
Federal University of Campina Grande
Center for Electrical Engineering and Informatics
Graduate Program in Electrical Engineering

Débora Nunes Pinto de Oliveira

# Heterogeneous and asynchronous optical tracking system for mobile vehicles surveillance

Campina Grande, Paraíba, Brazil

Débora Nunes Pinto de Oliveira

# Heterogeneous and asynchronous optical tracking system for mobile vehicles surveillance

*Master's Thesis submitted to the Coordination of the Electrical Engineering Graduate Program of the Federal University of Campina Grande - Campina Grande Campus, corresponding to the specialisation line in Information processing, in partial fulfilment of the requirements for the degree of Master of Science in Electrical Engineering.*

Antonio Marcus Nogueira Lima. Dr. – UFCG
Advisor
Marcos Ricardo Alcântara Morais, D.Sc. – UFCG
Advisor

Campina Grande, Paraíba, Brazil, August 3, 2022

O48h Oliveira, Débora Nunes Pinto de.

Heterogeneous and asynchronous optical tracking system for mobile vehicles surveillance / Débora Nunes Pinto de Oliveira. – Campina Grande, 2022.

91 f. : il. color.

Dissertação (Mestrado em Engenharia Elétrica) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2022.

"Orientação: Prof. Dr. Antonio Marcus Nogueira Lima, Prof. Dr. Marcos Ricardo Alcântara Morais".

Referências.

1. Processamento da Informação. 2. Sistema de Posicionamento Interno. 3. Odometria Visual. 4. Rastreamento Óptico – Assíncrono e Heterogêneo. 5. Captura de Movimento. I. Lima, Antonio Marcus Nogueira. II. Morais, Marcos Ricardo Alcântara. III. Título.

CDU 621.391:004.451.7:004.7(043)

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECÁRIA SEVERINA SUELI DA SILVA OLIVEIRA CRB-15/225

Débora Nunes Pinto de Oliveira

# Heterogeneous and asynchronous optical tracking system for mobile vehicles surveillance

*Master's Thesis submitted to the Coordination of the Electrical Engineering Graduate Program of the Federal University of Campina Grande - Campina Grande Campus, corresponding to the specialisation line in Information processing, in partial fulfilment of the requirements for the degree of Master of Science in Electrical Engineering.*

*Approved on 25th of July of 2022*

Antonio Marcus Nogueira Lima. Dr. – UFCG

Advisor

Marcos Ricardo Alcântara Morais, D.Sc. – UFCG

Advisor

José Sérgio da Rocha Neto, D.Sc.– UFCG

Member of Examining Board

Gutemberg Gonçalves dos Santos Júnior, Dr. – UFCG

Member of Examining Board

MINISTÉRIO DA EDUCAÇÃO
**UNIVERSIDADE FEDERAL DE CAMPINA GRANDE**
POS-GRADUACAO EM ENGENHARIA ELETRICA
Rua Aprigio Veloso, 882, - Bairro Universitario, Campina Grande/PB, CEP 58429-900

REGISTRO DE PRESENÇA E ASSINATURAS

1. ATA DA DEFESA PARA CONCESSÃO DO GRAU DE MESTRE EM ENGENHARIA ELÉTRICA, REALIZADA EM 25 DE JULHO DE 2022
**(Nº 742)**

CANDIDATA: **DÉBORA NUNES PINTO DE OLIVEIRA**. COMISSÃO EXAMINADORA: ANTONIO MARCUS NOGUEIRA LIMA, Dr., UFCG, MARCOS RICARDO DE ALCÂNTARA MORAIS, D.Sc., UFCG, Orientadores, GUTEMBERG GONÇALVES DOS SANTOS JÚNIOR, Dr., UFCG, JOSÉ SÉRGIO DA ROCHA NETO, D.Sc., UFCG. TÍTULO DA DISSERTAÇÃO: **Heterogeneous and asynchronous optical tracking system for mobile vehicles surveillance.** HORA DE INÍCIO: **14h00** – LOCAL: **Sala Virtual,** conforme o artigo 5º da PORTARIA SEI Nº 01/PRPG/UFCG/GPR, DE 09 DE MAIO DE 2022**).** Em sessão pública, após exposição de cerca de 45 minutos, o candidato foi arguido oralmente pelos membros da Comissão Examinadora, tendo demonstrado suficiência de conhecimento e capacidade de sistematização, no tema de sua dissertação, obtendo conceito APROVADO. Face a aprovação, declara o presidente da Comissão, achar-se o examinando, legalmente habilitado a receber o Grau de Mestre em Engenharia Elétrica, cabendo a Universidade Federal de Campina Grande, como de direito, providenciar a expedição do Diploma, a que o mesmo faz jus. Na forma regulamentar, foi lavrada a presente ata, que é assinada por mim, Leandro Ferreira de Lima, e os membros da Comissão Examinadora presentes. Campina Grande, 25 de Julho de 2022.

LEANDRO FERREIRA DE LIMA
Secretário

ANTONIO MARCUS NOGUEIRA LIMA, Dr., UFCG
Orientador

MARCOS RICARDO DE ALCÂNTARA MORAIS, D.Sc., UFCG
Orientador

GUTEMBERG GONÇALVES DOS SANTOS JÚNIOR, Dr., UFCG
Examinador Interno

JOSÉ SÉRGIO DA ROCHA NETO, D.Sc., UFCG
Examinador Interno

DÉBORA NUNES PINTO DE OLIVEIRA
Candidata

2 - APROVAÇÃO

2.1. Segue a presente Ata de Defesa de Dissertação de Mestrado da candidata **DÉBORA NUNES PINTO DE OLIVEIRA**, assinada eletronicamente pela Comissão Examinadora acima identificada.

2.2. No caso de examinadores externos que não possuam credenciamento de usuário externo ativo no SEI, para igual assinatura eletrônica, os examinadores internos signatários **certificam** que os examinadores externos acima identificados participaram da defesa da dissertação e tomaram conhecimento do teor deste documento.

Documento assinado eletronicamente por **LEANDRO FERREIRA DE LIMA**, **SECRETÁRIO (A)**, em 02/08/2022, às 09:22, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **JOSE SERGIO DA ROCHA NETO**, **PROFESSOR 3 GRAU**, em 02/08/2022, às 09:30, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **ANTONIO MARCUS NOGUEIRA LIMA**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 02/08/2022, às 13:57, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **GUTEMBERG GONCALVES DOS SANTOS JUNIOR**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 03/08/2022, às 08:54, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **MARCOS RICARDO ALCANTARA MORAIS**, **PROFESSOR 3 GRAU**, em 03/08/2022, às 09:11, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

A autenticidade deste documento pode ser conferida no site https://sei.ufcg.edu.br/autenticidade, informando o código verificador **2594399** e o código CRC **5DE94377**.

**Referência:** Processo nº 23096.050717/2022-08 SEI nº 2594399

# Dedication

To my family, who never ceased aiding me despite the fact that they couldn't understand a word of my engineering lingo.

# Thanks

First and foremost, I would like to express my gratitude to my advisers, Prof. Antonio Marcus and Prof. Marcos Morais, for their unwavering support, direction, and assistance in completing this work and all of the master course and side papers. Thank you to my family and friends for giving me the diversion to deal with the challenges of this endeavour. I am grateful to my coworkers Davi and Emanuel, who supported me daily in the lab. Furthermore, I would like to thank Prof. Gutemberg Gonçalves for supplying the computer and cameras used to get these results. Finally, I would like to thank CAPES for their financial aid and COPELE-DEE-UFCG for course administration, which allowed this study to be completed.

# Abstract

Given the need for low-cost ground truth in the laboratory with comparable performance to the commercially available motion capture systems, this study proposes off-the-shelf hardware and open-source software for a motion capture system. In the proposed scheme, cameras track passive markers based on their infrared reflection. Four V2 NoIR Raspberry cameras were applied with the goal of subcentimeter accuracy at 100 Hz. Algorithms for clock synchronization, quick contour extraction, and intrinsic camera calibration are addressed. Construction issues of non-uniform lighting or noisy reflections are also discussed. As a result of this study, a correction of the OpenCV extrinsics calibration library, a calibration toolbox for the generic lens model, and an innovative ordering algorithm for the markers were proposed. The system's performance achieved subcentimeter precision compared to industrial arm encoder data. Three drone flights are also included as examples. The code is already public, and further study will be in charge of fine-tuning the available configuration for multiple vehicles.

**Keywords**: indoor positioning system, visual odometry, optical tracking, motion capture, asynchronous, heterogeneous, RealSense, Raspberry Pi, infrared, extrinsics calibration, intrinsics calibration, subcentimetre accuracy.

# Resumo

Dada a necessidade de um rastreio de trajetória de baixo custo com desempenho comparável ao sistema de captura de movimento disponível comercialmente, este estudo propõe um conjunto de hardware de prateleira e softwares de código aberto para um sistema de captura de movimento. No esquema proposto, as câmeras rastreiam marcadores passivos a partir de sua reflexão à irradiação infravermelha. Quatro câmeras V2 NoIR Raspberry foram aplicadas para atingir precisão de subcentímetro em 100 Hz. Algoritmos para sincronização de relógio, extração rápida de contorno e calibração intrínseca da câmera são abordados em plataformas embarcadas. Questões de construção de iluminação não uniforme ou reflexos ruidosos também são discutidas. Como resultado deste estudo, foi proposta uma correção da biblioteca de calibração de lentes extrínsecas OpenCV, um algoritmo de calibração para o modelo genérico de lentes e um algoritmo inovador de ordenação para os marcadores. O desempenho do sistema alcançou uma precisão de subcentímetro em comparação com os dados do codificador de um braço industrial. Três vôos com drones também estão incluídos como exemplos. O código é público, e um estudo mais aprofundado será encarregado de afinar a configuração disponível para vários veículos.

**Keywords**: sistema de posicionamento interno, odometria visual, rastreamento óptico, captura de movimento, assíncrono, heterogêneo, RealSense, Raspberry Pi, infravermelho, calibração extrínseca, calibração intrínseca, precisão subcentimétrica.

# List of Figures

# List of Tables

# List of Abbreviations, Symbols, Initialisms, and Acronyms

## Abbreviations

IR: Infrared

PTS: Presentation Timestamp 46

SLAM: Simultaneous localisation and mapping 16

UWB: Ultra-wideband 16

## Initialisms

BCS: Body Coordinate System 23

BLE: Bluetooth Low Energy 15

CCS: Camera Coordinate System 23

COG: Centre of Gravity 22

DOF: Degrees of Freedom 29

EFCS: Earth Fixed Coordinate System 23

FPS: Frames per Second 21

IMU: Inertial Measurement Unit 15

IPS: Indoor Positioning System 15

KF: Kalman Filter 16

LIDAR: Light Detection and Ranging 16

MCS: Motion capture System 16

SVD: Singular Value Decompositon 30

# Acronyms

FOV: Field of view 21

# Summary

# Chapter 1

# Introduction

In this chapter, the requirement for a self-made motion capture system in the laboratory will be contextualised. Then, the objectives of this master thesis will be enumerated and further justified. The following section will go through the chapter structure and contents.

## 1.1 Motivation

There has been an increasing interest in robotics in academic and production line applications as the 4.0 industry thrives. Yet, due to weak, noisy or non-reliable GPS signals in indoor scenarios—such as shop floors or laboratories—to accurately detect a mobile robot's location is a complex task, directly affecting vehicles' control and path tracking algorithms. Localisation solutions are thus highly investigated and optimised.

Indoor positioning systems (IPS) may be built using either embedded sensors on the tracked vehicle or instrumentation of the environment. The most appropriate IPS technique is, however, highly dependent on the tracking system's purpose, the range and precision of the embedded sensors, and the infrastructure available at the workplace. For instance, when millimetre precision is required, it's unsuitable to apply conventional embedded Inertial Measurement Unit (IMU), since it predicts location through integration and is therefore prone to cumulative error[1]. WLAN and Bluetooth Low Energy (BLE) fingerprinting are no exception, with a maximum accuracy of 50 cm[2].

When centimetre precision is required, one must consider whether it is possible to modify

the vehicle's hardware and install a better-suited sensor. For instance, Light Detection and Ranging (LIDAR) sensors are more reliable than IMUs[3], particularly when their data is fused[4] using Kalman Filter (KF). This approach is however limited to terrestrial vehicles[5] because LIDARs can only collect two-dimensional locations. These sensors also weigh more than 150g, which is significant compared to vehicles like drones that weigh less than 100g.

Lighter sensors, such as ultra-wideband sensor (UWB) radars[6–10] or even RFID and NFC tags[11], are preferable for ensuring widespread localisation solutions. Nonetheless, these passive tags require the label to move continuously around the reader for measurement, making them unsuitable for particular tasks, such as pick and place, where the vehicle must remain stationary. The scalability to larger scenarios may also be a concern, given the difficulty in determining the best mix of beacons or broadcast settings[12].

When using an off-the-shelf closed platform, a vision-based IPS is thus a more practical choice. Simultaneous localisation and mapping (SLAM) solutions[13–15] that reconstruct 3D maps of indoor scenes are suitable for vehicles with onboard cameras. However, this technique is limited to circumstances in which distinct feature points are visible between frames[16] and is therefore inappropriate for indoor scenes with minimal roughness and surfaces devoid of trackable features[17, 18].

In spite of this restriction, synthetic trackable features may be added to the scene objects under controllable settings and enable motion capture systems (MCS) as a positioning technique. In this approach, artificial markers—such as infrared LEDs[19, 20] or reflective tags[21]—are attached to the vehicle and detected by cameras in a paired arrangement. The position of the markers in 3D space may then be recreated using matching and triangulation algorithms[22]. The calibration of this system is rigid, but it allows the user to relocate the cameras to the ideal capturing configuration to observe the scene. Commercial MCS, such as OptiTrack[23] and VICON[24–26], achieve millimetre accuracy, but are yet expensive (5 to 160 thousand dollars), since they require the purchase of an entire package (e.g. cameras, calibration wand, active or passive markers, lenses, rendering software, external hardware trigger) incompatible with any open-source device.

Accordingly, efforts are being made to develop robust optical tracking using commercially available off-the-shelf hardware[27, 28]. This work focuses on the mock-up of an MCS using heterogeneous and asynchronous off-the-shelf capture platforms in computationally

restricted settings. We also employ a networked trigger system to guarantee no cable arrangement limitations. A prototype is constructed with store-bought hardware, and the calibration, image segmentation, and performance are validated. This setup will be available at the laboratory for further vehicle control and navigation algorithms experiments.

## 1.2 Objective

The primary purpose of this master thesis is to propose the hardware and software bundle for an optical IPS. Although it appears straightforward at first glance, the concatenation of solutions (e.g. synchronised shutter timing, contour extraction, picture segmentation, and camera calibration) is a technological problem that comes from different combinations of algorithm settings and setup circumstances (e.g. illumination, reflective surface construction). In addition, given the scarcity of practical information in the open technical literature, this study describes how to avoid trial loops in order to obtain a calibration setup with subcentimetre precision. The following research will be responsible for fine-tuning the proposed system to the dynamic rate of various robots and increasing the tracking for numerous vehicles.

### 1.2.1 Specific objectives

To attain effectiveness and robustness, the suggested methodology seeks to:

- Elect appropriate camera's models and suitable framework for image processing and camera's intrinsic calibration;
- Examine the limits of triangulation algorithms and how to improve camera's extrinsic parameters calibration;
- Investigate the suitability and implementation of centralised and decentralised architectures for the proposed system;
- Implement global software trigger to compose a network-activated visual IPS;
- Test multiple artificial marker sizes and layouts to identify the relation between accurateness in 3D reconstruction and the cameras' disposition, focal length and resolution;

- Validate and assess the system's performance by computing the distance error of a test trajectory to a ground truth.

## 1.3 Structure

This document is structured in the following sections: chapter 2 describes a bibliography review of MCS constraints and mathematical foundations, chapter 3 presents the methodology of the planned experiments, and chapter 4 describes experimental results and discussions. Lastly, chapter 5 resumes the conclusion of this project.

# Chapter 2

# Literature review

The literature review is described in this chapter. The limits and assumptions of motion capture systems are outlined first. Following that, the workflow's notation and mathematical concepts are given.

## 2.1 Constraints of optical tracking systems

Unlike MCS for film production, where the markers reconstruction may be offline, IPS must follow the dynamic rates of the vehicle's control loop. Mehling[27] defines the main constraints of a visual IPS as latency, jitter, accuracy, frame rate, range and illumination.

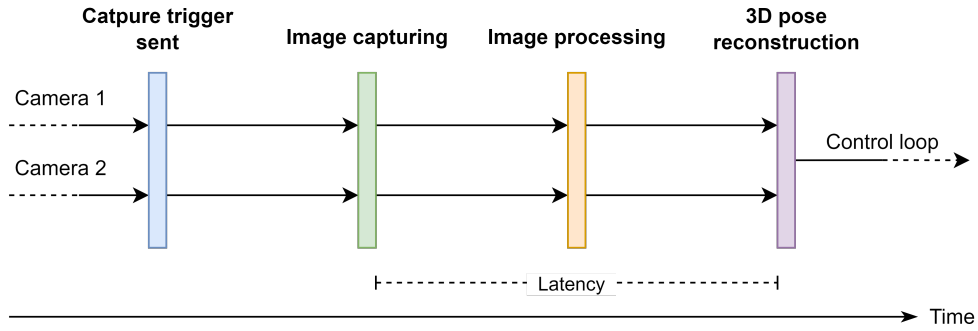
### Latency

The time between capture and transfer to the feedback loop is determined by the controller's ability to forecast the vehicle's pose. The route tracking of the robots will be hampered if the delay is too significant. To compensate for long latencies, OptiTrack systems predict the future position of the markers (that for the Flex 3 camera is 10ms). In asynchronous systems, the latency varies between cameras, as depicted in Figure 2.1.

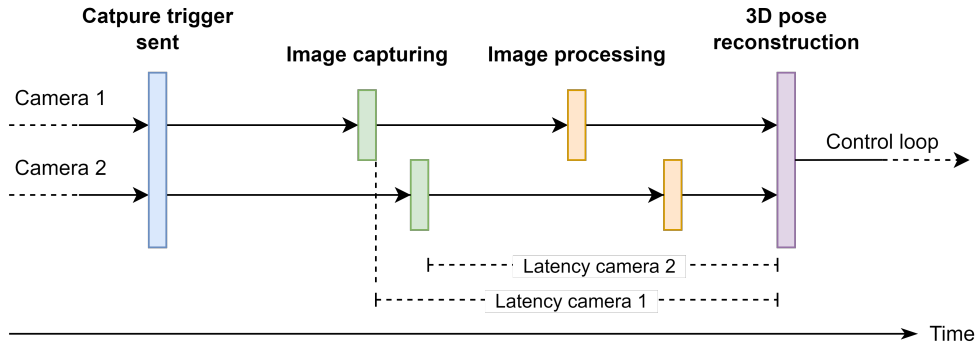
### Jitter

Jitter is the noise in tracking data that occurs when all objects in a scene remain still, as depicted in Figure 2.2. The camera's exposure and white balance gains, and the scene

**Figure 2.1:** Time diagram from the capture trigger to the reconstruction of the 3D pose in a synchronous (a) and asynchronous (b) motion caption system.




**(a)** Synchronous capture system




**(b)** Assynchronous capture system

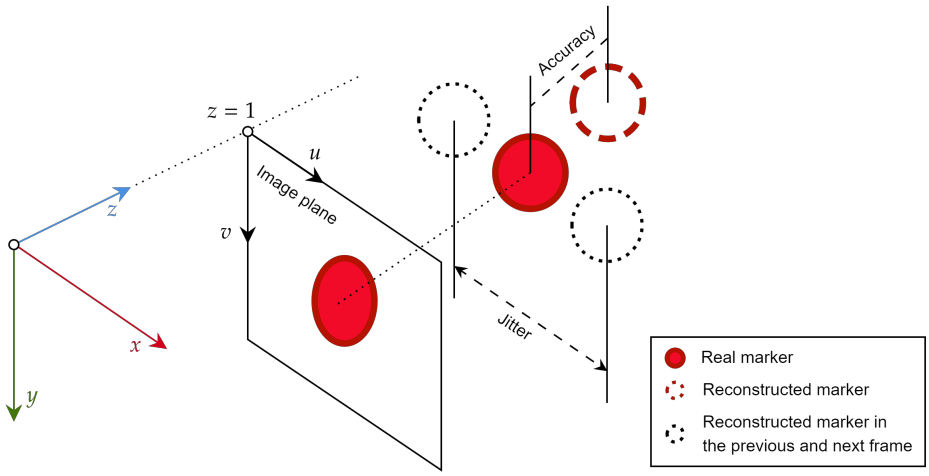
Source: Prepared by the author.

lighting must remain consistent to minimise this disturbance. Fixing the exposure that provides the most stable 3D reconstruction is the key to achieving minimal jitter. For the OptiTrack system, Teather, Pavlovych and Stuerzlinger[29] reported a 0.3 mm jitter to an optical mouse tracker.

### Accuracy

The application's purpose mainly determines the accuracy required of an MCS. For example, subject pose authenticity is not as crucial as during rescue efforts in underground chambers. According to the most recent survey of non-commercial infrared-based IPS[30], the best-achieved accuracy was about 1 cm. The lowest-cost OptiTrack system (four Flex 3 cameras) reports an accuracy of 0.5 mm for a 14 mm marker on a 9 m×9 m capture area using six cameras. An example of accuracy in the 3D reconstruction of a circular marker is depicted in Figure 2.2.

**Figure 2.2:** Illustration of accuracy and jitter for an still circular marker.




Source: Prepared by the author.

### Frame rate

The update rate of the subject's position must be compatible with the vehicle's maximum velocity. For instance, Flex 3 captures 100 frames per second (FPS) with 640×480 pixels and MJPEG format. When accuracy is more important than update frequency, however, the trade-off for high-resolution images leads FPS ratio to decay to roughly 30Hz[31]. The frame rate is constrained by the data processing time and communication bandwidth.

### Range

The size of the interaction volume also relies on the purpose of the application. However, it is also dependent on the focal length of the infrared (IR) emitters, the field of view (FOV) of the cameras, and the reach of the communication protocol between triggered platforms. The range should be sufficient without impacting the camera distribution, the size or arrangement of the markers, or the system's accuracy. For instance, the Flex 3 camera can detect passive markers up to 6 metres away.

### Illumination

The ability to add a new sensor to the vehicle determines whether to use passive or active IR markers. Given the goal of developing a generic MCS, this study treats the vehicle as a closed platform and chooses passive reflecting tags to designate the rigid body. As a result,

IR emitters must light the scene, and the correct balance between the number of emitters and camera positioning must be found to eliminate jitter. It is also a good idea to utilise IR-pass filters to avoid outliers. The Flex 3, for example, is outfitted with a 26 LED 850 nm IR ring light with an 800 nm long-pass filter.

## 2.2 Assumptions

You et al.[9] assumes following conditions for a project of a MCS:

**Assumption 1:** Without loss of generality, the 1st camera is regarded as the reference camera.

**Assumption 2:** At least two cameras share a common overlap, because a marker can only be reconstructed when it is visible from a pair of cameras.

**Assumption 3:** The empirically established cameras' number and placement satisfy the visibility demands. However, the optimal position remains to be searched in the future.

**Assumption 4:** The markers layout must be asymmetric, and the centre of the markers must also be as close as possible to the centre of gravity (COG) of the vehicle.

**Assumption 5:** The camera intrinsics parameters (i.e. distortion coefficients, focal length and principal point) remain unaltered during calibration.

## 2.3 Notation and definitions

Basic notions of perspective and multi-view geometry are essential to comprehend the triangulation process. Similar to Deng et al.[28], $\mathbb{R}^{m\times n}$ as a real matrix with $m$ lines and $n$ columns, and $\mathbb{R}^n$ as a real $n$-dimensional column-vector. Note $\boldsymbol{A}^T$ and $\boldsymbol{A}^{-1}$ as the transpose and inverse matrix of $\boldsymbol{A}$. The symbol $\tilde{\boldsymbol{x}}$ indicates the homogeneous coordinate of $\boldsymbol{x}$ by adding a scalar as the last element; i.e. expanding one dimension, so $\dim(\tilde{\boldsymbol{x}}) = \dim(\boldsymbol{x}) + 1$[22]. Uppercase vectors $\boldsymbol{X}$ are in 3D-space, whilst lowercase $\boldsymbol{x}$ are on a 2D plane. The cross product $\boldsymbol{a} \times \boldsymbol{b}$ is noted as $[\boldsymbol{a}]_\times \boldsymbol{b}$, for $\boldsymbol{a}^{1\times 3}$ skew-symmetric.

Consider the coordinate systems illustrated in Figure 2.3: the earth fixed coordinate system (EFCS), the body coordinate system (BCS) of the mobile vehicle, and each cameras coordinate system (CCS). The main goal of the calibration is to determine the transformation between each CCS and the EFCS. Then, with the image data from each CCS, it's possible to estimate the pose between the markers attached to the robot's body and the ECFC. These markers' position concerning the BCS origin is known.

To calibrate, the system consisting of $M \in \mathbb{Z}_+$ cameras captures three markers $(A, B, C)$ fixed on a stick, as in Figure 2.4. Each markers' 3D pose is written as $\boldsymbol{X}_j^k = [X_j^k, Y_j^k, Z_j^k]^T$, for $j$ the image frame index and $k = A, B$ or $C$. The projection of these markers in the $j$th frame on the image plane of the $i$th camera is denoted $\boldsymbol{x}_j^{k,i} = [x_j^{k,i}, y_j^{k,i}]^T$. Note on Figure 2.3 that the origin $o_{c_i}$ of each camera is located over the camera's optical centre whilst the $o_{c_i} z_{c_i}$ axis is aligned with its optical axis.

