TÉCNICO LISBOA

Monocular Ball Detection and Tracking with Noise Characterization

Bernardo Guilherme de Almeida Pitrez Pina Ferreira

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor: Prof. Alexandre José Malheiro Bernardino

Examination Committee

Chairperson: Prof. João Fernando Cardoso Silva Sequeira
Supervisor: Prof. Alexandre José Malheiro Bernardino
Member of the Committee: Prof. Pedro Daniel dos Santos Miraldo

January 2021

Declaração

Declaro que o presente documento é um trabalho original da minha autoria e que cumpre todos os requisitos do Código de Conduta e Boas Práticas da Universidade de Lisboa.

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

First of all I would like to thank my thesis coordinator Alexandre Bernardino for the availability, insight, support and opportunity provided for the realization of this thesis, without his contributions and the sharing of knowledge this work would not have been possible. I would also like to thank all my family for the encouragement, caring and understanding all these years because without them i wouldn't be where i am today. Finally, i would like to thank my friends that belonged in this path for being present and help me grow as a person.

To each and everyone of you - Thank you.

Abstract

Ball tracking has been a problem with extensive study done, but the accuracy of such trackers still gets affected by errors and there is a lack of uncertainty characterization in the observation models utilized.

In this thesis, we aim to develop an algorithm that provides 3D ball locations with the corresponding noise characterization from 2D segmented blobs. These outputs are crucial to cope with distance varying characteristics of the noise associated and can serve as an input to tracking algorithms in order to increase their robustness. This approach uses the known color and ball size information and through image segmentation it is able to generate an estimation of its 3D location in space. Furthermore, two methods which are Monte Carlo (MC) and Unscented Transform (UT), are used to propagate the noise uncertainty which is then used as input into a tracker to test its performance.

The errors, when using the measurements and covariances from the proposed methods, were inferior when comparing to a covariance that remained fixed throughout the whole tracking process, showing a decrease on the error up to 76% in some trajectories.

Keywords

Projection; Ellipse; Covariance; Noise characterziation; RANSAC.

Resumo

O rastreamento de bolas tem sido um problema em que extenso estudo já foi feito, mas a precisão destes sistemas ainda é afetada por erros e existe uma falta de caracterização de ruído nos modelos de observação utilizados.

Nesta tese, o objectivo é desenvolver um algoritmo que providencie localizações da bola em 3D com a correspondente caracterização de ruído a partir de blobs segmentados em 2D. Estes resultados são cruciais para lidar com as características que variam com a distância do ruído associado e podem servir como entrada em algoritmos de rastreamento para aumentar a sua robustez. Esta abordagem usa a conhecida cor e tamanho da bola e através segmentação de imagem é possível gerar estimações em 3D. Além disso, dois métodos que são MC e UT, são utilizados para propagar a incerteza de ruído que depois é utilizada como entrada em um algoritmo de rastreamento para avaliar o seu desempenho.

Os erros, quando utilizados as estimativas e covariâncias dos métodos propostos, foram inferiores quando comparados a uma covariância que se manteve fixa ao longo de todo o processo de localização, mostrando uma diminuição no erro até 76% em algumas trajetórias.

Palavras Chave

Projeção; Ellipse; Covariância; Caracterização de ruído; RANSAC.

Contents

List of Figures

List of Tables

Acronyms

CV	Computer Vision
RGB	Red-Green-Blue
HSI	Hue-Saturation-Intensity
HSV	Hue-Saturation-Value
2D	Two-dimensional
3D	Three-dimensional
KF	Kalman Filter
RANSAC	Random Sample Consensus
MC	Monte Carlo
UT	Unscented Transform
MSE	Mean Squared Error
GT	Ground Truth

1

Introduction

Contents

1.1 Motivation

Humanity has always relied on the ability to observe the world in order to interact with it, and as it has developed itself, computers started aiding it with such task. Henceforth Computer Vision (CV) was born. The potential of this area of expertise has been growing swiftly throughout the years [Cagnoni and Zhang, 2016], because it is one of the main technologies that allows communication between the digital and physical world.

The constant development of cameras has allowed a decrease in their cost and an increase in image resolution, which implies that CV algorithms have far more pixels to work with. With the increase in the development of cameras and the extensible area of application, this field of study has an enormous potential of growth. It is a necessary tool that has various applications such as human-computer interaction [Agren, 2017] and medicine [Huang, 2019].

