goal of performing the full 6 degrees of freedom localization of features from a single image (figure A.1). Differently from QR Codes, a visual fiducial tag has a small information payload and it can be automatically detected even when the resolution of the image is low or when it is rotated or tilted. QR Codes do not allow such flexibility, since they require a perfect alignment between the camera and the code. Plus, since the amount of information they contain is enormous with respect to AprilTags, the resolution of the picture should be fairly high.

A.2 The working principle

In order to detect the AprilTags, the system must be composed by two elements: the tag detector and the coding system.

The detector attempts to find four-sided regions that have a darker interior than the exterior. In order to facilitate this process, the tags have black and white borders. The first step is the detection of the quad lines from an image. Once they are found, the homography can be computed with the Direct Linear Transform (DLT) algorithm [16].
The final task of the detector is to read the bits from the payload field. This can be achieved by computing the tag relative coordinates of each bit field, transforming them into image coordinates using the homography, and finally thresholding the pixels.

Once the data payload is decoded from a quad, the coding system determines whether it is valid or not. The ultimate goals of the coding system is to maximize the number of tags that can be distinguished and to minimize the number of false detections.

This appendix has described briefly the working principle of AprilTags. Overall, they represent a fiducial system that has numerous advantages, such as reliability. Therefore, they are a robust method for localization and this characteristic make them fundamental for the robotics field.