### 2.3.1 Notions on perspective geometry

A camera is usually modelled by the pinhole camera model[32], for which the relationship between a 3D point $\tilde{\boldsymbol{X}}$ and its image projection $\tilde{\boldsymbol{x}}$ is given by:

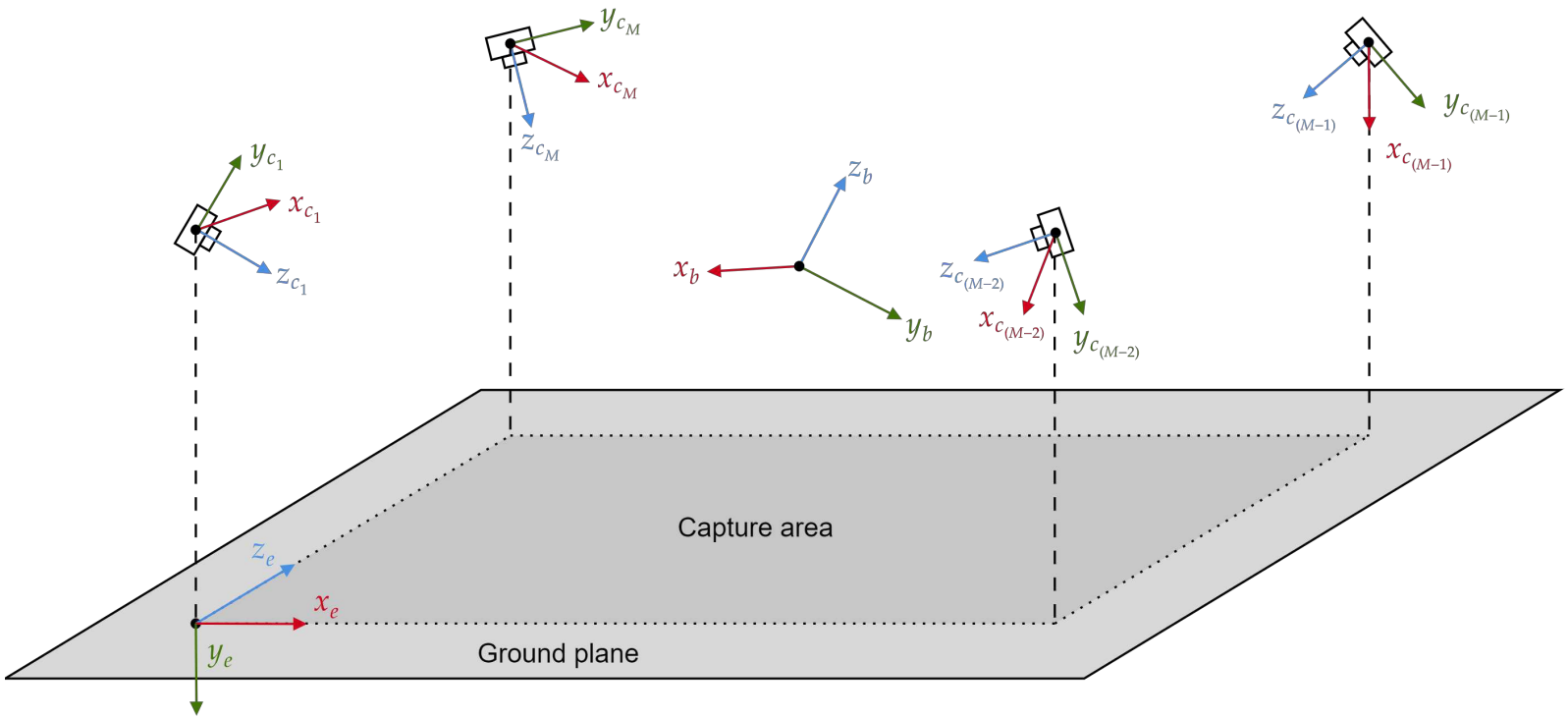
$$\tilde{\boldsymbol{x}} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \boldsymbol{K}[\boldsymbol{I}^{3\times3} \,|\, \boldsymbol{0}^3]\tilde{\boldsymbol{X}} = \begin{bmatrix} \alpha & \gamma & u_0 & 0 \\ 0 & \beta & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.1}$$

where $\boldsymbol{K}$ is the camera intrinsics matrix, $(\alpha, \beta)$ represents the focal length in pixels in the horizontal and vertical direction, $(u_0, v_0)$ is the principal point coordinate, and $\gamma$ is the skew factor. The projection of $\boldsymbol{X}$ to $\boldsymbol{x}$ is depicted in Figure 2.5. To dislocate $\tilde{\boldsymbol{x}}$ from the CCS to the EFCS, Equation 2.1 is rewritten to:

$$\lambda\tilde{\boldsymbol{x}} = \boldsymbol{K}[\boldsymbol{R} \,|\, \boldsymbol{t}]\tilde{\boldsymbol{X}} \Rightarrow \tilde{\boldsymbol{x}} = \boldsymbol{P}\tilde{\boldsymbol{X}} \tag{2.2}$$

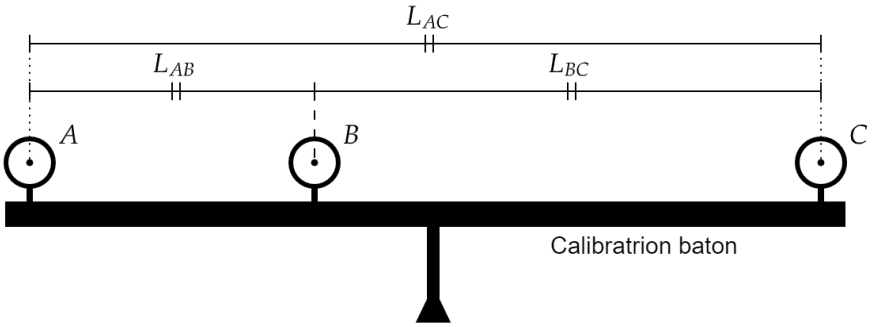
where $(\boldsymbol{R}^{3\times3}, \boldsymbol{t}^3)$ is the extrinsic parameters, i.e. the rotation and translation of the CCS to the EFCS, $\lambda$ a scale factor, and $\boldsymbol{P}^{3\times4}$ is the projection matrix. For actual non-pinhole lenses, the assumption of linearity between the world point, optical centre and image point

**Figure 2.3:** Illustration of the EFCS, BCS and CFF coordinate system.



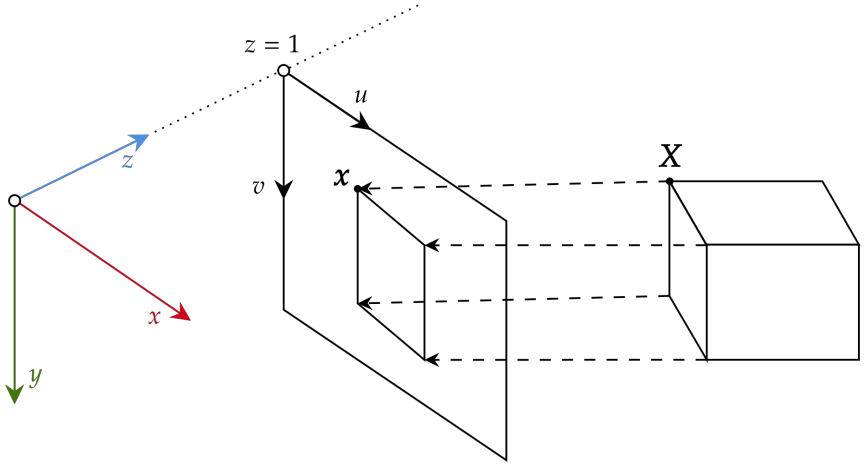

Source: Adapted from Deng et al.[28].

**Figure 2.4:** Calibration 1D wand prototype for three markers.




Source: Prepared by the author.

**Figure 2.5:** Projection of the side of a box to an image plane.

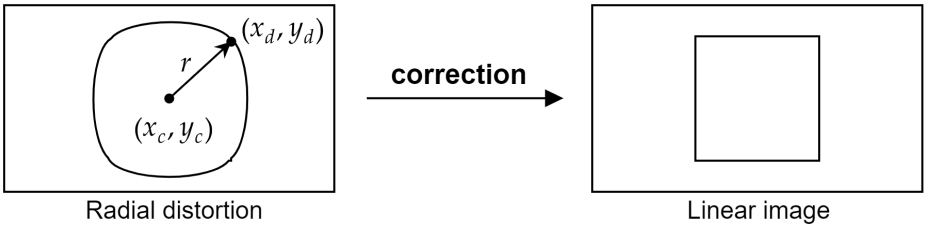



Source: Prepared by the author.

does not hold. The correction for this lens distortion (see Figure 2.6) is to adopt a distortion factor $L(r) = (1 + k_1 r^2 + k_2 r^4 + k_3 r^6)/(1 + k_4 r^2 + k_5 r^4 + k_6 r^6)$, for $r$ the radial distance from the centre of radial distortion $(x_c, y_c)^T$ and $(x_d, y_d)$ the distorted coordinates:

$$\begin{cases} x = x_c + L(r)(x_d - x_c) \\ y = y_c + L(r)(y_d - y_c) \end{cases} \tag{2.3}$$

**Figure 2.6:** The image of a square capture with lens distortion (left) and correct to a linear model (right).




Source: Adapted from Hartley and Zisserman[22].

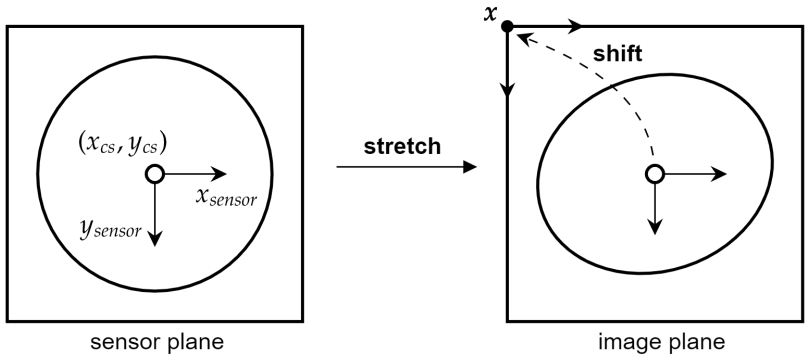
The intrinsic parameters must also count for sensor-to-lens misalignment. The following equation maps an image point into its corresponding 3D vector trough an affine transformation, as displayed in Figure 2.7:

$$\tilde{\boldsymbol{x}} = \begin{bmatrix} c & d \\ e & 1 \end{bmatrix} \begin{bmatrix} x_{\text{sensor}} \\ y_{\text{sensor}} \end{bmatrix} + \begin{bmatrix} x_{cs} \\ y_{cs} \end{bmatrix} = \lambda \begin{bmatrix} x_{\text{sensor}} \\ y_{\text{sensor}} \\ f(\rho) \end{bmatrix} = \boldsymbol{P}\tilde{\boldsymbol{X}} \tag{2.4}$$

where $(c, d, e)$ compose the stretch matrix[33], which is represented as the polynomial transformation $f(\rho) = a_o\rho + a_2\rho^2 + ... + a_n\rho^n$[34], for $\rho$ the radial distance from the centre of the sensor $(x_{cs}, y_{cs})^T$ and $(x_{\text{sensor}}, y_{\text{sensor}})^T$ the corresponding image point on the sensor plane. The unknown intrinsic and distortion parameters are calibrated by an analytical solution followed by an non-linear least squares minimisation. Further details about this procedure can be found in Zhang[32] and Bouguet[35].

In this work, however, the generic camera model proposed by Kannala and Brandt[36] is used. In this model, the lens is a spherical dome parallel to the image plane, as shown in

**Figure 2.7:** Sensor plane in metric coordinates (left) and camera image plane expressed in pixel coordinates (right) when tangential distortion is non negligible.




Source: Adapted from Zhang[32].

Figure 2.8. The 2D coordinate of a pixel in the image can be found as:

$$\tilde{\boldsymbol{x}} = \begin{bmatrix} \alpha & 0 & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tilde{\boldsymbol{x}}_{\boldsymbol{d}} \tag{2.5}$$

The mapping of the incoming rays to the distorted image coordinates $\tilde{\boldsymbol{x}}_{\boldsymbol{d}}$ is:

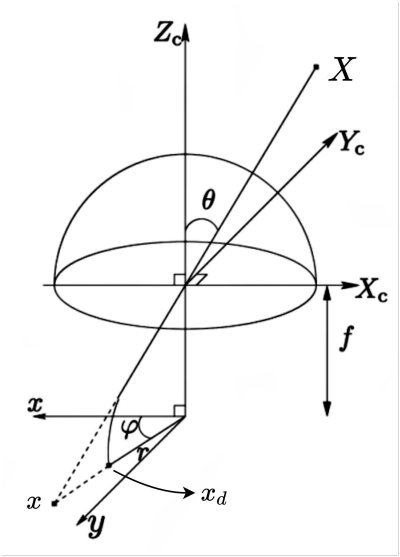
$$\tilde{\boldsymbol{x}}_{\boldsymbol{d}} = r(\theta)\boldsymbol{u}_r(\varphi) \tag{2.6}$$

for $r(\theta) = \theta + k_1\theta^3 + k_2\theta^5 + k_3\theta^5 + k_4\theta^9$ and $\boldsymbol{u}_r(\varphi)$ the unit vector in the direction of $r(\theta)$.

The odd powers are dropped without loss of generality, and the coefficients $k_i$ are such that $r(\theta)$ is monotonically increasing on the interval $[0, \theta_{\max}]$, for $\theta_{\max}$ the FOV. The calibration of $k_i$ involves planar homography estimation and external parameters decomposition, which will be shown in the following section. Further details about this procedure can be found in Kannala and Brandt[36] and Deng et al.[21]. The tangential distortion was ignored.

Without redoing the capture of the calibration board, the intrinsic matrix may be recomputed to a new resolution. If the entire picture is resized, the focal length must only be rescaled to the ratio between the new and the old image. However, if the image is cropped, the principal point must be recentered. The order of these steps must be respected while both rescaling and cropping. For example, if the original picture was redimensioned from a previous height $H_p$ to $H$, and then cropped $U$ pixels in the horizontal direction and $V$

**Figure 2.8:** Generic camera model considering a spherical lens distant $f$ millimetres from the projected image plane. The image of $X$ is $x$ whereas it would be $x^{'}$ by a pinhole camera.




Source: Adapted from Kannala and Brandt[36].

pixels in the vertical direction on both sides of the image, the new principal point $(u_{0_n}, v_{0_n})$ would be:

$$\begin{cases} u_{0_n} = \frac{H}{H_p} u_0 - 0.5U \\ v_{0_n} = \frac{H}{H_p} v_0 - 0.5V \end{cases} \tag{2.7}$$

### 2.3.2 Notions on epipolar geometry

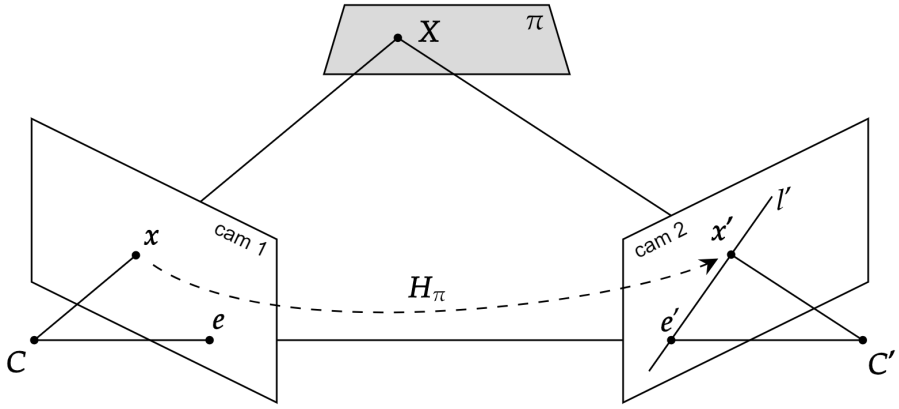
Suppose a point $\boldsymbol{X}$ in a 3D-space is imaged in two views: $\boldsymbol{x}$ and $\boldsymbol{x}^{'}$ for the first and second cameras. As shown in Figure 2.9, the centre of the cameras $\boldsymbol{C}$ and $\boldsymbol{C}^{'}$ are coplanar at $\pi$ and thus are also the rays projected from $\boldsymbol{X}$ to $\boldsymbol{x}$ and $\boldsymbol{x}^{'}$. The epipoles $\boldsymbol{e}$ and $\boldsymbol{e}^{'}$ are the intersection of the line joining $\boldsymbol{C}$ and $\boldsymbol{C}^{'}$ with the respective image planes. The epipolar line $l^{'}$ is the image of the ray from $\boldsymbol{X}$ to $\boldsymbol{x}$ back-projected on the second image plane.

For a point $\tilde{\boldsymbol{x}}$ in one image, there is a corresponding epipolar line $l^{'}$ in the other image plane, on which any matching point $\tilde{\boldsymbol{x}}^{'}$ must lie. Hence, there is a 2D homography $\boldsymbol{H}_\pi$ mapping such that $\tilde{\boldsymbol{x}}^{'} = \boldsymbol{H}_\pi \tilde{\boldsymbol{x}}$ and $[\tilde{\boldsymbol{x}}^{'}]^T \cdot l^{'} = 0$, for $\cdot$ the inner product. Given that both $\tilde{\boldsymbol{e}}^{'}$ and $\tilde{\boldsymbol{x}}^{'}$ pass through $l^{'}$:

$$l^{'} = [\tilde{\boldsymbol{e}}^{'}]_\times \tilde{\boldsymbol{x}}^{'} = [\boldsymbol{e}^{'}]_\times \boldsymbol{H}_\pi \tilde{\boldsymbol{x}} = \boldsymbol{F} \tilde{\boldsymbol{x}} \tag{2.8}$$

for $\boldsymbol{F}^{3\times3} = [\tilde{\boldsymbol{e}}^{'}]_\times \boldsymbol{H}_\pi$ the fundamental matrix.

**Figure 2.9:** Epipolar multi-view geometry relation between two cameras' image plane.




Source: Adapted from Hartley and Zisserman[22].

The correspondence between $\tilde{\boldsymbol{x}}^{'}$ and $\tilde{\boldsymbol{x}}$ is:

$$[\tilde{\boldsymbol{x}}^{'}]^T \cdot l^{'} = [\tilde{\boldsymbol{x}}^{'}]^T \boldsymbol{F} \tilde{\boldsymbol{x}} = 0 \tag{2.9}$$

Recalling that the last element of $\boldsymbol{x}$ and $\boldsymbol{x}^{'}$ is unitary, Equation 2.9 can be rewritten as $\boldsymbol{A}\boldsymbol{f} = 0$, where $\boldsymbol{f}^9$ is a vector made up of the entries of $\boldsymbol{F}$ in row-major order and $\boldsymbol{A}^{N\times 9}$, a matrix with rank 8. $N$ is the number of point matches. Therefore, the solution for $\boldsymbol{F}$ is unique up to scale and can be found by linear methods (i.e. generator of the right null-space of $\boldsymbol{A}$). In the case of noisy data, the rank of $\boldsymbol{A}$ maybe 9, and thus $\boldsymbol{F}$ is a least-squares solution (i.e. singular vector corresponding to the smallest singular value of $\boldsymbol{A}$)[22].

It is worth noting that $\boldsymbol{F}$ has only seven independent values (i.e. degrees of freedom, DOF). When counting the unique equations from $\boldsymbol{A}$, there are eight DOF, but one is subtracted from the restriction $\det \boldsymbol{F} = 0$. Another way of computing $\boldsymbol{F}$ DOF is to sum two DOF for the epipole $\boldsymbol{e}$, two DOF for the epipole $\boldsymbol{e}^{'}$, and three DOF for the homography $\boldsymbol{H}_\pi$ which maps a line through $\boldsymbol{e}$ to a line through $\boldsymbol{e}^{'}$[22].

Due to an projection ambiguity, one may not recover $\boldsymbol{R}$ and $\boldsymbol{t}$ from the fundamental matrix. Suppose that $\boldsymbol{F}$ corresponds to two different pairs $(\boldsymbol{P}, \boldsymbol{P}^{'})$ and $(\hat{\boldsymbol{P}}, \hat{\boldsymbol{P}}^{'})$, with $\hat{\boldsymbol{P}} = \boldsymbol{P}\boldsymbol{H}$ and $\hat{\boldsymbol{P}}^{'} = \boldsymbol{P}^{'}\boldsymbol{H}$, for $\boldsymbol{H}^{4\times 4}$ a projective 3D transformation. Note that $\lambda \boldsymbol{x}^{'} = \boldsymbol{P}^{'}\boldsymbol{X} = (\boldsymbol{P}^{'}\boldsymbol{H})(\boldsymbol{H}^{-1}\boldsymbol{X}) = \hat{\boldsymbol{P}}^{'}\hat{\boldsymbol{X}}$, for $\lambda$ a scalar. Therefore, if $\boldsymbol{x}$ and $\boldsymbol{x}^{'}$ are the matched points to the cameras $(\boldsymbol{P}, \boldsymbol{P}^{'})$ corresponding to a 3D point $\boldsymbol{X}$, they are also matched points between the pair $(\hat{\boldsymbol{P}}, \hat{\boldsymbol{P}}^{'})$ to the point $\hat{\boldsymbol{X}}$. The solution of Equation 2.9 is not unique.

The fundamental matrix can also be written as a function of the projection matrices $\boldsymbol{P} = \boldsymbol{K}[\boldsymbol{I}^{3\times3} \,|\, \boldsymbol{0}^3]$ and $\boldsymbol{P}' = \boldsymbol{K}'[\boldsymbol{R} \,|\, \boldsymbol{t}]$. Admit $\boldsymbol{P}^+ = \left[\boldsymbol{K}^{-1} \,|\, (\boldsymbol{0}^3)^T\right]^T$ as the pseudo-inverse of $\boldsymbol{P}$, thus $\boldsymbol{H}_\pi = \boldsymbol{P}'\boldsymbol{P}^+$. The epipole $\tilde{\boldsymbol{e}}'$ is written as $\tilde{\boldsymbol{e}}' = \boldsymbol{P}'\tilde{\boldsymbol{C}}$, where $\tilde{\boldsymbol{C}} = [0, 0, 0, 1]^T$ is the 1st camera position in the EFCS. Hence, Equation 2.8 can be revised to:

$$l' = [\tilde{\boldsymbol{e}}']_\times \boldsymbol{H}_\pi \tilde{\boldsymbol{x}} = [\boldsymbol{P}'\tilde{\boldsymbol{C}}]_\times \boldsymbol{P}'\boldsymbol{P}^+\tilde{\boldsymbol{x}} = [\boldsymbol{K}'\boldsymbol{t}]_\times \boldsymbol{K}'\boldsymbol{R}\boldsymbol{K}^{-1}\tilde{\boldsymbol{x}} \tag{2.10}$$

Hartley and Zisserman[22] defines that $[\boldsymbol{M}\boldsymbol{p}]_\times \boldsymbol{M} = \boldsymbol{M}^{-T}[\boldsymbol{p}]_\times$, where $\boldsymbol{M}$ a non-singular invertible matrix and $\boldsymbol{p}$, a skew-symmetric matrix. Comparing Equation 2.10 to Equation 2.8, one can infer that:

$$\boldsymbol{F} = [\boldsymbol{K}'\boldsymbol{t}]_\times \boldsymbol{K}'\boldsymbol{R}\boldsymbol{K}^{-1} = \boldsymbol{K}'^{-T}[\boldsymbol{t}]_\times \boldsymbol{R}\boldsymbol{K}^{-1} = \boldsymbol{K}'^{-T}\boldsymbol{E}\boldsymbol{K}^{-1} \tag{2.11}$$

where $\boldsymbol{E}^{3\times3} = [\boldsymbol{t}]_\times \boldsymbol{R} = \boldsymbol{K}'^T\boldsymbol{F}\boldsymbol{K}$ is the essential matrix. Likewise to the fundamental matrix, the $\boldsymbol{E}$ has rank 2. Nevertheless, matrix $\boldsymbol{E}$ has only five DOF. It would initially be 7 DOF as the fundamental matrix, but one DOF is subtracted for each normalisation by the intrinsic matrix $\boldsymbol{K}$ and $\boldsymbol{K}'$.

Since both $\boldsymbol{t}$ and $\boldsymbol{R}$ have three DOF, there is an overall scale ambiguity $\lambda$. One can, however, retrieve $\boldsymbol{P}'$ from $\boldsymbol{E}$ using singular value decomposition (SVD)[21, 22]. Given an square matrix $\boldsymbol{M}$, SVD can be used to factor $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T$, where $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal matrices and $\boldsymbol{D}$ is a diagonal matrix with non-negative values. This decomposition is done in a way that the columns of $\boldsymbol{V}$ represent the singular values of $\boldsymbol{M}$ in descending order[22].