However, with the unpredictability of the world, complex situations emerge with plenty of noise, that require robust algorithms in order to be properly analyzed and drawn conclusions from. An example of one of those situations is object tracking. It is a very well studied problem in the computer vision environment and allows following objects moving throughout the 3D space. This technology although it has been studied for a long time and considerable progress has been made [Wu et al., 2015], it still remains a challenging problem with the lack of an algorithm that can tackle all scenarios, because of the infinite number of possibilities. A few examples of these situations can be such as video surveillance [Verma et al., 2015] and automotive [Zhang et al., 2009].

One of the simplest objects to track are balls, due to their homogeneous form. Ball tracking is a tool which its accuracy relies mainly on the quality of the camera and it is necessary for a lot of industries. These industries are, for the most part, related to sports, such as football [Ren et al., 2004], basketball [Chakraborty, 2013], volleyball [Huang et al., 2012] and golf [Woodward and Delmas, 2005], where computer vision has overcame traditional methods (mostly manual), because statistical data is displayed in a much faster pace. It is also an important branch in robotic navigation [Olufs et al., 2007], for example for soccer or social robots to be able to gather information from their surroundings and take action.

The proposed work focuses on developing an algorithm that can track a ball in Three-dimensional (3D) throughout the frames of a video. This is obtained through a method that exploits the known color and dimensions of the ball, thus it suffices the use of a single camera. Finally, the video will be captured from a camera that captures images in Red-Green-Blue (RGB) and then deconstructed into frames.

1.2 Objectives

This thesis has one main objective:

- Detection of a ball knowing its color and size from a single camera with characterization of the uncertainty from a monocular reconstruction of its location.

This is accomplished by analysing each frame of the corresponding video and proceeding to the identification of the whereabouts of the ball. Then noise in the Two-dimensional (2D) image plane is propagated to the 3D world for the uncertainty characterization. The characterization of uncertainty is essential to cope with the distance varying characteristics of the noise and serves as input to tracking algorithms. The proposed method is applied recurring to a Kalman Filter (KF) architecture, but it can also be applied to more advanced probabilistic trackers, such as Particles Filters.

1.3 Thesis Outline

The presented work is organized by the following structure: Chapter 1 provides a short introductory discussion and motivates the problem which this thesis addresses. Chapter 2 provides theoretical background that is necessary to understand the scope, along with the notations that will be used throughout this work. In Chapter 3, the previous work done regarding this subject is addressed and the different approaches taken into consideration by those works are analyzed and briefly described, presenting after the novelty proposed by this thesis. Chapter 4 bestows the algorithm in a chronological order where the steps implemented are described by the order in which they were implemented and every detail is thoroughly explained. Following, Chapter 5 takes into consideration the steps that were required to implement the algorithm and how they were executed. In Chapter 6 the results are submitted and the evaluation of the efficiency and accuracy of the algorithm is tested according to different metrics. Finally in Chapter 7, the conclusions related to this work are presented and are suggested the possible improvements for future work regarding this area of investigation.

2

Background

Contents

This chapter will focus on providing a theoretical background on the necessary subjects involved in the tracking process. This revolves around projection and pixel coordinates, which is how a 3D object is visualized in the image plane. Furthermore, a narrowed down point of view will be taken for spherical objects that is in the scope of this project and will be later used in Chapter 4.

2.1 Projection

The process of 3D projection consists in mapping points from the Three-dimensional (3D) world into a Two-dimensional (2D) plane. The usual way of modelling this process is where a ray is drawn from a point in space from the 3D world and it will intersect a specific plane chosen as the *image plane*.

2.1.1 Perspective projection

The image plane is defined by the plane that contains the two dimensional array of pixels sensed by the camera. The x and y axis form the image plane basis, and often represent the horizontal and vertical direction respectively. The z axis is considered to be perpendicular to the image plane and intersects it at its center, being usually denominated as *principle point*. The origin of this coordinate system is positioned at a distance λ, considered the *focal length* of the camera, behind the image plane and is usually referred as *center of projection*. This coordinate system can be observed in Figure 2.1 where it coincides with the world coordinate frame.