It is simple to decompose $\boldsymbol{E}$ as $[\boldsymbol{t}]_\times \boldsymbol{R} = \boldsymbol{S}\boldsymbol{R}$, for $\boldsymbol{S}$ a skew symmetric matrix. Hartley and Zisserman[22] defines that the skew symmetric matrix $\boldsymbol{S}$ may be written as:

$$\boldsymbol{S} = \boldsymbol{U}\mathrm{diag}(1, 1, 0)\boldsymbol{W}\boldsymbol{U}^T \tag{2.12}$$

for $\boldsymbol{U}$ and $\boldsymbol{W}$ orthogonal matrices, and $\mathrm{diag}(1, 1, 0)\boldsymbol{W}$ an skew symmetric matrix. So let $\boldsymbol{E} = \boldsymbol{U}\mathrm{diag}(1, 1, 0)(\boldsymbol{W}\boldsymbol{U}^T\boldsymbol{R})$, or $\boldsymbol{E} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T$. Due to the unitary elements in the diagonal of $\boldsymbol{D}$, this is an SVD of two equal singular values. Therefore, the solution of the SVD is not unique and given by:

$$\boldsymbol{E} = \boldsymbol{U}\text{diag}(\boldsymbol{R}^{2\times2}, 1)\text{diag}(1, 1, 0)(\boldsymbol{R}^{2\times2}, 1)\boldsymbol{V}^T \tag{2.13}$$

for any $2 \times 2$ rotation matrix $\boldsymbol{R}$[22]. There are thus four possible solutions for $\boldsymbol{P}' = [\boldsymbol{R} \,|\, \boldsymbol{t}]$[22]:

$$\boldsymbol{P}' = \begin{cases} [\boldsymbol{U}\boldsymbol{W}\boldsymbol{V}^T \,|\, \boldsymbol{U}|] \\ [\boldsymbol{U}\boldsymbol{W}\boldsymbol{V}^T \,|\, -\boldsymbol{U}|] \\ [\boldsymbol{U}\boldsymbol{W}^T\boldsymbol{V}^T \,|\, \boldsymbol{U}|] \\ [\boldsymbol{U}\boldsymbol{W}^T\boldsymbol{V}^T \,|\, -\boldsymbol{U}|] \end{cases}, \text{ for } \boldsymbol{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.14}$$

where $\boldsymbol{U}|$ is the last column of $\boldsymbol{U}$. The correct result is chosen testing if $\boldsymbol{X}$ is reconstructed in front of both cameras ($Z > 0$). This check is not performed by the `OpenCV` library and must consequently be manually implemented. The equations $\lambda\boldsymbol{x} = \boldsymbol{P}\boldsymbol{X}$ and $\lambda'\tilde{\boldsymbol{x}}' = \boldsymbol{P}'\tilde{\boldsymbol{X}}$ can be combined to into a form $\boldsymbol{A}\tilde{\boldsymbol{X}} = \boldsymbol{b}$ by a cross product:

$$\tilde{\boldsymbol{x}} \times \boldsymbol{P}\tilde{\boldsymbol{X}} = \begin{bmatrix} x\boldsymbol{p}^{3T} - \boldsymbol{p}^{1T} \\ y\boldsymbol{p}^{3T} - \boldsymbol{p}^{2T} \\ x\boldsymbol{p}^{2T} - y\boldsymbol{p}^{1T} \end{bmatrix} \tilde{\boldsymbol{X}} \tag{2.15}$$

for $\boldsymbol{p}^{iT}$ the ith row of $\boldsymbol{P}$. These equations are linear in $\boldsymbol{X}$, but only two are linearly independent. Hence, for two views, Equation 2.15 is written as:

$$\begin{bmatrix} x'\boldsymbol{p}'^{3T} - \boldsymbol{p}'^{1T} \\ y'\boldsymbol{p}'^{3T} - \boldsymbol{p}'^{2T} \\ x'\boldsymbol{p}'^{3T} - \boldsymbol{p}'^{1T} \\ y'\boldsymbol{p}'^{3T} - \boldsymbol{p}'^{2T} \end{bmatrix} \tilde{\boldsymbol{X}} = \boldsymbol{A}\tilde{\boldsymbol{X}} = 0 \tag{2.16}$$

for $A \in \mathbb{R}^{N\times4}$. One may reconstruct $\boldsymbol{A}\boldsymbol{X} = \boldsymbol{b}$ using linear least-squares:

$$\boldsymbol{X} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{b} \tag{2.17}$$

As $N > \text{rank}(\boldsymbol{A}) = 3$, the solution is not unique, and $\boldsymbol{A}^T\boldsymbol{A}$ is not invertible. Thus, the usual choice is to minimise the Euclidian norm, i.e. the smallest singular value of $\boldsymbol{A}$.

$\boldsymbol{X}$ is the triangulation result (or 3D reconstruction) of the 2D points $\boldsymbol{x}$ and $\boldsymbol{x}'$.The scale ambiguity $\hat{\boldsymbol{t}} = \lambda \boldsymbol{t}$ is solved from the known distances between the markers on the wand:

$$\lambda = \frac{1}{3N} \sum_{j=1}^{N} \left[ \frac{L_{AC}}{\|\boldsymbol{X}_j^A - \boldsymbol{X}_j^C\|} + \frac{L_{AB}}{\|\boldsymbol{X}_j^A - \boldsymbol{X}_j^B\|} + \frac{L_{BC}}{\|\boldsymbol{X}_j^B - \boldsymbol{X}_j^C\|} \right] \tag{2.18}$$

for $\|\boldsymbol{a}\|$ the 2-norm of the vector $\boldsymbol{a}$.

## 2.4 Final considerations

This chapter described the constraints and assumptions of optical IPS and the mathematical basis under the 3D reconstruction of a 2D marker. The next chapter must thus clarify how these assumptions were assured and the elements used to design the testbed.

# Chapter 3

# The proposed indoor positioning system

The hardware and architecture of each experimental phase of the system are discussed in this chapter.

## 3.1 Hardware design

The system's physical architecture comprises five principal components: target subjects, onboard reflective markers, scenario constraints, cameras and arena workspace. Each of these components will be discussed below.

### 3.1.1 Monitored mobile vehicle

A Tello DJI and the Parrot Mambo quadrotor are defined as our monitored mobile vehicles for this work. These drones are commercially available and have support within MATLAB and `python`, allowing future studies to feedback on the estimated pose to the onboard controller. Taking the bumpers into account, both drones are around 18 cm broad.

A test wand is constructed similar to where the markers would be attached to the drone, as shown in Figure 3.1. The alignment of the markers is used to minimise reprojection errors, whilst the fourth sphere enables orientation estimation. For extrinsics calibration, one may use only the aligned markers similar to the wand illustrated in Figure 2.4. It should be mentioned that the markers may be configured in any way wanted; this was only the decision of our work.

**Figure 3.1:** Picture of the Tello DJI (a), Parrot Mambo (b) and the four maker wand (c).

**(a)** Tello DJI — 80g, 18 cm × 18 cm

**(b)** Parrot Mambo — 60g, 18 cm × 18cm

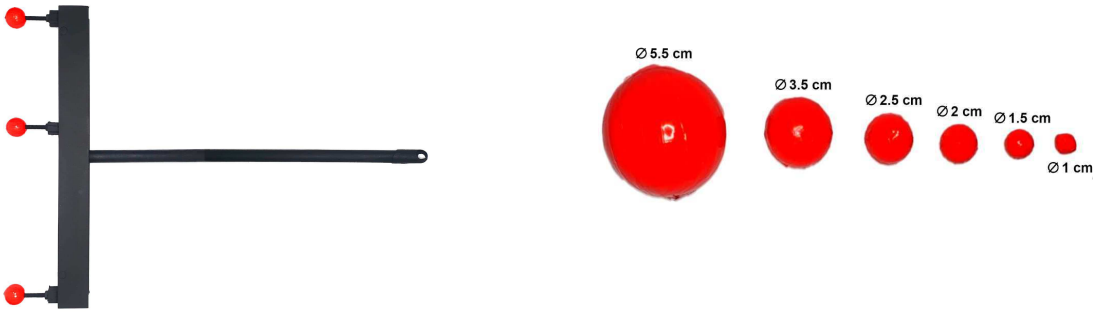
**(c)** Test wand — 15 cm × 15 cm

Source: Prepared by the author.

### 3.1.2 Reflective markers

The markers are built by sticking reflective car wrapping paper on polystyrene spheres. Considering the drones' width and height, the chosen markers have a 20 mm diameter. However, we coated spheres of multiple diameters (Figure 3.2b) to discuss the marker's visibility and reconstruction accuracy in various resolutions, which will be presented in chapter 4. A larger 50 cm wand was constructed for these experiments (Figure 3.2a).

**Figure 3.2:** Picture of the 50 cm wand (a) and the markers used to test the MCS (b).




**(a)** Larger three marker 1D wand

**(b)** Multiple size markers.

Source: Prepared by the author.

### 3.1.3 Scene settings

After capture, sequential filters segment the markers from the image background. To prevent continuous threshold adjustments, the illumination must be constant. Therefore, all light sources should be artificial, and the camera's FPS should be slower than the utility

frequency (60Hz) if no IR low or band-pass filter is used. The camera's auto exposure and white balance settings must also be disabled since they affect the intensity of the image and shift the timing to a dynamic rate.

Another practical tip is to wear darker clothing and painting the wand matte black when calibrating since this creates greater contrast between the markers background. The windows were also blocked and blackout curtains were added to guarantee no outer IR light leakage. The floor was covered with a rubber mat to avoid breaking the drones upon collision. Pictures of inside and outside the arena are shown in Figure 3.3.

**Figure 3.3:** Images from both inside (a) and outside (b) the arena. The shelves are 2.6 meters above the ground and house the Raspberry Pi, cooler, and camera. Curtains were installed to block IR light and shield laboratory users from fast-moving vehicles. Opaque paper was also used to cover the windows to prevent IR light from entering the arena.

**(a)** Inside the arena.

**(b)** Outside the arena.

Source: Prepared by the author.

### 3.1.4 Cameras and arena setup

The camera arrangement divided the arena design into three phases. In the initial phase, two RealSense cameras were used. These cameras already offer intrinsic fabric calibration (pinhole model in Equation 2.1), simplifying the workflow to only extrinsic calibration (as in Equation 2.14). They cost, however, over $300 since they also include an stereo pair and an inbuilt laser aside from the RGB camera.

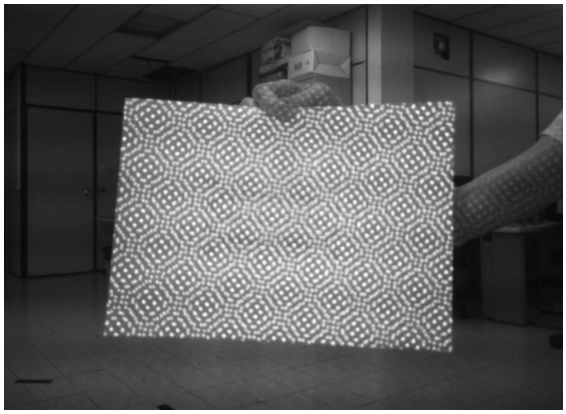
In the second phase, two Raspberry NoIR V1 cameras were employed. These cameras cost $15 each and have a broader range of low-level access settings than the high-end RealSense cameras. Nevertheless, they have a sharpness issue when operating at over 40FPS. This issue is not present at the second version (V2) of the same camera.

The final of the setup runs over four Raspberry NoIR V2 cameras. Following, each phase will be thoroughly discussed.

### Phase 1

In the first phase, the system was composed two RealSense D415 cameras. Since these cameras contain embedded intrinsic calibration (pinhole model in Equation 2.1) and synchronous software triggers, the output image is already rectified and simultaneous between all devices. They are also equipped with one RGB and two IR sensors, besides an IR beam to lighten the scene. Nevertheless, this laser has a projection pattern (see Figure 3.4) that challenges the segmentation of the markers. Hence, the internal beam is switched off, and 830nm IR reflectors are aligned with each camera's optical axis (see Figure 3.5).

**Figure 3.4:** Picture of the Real Sense D415 laser projection pattern.



Source: Prepared by the author.

**Figure 3.5:** Picture of RealSense camera aligned with an IR emitter (a,b) and the Raspberry V1 camera attached to an IR pass-filter (c).

**(a)** RealSense setup side-view

**(b)** RealSense setup front-view

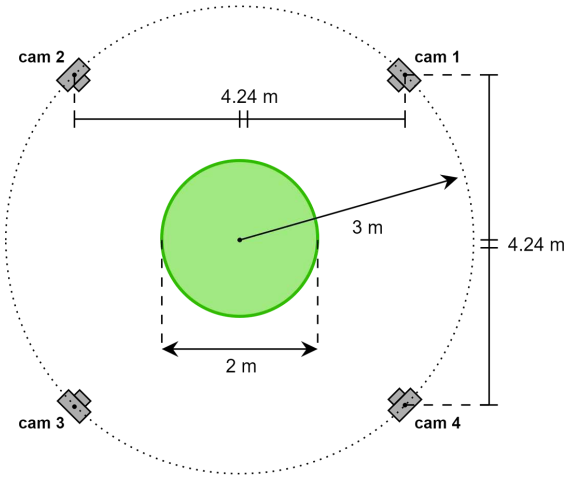
**(c)** Raspberry V1 camera

Source: Prepared by the author.

All RealSense cameras were configured with identical factory $\boldsymbol{K}$ matrices. Because of the minimal 0.25V jitter necessary to avoid frame drops, one camera must be powered externally through an USB hub. Note that a RealSense camera is costly and underused as a monocular night vision camera. In this work, they are employed solely to test the extrinsic calibration (Equation 2.14) before considering off-the-shelf, low-cost cameras susceptible to inaccurate intrinsic calibration using Equation 2.5 and Equation 2.6.

The capture area was inspired by the OptiTrack Flex 3 four-camera MCS, arranged for a 2-meter diameter arena. The cameras were arranged in the scene as shown in Figure 3.6. A 3m 2.0 USB cable was used to connect each camera to the central processing unit and the cameras, for which the maximum update rate is 30 FPS for a 640×480 pixels resolution. A faster FPS or greater resolution would only be achieved with 3.0 USB active cables. All cameras are positioned at 1.5 meter high from the ground.

**Figure 3.6:** Scene layout for a circular 2-meter diameter arena similar to the OptiTrack MCS. The cameras are equally spaced and positioned over a 6-meter diameter circumference.




Source: Prepared by the author.

### Phase 2

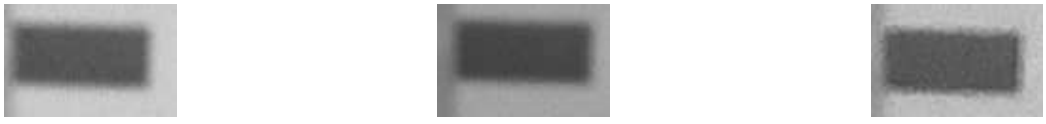
The system was tested in a second phase with Raspberry Pi V1 NoIR cameras with 60-degree FOV lenses. They were place in the same setup as the phase 1 at 1.5 meter high, like depicted in Figure 3.6. A flat cable connects each camera to a Raspberry, then connected to the server via Ethernet. They ran at 40 FPS for a 640× 480 pixels resolution.

In contrast to RealSense, V1 cameras require intrinsic calibration (using the generic

model in Equation 2.5 and Equation 2.6) and handling of asynchronous capture. They also do not offer IR-pass filters, which should be externally placed in front of the lenses. An easy way to construct an IR-pass filter is to use floppy disks, but HWB780 glass filter was used instead. Two 850nm IR LEDs are placed next to the camera module, as illustrated in Figure 3.5. The photoresistor was covered to ensure constant and maximum lightning.

The NoIR cameras have a discrete set of modes (FPS, resolution, FOV and aspect ratio sets) to output the data from the GPU. According to the `raspicam` documentation[37], any frame rate above 30FPS initiates binning, a procedure that combines a window of pixels. This technique reduces resolution and lowers latency to output the final image. At 40 FPS, the binning is 2×2 (horizontal and vertical), whereas, at 90FPS, the binning is 4x4. Because the resolution downscale at 90FPS impacts the sharpness of the marker boundaries (see Figure 3.7), the camera was upgraded to the V2 version in the third phase.

**Figure 3.7:** Border quality comparison of V1 Raspberry camera binning modes using a Koren 2003 lens test chart band 3 (a,b,c) and the constructed markers (c,d,e). The difference in border sharpness between modes 1 (10FPS) and 4 (10FPS) is not visible (40FPS). However, while utilising mode 7 binning for 90 FPS, notice the border erosion. The blob contour's ability to estimate the centroid is hampered by this indefiniteness.



**(a)** No binning, mode 1 **(b)** 2×2 binning, mode 4 **(c)** 4×4 binning, mode 7

**(d)** No binning, mode 1 **(e)** 2×2 binning, mode 4 **(f)** 4×4 binning, mode 7

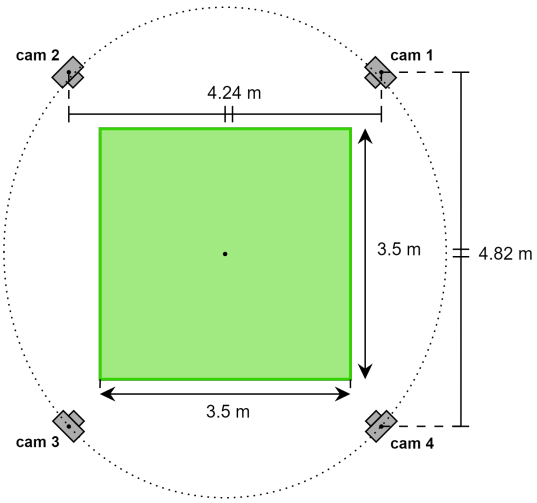
Source: Prepared by the author.

### Phase 3

The third phase expanded the system to four Raspberry V2 NoIR. Each one was placed over shelves at 2.6 meters from the ground to expand the height of the working space. This setup now depends on the walls' configuration, which is depicted in Figure 3.8.

According to the Raspberry documentation, the Sony IMX219 sensor captures until 90 FPS with a 2×2 binning. However, preliminary tests showed that a Raspberry 3b+ could

**Figure 3.8:** Scene layout for a circular arena using shelves as support for the capture station. Similar to the OptiTrack Flex 3 arena, the cameras are positioned 4.24 m apart on the same wall. Cameras are 60 cm further away on opposite walls. The working space is a square with a side length of 3.5 meters.



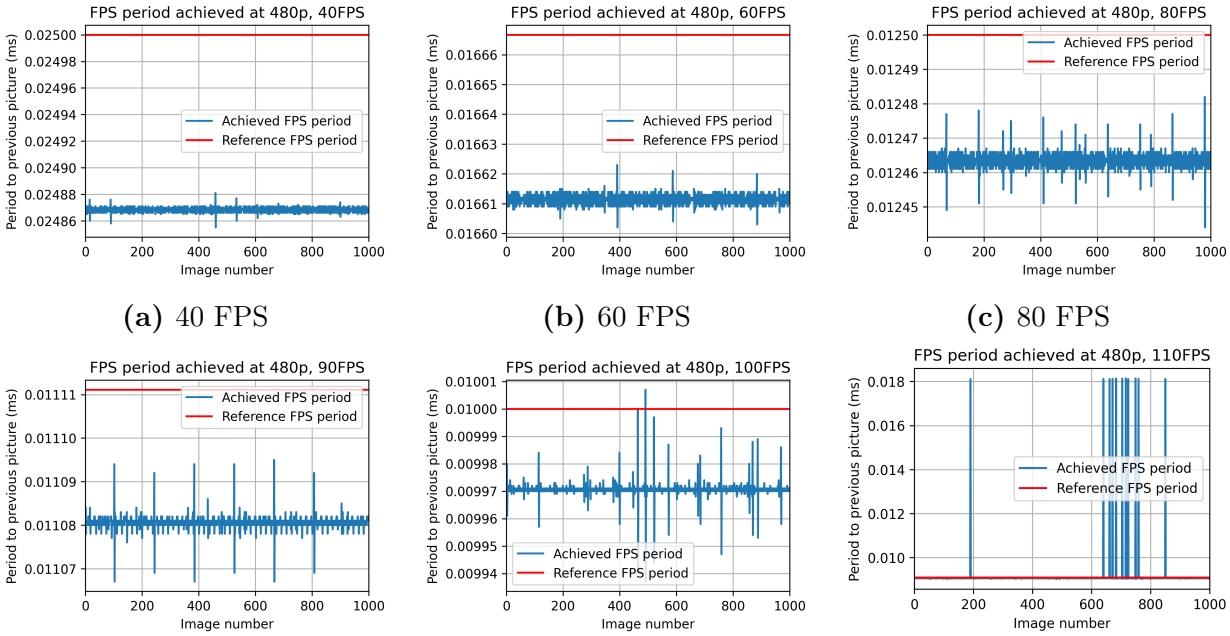

Source: Prepared by the author.

operate at 100 FPS, while a Raspberry 4 could operate at 150 FPS, both for a resolution of 640×480 pixels using YUV/bitmap. For this test, a sequence of 1000 images with a reference period of 0.01 ms was collected at mode seven, and the mean PTS was 0.00997 ms. Following testing revealed that the obtained frame rate is always faster than the reference frame rate until 100 FPS (see Figure 3.9). The problem now is accomplishing the same pace of 100 FPS on image processing when running in a constrained processing environment.

The resolution may also be increased to 720p (equivalent to mode six) without loosing quality. In this resolution, the Raspberry 3b+ achieved 60 FPS, and the Raspberry 4, 85 FPS, as shown in Figure 3.10. The cameras' intrinsic matrix must be recalculated when the resolution is changed due to the FOV update, as described in chapter 2. There is a shrinking of the FOV in mode six (see Figure 3.11) in relation to mode 5, and thus the maximum flight height must be restricted.

Unlike the V1 module, the NoIR V2 cameras do not offer pins to power the infrared reflectors. So a circuit was developed to trigger an IR LED ring via the GPIO pins of the Raspberry (see Figure 3.12 and Figure 3.13). Our ring was a leftover from a surveillance camera, and the LDR was removed to avoid needing to cover it. It must be aligned with the camera axis, as shown in Figure 3.14. Note that the black ribbon internally placed in the

**Figure 3.9:** Time difference between frames in mode 7 and 480p resolution for a V2 camera at various FPS ratios. The maximum attainable frame rate is 100 FPS. The camera begins to lose frames at 110 FPS, as seen by the oscillation in (f).

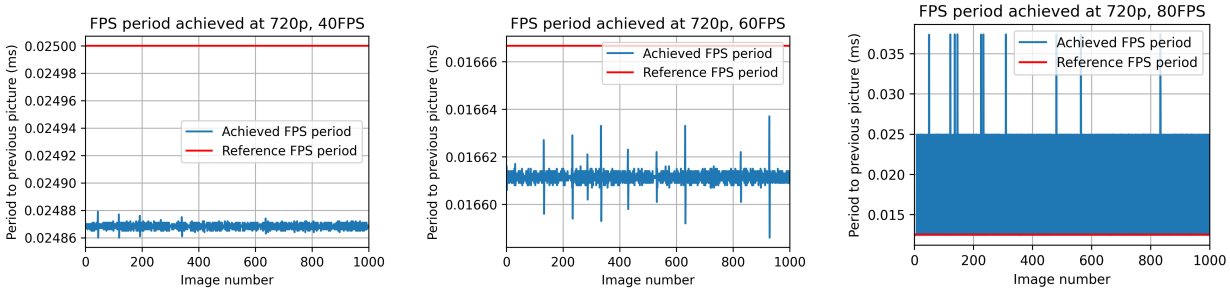



**(a)** 40 FPS **(b)** 60 FPS **(c)** 80 FPS

**(d)** 90 FPS **(e)** 100 FPS **(f)** 110 FPS

Source: Prepared by the author.

**Figure 3.10:** Time difference between frames in mode 6 and 720p resolution for a V2 camera at varying frame rate. The highest achievable FPS is 60 FPS. Disregarding outliers, the period between frames for 80 FPS oscillates at 16 ms (60 FPS).



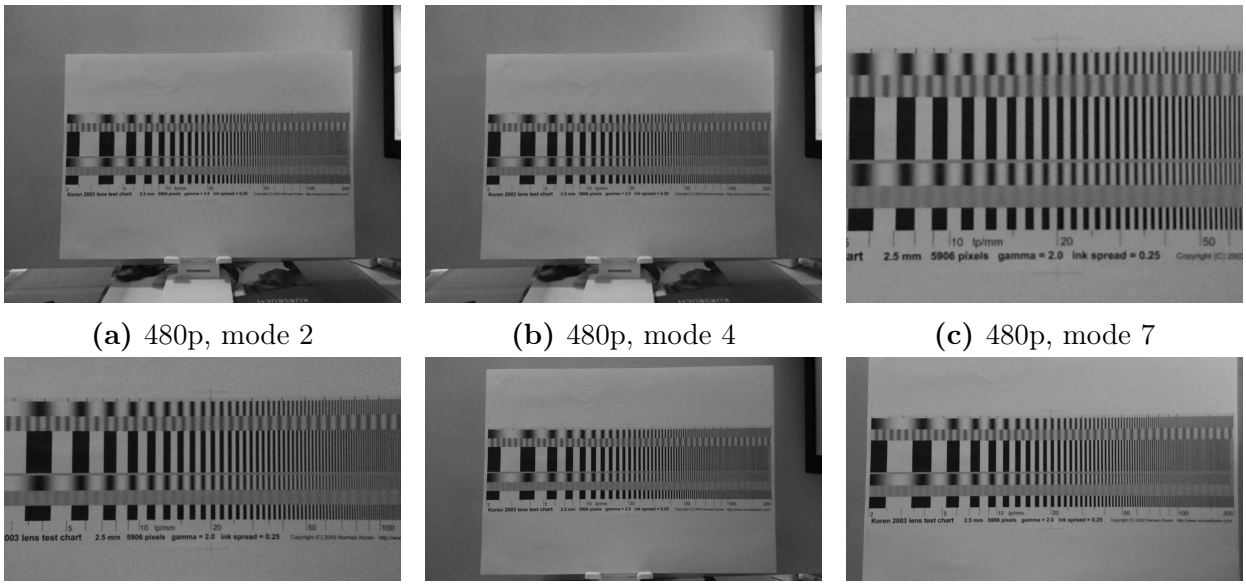

**(a)** 40 FPS **(b)** 60 FPS **(c)** 80 FPS

Source: Prepared by the author.

ring is to avoid light leakage. The LEDs turn on only during capture to avoid overheating.

Furthermore, due to the high FPS, the temperature of the Raspberry approaches 80°C, triggering frequency capping and resulting in frame loss. So a cooler was installed on the setup station to keep the processor temperature stable at 35°C, as depicted in Figure 3.14.

**Figure 3.11:** Exemplifications of Raspberry capture modes utilising a V2 camera and Koren 2003 lens test chart. It should be noted that the FOV of the sensor varies across modes. Modes 2 and 4 record the whole sensor, with the latter having a 2x2 binning in comparison to the first. Modes 6 and 7 are the fast capture modes, with 15 percent and 45 percent ratios from the original sensor resolution, respectively, and both having 2x2 binning. Mode 1 is the natural default mode and takes up just 25% of the original sensor area. Mode 5 is a cropped variant of mode 4 with a 16:9 aspect ratio.

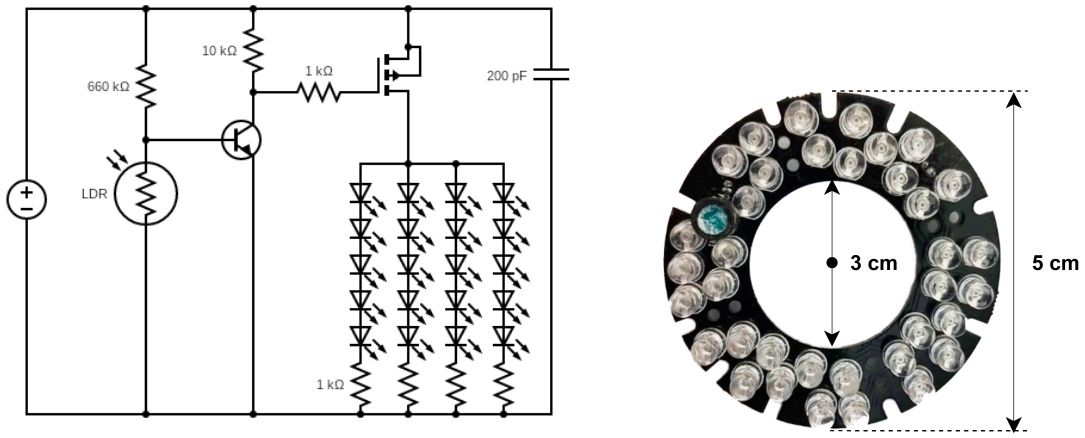


**(a)** 480p, mode 2 **(b)** 480p, mode 4 **(c)** 480p, mode 7

**(d)** 720p, mode 1 **(e)** 720p, mode 5 **(f)** 720p, mode 6

Source: Prepared by the author.

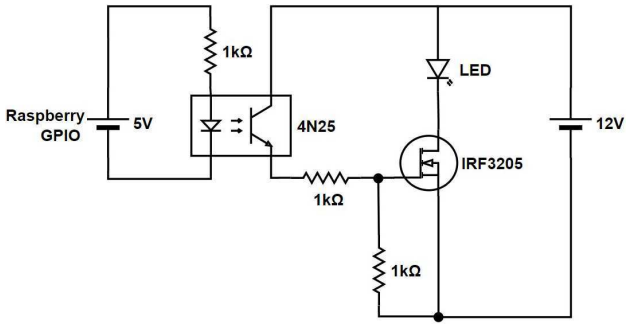
**Figure 3.12:** Circuit diagram and picture of the IR LED ring. In this example, four lines of five LEDs were driven by a p-channel FET, activated by the NPN transistor when the LDR is open. The ring used has 36 IR 830nm LEDs summing 3W.




**(a)** Circuit of the LED ring **(b)** Dimensions of the LED ring

Source: Prepared by the author.

**Figure 3.13:** Circuit diagram that drives the IR LED ring. The GPIO pins of the Raspberry Pi are used to apply 5V to the photodiode, which activates the transistor and shuts the NMOS MOSFET. The load (in this case, the LED ring) is then directly polarised.




Source: Prepared by the author.

**Figure 3.14:** Montage of the camera box and the cooler. In(a), the camera NoIR V2 was put within a box 8 cm×8 cm×5 cm, alongside the driving circuit. A low-pass filter was placed on the camera lens and wrapped with insulating tape to prevent leakage. The box was also mounted on the top of a gimbal to facilitate movement and positioning. In (b), a PC cooler was put over the Raspberry to avoid heating. The flat cable was used to pull the cables connected to the GPIO pins away from the ventilator.

**(a)** Front of the camera box.

**(b)** Setup of the cooler and Raspberry.

Source: Prepared by the author.

## 3.2 Software design

The system logic may be divided into two steps: the communications protocol between the capture stations and the server, and how to rebuild the marker's 3D poses.

### 3.2.1 Communications protocol

In the first and second phases, image processing is done offline. Therefore, all pictures are captured before the image processing is executed on the server. In the third phase, however, the system is functional real time. As a result, the photos must be processed

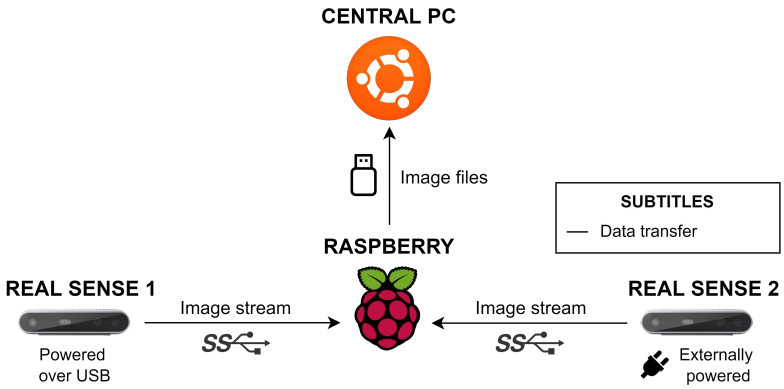
and delivered to the server as quick as the cameras' frame rate. This condition requires distributed image processing, and thus alters the protocol between the cameras and the central server.

The communication between the capture stations in each phase and the server will be discussed next. Only later in this chapter will we look at how the server processes the data.

### Phase 1

In the first phase, all RealSense cameras are connected to one Raspberry Pi 4, which polls synchronous images using Intel `pyrealsense`[38] python wrapper cross-compiled to Debian OS on ARM architecture. When frames are dropped, both streams are blocked, and the flow waits for the subsequent available images. The images are then transferred to a Desktop, where OpenCV and Numpy are applied for image processing. Because the processing is done offline, using a USB stick to transfer the pictures has no effect. The connection scheme between the cameras and the central PC is shown in Figure 3.15.

**Figure 3.15:** Scheme of the connection between the RealSense and the Desktop for phase 1 experiments. Each camera is connected to a Raspberry Pi, which stores the photos to disk. The dataset is then transmitted to the desktop via USB stick.




Source: Prepared by the author.

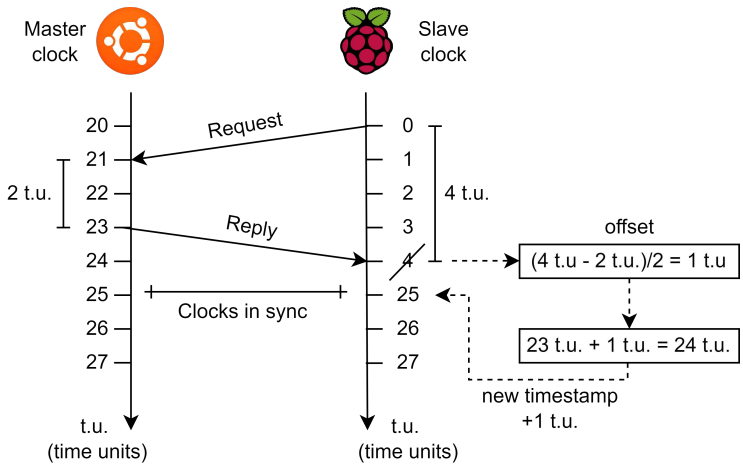
### Phase 2

In the phase 2, two Raspberry 4 are configured as a server-client pair. They are both connected to the server via Ethernet. When the client Raspberry connects to the server, the master Raspberry sends a UDP package with the start timestamp and recording length.

Even though the cameras are configured to achieve the same FPS, there is no guarantee that the images are taken simultaneously (see Figure 2.1). As a result, corresponding image points from different views may not represent the same point in space. The images are then artificially synced, and a match is considered if the timestamps deviate by less than 0.5/FPS. This avoids self-designed trigger platforms (like OptiTrack eSync).

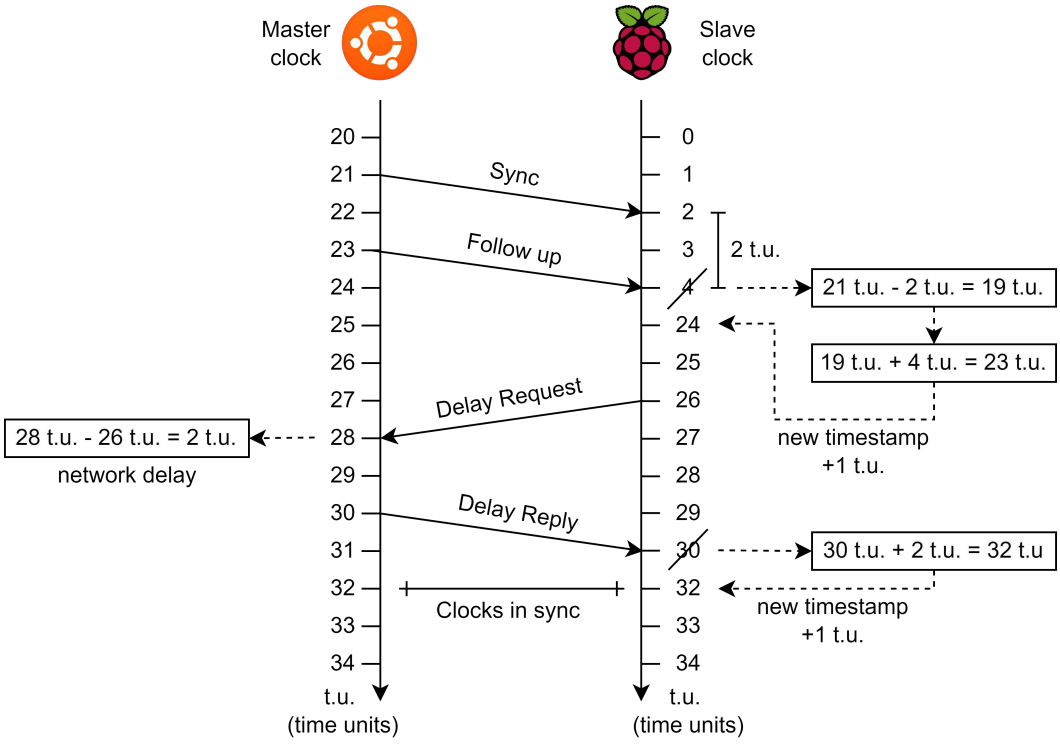
To trigger the cameras as close in time as possible, the Raspberries clocks were synced through a PTP[39] (Precision Time Protocol) using `ptpd`[40]. In this protocol, a master periodically broadcasts the current time to the other slave clocks, which are adjusted to compensate for the network transit. To ensure that the PTP worked adequately, the Raspberries' default NTP[41] server was disabled. The difference between the PTP and NTP clock synchronisation protocols is how they account for network latency, as illustrated in Figure 3.17 and Figure 3.16. The connection between the Raspberries and the Desktop is depicted in Figure 3.18. The Desktop was used as the grandmaster and Raspberries, as slaves. In the event of a master clock jitter or loss, a redundant Raspberry is available.

**Figure 3.16:** Scheme of clock synchronisation using NTP. A round-trip delay must be computed to adjust the slave clock. First, the slave request a delayed package to the master, which arrives at 21 t.u. local master time. The master then answers with a package containing receiving and sending timestamps (21 and 23 t.u.) to the slave. We now have the delay between sending and receiving the message in the master (2 t.u.) and the slave (4 t.u.). Following, the offset of the slave clock from the master clock is computed as half the difference between these delays (1 t.u.) since they count for a round trip. The slave then adjusts its timestamp to the sum of the reply timestamp of the master and the single path delay (25 t.u.). Note that this protocol only works for networks with a symmetrical delay between the slave and the master.




Source: Prepared by the author.

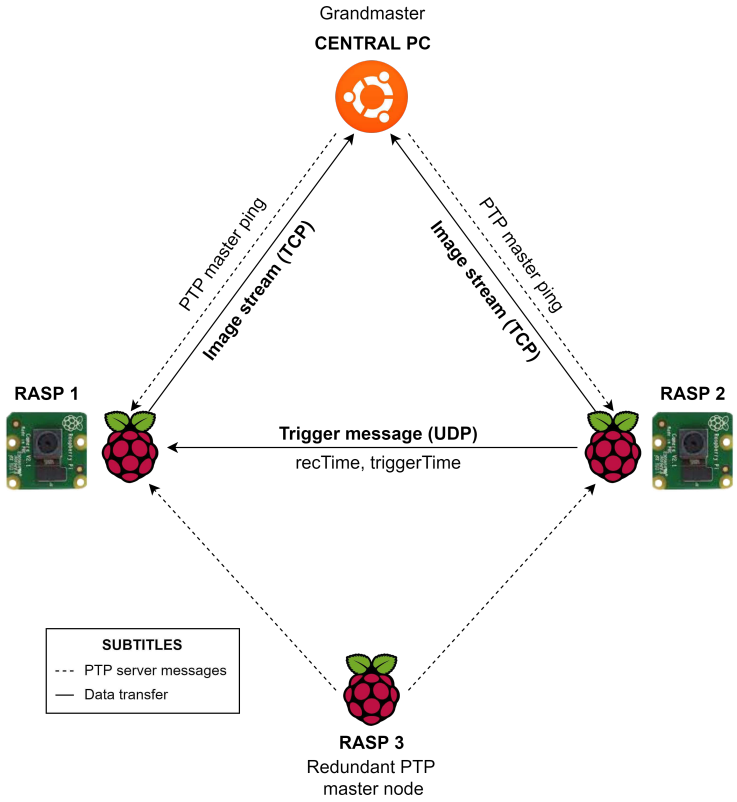
**Figure 3.17:** Scheme of clock synchronisation using PTP. To adjust the slave clock accounting for network latency, two round-trip delays must be computed. The master broadcasts a sync message at 21 t.u., which is received at 2 t.u. local slave time. A follow-up message is sent containing the previous timestamp (21 t.u.) from the master, which is received at 4 t.u. local slave time. The slave then computes the difference between the slave and master timestamps and adds this offset to its present timestamp. Following this, the slaves send a delay request to the master. The master answers with a package comprising when the delay request was received (28 t.u. master local time). The slave can now determine the network delay by subtracting the time it sent the delay request (26 t.u.) from when the master received it (28 t.u.), resulting in a 2 t.u. offset. The slave then adjusts its clocks by the offset and is now in sync with the master. Note that the master will periodically perform the pool to maintain accuracy.




Source: Adapted from Perle Systems[39].

The captured JPEG images are then streamed through a TCP port to the Desktop, alongside a CSV file containing the timestamps. This CSV is transmitted from each Raspberry to the Desktop when the final image is captured. It should be noted that phase 2 is done offline, so getting the timestamp after the capture is not an issue. These timestamps were equal to the CPU timestamp when the image was saved informed through the python library `time`. The `piCamera`[37] python wrapper was applied, as other wrappers (e.g. `imutils`, `opencv`) do not provide frame rate input.

**Figure 3.18:** Scheme of the connection between the Raspberries and the Desktop for phase 2 experiments. Both capture Raspberries and the redundant Raspberry are connected to the Desktop through a PTP server. The TCP image stream port is opened after the trigger timestamp has been reached on the Raspberries with the cameras. This timestamp and the recording duration are transferred between Raspberries in a master-slave configuration.



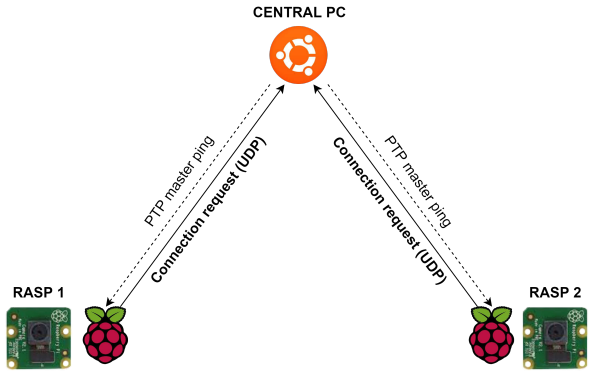

Source: Prepared by the author.

### Phase 3

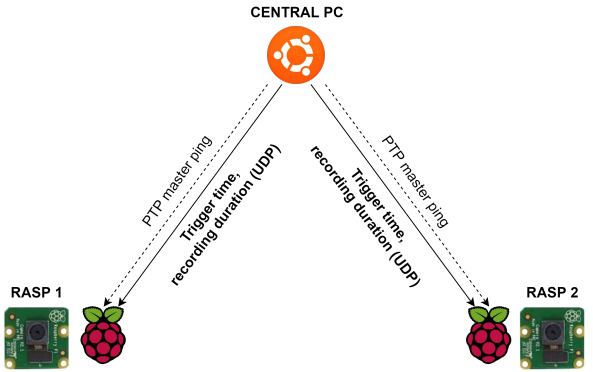
In the third phase, all Raspberries function as clients, while the Desktop acts as the server, broadcasting the trigger/recording time. Image processing was decentralised and performed on each Raspberry Pi, which have limited memory and processing capability. How image processing is done will be covered later in this chapter. Finally, the centroids coordinates of the candidate markers are delivered to the server over UDP. The connection scheme between the Raspberries and the PC in phase 3 is shown in Figure 3.19.

In phase 3, the low-level `raspicam` code inside `userland` module was changed[42] and recompiled to alternate the buffer between the image arrays and the presentation timestamp (PTS), which is the timestamp of the trigger exported from the camera chipset. This PTS

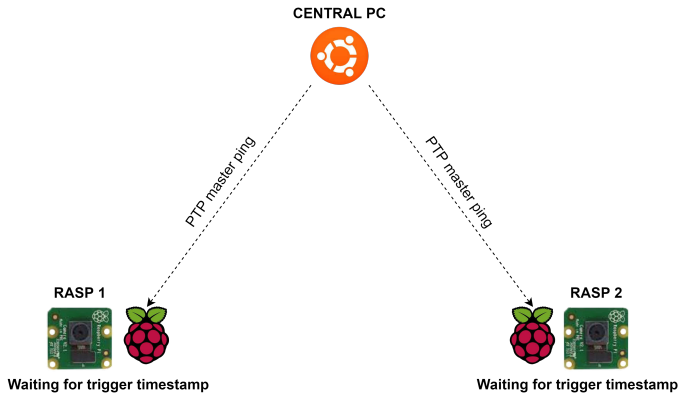
**Figure 3.19:** Scheme of the connection between the Raspberries and the Desktop for phase 3 experiments. The activation sequence follows the order (a), (b), (c), (d).



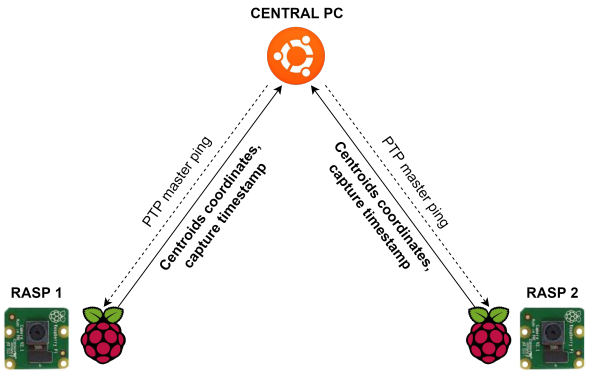

**(a)** Both Raspberries transmit a random message to the server, which records the IP and open port of the Raspberries.




**(b)** When the maximum supported camera number is reached, the server sends the trigger timestamp and recording length to the clients.




**(c)** Raspberries then enter standby mode until the trigger timestamp is reached, at which point they begin the capture procedure.




**(d)** Only the centroids coordinates and presentation timestamps are supplied to the server once the images have been analysed on board.

Source: Prepared by the author.

generated by the camera sensor contains a variable number of leading null bytes for the image sizes 960×540 and 1280×960 pixels. As a solution, an inline character was added after the PTS value and the entire line was scanned. Then the final 10 bytes were checked as the PTS value, ignoring the leading zeros.

With the PTS being the exact timestamp of the image capture, cameras may now run at various frame rate and be later interpolated to a standard time vector. This technique, in conjunction with the clock synchronisation approach, allows any Linux platform (e.g., Orange Pi, Jetson Nano) to be employed on the proposed system. The final architecture is thus heterogeneous, asynchronous, distributed, and not limited to real-time kernels. Undoubtedly, more research is needed to explore the limit of the difference between the cameras' FPS and the virtual dataset and the implications of this discrepancy on the triangulated vehicles' pose. At least half of the vehicle's control loop was attempted to be accomplished via interpolation, which was done over a standard time vector of 100 FPS.

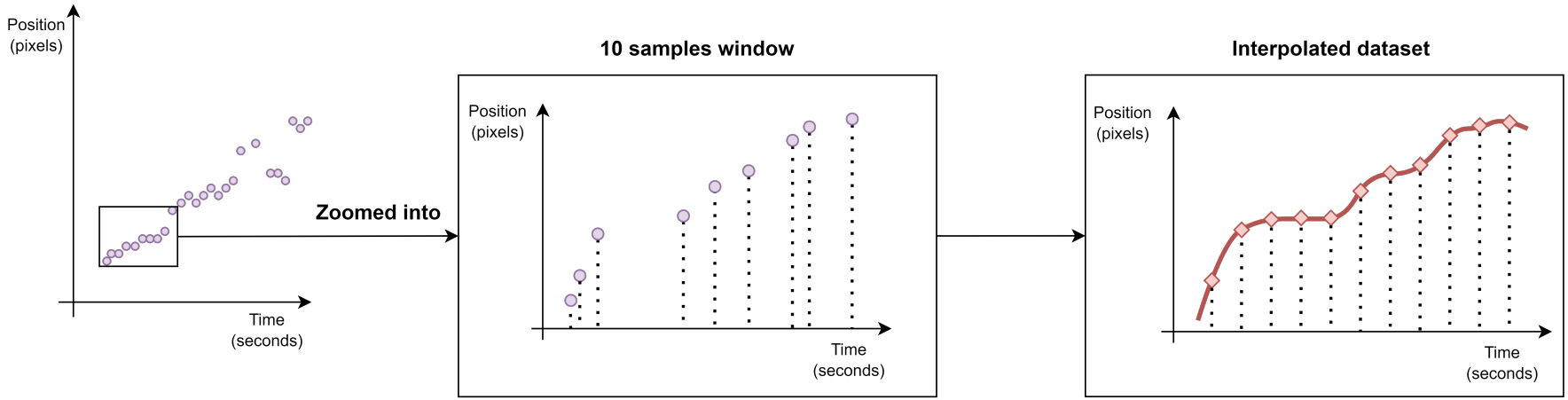
In this work, interpolation was performed on ten samples using a cubic spline[43], as depicted in Figure 3.20. Thus, a warm-up of ten valid images is necessary. Because interpolation is executed on each marker's 2D location, the correct association of the markers between images from the same camera must be assured. The suggested ordering approach and the validity concept will be studied further in this chapter.

Note that the segmentation and ordering procedure is carried out online. The solution's scalability is an advantage since several cameras may be employed as long as the bandwidth is within the LAN's range. Table 3.1 shows a summary of each phase flow.

### 3.2.2 3D reconstruction workflow

Firstly, the intrinsic parameters must be calibrated. The matrices $\boldsymbol{K}$ and $\boldsymbol{K}'$ are achieved using the procedure described in chapter 2, matching 2D coordinates of crossing points on the chessboard to the actual 3D square pattern length. The intrinsic calibration was searched using the generic camera model in Equation 2.5 through MATLAB and double-proofed with `OpenCV`. The code[45] was adapted from Deng et al.[21] radial model, who created the calibration toolbox using the model of Kannala and Brandt[36] but employing chessboard instead of circular patterns. The initial guess of the intrinsic parameters was corrected to

**Figure 3.20:** Diagram of the interpolation flow to virtually synchronise the capture stations. As previously declared, there is no assurance that the cameras trigger simultaneously, nor if the FPS period is constant. For instance, see the purple graphic below for the X position, in pixels, of marker A in the first camera. Generally, for each marker's X (or Y) position in each camera, we capture subsequent 10 sample windows. Each 10-sample window is interpolated using a cubic spline to a standard time vector of 100 FPS, as illustrated in red. Note that the captured positions of the markers are now equally spaced in time. We then have an equally spaced virtual dataset with equal timestamps for all cameras. This is the proposed solution for the heterogeneity and asynchrony of the system.




Source: Prepared by the author.