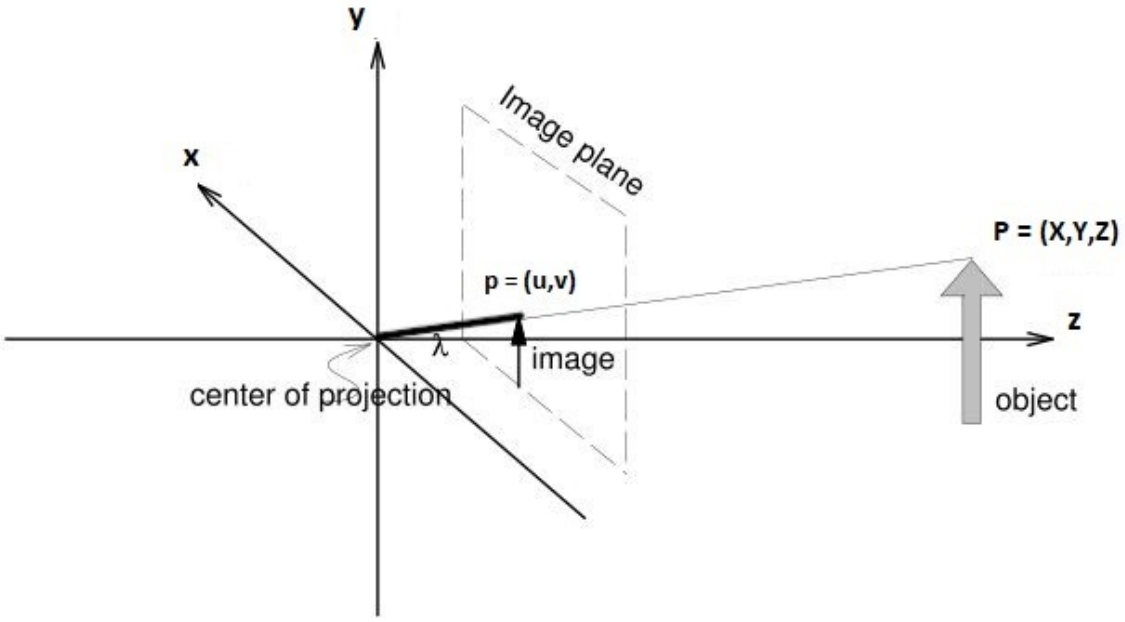


Figure 2.1: Schematic representation of the projection of a point P in the image plane, from [Spong et al., 2006]

The image plane is often modeled by the the pinhole lens approximation. This model states that the lens is considered an ideal pinhole which is positioned at the focal center. Therefore, light rays pass this pinhole and intersect the image plane. Note that in the schematic representation in Figure 2.1 the pinhole is placed is behind the image plane for illustration purposes.

Let there be a point P in the 3D world with coordinates (X, Y, Z) and its projection p with coordinates (x, y, λ), both represented in Figure 2.1. Under the pinhole assumption, P and p are collinear and thus, for some unknown positive K we have

$$K \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \sim \begin{pmatrix} x \\ y \\ \lambda \end{pmatrix} \tag{2.1}$$

which when converted to a system of equations is possible to obtain the equations for perspective projection [Spong et al., 2006]

$$x = \lambda \frac{X}{Z}, \qquad y = \lambda \frac{Y}{Z}. \tag{2.2}$$

The equations described in (2.1) and (2.2) can also be written as a linear mapping between homogeneous coordinates,

$$\begin{pmatrix} x \\ y \\ \lambda \end{pmatrix} = K \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.3}$$

where a $3\text{x}4$ matrix projection represents a map from the 3D to 2D. Note that it is assumed that the camera has its coordinate frame coincident with the world frame.

2.1.2 Conversion to pixel coordinates

In order to relate a digital image to the 3D world and reconstruct the position on the point p, the connection between image plane coordinates and pixel coordinates must be known. Considering the pixels as a discrete set of points, the x and y values attained must be rounded to integer values to achieve the corresponding pixel point. Let the pixel coordinates of the *principal point* be represented by (u_0, v_0) as it is possible to visualize in Figure 2.1. In general, the pixels are not necessary square, thus its height and width must be compensated for by scaling its s_x and s_y, that represent its horizontal and vertical dimensions respectively. Therefore, it is possible to discover the pixel coordinates from

$$u = s_x.x + u_0, \qquad v = s_y.y + v_0. \tag{2.4}$$

Accordingly, with Equations (2.1) and (2.4) it is possible to arrive at the relation given by (2.5) where K is the camera intrinsic matrix [Spong et al., 2006].

$$Z \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{2.5}$$

$$K = \begin{bmatrix} a_x & 0 & u_0 \\ 0 & a_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

Note that the parameters a_x and a_y in the matrix (2.6) are $\lambda.s_x$ and $\lambda.s_y$ respectively. For the rest of this work and for representation purposes of the 2D world instead of u and v, x and y are used.