**Table 3.1:** Comparison the phases hardware and testbed architecture.

| | **Phase 1** | **Phase 2** | **Phase 3** |
|---|---|---|---|
| **Camera hardware** | RealSense D415 | OmniVision OV4647 | Sony IMX219 |
| **Camera set** | Homogeneous | Homogeneous | Heterogeneous (Model 3, 3B+ and 4) |
| **Auto exposure and white balance shutoff** | Available | Available (after modification[44]) | Available (after modification[44][42]) |
| **Image output** | Rectified | Distorted | Distorted |
| **Image dimensions** | 640×480 pixels | 640×480 pixels | Multiple |
| **Frame rate** | 30 FPS | 40 FPS | Multiple |
| **External IR-pass filter** | No | Yes | Yes |
| **Trigger type** | Synchronous and centralised | Asynchronous and distributed | Asynchronous and centralised |
| **Image processing** | Centralised | Centralised | Decentralised |

Source: Prepared by the author.

use the nominal values of the focal distance in pixels instead of the pixel/millimetre conversion factor. The chessboard used in the intrinsic calibration has 11×12 squares with 30 mm length. The calibration pattern was close to the image's boundaries, and the board angle and rotation varied throughout the images used in the calibration.

Afterwards, the extrinsic calibration is split as marker detection, ordering, computation of the essential matrix, decomposition into rotation and translation and scale estimation. The last three steps are already described in chapter 2, and the others will be described next. After full calibration, the pose $\boldsymbol{X}$ of the monitored vehicle during a mission is triangulated based on Equation 2.17 for the computed values of $\boldsymbol{x}$, $\boldsymbol{x}^{'}$, $\lambda$ and $\boldsymbol{P}^{'}$.

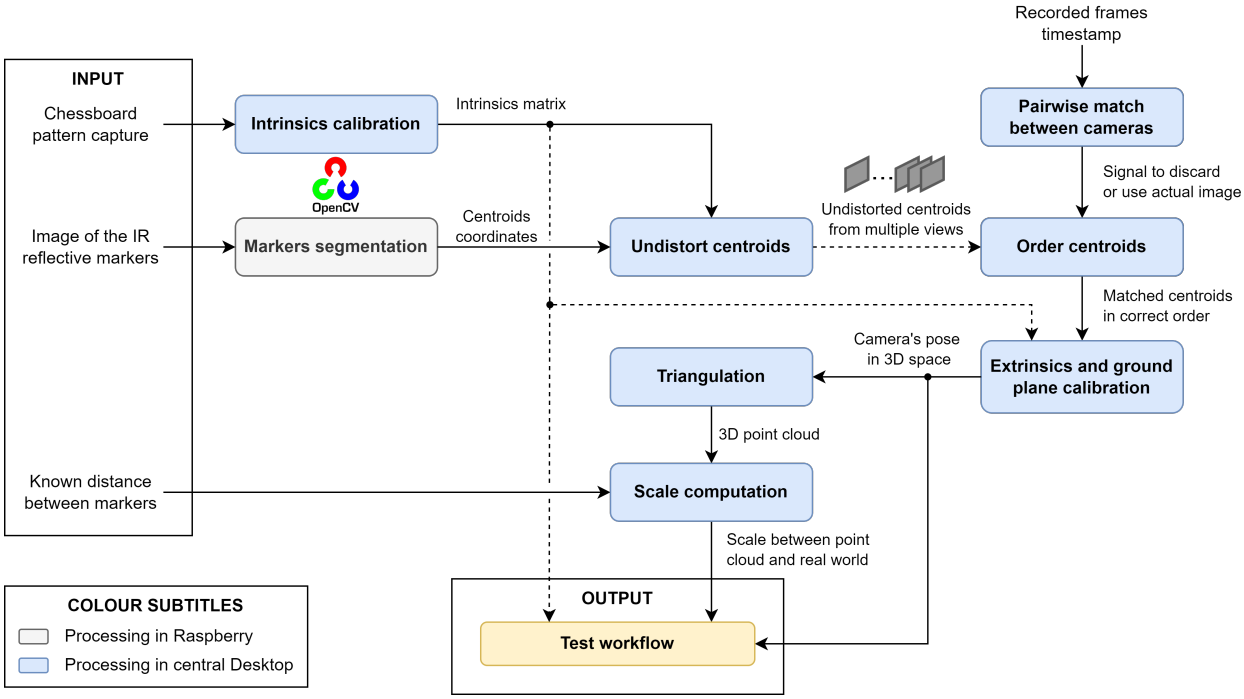
After calibrating, the user can test the system using the same approach of pose reconstruction. The calibration and test flows are depicted in Figure 3.21 and Figure 3.22, respectively. The difference between these flows is the extrinsic and intrinsic calibration.

### Marker detection

The marker detection is to identify and extract the subpixel location of the markers. This segmentation varies between centralised and decentralised for the offline (phase 1 and 2) and online phases (phase 3), respectively.

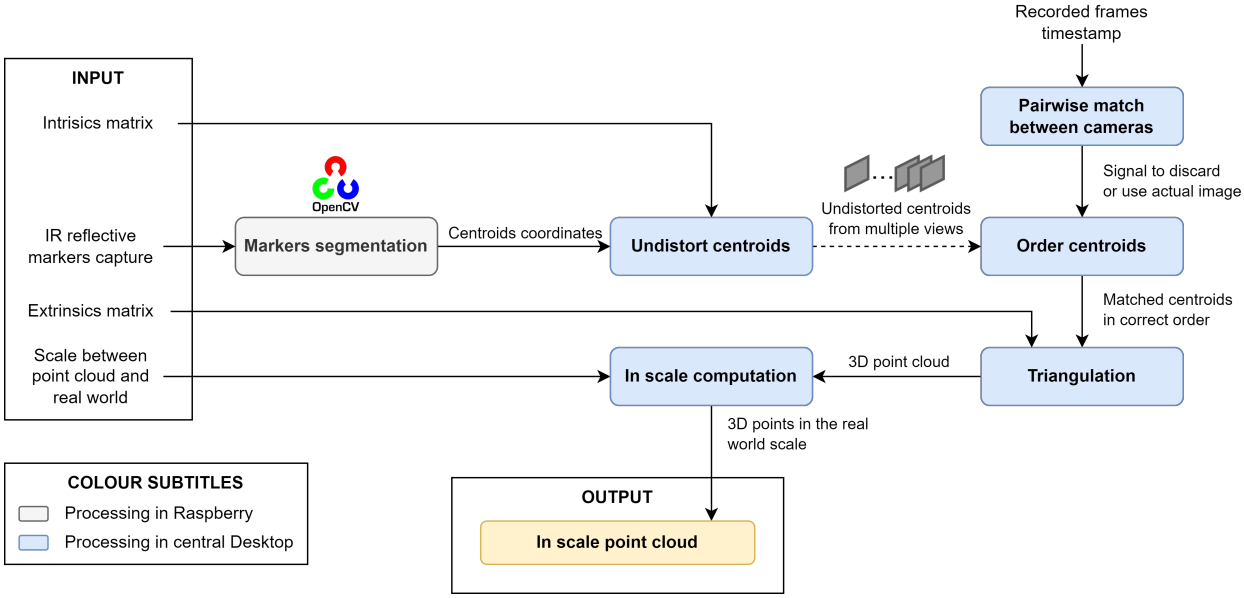
In phase 1, a picture of the empty arena was captured and then subtracted from each

**Figure 3.21:** Illustration of the calibration flow. As input, checkerboard pictures, markers images, and the known distance between markers are supplied. The flow generates the intrisics parameters, 3D camera postures, and the scale between the point cloud and the actual world for use in the test flow.




Source: Prepared by the author.

**Figure 3.22:** Illustration of the test flow. As input, intrinsics/extrinsics parameters, markers images, and the scale between the point cloud and the actual world are supplied. The flow generates the markers 3D pose in relation to the 1st camera.



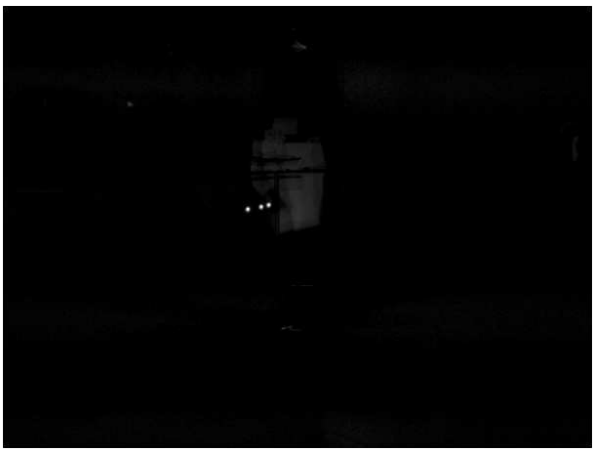

Source: Prepared by the author.

subsequent image. The outcome is a picture that only includes the objects not previously in the scene, namely the calibration wand, the markers, and the person caring for the wand. In phase 2, the original picture is sent to the threshold filter since contrast control is available and the backdrop remains dark compared to the marker's brightness.

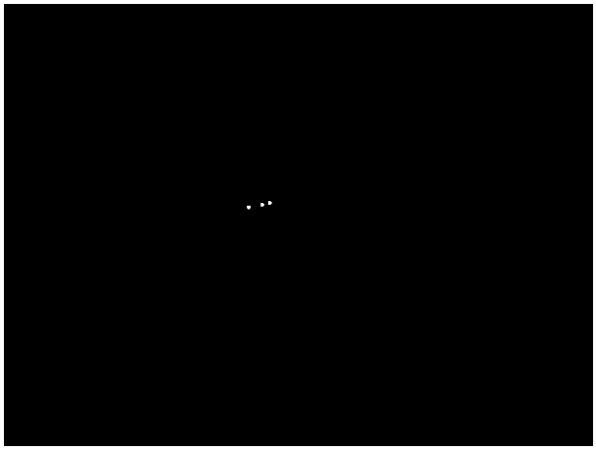
The area of the markers was then identified using a threshold filter with a cut intensity of 127, as shown in Figure 3.23. Firstly, the far most upper-left $(x_{UL}, y_{UL})$ and lower-right $(x_{LR}, y_{LR})$ non-null pixels are determined. Secondly, a bounding box with a margin of 5 pixels is stamped around these points and used to mask the original image.

**Figure 3.23:** Workflow of marker segmentation from the original image (a) to subtraction the static background (b), passing through 127 cut threshold filter (c), and masking with a bounding box (d) using RealSense cameras.

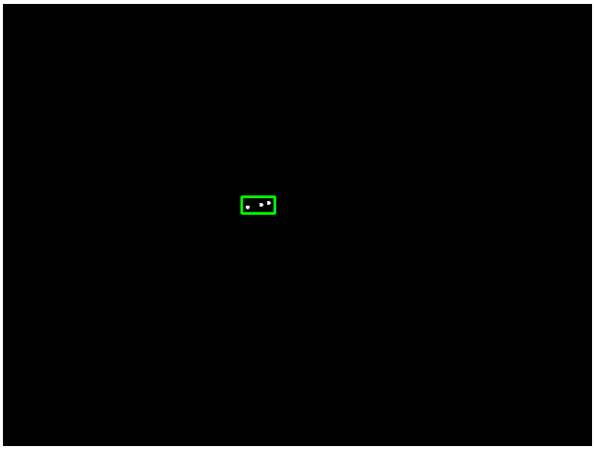
**(a)** Original image



**(b)** Subtraction of clean plate



**(c)** Threshold filter



**(d)** Bounding box on the difference image

Source: Prepared by the author.

The masked picture is then inverted and transformed to binary using a set of threshold filters. The step between the threshold filters, the maximum and minimum thresholds are all adjustable parameters. The connected components from every binary image are extracted and stored. As close centres correspond to the same blob, these components from several images are grouped by their coordinate centre. The groups are then filtered by convexity and area. To calculate the marker centre with subpixel precision, the image moment $M_{pq}$ was employed:

$$M_{pq} = \sum_{u} \sum_{v} u^p v^q I(u, v) \tag{3.1}$$

for $p, q \in [0, 1]$ and $I(u, v)$ the intensity of image point $(u, v)$ on the masked image. The centre is computed as:

$$\begin{cases} \hat{x} & = M_{10} \,/\, M_{00} \\ \hat{y} & = M_{01} \,/\, M_{00} \end{cases} \tag{3.2}$$

After obtaining $(\hat{x}, \hat{y})$ of all blobs, if more than $N$ blobs are found, the $N$ largest are elected as valid markers. For the calibration and test flows, $N = 3$ and $N = 4$, respectively.

The markers are considered collinear if all centroids are less than 1 pixel away from the line fit estimated by the least-squares method. Moreover, the image is discarded if there are no three collinear blobs, if less than $N$ blobs are detected, or if the collinear blobs' radiuses is higher than the distance between them (e.g. obstruction). Note that, in phases 2 and 3, the coordinates of the markers must be undistorted before verifying collinearity. The pseudocode for the final marker segmentation algorithm is depicted in Algorithm 1.

In phase 3, the image is processed onboard the Raspberry, a constrained computing environment. So, instead of masking, the image is cropped to reduce the bytes the blob detector inspects. The number of filters of the blob detector was also reduced. Then all coordinates of the visible centroids are sent to the server, along with the image's PTS.

Two python instances run on the Raspberry Pi simultaneously (see Figure 3.24). When the trigger timestamp is reached, the YUV stream is opened, and the pictures are stored as bitmaps with their PTS as the filename in the RAM shared memory folder. The image is then sent to the processing coroutine, which transmits the coordinates of the centroids, the PTS, and the superior left coordinates of the cut mask to the server. The use of a coroutine

**Algorithm 1** Marker segmentation and centroid extraction

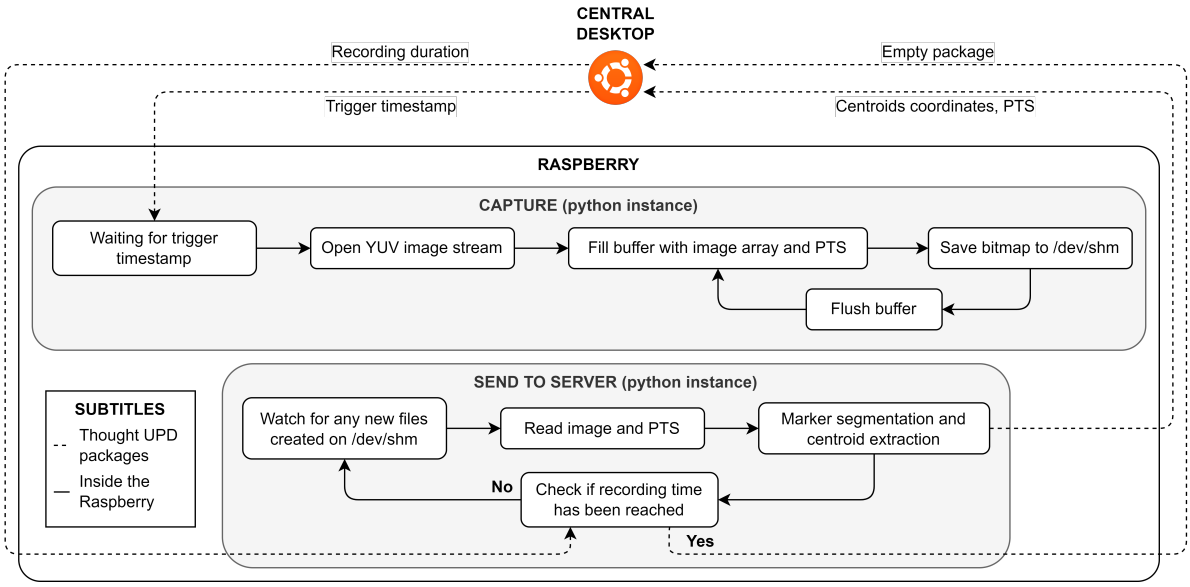
```
I ← captured image matrix with size M × N
T ← high threshold value
maxT, minT, s ← maximum, minimum and step of the set of threshold filters
for i in range(M) and j in range(N) do
  if I[i, j] > H then
    save coordinate to a list L
  end if
end for
C1, C2 ← minimum and maximum coordinates in L
for i in range(M) and j in range(N) do
  if [i, j] > C1 and [i, j] < C2 then
    add I[i, j] to a new image array i
  end if
end for
i ← image after cut mask with size m × n
t = minT
while t < maxT do
  for i in range(m) and j in range(n) do
    if i[i, j] > t then
      add [i, j] to a list K
    end if
  end for
  for u, v in K do
    M_pq = Σ_u Σ_v u^p v^q I(u, v)
  end for
  [x̂, ŷ] = [M10 / M00, M01 / M00]
  out ← list of x̂, ŷ if present in previous threshold filter
  t+ = s
end while
N ← minimum number of markers required
if size(out)> N and all coordinates then
  return out
end if
```

instead of threaded or multi-processing code allows for asynchronous execution, which has no effect on the FPS. There is no verbose to prevent frame loss.

### Marker ordering

After segmentation, the points must be correctly matched between simultaneous images of each camera. The ordering process also differs between the offline and real-time flows, as the camera's height in these setups changes.

**Figure 3.24:** Schematic of Raspberry flow. One python instance is in charge of waiting for the timestamp trigger, as well as opening and storing the picture stream to the RAM shared memory folder. In the background, separate python instance is checking to see whether there are any new photos in the shared memory folder. If yes, the photos are analyzed, and the markers' centroids are provided to the server along with the PTS.



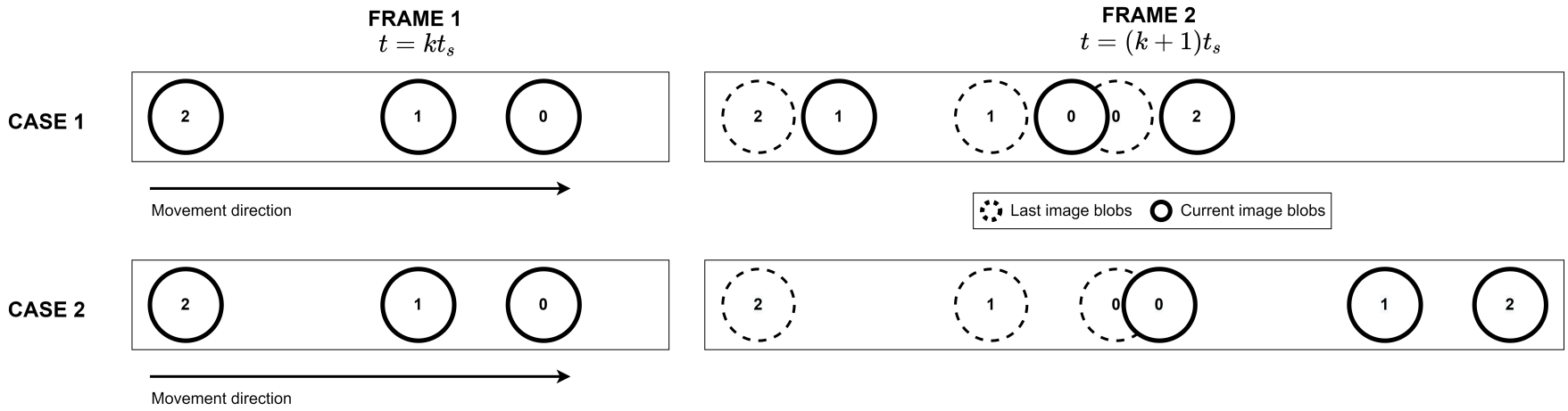

Source: Prepared by the author.

For the first and second phases, the wand was moved at a pendulum movement around a 1-meter radius semicircle. Therefore, the blobs on the first image of each camera stream can be ordered by rising $x$ or $y$, depending on which axis they were further apart. The same order is applied to the other $N-1$ views, considering the known disposition of cameras. For instance, the order must be inverted between views if the cameras face each other.

The markers in the following images were then arranged by their closeness to the blobs in the latest image. Invalid images have no visible collinear markers or are asynchronous to the other camera. If more than ten consecutive photos are invalid, the next image is considered the first in the sequence.

Ordering problems may arise in frame rates lower than 40 FPS and fast-moving markers. As illustrated in Figure 3.25, if the wand moves in the opposite direction of the markers ascending ordering, comparing to the last image blobs may result in an incorrect association. So our system assures whether the centre blob is labelled as "1" or if they are arranged in the same ascending/descending order like the last picture. If not, the image is marked as the first in the valid sequence.

**Figure 3.25:** Example of erroneous ordering when utilising the proximity method. In both cases 1 and 2, the wand is travelling in the opposite direction of the markers' ascending sequence. The output ordering will be incorrect if the nearest blob to the previous "0" marker is in the center (case 1) or on the opposite end of the wand (case 2).




Source: Prepared by the author.

In the third phase, the server stores each received package, but only if at least three separate blobs are visible and collinear. Similarly to the first and second phases, the server organises the following images' markers based on their closeness to the previous image's markers. When the markers from subsequent images from the same camera stream are correctly related, the 2D location of each marker is then interpolated over 100 FPS.

The cameras were also placed 2.6 meters above the ground, the maximum distance allowed by the laboratory ceiling. This height increases working area coverage taking into account the narrowing of the FOV when utilising rapid camera settings. Nevertheless, as shown in Figure 3.26, the blobs can no longer be associated by $x$ or $y$ ascending order. The calibration and test flow employ distinct strategies to solve this problem.

In the calibration flow, the markers' 3D location is not required in real-time. As a result, the 2D coordinates from different cameras are matched only after the capture. To correctly order the markers, it is checked if the distance between the three markers follows the proportion 2:1 (similar to the wand depicted in Figure 3.2a). If yes, the order 0-1-2 is associated to the markers A-B-C (see Figure 2.4) and backtracked per proximity to the markers of the first valid picture of the sequence (see Figure 3.27). The time intervals when the sequence of the markers is known were then crossed between the pair cameras. The markers in these crossing periods were finally interpolated at 100 FPS and used to generate the fundamental matrix. The pseudocode for the marker ordering algorithm during calibration is depicted in Algorithm 2.

In the test flow, the fundamental matrix is already known. So epipolar lines can be used to relate the markers on the first valid images in each camera's stream[27]. The epipolar line coefficients for a pair of cameras are computed such as in Equation 2.8. Note that $l = \boldsymbol{F}^T \tilde{\boldsymbol{x}}'$ is used to compute line $l$ on the first camera plane.

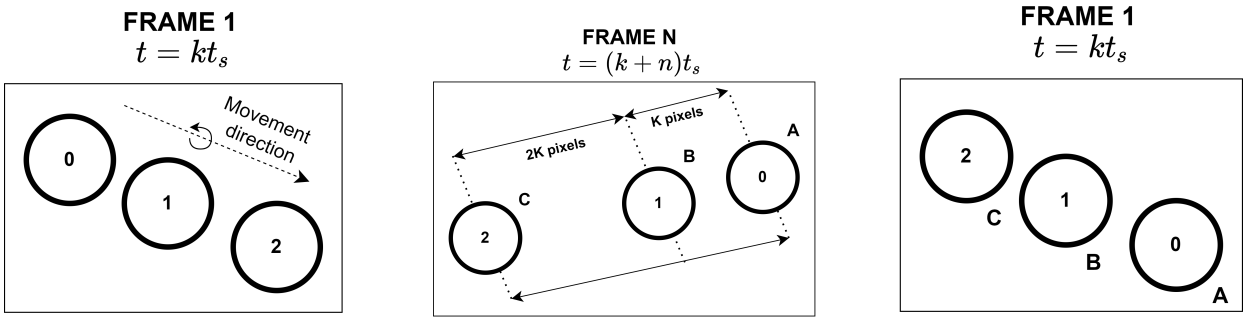
Then, the distance between each epipolar line and the markers in every possible permutation is computed. In Figure 3.28, three epipolar lines in the right camera were plotted, connected to each of the left image markers. Then, in the right picture, the distances between these epipolar lines and the markers in all possible combinations were added. The combination with the least sum was chosen as the proper relationship between each cameras' markers. The mean frequency achieved for this ordering loop was 200 Hz. The pseudocode for the marker ordering algorithm during testing is depicted in Algorithm 3.

**Figure 3.26:** Picture of the wand at 1.1 meter from the ground plane and parallel to the floor by the left (a) and right (b) cameras, which are 2.6m from the ground. Note that, due to the perspective, the markers can not be matched between cameras using the father left, right, up or down pixels. Since the spheres' arrangement on the wand is known, the markers can be associated via the distance between the centroids. The pedestal and arena floor were visible in these images because they were recorded with maximum analog and digital gain.

**(a)** Left camera **(b)** Right camera

Source: Prepared by the author.

**Figure 3.27:** Example of markers ordering correction when using the forth-back in time approach. The blobs are ordered arbitrarily in the first image (a), with the blob "1" always serving as the central marker. The markers in the subsequent images are then arranged by proximity until the wand achieves a configuration that validates the placement of the blobs A-B-C (2:1 proportional distance between the extremities markers) as depicted in (b). Finally, the blobs "0" and "2" of the stored dataset are flipped until the first valid image of the stream is reached. (c). This sequence is maintained for the subsequent images using the proximity approach.



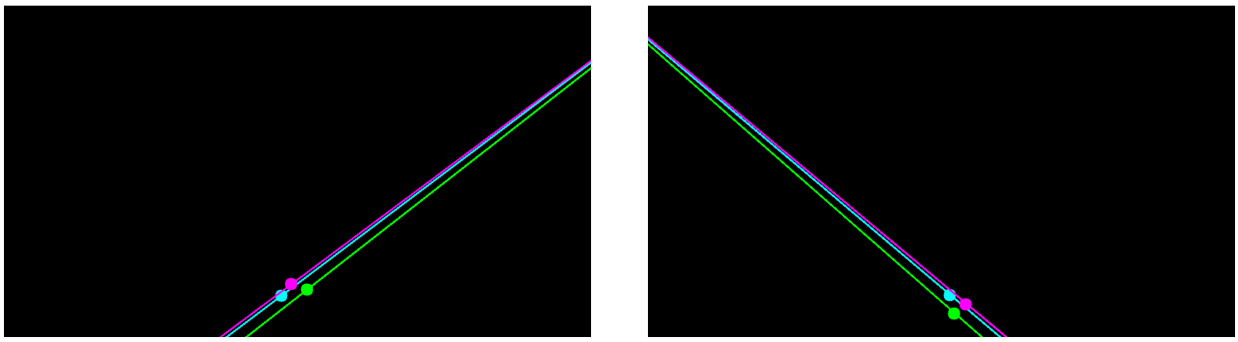

**(a)** First image of the valid time interval with the original marker sequence.

**(b)** Image in which the marker order is confirmed. Note that the numerical order was inverted.

**(c)** Correction of the order of the markers in the first image.

Source: Prepared by the author.

**Figure 3.28:** Picture of the epipolar lines between two camera images. The markers and the corresponding lines can be linked by colour between the views (a) and (b). The lines in (b) were projected from the markers in (a) of the same colour, and vice versa.. The wand was placed on the ground, and the markers were arranged according to their closeness to the epipolar lines. Note that the cyan and magenta markers are on collinear epipolar lines. This uncertainty was resolved by ordering these markers per ascending $x$ coordinate.



**(a)** Left camera

**(b)** Right camera

Source: Prepared by the author.

**Algorithm 2** Centroid ordering in calibration

```
K, d ← intrinsics matrix and distortion coeficients
inv = 0 ← number of invalid images
k = False ← known markers order
for each captured image do
  ts ← timestamp of the capture image
  N ← image number sent by the capture station
  C ← list of distorted centroids coordinates
  U ← list of undistorted centroids coordinates using K and d
  if N > lastN + 1 then
    inv+ = N − lastN
  end if
  if U are collinear and different coordinates then
    if N > 0 then
      compare U to last image's U and order markers per proximity
    end if
  else
    wait for next image
  end if
  if k = False and ratio of coordinates in U is 1:2 and inv < 10 then
    k = True
    reorder A-B-C to 0-1-2 in all previous images
  end if
  lastN = n
  inv = 0
end for
interpolate the ordered marker using cubic spline
out ← list of ordered and interpolated markers
return out
```

There is some ambiguity when two or more markers in the left image project almost collinear epipolar lines on the right image and vice-versa. However, because of the rigid construction of the wand, these markers can be linked per increasing $x$ in this case. To save time, this comparison analysis is executed only with ambiguous markers. Like calibration, the subsequent images are ordered by proximity to the preceding image's centroids. If more than ten invalid packages are received, the valid interval is reset.

### Camera pose estimation

After the fundamental matrix, the essential matrix is computed by Equation 2.11, each camera projection matrix $\mathbf{P}$ is estimated and the scale $\lambda$ is found. As $||\mathbf{t}|| = \mathbf{1}$, this

**Algorithm 3** Centroid ordering in testing

```
K, d ← intrinsics matrix and distortion coeficients
inv = 0 ← number of invalid images
for each captured image do
  ts ← timestamp of the capture image
  N ← image number sent by the capture station
  C ← list of distorted centroids coordinates
  U ← list of undistorted centroids coordinates using K and d
  if N > lastN + 1 then
    inv+ = N − lastN
  end if
  if U are collinear and different coordinates then
    if N > 0 then
      compare U to last image's U and order markers per proximity
    end if
  else
    wait for next image
  end if
  if N > 10 and inv < 10 then
    interpolate last 10 available ordered markers
    if any other camera has already submited markers in the window timeframe then
      compute and order markers epipolar lines
      triangulate 3D markers' position
    end if
  end if
  lastN = n
  inv = 0
end for
```

scale must represent the Euclidean distance between the cameras while also accounting for compensations for focal length calibration errors. The 3D position of each tag is then triangulated in scale, and in phase 3, the images whose reconstructed distance $L_{AC}$ is 1% larger than the actual distance are eliminated as outliers[32]. Finally, the fundamental and essential matrices are recalculated using the inline markers, and the final rotation and translation between cameras are obtained. Using matrix multiplication, these relative poses may be turned absolute (i.e. regarding the 1st camera). A bundle adjustment is recommended to enhance extrinsics calibration when employing a large number of cameras.

The ground plane must now be rotated to the XZ plane, and the world's origin must be translated to precisely below the 1st camera and coplanar to the XZ plane (see Figure 3.29). To capture a valid dataset, the triangular wand was placed on the ground with three spheres

over the triangle vertices. As long as the wand remains still, any image from each stream may be used to triangulate the 3D position of the markers without interpolating. Because the fundamental matrix is known, the epipolar lines were employed to relate the markers between the cameras.

The equation of the plane that goes across the three rebuilt 3D markers was defined as $ax + by + cz + d = 0$, for $(a, c) \in \mathbb{Q}^+$. Firstly, the vector $\boldsymbol{v} = [a, b, c]^T$ was translated to the same origin as the vector $\boldsymbol{k} = [0, 1, 0]^T$ orthogonal to the plane XZ. Therefore, the equation of the new translated plane by the vector $\boldsymbol{t}_P = [0, +d/b, 0]^T$ is $ax + by + cz = 0$. Secondly, the rotation angle $\theta$ and the rotation axis $\boldsymbol{u}$ were then computed as:

$$\boldsymbol{u} = \begin{bmatrix} u_1 & u_2 & u_3 \end{bmatrix}^T = \frac{\boldsymbol{v} \times \boldsymbol{k}}{|\boldsymbol{v} \times \boldsymbol{k}|}\ ,\ \theta = \arccos\left(\frac{\boldsymbol{v} \cdot \boldsymbol{k}}{|\boldsymbol{v}|}\right) \tag{3.3}$$

The rotating matrix $Q_P(\theta, \boldsymbol{u})$ is then computed as:

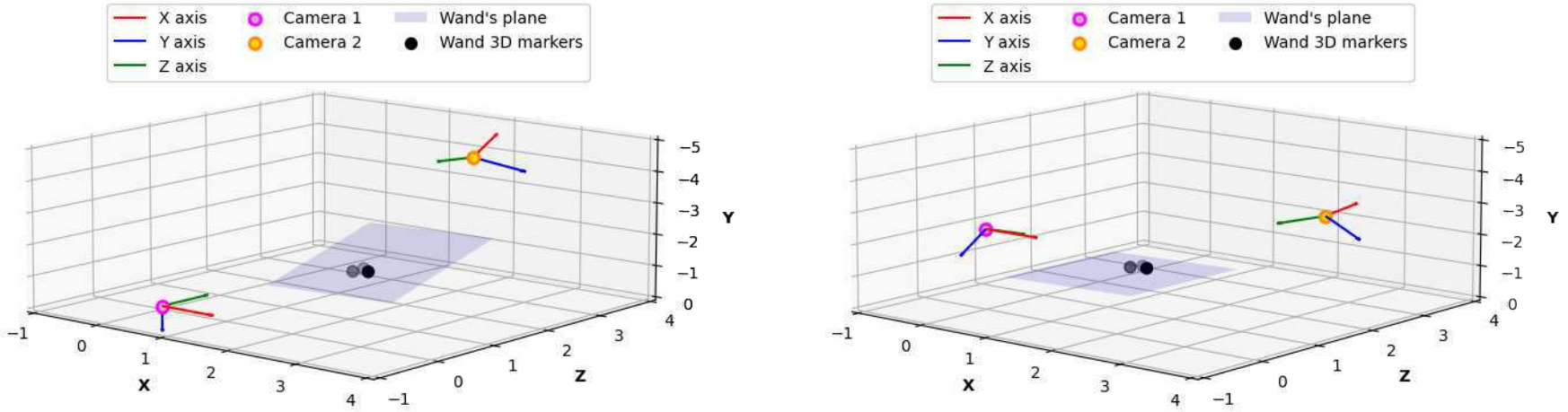
$$\boldsymbol{u} = \begin{bmatrix} \cos\theta + u_1^2(1-\cos\theta) & u_1u_2(1-\cos\theta) - u_3\sin\theta & u_1u_3(1-\cos\theta) + u_2\sin\theta] \\ u_1u_2(1-\cos\theta) + u_3\sin\theta & \cos\theta + u_2^2(1-\cos\theta) & u_2u_3(1-\cos\theta) - u_1\sin\theta] \\ u_1u_3(1-\cos\theta) - u_2\sin\theta & u_2u_3(1-\cos\theta) + u_1\sin\theta & \cos\theta + u_3^2(1-\cos\theta) \end{bmatrix}$$

This translation and rotation procedure is performed not just with the plane but also with each camera pose and the point cloud of the 3D markers. As last, the system is translated so that it has the origin is directly below the 1st camera.

## 3.3 Final considerations

This chapter initially described the calibration wand, scenarios setup and reflective markers used in the proposed IPS. We then defined the architecture and hardware used in three experimental phases: the first phase with RealSense cameras, where only extrinsics calibration was done with centralised image processing; a second phase with Raspberry V1 NoIR cameras, with centralised image processing while with both intrinsic and extrinsics calibration; and a final phase with Raspberry V2 NoIR cameras, accompanied by intrinsic and extrinsics calibration and decentralised image processing. The final proposed system is heterogeneous and asynchronous since we employ different Raspberry Pi as capture platforms.

**Figure 3.29:** Reconstructed 3D camera poses and point cloud from a pair of cameras dataset using a triangle wand on the ground plane. In (a), the origin of the coordinate system is equivalent to the first camera's axes, with the Z axis piercing the picture plane. In contrast, in (b), the wand's plane and the cameras' 1 and 2 axes have been adjusted to match the XZ ground plane. Take note that the system's origin is now exactly below camera 1.




**(a)** Before ground plane calibration

**(b)** After ground plane calibration

Source: Prepared by the author.

The PTP protocol and an interpolated database were used to virtually synchronise each Raspberry's different FPS and trigger response. Following, the threshold filters, centroid search techniques and tracking issues caused by fast-moving markers were addressed, leading to the development of an innovative and generic sorting system. During calibration, the order of the markers is validated when the wand's physical disposition in space is known. In contrast, the ordering was found in real-time (about 200 Hz) during testing using an epipolar line comparison technique. The next chapter will present the experiments' results and discussion for each phase.

# Chapter 4

# Experimental results and discussion

This chapter will discuss the results obtained through all three phases experiments. In the first phase, a RealSense camera is used, while in the second and third phases, Raspberry cameras are used. During stages 1 and 2, processing is centralised and the architecture is homogeneous; however, in phase 3, the system is heterogeneous and processing is distributed.

## 4.1 Phase one experiments

The RealSense setup was used to test different sizes of markers, batons and reconstruction flow. The auto exposure, auto white balance, and laser projection were disabled. The exposure was decreased by half (from 30ms to 15 ms), increasing the gain three times (from 30 to 90). Each configuration was separately sent to each camera during the warm-up period before capturing. One may also upload these configurations through a JSON file. The calibration parameters used in the pinhole model (Equation 2.1) are depicted in Table 4.1.

The results achieved from the different marker sizes and baton width are shown in Table 4.2. Note that the cameras were repositioned; thus, the measured real distance between cameras differs between datasets. The results were double-checked with the MATLAB image processing toolbox and are equal up to four decimal cases. All cameras were positioned around 1.5m above the ground and perpendicular to the floor plane, as depicted in Figure 3.6. Each operation step shown in Table 4.2 may be evaluated independently.

Larger spheres reflect more light and are brighter than the background. Therefore, the

**Table 4.1:** Intrinsic calibration parameters used in the three experimental phases. For phase 1, the RealSenses were calibrated using the Intel SDK and a pinhole model like Equation 2.1. In the second and third phases, the generic camera model[36] was employed such as in Equation 2.5. Note the difference in image resolution.

| **Camera type** | **Resolution (pixel$^2$)** | **Camera ID** | $\alpha$ (pixel) | $\beta$ (pixel) | $u_0$ (pixel) | $v_0$ (pixel) | $k_1$ | $k_2$ | $k_3$ | $k_4$ |
|---|---|---|---|---|---|---|---|---|---|---|
| **RealSense** | 640 × 480 | **Camera 1** | 626.744 | 626.744 | 307.936 | 242.777 | - | | | |
| | | **Camera 2** | 626.744 | 626.744 | 307.936 | 242.777 | | | | |
| **V1 NoIR** | 640 × 480 | **Camera 1** | 816.188 | 814.325 | 318.382 | 250.263 | -0.292 | 0.199 | 0.387 | -6.514 |
| | | **Camera 2** | 684.117 | 683.027 | 313.435 | 262.314 | -0.202 | 0.657 | -1.644 | 0.418 |
| **V2 NoIR** | 960 × 640 | **Camera 1** | 720.313 | 719.521 | 481.014 | 360.991 | 0.396 | 0.634 | -2.417 | 2.111 |
| | | **Camera 2** | 768.113 | 767.935 | 472.596 | 350.978 | 0.369 | 1.501 | -7.941 | 11.917 |
| | | **Camera 3** | 728.237 | 729.419 | 459.854 | 351.59 | 0.276 | 2.095 | -9.980 | 14.192 |
| | | **Camera 4** | 750.149 | 748.903 | 492.144 | 350.213 | 0.401 | 1.160 | -7.103 | 11.415 |

Source: Prepared by the author.

**Table 4.2:** Results for the calibration in phases 1, 2 and 3 between cameras 1 and 2.

| | | Phase 1 | | | | | | Phase 2 | Phase 3 | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Operation** | **Parameter** | **Dataset 1** | **Dataset 2** | **Dataset 3** | **Dataset 4** | **Dataset 5** | **Dataset 6** | **Dataset 7** | **Dataset 8** | **Dataset 9** |
| **Setup config.** | **Marker size** | 55 mm | 35 mm | 25 mm | 20 mm | 15 mm | 10 mm | 20 mm | 20 mm | 20 mm |
| | **Wand width** | 50 cm | 50 cm | 15 cm | 15 cm | 15 cm | 15 cm | 15 cm | 15 cm | 15 cm |
| | **Total images** | 999 | 900 | 999 | 1000 | Not visible | | 4156 | 5395 | 3923 |
| **Segmentation** | **Bounding box threshold** | 150 | 150 | 127 | 100 | | | 127 | 127 | 127 |
| **Marker detection** | **Min. threshold** | 0 | 0 | 0 | 0 | | | 0 | 0 | 0 |
| | **Max. threshold** | 150 | 150 | 200 | 200 | | | 200 | 151 | 151 |
| | **Threshold step** | 1 | 1 | 1 | 1 | | | 1 | 50 | 50 |
| | **Min. convexity** | 0.9 | 0.9 | 0.95 | 0.95 | | | 1 | - | - |
| | **Min. distance between blobs** | 10 pixels | 0 pixel | 0 pixel | 0 pixel | | | 0 pixel | 0 pixel | 0 pixel |
| | **Min. blob area** | 10 pixels$^2$ | 5 pixels$^2$ | 2 pixels$^2$ | 2 pixels$^2$ | | | 2 pixels$^2$ | 2 pixels$^2$ | 2 pixels$^2$ |
| | **Min. blob repeatability** | 10 | 20 | 20 | 20 | | | 20 | 3 | 3 |
| **Results** | $L_{AC}$ | 48 cm | 48 cm | 15.7 cm | 15.7 cm | | | 15.7 cm | 15.7 cm | 15.7 cm |
| | $\|\boldsymbol{X}_j^A - \boldsymbol{X}_j^C\|$ **mean** | 48.0388 cm | 47.9940 cm | 15.6587 cm | 15.7188 cm | | | 15.7099 cm | 15.6591 cm | 15.6429 cm |
| | $\|\boldsymbol{X}_j^A - \boldsymbol{X}_j^C\|$ **sdt. dev.** | 0.2720 cm | 0.2282 cm | 0.1825 cm | 0.2658 cm | | | 0.1296 cm | 0.0771 cm | 0.1746 cm |
| | $L_{BC}$ | 29 cm | 29 cm | 10.3 cm | 10.3 cm | | | 10.3 cm | 10.3 cm | 10.3 cm |
| | $\|\boldsymbol{X}_j^B - \boldsymbol{X}_j^C\|$ **mean** | 29.1605 cm | 28.9773 cm | 10.1771 cm | 10.351 cm | | | 10.3333 cm | 10.0905 cm | 10.0190 cm |
| | $\|\boldsymbol{X}_j^B - \boldsymbol{X}_j^C\|$ **sdt. dev.** | 0.1709 cm | 0.1629 cm | 0.1228 cm | 0.1484 cm | | | 0.0672 cm | 0.0713 cm | 0.1357 cm |
| | $L_{AB}$ | 19 cm | 19 cm | 5.4 cm | 5.4 cm | | | 5.4 cm | 5.4 cm | 5.4 cm |
| | $\|\boldsymbol{X}_j^B - \boldsymbol{X}_j^A\|$ **mean** | 18.8824 cm | 19.0187 cm | 5.4831 cm | 5.3728 cm | | | 5.3807 cm | 5.5761 cm | 5.6249 cm |
| | $\|\boldsymbol{X}_j^B - \boldsymbol{X}_j^A\|$ **sdt. dev**. | 0.1234 cm | 0.0868 cm | 0.0713 cm | 0.1294 cm | | | 0.0689 cm | 0.0563 cm | 0.0770 cm |
| | **Real distance between cameras** | 270 cm | 424 cm | 425 cm | 423 cm | | | 425 cm | 428 cm | 428 cm |
| | **Computed** $\lambda$ | 282.62 cm | 419.88 cm | 426.08 cm | 419.30 cm | | | 426.05 cm | 400.10 cm | 403.06 cm |
| | **Discarded images** | 3 | 1 | 1 | 17 | | | 0 | 2947* | 1446* |

∗Pictures dropped during the fundamental matrix refining

Source: Prepared by the author.

threshold for creating the bounding box may increase when the size of the markers expands. Note that different markers require distinct configurations of the marker detector.

The image passed to the marker detector described in chapter 3 is inverted (markers are black, and the background is white). Hence, the minimum threshold ($T_m$) for segmenting the blob is zero (pure black), and the maximum ($T_M$) is inverse to the size of the markers. To be considered a blob, the marker must be present in at least $k$ images from all $(T_M - T_m)/s$ filters, for $s$ the step between filters. When the sphere's diameter increases, the makers are darker, and the background is whiter, enabling lower threshold values to identify the tag.

In this work, convexity is defined as the ratio of the blob and the convex hull area. The smaller the diameter, the more alike a circle the marker is. Hence, larger markers must be filtered through lower convexity values than smaller ones. For the 55mm marker, the minimum distance between blobs was increased to avoid double identification.

Observing the computed $\lambda$ and the actual distance between cameras, it is clear that the smaller markers provide more accurate results than the larger diameters. The wider the diameter, the broader the area for selecting the incorrect centroid. So it is harder to estimate the sphere's centre with subpixel precision. This error then propagates to the $\boldsymbol{F}$ matrix estimation and the scale of the 3D reconstruction.

Comparing the standard deviations of the reconstructed distance between markers, the optimal marker size for the triangulation is the 25mm maker. However, the UAV structure supports 20mm spheres at maximum due to the rotating blades vibrations. The 15mm and 10mm markers are too small to be detected with the 30ms exposure.

## 4.2 Phase two experiments

Phase two was initially configured at the RealSense layout, as depicted in Figure 3.6. Similarly to phase one, the camera auto white balance and exposure was disabled.

While Deng et al.[21] reports an reprojection error of 0.05 pixel, we achieved a minimum of 0.4 pixels in the intrinsic parameters calibration. This under-performance is due to the quality of the image difference, shown in Figure 4.1. The blur on the pattern introduces noise to the results. This blur could be improved using a lens with a narrower shutter, yet the cost of the camera would rise. Note that the focus was set to the chessboard pattern,

50 cm away from the camera. This focus length must not change to when capturing the datasets. The final calibration parameters used in the generic model (Equation 2.5) are depicted in Table 4.1. The summary of the results is shown in Table 4.2.

**Figure 4.1:** Comparison of 1600% zoom in Deng et al.[21] (a) and the collected dataset (b).

**(a)** Deng et al.[21] **(b)** Our dataset.

Source: Prepared by the author.

Compare the dataset 7 with the fourth dataset of Table 4.2, and the accuracy is greater than the RealSense setup for the same markers. This finding may be explained by the OmniVision sensor's greater sensitivity to light as compared to the RealSense sensor. As a result, a better contour of the blob is captured and a more accurate centroid is found.

## 4.3 Phase three experiments

In phase three, the setup was initially tested using two V2 cameras linked to two Raspberries 4 8GB. In a second step, two more cameras were connected to a Raspberry 3b+ and a Raspberry 3. The chosen resolution was 960×640 pixels comprising the full sensor extent. The cameras were placed such as in in Figure 3.8.

In the calibration of the intrinsic parameters for the V2 cameras, a low reprojection error of 0.35 pixels was achieved. However, the V2 camera has a broad depth of view that focuses on everything more than one meter away from the camera. As a result, the calibration pattern must be held away from the camera and cannot cover the entire sensor in a single shot. It is thus not expected for $\lambda$ to equal the distance between the cameras since this scale compensates for the error in the focal distance calibration. A bigger calibration pattern may be preferred to attain a better outcome. The final calibration parameters used in the generic model (Equation 2.5) are depicted in Table 4.1.

The IR led rings used in phase three have a range of 4 meters, which is 1 meter longer than the IR reflectors attached to the V1 cameras. So the blobs are brighter, and sensor

output contrast may be increased to 100% without gathering noise (such as clothes or wand pieces). Because the Raspberry Pi has limited computing power, the circularity filter was disabled to save time.

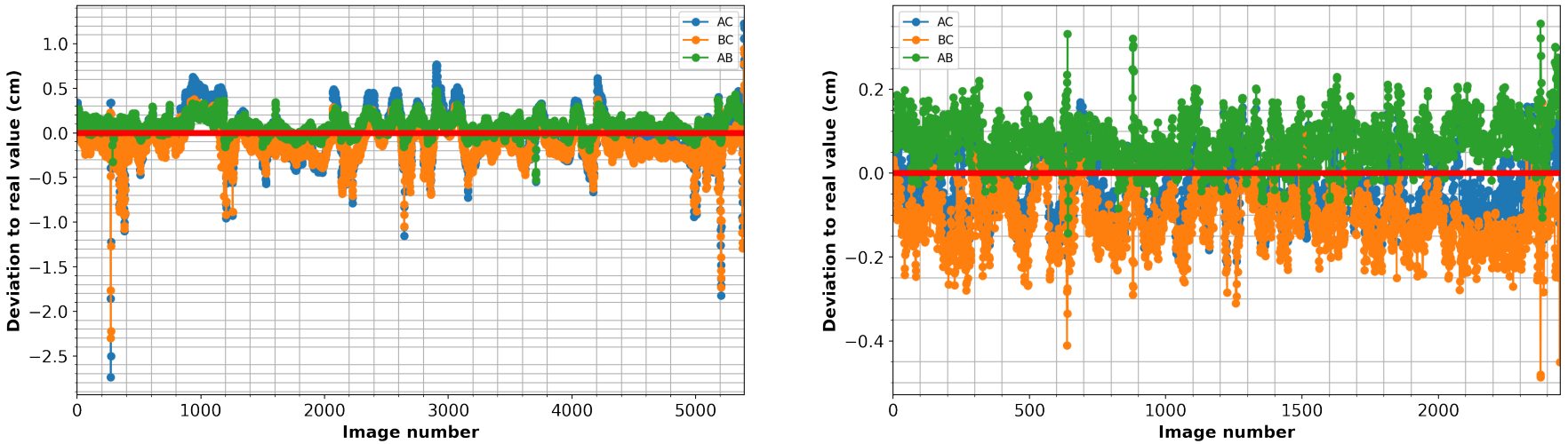
Unlike the first and second phases, the invalid images in phase three account for those eliminated as outliers for the fundamental matrix refinement. In dataset 8, for example, the entire capture length was 90 seconds at 40 frames per second for two cameras, yielding 3600 pictures per camera. Specific motions, however, obscured the display of all the blobs, and only 2717 and 2596 packages with at least one blob were received from each Raspberry. Following that, due to temporal mismatches or non-collinear centroids, 166 and 351 photos from camera 1 and 2 streams were removed, respectively. Then, at 100 frames per second, 5395 valid interpolation images were generated. Finally, 2947 images were dismissed because their reconstructed 3D markers A and C were more than 1.57 cm apart (or 1% of the actual distance of 15.7 cm, see Figure 4.2). The remaining 2448 images were used to recalculate the fundamental matrix, as stated in Table 4.2. It is worth noting that there was no need to boost the FPS or switch to a lower FOV because a 100Hz interpolated database was discovered, which is half the frequency of the integrated drone controller.

Following that, the ground plane was calibrated. Figure 4.3 depicts the camera's poses after calibration. It is worth noting that the new ground plane equals $y = 0$, and the first camera is at $(0, -h, 0)$, where $h$ is the camera's height above the floor.

The system was then tested using a drone. The Tello DJI can fly steadily with all four markers and one for counterbalancing. However, this drone calculates its height by sending an infrared pulse to the ground, causing the cameras to see its reflection. When the drone is flying, applying the high threshold filter to remove this noise is simple since the markers are brighter than the IR reflection of the ground. Nonetheless, when the drone is near 30 cm of the ground, the marker detector recognises the reflection as a marker and the track of the drone's pose is lost. Because this study does not attempt to address this issue, these pictures are discarded.