2.2 Conic sections

A 3D spherical object, when projected into the 2D world using perspective projection is an ellipse, which is a type of a conic. It is necessary to understand the fundamentals of conic sections, in order to analyze how to represent one and how that can help the problem presented in this work. A conic section consists in a curve that is obtained by intersecting a plane with a cone. There are different types of conics that can be obtained by this method and they can be represented by the implicit Equation (2.7) where A, B and C are not all zero.

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0 \tag{2.7}$$

Note that the Equation (2.7) has scale indeterminacy, meaning that all variables can be multiplied by a scalar non zero and the result would be the same. and can be rewritten in matrix form

$$\begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} A & B/2 \\ B/2 & C \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} D & E \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + F = 0 \tag{2.8}$$

or also written as

$$\mathrm{x}^T A_k \mathrm{x} = 0 \tag{2.9}$$

where x is the homogeneous coordinate vector and A_k the matrix with the Equation (2.7) coefficients

$$\mathrm{x} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}, \qquad A_k = \begin{pmatrix} A & B/2 & D/2 \\ B/2 & C & E/2 \\ D/2 & E/2 & F \end{pmatrix} \tag{2.10}$$

The sum of the first three terms of the Equation (2.7) is the quadratic form and the matrix (2.11)

$$A_{2x2} = \begin{pmatrix} A & B/2 \\ B/2 & C \end{pmatrix} \tag{2.11}$$

is called the matrix of the quadratic form, which is obtained by removing the last row and column of A_k. The trace and determinant of (2.11) are both invariant to rotation and translation [Pettofrezzo, 1978] [Spain, 2007].

Therefore, with both matrices from (2.11) and (2.10), it is possible to classify conics surfaces into to different groups [Lawrence, 1972]. The scope of this work is only interested in non degenerate conics, consequently the $detA_k \neq 0$.

Accordingly, non degenerate conics can be classified, by recurring to the determinant of the matrix A_{2x2}, into:

- Hyperbola if and only if $detA_{2x2} < 0$
- Parabola if and only if $detA_{2x2} = 0$
- Ellipse if and only if $detA_{2x2} > 0$

The condition that is for the interest of this work is the one regarding the ellipse and its form is

$$B^2 - 4AC < 0. \tag{2.12}$$

2.2.1 Projection applied to conics

The principles of perspective projection have been discussed in section 2.1 and now it is possible to apply them in spherical objects. A sphere is a form of quadratic surface described in section 2.2, considering it has radius R and it is positioned in the origin, any point on its surface from the projective space can be described by

$$X^2 + Y^2 + Z^2 = R^2. \tag{2.13}$$

Accordingly, (2.13) can be represented in matricial form, which is portrayed in Equation (2.14), where M and Q_0 are

$$M^T Q_0 M = 0 \tag{2.14}$$

$$M = \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \quad , \quad Q_0 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -R^2 \end{pmatrix}. \tag{2.15}$$

Implementing a translation to the sphere represented by (2.14), to any arbitrary location $t = [t_x \quad t_y \quad t_z]^T$, can be done by applying a homogeneous coordination transformation T to Q_0 as $Q = T^T Q_0 T$, with T as

$$T = \begin{pmatrix} 1 & 0 & 0 & -t_x \\ 0 & 1 & 0 & -t_y \\ 0 & 0 & 1 & -t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{2.16}$$

With Equation (2.14) and (2.16) its possible to derive the relation (2.17), which represents any point on the sphere's surface at a given location t.

$$M^T Q M = 0 \quad , \quad Q = \begin{pmatrix} 1 & 0 & 0 & -t_x \\ 0 & 1 & 0 & -t_y \\ 0 & 0 & 1 & -t_z \\ -t_x & -t_y & -t_z & t_x^2 + t_y^2 + t_z^2 - R^2 \end{pmatrix} \tag{2.17}$$

Therefore, applying perspective projection on the points of the sphere's surface, the projected figure in the image plane is represented by a conic [Cross and Zisserman, 1998], as illustrated in Figure 2.2. The

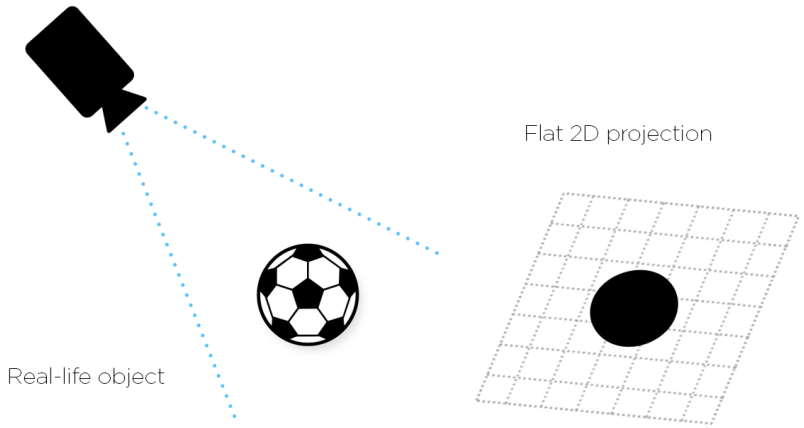


Figure 2.2: Schematic representation of a sphere's projection into the image plane, representing a conic section, from [https://www.zivid.com/, 2018]