The reconstructed position of the drone's central marker during a 60-second test flight with a square 0.5m width and an 80 cm height is shown in Figure 4.4. The X and Z axes in this plot were also adjusted to the front and right of the drone using the two frontal markers, as depicted in Figure 3.1a and Figure 3.1b. It should be noted that the origin of

**Figure 4.2:** Error between the reconstructed positions of the collinear markers and the actual distance between the spheres in dataset 8. Before the fundamental matrix refinement (a), the markers in 5395 pictures were triangulated with a maximum error to the actual distance of 2.5 cm. Inaccurate centroid estimates produce these errors as a result of non-uniform lighting. After removing the outliers (b), only 2448 images remain, all of which had AC markers within 0.5 cm of the actual distance. The offset between the AB and BC mean values demonstrates that the real value utilised is inaccurate.

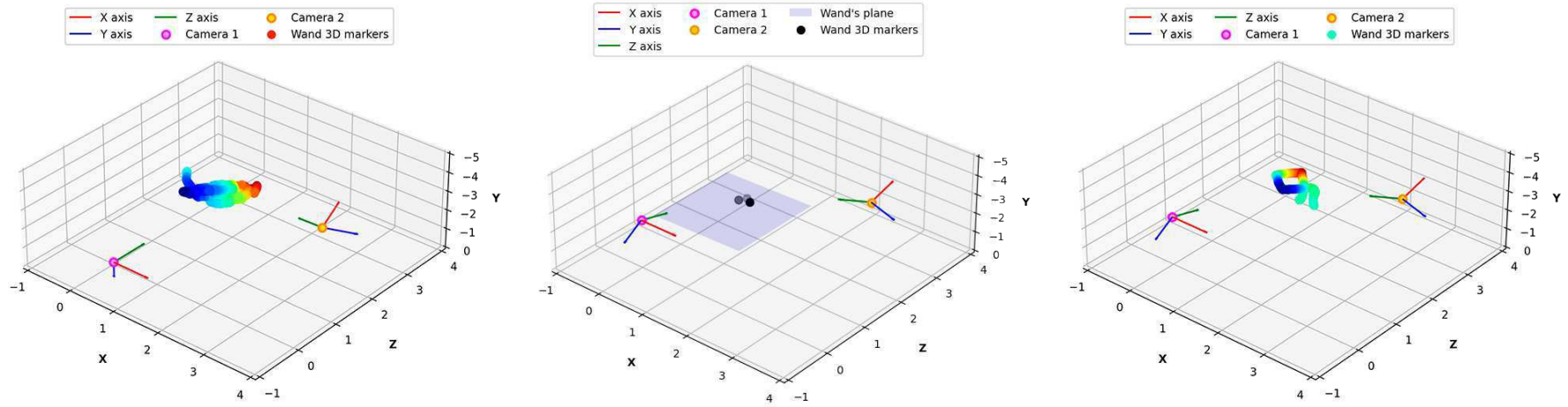



**(a)** Before fundamental matrix refining

**(b)** After fundamental matrix refining

Source: Prepared by the author.

**Figure 4.3:** Reconstructed 3D camera poses and point cloud during extrinsic calibration, ground plane calibration, and testing in dataset 8. The first camera in the pair (left) is the source of the world coordinate system in the extrinsics calibration (a). Following ground plane calibration (b), the wand's plane equals XZ, and the first camera is located at $(0, -h, 0)$, with $h$ being the camera's height above the floor. The system is now ready to capture a drone's movement, as seen in (c) a square 0.5m wide and 80 cm tall.




**(a)** After extrinsic calibration

**(b)** After ground calibration

**(c)** Flight capture

Source: Prepared by the author.

the coordinates is the location of the drone's central marker in the first triangulated image.

Observing Figure 4.4, the measured altitude reaches about 80 cm, and the mean value of the drone's plane around the Y-axis is 88°. Since the embedded estimator's inherent inaccuracy is unknown and the DJI SDK does not provide access to the drones' states, it is unclear if the drone has attained the target height (80 cm) and yaw (90°). Furthermore, because the IR reflection from the rubber surface is brighter than the markers, the sight of the drone near the ground is lost. Nonetheless, the peaks of the angles $\theta$ and $\psi$ correspond to the ramps on the X and Z axes, as the drone revolves around the rotor axis to perform curves. In Figure 4.5, the take-off and landing locations differ, and the square sides are larger than 50 cm probably because of the cumulative IMU error that feeds the estimator.

The system was then tested with all four cameras in the second step. As indicated by Olague and Mohr[19], calibration was performed in pairs with sequential cameras (i.e. cameras 1 and 2, 2 and 3, and so on) since the best results are obtained when the picture planes are 90 degrees apart. In this configuration, the arena is rectangular, with two pairs separated by 4.24m and the other two separated by 4.82m. After the fundamental matrix refinement, the final calibration errors for the three-camera pairs are presented in Figure 4.6. The numerical values obtained between cameras 1 and 2 are displayed in Table 4.2 as dataset 9. Compared to dataset 8, where identical cameras are used, the inverse relationship between the number of pictures used to calibrate and the standard deviation is clear.

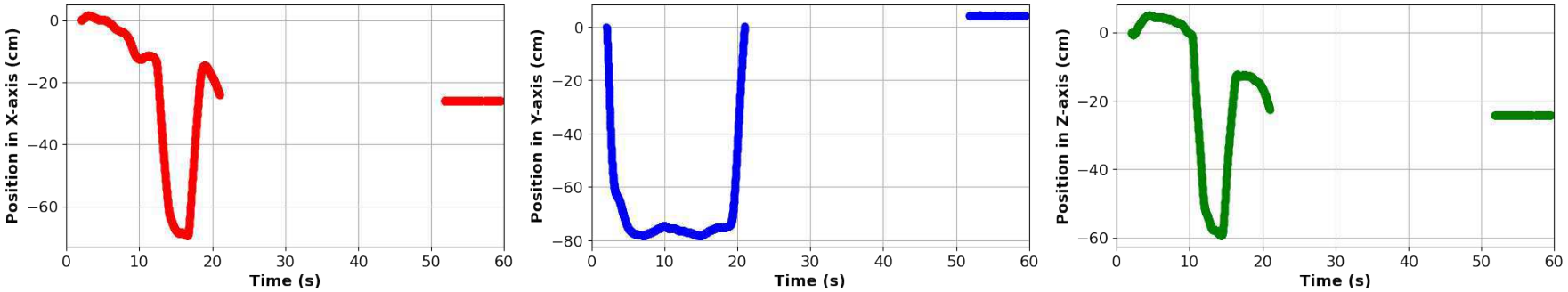
The point clouds from the second and third camera pairs must now be rotated and translated to fit the global coordinate system of the 1st camera. Therefore, the position of camera $k$ in reference to the 1st camera is:

$$\boldsymbol{x}_k = \Pi_{j=1}^{k} \begin{bmatrix} \boldsymbol{R}_j^T & -\boldsymbol{R}_j^T \boldsymbol{t} \\ \boldsymbol{0}_{3\times 1} & 1 \end{bmatrix} \boldsymbol{x}_{j-1} \tag{4.1}$$

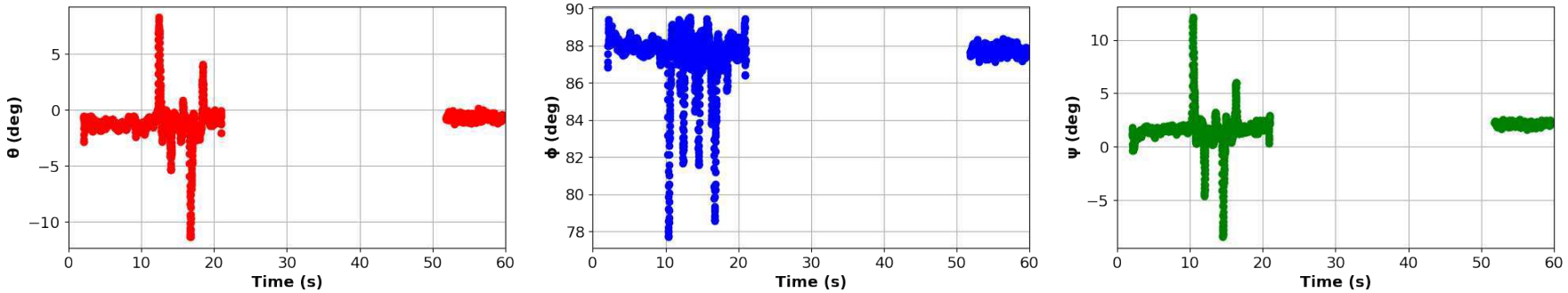
for $\boldsymbol{R}_k$ and $\boldsymbol{t}_k$ achieved by the essential matrix decomposition. Remember that $x_0 = (0, 0, 0)$. The point cloud produced by each pair can be transformed similarly (see Figure 4.7).

The system was tested again with a flight of 0.5m square width and 80cm height. This time, the drone started at the right square's edge. The reconstructed position of the drone's central marker during a 30-second test flight is shown in Figure 4.8. Firstly, it is worth noting a noise in the estimation caused by switching between distant camera pairs, as we

**Figure 4.4:** Drone pose during a 60-second Tello flight using dataset 8 calibration. (a) and (b) depict the position and attitude of the central marker, respectively.



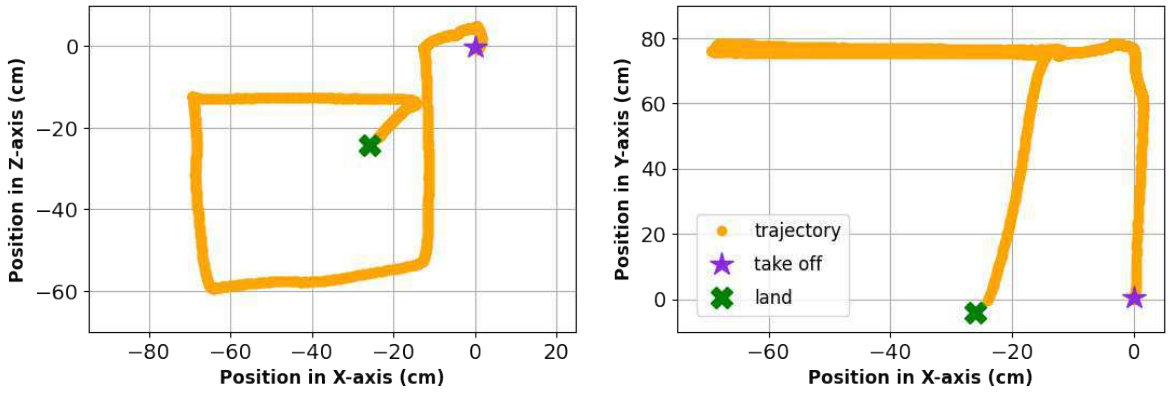

**(a)** Position of the drone in flight




**(b)** Orientation of the drone in flight

Source: Prepared by the author.

**Figure 4.5:** Drone trajectory upper and lateral map during a 60-second Tello flight using dataset 8 calibration.



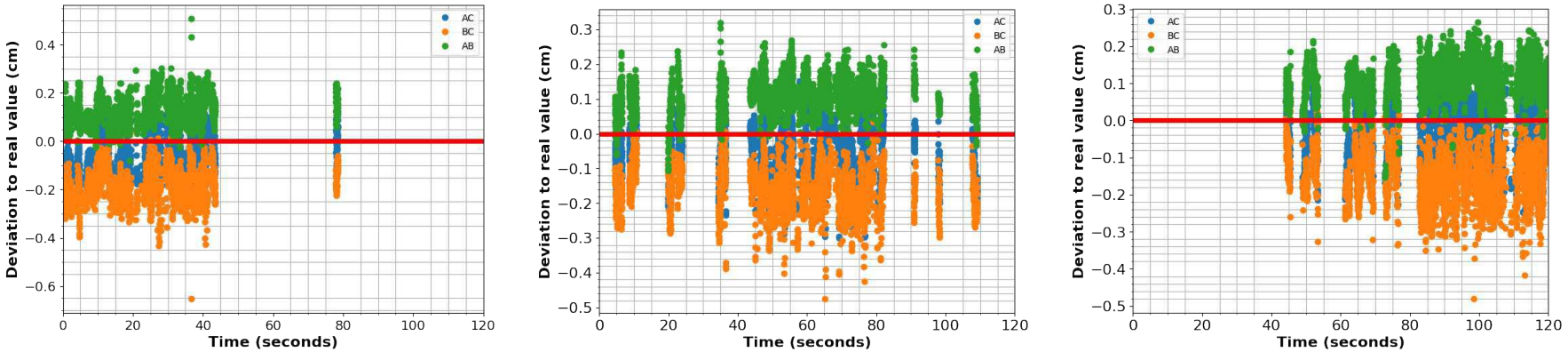

Source: Prepared by the author.

swap between the pairs that send the package the quickest for a given timestamp. The smoother the transition, the closer the pairings swapped were (e.g. between pair 1-2 and 2-3 in comparison to pair 1-2 and 3-4). Systematic errors in the intrinsic calibration might explain the disparity between the pairings (e.g. focal distance or scale). However, the data inaccuracy might be reduced via sensor fusion or a prediction filter before being sent back to the drone's controller. Note that the drone can be monitored even close to the ground because the lighting is uniform throughout the arena.

In Figure 4.9, it is also possible to verify the low altitude, which may have resulted from an inconsistent reflection of the IR laser. An erroneous reading might be the case, given the drone's instability immediately after takeoff (see the upper trajectory map) and that the drone reaches 1-meter altitude before backing down (see the side trajectory map). On the other sides of the square, the drone stabilises at about 60 cm in height.

A third test was then performed using the Amatrol Pegasus II robotic arm to ensure that the optical tracking system's readings were correct. A three-arc trajectory was designed, with one joint moving at a time. The data was then used as ground truth to evaluate the performance of the optical tracking system. The drone was fastened to the claw 2.5 cm away from the tip. This distance was eventually added to the ground truth data as a horizontal radial offset. As depicted in Figure 4.10, to prevent introducing noisy reflections into the optical system, all metallic parts of the arm were covered. The Parrot Mambo was used in this experiment since its body is thinner and than the Tello DJI.

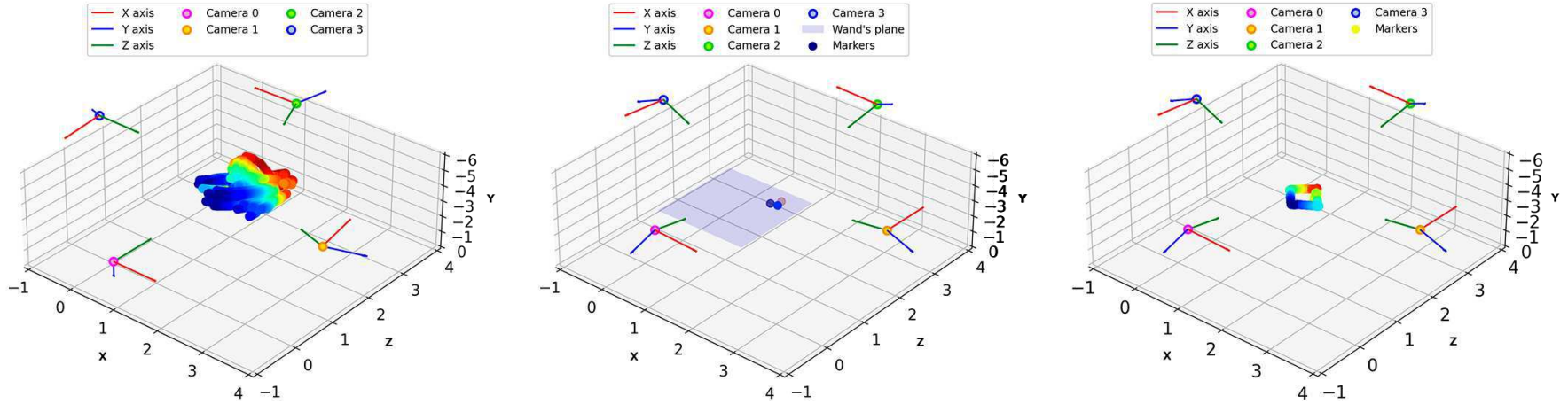
**Figure 4.6:** Error between the reconstructed positions of the collinear markers and the actual distance between the spheres in dataset 9. All errors are less than 0.5 cm after outlier removal. The discrepancy between the AB and BC mean values reveals that the actual value is still incorrect. The Y-axis was displayed in time units to emphasize the overlapping intervals where the wand is visible to both cameras in the pair.




**(a)** Cameras 1 and 2 **(b)** Cameras 2 and 3 **(c)** Cameras 3 and 4

Source: Prepared by the author.

**Figure 4.7:** Reconstructed 3D camera poses and point cloud during extrinsic calibration, ground plane calibration, and testing in dataset 9. The extrinsics calibration results, ground plane calibration and the captured square flight test in the 3D map are shown in (a), (b) and (c).



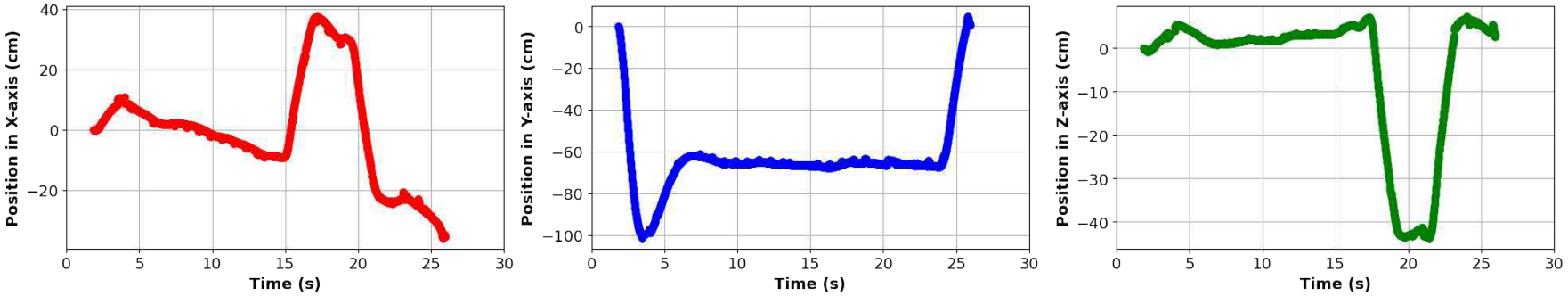

**(a)** After extrinsic calibration

**(b)** After ground calibration

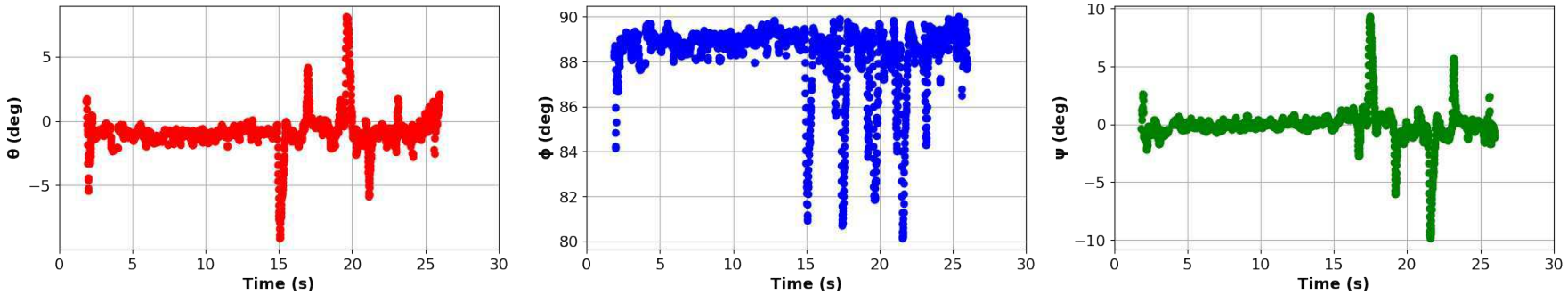
**(c)** Test dataset

Source: Prepared by the author.

**Figure 4.8:** Drone pose during a 30-second Tello flight using dataset 9 calibration. (a) and (b) depict the position and attitude of the central marker, respectively.



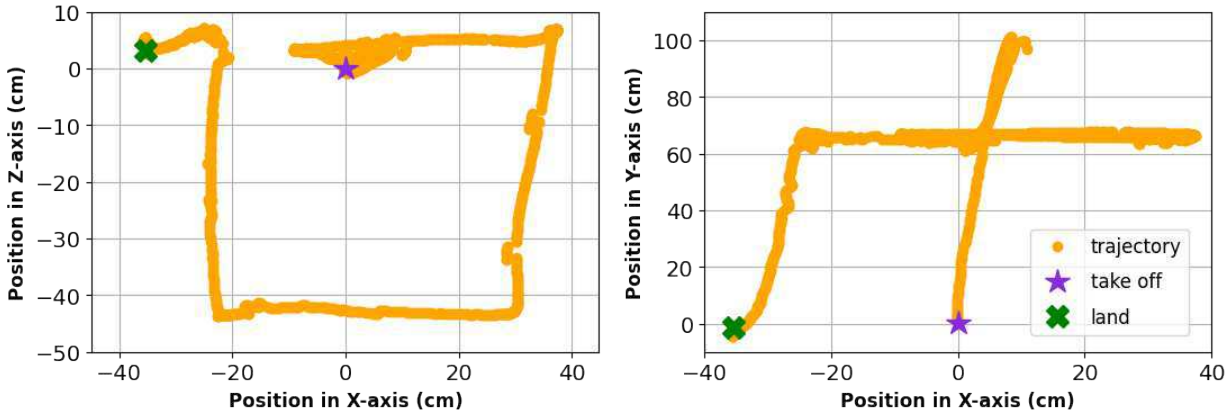

**(a)** Position of the drone in flight




**(b)** Orientation of the drone in flight

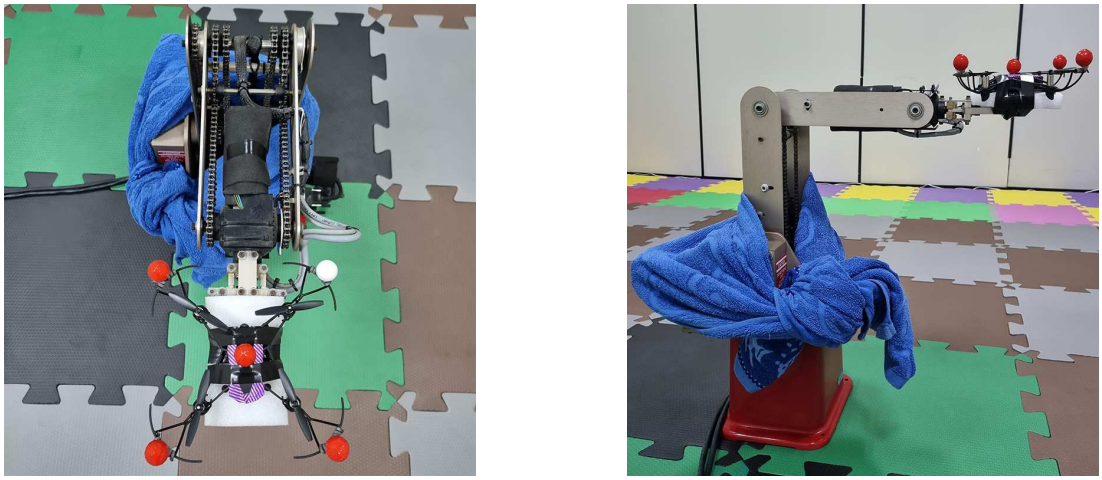
Source: Prepared by the author.

**Figure 4.9:** Drone trajectory upper and lateral map during a 30-second Tello flight using dataset 9 calibration.




Source: Prepared by the author.

**Figure 4.10:** Picture of the Parrot Mambo attached to the Amatrol Pegasus II claw.



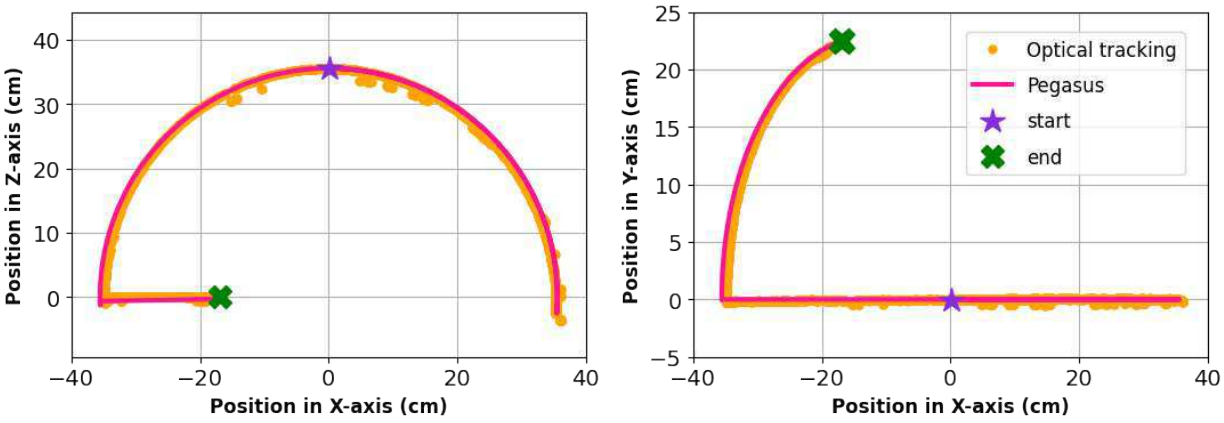
**(a)** Upper view **(b)** Side view

Source: Prepared by the author.

The trajectory arcs were $\pi/2$ to 0 in the XZ plane, 0 to $\pi$ in the XZ plane, and $\pi$ to $5\pi/9$ in the XY plane. Because of the fabric restricting the motors' rotation, the last arc was terminated before $\pi/2$. Both trajectories reconstructed from the Pegasus encoder and the optical tracking system are illustrated in Figure 4.11. When comparing the two trajectories, the divergence is only noticeable when the camera switches between distant pairs.

The numerical error between the optical tracking and the ground truth trajectories are shown in Figure 4.12. Each Pegasus arch was translated in time to match the capture of

**Figure 4.11:** Arm and drone upper and lateral trajectory map during a 40-second test using dataset 9 calibration.



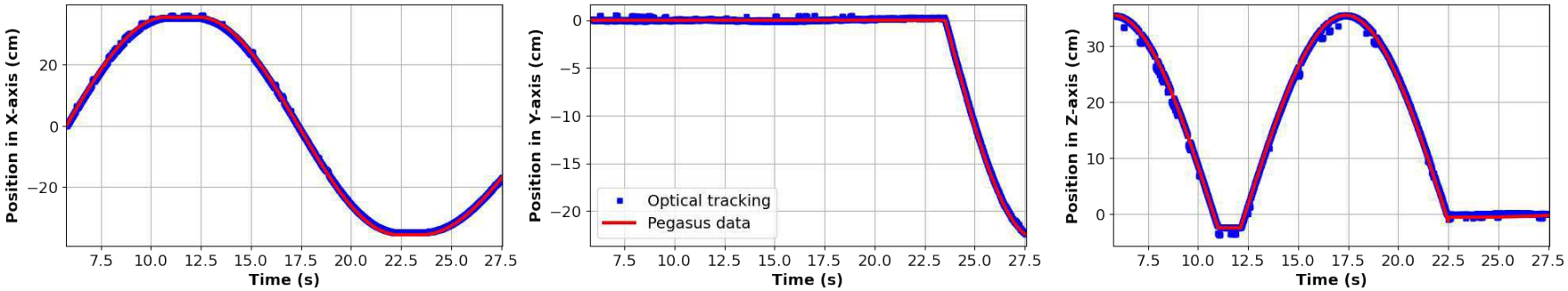

Source: Prepared by the author.

the optical tracking system since the timestamp of its journey was not available. Observing Figure 4.12, it is pertinent to mention that the errors between the two trajectories are within 1 cm in both the X and Y planes and the errors in the Z plane above this limit match when the camera pair switches between 1-2 and 3-4. Hence, subcentimeter accuracy can be assumed if later research use filtering procedures to smooth these curves. The average error was 0.003 cm, 0.074 cm and 0.019 cm for the X, Y and Z axes. Likewise, the standard deviation of the error was 0.513 cm, 0.120 cm and 0.491 cm for the X, Y and Z axes, respectively. It is feasible to say that the Tello DJI flew a trajectory very similar to those shown in Figure 4.5 and Figure 4.9 in the prior testing examples.

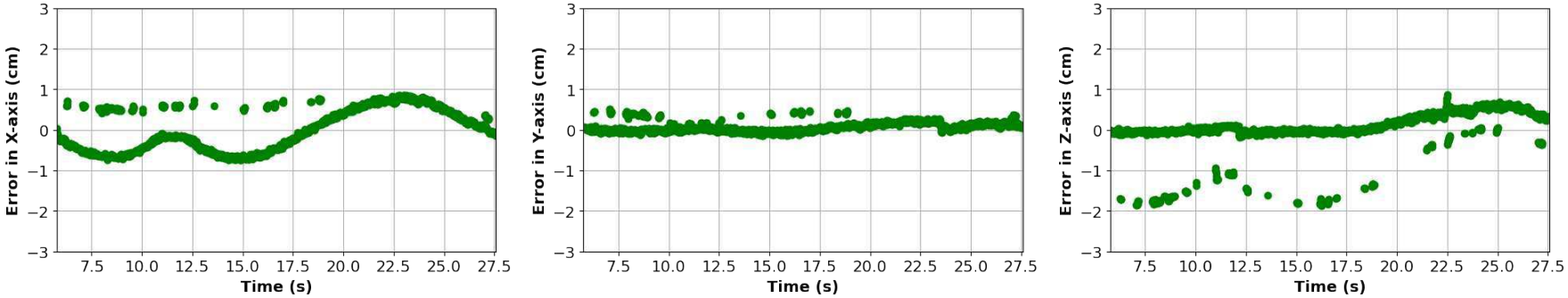
Finally, as the MATLAB SDK gives access to the drone's state space, an end test was performed using the Parrot Mambo. Unfortunately, this non-commercial controller is non-stable while performing a square trajectory, owing to unsatisfactory lateral position estimation and inadequate controller tuning. Therefore, the flight consisted of a takeoff, a hover, and a landing with no lateral movement.

Figure 4.13 depicts the reconstructed poses of the drone's central marker using optical tracking and data collected from the estimator after the flight. Both plots are aligned to associate the X and Z axes to the right and front of the drone, respectively. The Mambo computes displacement in the X and Z axes using an optical flow sensor, while altitude is estimated using a barometer and an ultrasonic sensor fusion. When the two trajectories

**Figure 4.12:** Drone and arm pose during a 40-second test trajectory. (a) and (b) depict the position and error of optical tracking system and the Pegasus encoder.




**(a)** Position of the drone in flight

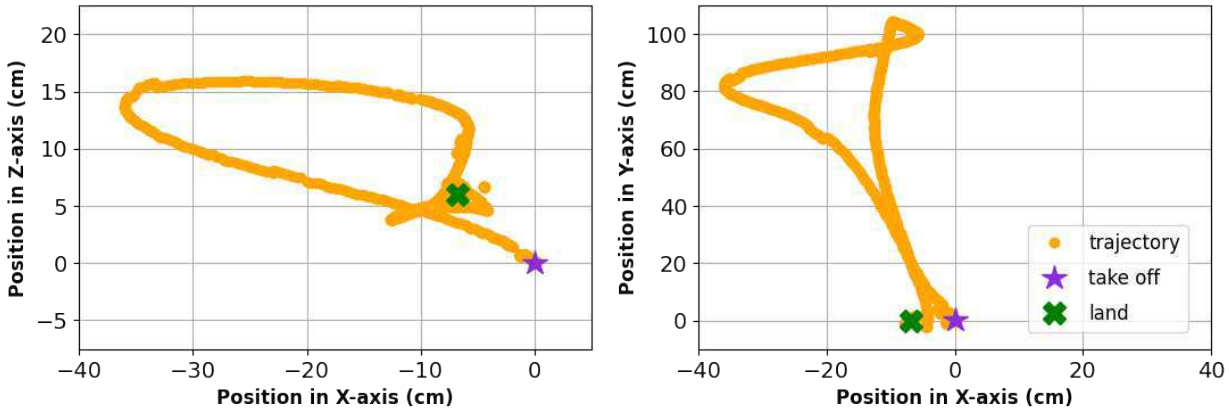



**(b)** Error between the two trajectories

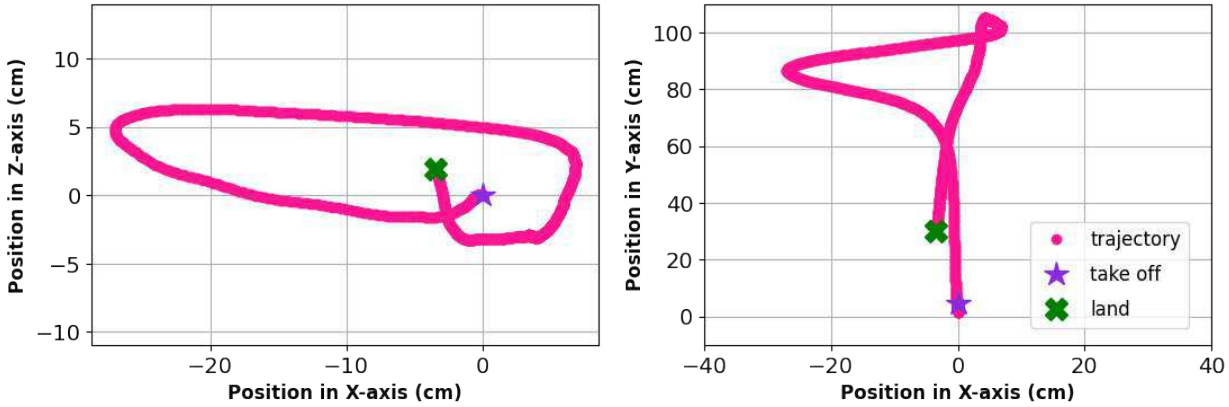
Source: Prepared by the author.

in Figure 4.13 and the point clouds in Figure 4.14 are compared, it is clear that the data in the X and Z axes feedbacked to the controller are incorrect. However, the altitude is consistent across both time series. Previous research[25] has demonstrated similar difference when utilising commercial optical tracking devices.

**Figure 4.13:** Optitrack and Mambo estimator upper and lateral trajectory map during a flight test using dataset 9 calibration.




**(a)** Optitrack trajectory



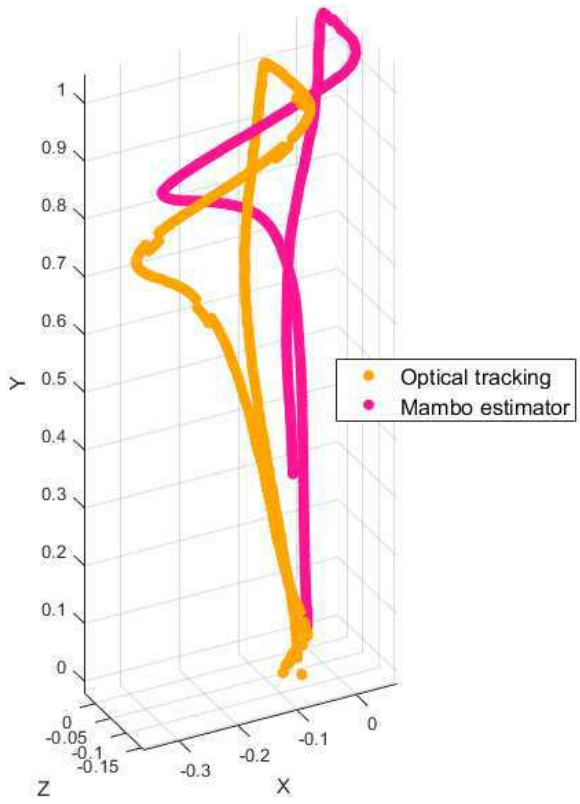

**(b)** Mambo estimator trajectory

Source: Prepared by the author.

## 4.4 Final considerations

In this chapter, we presented the results of the 3D reconstruction for a pair of RealSense cameras. Afterwards, we acquired comparable findings with two Raspberry NoIR cameras, completing the intrinsic and extrinsic calibration loop. The test flow was only finished

**Figure 4.14:** Optitrack and Mambo estimator point clouds during a flight test using dataset 9 calibration.




Source: Prepared by the author.

in the third phase, when it was implemented online between the main Desktop and the Raspberries via a server-client protocol. During this phase, we conducted two test flights using the Tello DJI, one for one camera and one for four cameras. Finally, we performed a movement using an industrial arm and compared the performance to the output of the visual tracking system, which was validated to subcentrimetre. In the next chapter, a summary of the findings of this work will be presented.

# Chapter 5

# Conclusions & future work

The theoretical basis of optical tracking systems was explored in earlier chapters. This study resulted in the development of a hardware and software bundle for an motion capture system that can be easily implemented using off-the-shelf cameras and Linux embedded systems. In the final architecture, photos are received with acceptable quality (960×640 pixels) and processed in real-time at 100 FPS. The markers attached to the mobile vehicle are then segmented effectively in a limited computational environment, and a new ordering technique general to any movement sequence was developed. Side problems like clock synchronisation, lens model fit and markers' size were also addressed. The system was ultimately tested on a ground truth trajectory, and the maximum latency was ten milliseconds, while the accuracy was less than one centimetre when disregarding noise. Overall, the proposed asynchronous and heterogeneous system achieved comparable performance to the commercially available OptiTrack Flex 3, yet using network triggered capture stations.

It is worth noting that this work emphasises the technical challenges of establishing a permanently visible marking across an ample working space. So good practices such as light insulation, cooling, and highly optimised code structures are suggested throughout this report to reduce future efforts to adapt the platform either to alternative tracking subjects or to other hardware that may be available.

The suggested system's restriction is the accuracy of centroid estimation, which is strongly dependent on homogeneous lighting, a condition hardly achieved using low-range LED rings. A wider intrinsic calibration pattern or depth of vision adjustment should also

be made to lessen the disparity between the point clouds from several camera pairs and approximate the scale of the translation vector to the real distance between the cameras. Due to time constraints, these concerns could not be resolved, although the following suggestions may be pursued in future work.

## Marker reconstruction refinement

To enhance the number of threshold filters in the blob detector, hardware-level processing might be employed. Instead of using hard thresholds, gradient filters can also be used identify the contour. These changes are intended to improve centroid estimation and fundamental matrix computing, affecting extrinsics and ground plane calibration performance. On the assumption that the intrinsic calibration has been addressed, it would also reduce the difference between the point clouds from the various camera pairs. Constructing a brighter IR LED with a broader range might help eliminate any noise caused by poor marker lighting.

## Prediction of future poses

The noise removal must be done in order to achieve smoother pose tracking using the optical tracking system. Predicting future positions of the observed vehicles is a good solution that also reduces latency, as applied by the OptiTrack Flex 3 system.

## Treating marker obstruction

The number of cameras can be increased to ensure consistent monitoring of the markers. Another option is to associate images with less than the number of markers in the arena. An outer loop in the ordering algorithm will be required to detect which markers are present in each image and triangulate with the corresponding camera pair.

## Multiple vehicles tracking

The presented system tracks the absolute position of only one vehicle. Further development would be using clusterisation techniques to identify multiple vehicles in the working

space. Since the rotation and translation to a single-vehicle reference image have already been accomplished, one can replicate this effort across the number of robots. This improvement would allow for relative vehicle positioning and the assessment of formation techniques. It should be noted that changing the validity constraints of the photos, which considers only frames with exactly four visible blobs valid, would be required.

# References


1 LIU, Xinyu et al. Bias-error accumulation analysis for inertial navigation methods. *IEEE Signal Processing Letters*, Institute of Electrical and Electronics Engineers (IEEE), 2021. Available at: <https://doi.org/10.1109/lsp.2021.3129151>.

2 FENG, Xu; NGUYEN, Khuong An; LUO, Zhiyuan. A survey of deep learning approaches for WiFi-based indoor positioning. *Journal of Information and Telecommunication*, Informa UK Limited, p. 1–54, sep. 2021. Available at: <https://doi.org/10.1080/24751839.2021.1975425>.

3 KARAM, Samer; LEHTOLA, Ville; VOSSELMAN, George. Strategies to integrate IMU and LiDAR SLAM for indoor mapping. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Copernicus GmbH, V-1-2020, p. 223–230, aug. 2020. Available at: <https://doi.org/10.5194/isprs-annals-v-1-2020-223-2020>.

4 JIANG, Ping et al. Novel indoor positioning algorithm based on lidar/inertial measurement unit integrated system. *International Journal of Advanced Robotic Systems*, SAGE Publications, v. 18, n. 2, p. 1729, mar. 2021. Available at: <https://doi.org/10.1177/1729881421999923>.

5 PROPHET, S.; SCHOLZ, G.; TROMMER, G. F. Collision avoidance system with situational awareness capabilities for autonomous MAV indoor flights. In: *2017 24th Saint Petersburg International Conference on Integrated Navigation Systems (ICINS)*. IEEE, 2017. Available at: <https://doi.org/10.23919/icins.2017.7995565>.

6 ALARIFI, Abdulrahman et al. Ultra wideband indoor positioning technologies: Analysis and recent advances. *Sensors*, MDPI AG, v. 16, n. 5, p. 707, may 2016. Available at: <https://doi.org/10.3390/s16050707>.

7 QI, Yang; ZHONG, Yisheng; SHI, Zongying. Cooperative 3-d relative localization for UAV swarm by fusing UWB with IMU and GPS. *Journal of Physics: Conference Series*, IOP Publishing, v. 1642, n. 1, p. 12028, sep. 2020. Available at: <https://doi.org/10.1088/1742-6596/1642/1/012028>.

8 RUIZ, Antonio R.J.; GRANJA, Fernando S. Comparing ubisense, BeSpoon, and DecaWave UWB location systems: Indoor performance analysis. *IEEE Transactions on Instrumentation and Measurement*, Institute of Electrical and Electronics Engineers (IEEE), v. 66, n. 8, p. 2106–2117, aug. 2017. Available at: <https://doi.org/10.1109/tim.2017.2681398>.

9 YOU, Weide; LI, Fanbiao; LIAO, Liqing; HUANG, Meili. Data fusion of UWB and IMU based on unscented kalman filter for indoor localization of quadrotor UAV. *IEEE Access*, Institute of Electrical and Electronics Engineers (IEEE), v. 8, p. 64971–64981, 2020. Available at: <https://doi.org/10.1109/access.2020.2985053>.

10 BAHAROM, Abdul K.; ABDUL-RAHMAN, Shuzlina; JAMALI, Rafidah; MUTALIB, Sofianita. Towards modelling autonomous mobile robot localization by using sensor fusion algorithms. In: *2020 IEEE 10th International Conference on System Engineering and Technology (ICSET)*. IEEE, 2020. Available at: <https://doi.org/10.1109/icset51301.2020.9265372>.

11 BRENA, Ramon F. et al. Evolution of indoor positioning technologies: A survey. *Journal of Sensors*, Hindawi Limited, v. 2017, p. 1–21, 2017. Available at: <https://doi.org/10.1155/2017/2630413>.

12 MENDOZA-SILVA, Germán Martín; TORRES-SOSPEDRA, Joaquín; HUERTA, Joaquín. A meta-review of indoor positioning systems. *Sensors*, MDPI AG, v. 19, n. 20, p. 4507, oct. 2019. Available at: <https://doi.org/10.3390/s19204507>.

13 CAMPOS, Carlos et al. ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM. *IEEE Transactions on Robotics*, Institute of Electrical and Electronics Engineers (IEEE), v. 37, n. 6, p. 1874–1890, dez. 2021. Available at: <https://doi.org/10.1109/tro.2021.3075644>.

14 ROSINOL, Antoni; ABATE, Marcus; CHANG, Yun; CARLONE, Luca. Kimera: an open-source library for real-time metric-semantic localization and mapping. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020. Available at: <https://doi.org/10.1109/icra40945.2020.9196885>.

15 SARLIN, Paul-Edouard; DETONE, Daniel; MALISIEWICZ, Tomasz; RABINOVICH, Andrew. SuperGlue: Learning feature matching with graph neural networks. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2020. Available at: <https://doi.org/10.1109/cvpr42600.2020.00499>.

16 MORAR, Anca et al. A comprehensive survey of indoor localization methods based on computer vision. *Sensors*, MDPI AG, v. 20, n. 9, p. 2641, may 2020. Available at: <https://doi.org/10.3390/s20092641>.

17 BILLS, Cooper; CHEN, Joyce; SAXENA, Ashutosh. Autonomous MAV flight in indoor environments using single image perspective cues. In: *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011. Available at: <https://doi.org/10.1109/icra.2011.5980136>.

18 OLIVEIRA, Débora N.P.; SOUSA, Davi J.G.; MORAIS, Marcos R.A.; LIMA, Antonio M.N. Void detection for UAV based on optical flow and vanishing points. In: *2021 Intelligent Automation Brazilian Symposium*. Rio Grande: SBA, 2021. Available at: <https://doi.org/10.20906/sbai.v1i1.2586>.

19 OLAGUE, Gustavo; MOHR, Roger. Optimal camera placement for accurate reconstruction. *Pattern Recognition*, Elsevier BV, v. 35, n. 4, p. 927–944, apr. 2002. Available at: <https://doi.org/10.1016/s0031-3203(01)00076-0>.

20 MASON, Scott O.; GRüN, Armin. Automatic sensor placement for accurate dimensional inspection. *Computer Vision and Image Understanding*, Elsevier BV, v. 61, n. 3, p. 454–467, may 1995. Available at: <https://doi.org/10.1006/cviu.1995.1034>.

21 DENG, Heng; YANG, Kun; QUAN, Quan; CAI, Kai-Yuan. Accurate and flexible calibration method for a class of visual sensor networks. *IEEE Sensors Journal*, Institute of Electrical and Electronics Engineers (IEEE), v. 20, n. 6, p. 3257–3269, mar. 2020. Available at: <https://doi.org/10.1109/jsen.2019.2958104>.

22 HARTLEY, Richard; ZISSERMAN, Andrew. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004. Available at: <https://doi.org/10.1017/cbo9780511811685>.

23 FURTADO, Joshua S. et al. Comparative analysis of OptiTrack motion capture systems. In: *Lecture Notes in Mechanical Engineering*. Springer International Publishing, 2019. p. 15–31. Available at: <https://doi.org/10.1007/978-3-030-17369-2_2>.

24 GOMES, L.L. et al. Unmanned quadcopter control using a motion capture system. *IEEE Latin America Transactions*, Institute of Electrical and Electronics Engineers (IEEE), v. 14, n. 8, p. 3606–3613, aug. 2016. Available at: <https://doi.org/10.1109/tla.2016.7786340>.

25 HABSI, Shaima Al et al. Integration of a vicon camera system for indoor flight of a parrot AR drone. In: *2015 10th International Symposium on Mechatronics and its Applications (ISMA)*. IEEE, 2015. Available at: <https://doi.org/10.1109/isma.2015.7373476>.

26 MASHOOD, Ahmed et al. Quadrotor object tracking using real-time motion sensing. In: *2016 5th International Conference on Electronic Devices, Systems and Applications (ICEDSA)*. IEEE, 2016. Available at: <https://doi.org/10.1109/icedsa.2016.7818504>.

27 MEHLING, Michael. *Implementation of a Low Cost Marker Based Infrared Optical Tracking System*. Dissertation (Audiovisual Media) — Fachhochschule Stuttgart, Vienna, fev. 2006.

28 DENG, Heng et al. Indoor multi-camera-based testbed for 3-d tracking and control of UAVs. *IEEE Transactions on Instrumentation and Measurement*, Institute of Electrical and Electronics Engineers (IEEE), v. 69, n. 6, p. 3139–3156, jun. 2020. Available at: <https://doi.org/10.1109/tim.2019.2928615>.

29 TEATHER, Robert J.; PAVLOVYCH, Andriy; STUERZLINGER, Wolfgang. Effects of latency and spatial jitter on 2d and 3d pointing. In: *2009 IEEE Virtual Reality Conference*. IEEE, 2009. Available at: <https://doi.org/10.1109/vr.2009.4811029>.

30 ALI, Rashid et al. Systematic review of dynamic multi-object identification and localization: Techniques and technologies. *IEEE Access*, Institute of Electrical and Electronics

Engineers (IEEE), v. 9, p. 122924–122950, 2021. Available at: <https://doi.org/10.1109/access.2021.3108775>.

31 POST, Tjark. *Precise localization of a UAV using visual odometry*. Dissertation (Robotics and Mechatronics) — University of Twente, Enschede, dec. 2015.

32 ZHANG, Z. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Institute of Electrical and Electronics Engineers (IEEE), v. 22, n. 11, p. 1330–1334, 2000. Available at: <https://doi.org/10.1109/34.888718>.

33 URBAN, Steffen; LEITLOFF, Jens; HINZ, Stefan. Improved wide-angle, fisheye and omnidirectional camera calibration. *ISPRS Journal of Photogrammetry and Remote Sensing*, Elsevier BV, v. 108, p. 72–79, oct. 2015. Available at: <https://doi.org/10.1016/j.isprsjprs.2015.06.005>.

34 SCARAMUZZA, Davide; MARTINELLI, Agostino; SIEGWART, Roland. A toolbox for easily calibrating omnidirectional cameras. In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006. Available at: <https://doi.org/10.1109/iros.2006.282372>.

35 BOUGUET, Jean-Yves. *Camera Calibration Toolbox for Matlab*. 2004. Available at: <http://www.vision.caltech.edu/bouguetj/calib_doc/>.

36 KANNALA, Juho; BRANDT, Sami S. A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Institute of Electrical and Electronics Engineers (IEEE), v. 28, n. 8, p. 1335–1340, aug. 2006. Available at: <https://doi.org/10.1109/tpami.2006.153>.

37 JONES, Dave. *piCamera: Sensor modes*. 2015. Available at: <https://picamera.readthedocs.io/en/release-1.13/fov.html#sensor-modes>.

38 LORIETTE, Antoine. *PyRealSense's documentation*. 2016. Available at: <https://pyrealsense.readthedocs.io/en/master/#>.

39 Perle Systems. *Precision Time Protocol*. 2020. Available at: <https://www.perle.com/supportfiles/precision-time-protocol.shtml>.

40 NEVILLE-NEIL, George. *PTP deamon*. 2015. Available at: <https://github.com/ptpd/ptpd>.

41 MOSIMANN, Felix. *Summary of NTP and PTP informations*. 2022. Available at: <https://www.mobatime.com/article/ntp-and-ptp-informations/>.

42 OLIVEIRA, Débora N.P. *Userland module for PTS exportation*. 2022. Available at: <https://github.com/debOliveira/userland>.

43 LACHANCE, Michael. An introduction to splines for use in computer graphics and geometric modeling. *Computer Vision, Graphics, and Image Processing*, Elsevier BV, v. 50, n. 1, p. 125–126, apr. 1990. Available at: <https://doi.org/10.1016/0734-189x(90)90071-3>.

44 BOWMAN, Richard. *Lens shading control*. 2004. Available at: <https://github.com/waveform80/picamera/pull/470>.

45 OLIVEIRA, Débora N.P. *Generic Camera Calibrator*. 2021. Available at: <https://github.com/debOliveira/myCameraCalibrator>.