conic can now be represented as 2.18, where A_k is the matrix from 2.10 and the $m = [x \quad y \quad 1]^T$ points must satisfy

$$m^T A_k m = 0. \tag{2.18}$$

Then, the projection of the ball Q in (2.17) to the ellipse A_k [Cross and Zisserman, 1998] is

$$A_k^* \sim P Q^* P^T. \tag{2.19}$$

where P is the camera 3x4 projection matrix in (2.3), $A_k^* = adj(A_k)$, $Q^* = adj(Q)$, adj represents the classical adjoint, which is the transpose of its cofactor matrix, and $\sim$ denotes equality through a scale

factor.

2.2.2 Canonical and general ellipse form

The coefficients (A, B, C, D, E, F) that represent an ellipse can be acquired from its canonical form where a and b are the semi-major and semi-minor axis respectively

$$\frac{x'^2}{a^2} + \frac{y'^2}{b^2} = 1. \tag{2.20}$$

Through an application of an affine transformation

$$x' = (x - x_0)\cos\Theta + (y - y_0)\sin\Theta, \qquad y' = -(x - x_0)\sin\Theta + (y - y_0)\cos\theta \tag{2.21}$$

the coefficients present in the Equation (2.7) can be acquired from a known a, b, center coordinates (x_0, y_0) and the rotation angle Θ (from the positive horizontal axis to the ellipse major axis) through the subsequent equations [Wikipedia, the free encyclopedia, 2020]:

$$A = a^2(\sin\Theta)^2 + b^2(\cos\Theta)^2 \tag{2.22a}$$

$$B = 2(b^2 - a^2)\sin\Theta\cos\Theta \tag{2.22b}$$

$$C = a^2(\cos\Theta)^2 + b^2(\sin\Theta)^2 \tag{2.22c}$$

$$D = -2Ax_0 - By_0 \tag{2.22d}$$

$$E = -Bx_0 - 2Cy_0 \tag{2.22e}$$

$$F = Ax_0^2 + Bx_0y_0 + Cy_0^2 - a^2b^2 \tag{2.22f}$$

Conversely, the canonical form parameters can also be derived from the general form coefficients from the following equations [Wikipedia, the free encyclopedia, 2020]:

$$a, b = \frac{-\sqrt{2\Big(AE^2 + CD^2 - BDE + (B^2 - 4AC)F\Big)((A + C) \pm \sqrt{(A - C)^2 + B^2)}}}{B^2 - 4AC} \tag{2.23a}$$

$$x_0 = \frac{2CD - BE}{B^2 - 4AC} \tag{2.23b}$$

$$y_0 = \frac{2AE - BD}{B^2 - 4AC} \tag{2.23c}$$

$$\Theta = \begin{cases} arctg\left(\frac{1}{B}(C - A - \sqrt{(A-C)^2 + B^2}\right) & \text{for } B \neq 0 \\ 0 & \text{for } B = 0, A < C \\ 90^\circ & \text{for } B = 0, A > C \end{cases} \tag{2.23d}$$

2.2.3 Kanatani Ellipse Representation

Extracting elliptic edges from images and fitting ellipse equations to them is one of the most crucial tasks in computer vision and throughout this work, for their representation, the notation in (2.24) is the one used.

$$A'x^2 + 2B'xy + C'y^2 + 2f_0(D'x + E'y) + f_0^2 F' = 0 \tag{2.24}$$

The variable f_0 is a constant for adjusting the scale. Although theoretically it can be 1, for finite-length numerical computation it should be chosen, so that x/f_0 and y/f_0 have order approximately $O(1)$. This increases accuracy and reduces the loss of significant digits [Kanatani et al., 2016]. The notation represented in Equation (2.24) notation is preferred to (2.7), where $B' = B/2, D' = D/2$ and $E' = E/2$, so that is possible to obtain a matrix representation A_k in homogeneous coordinates stated in

$$\begin{pmatrix} x/f_0 & y/f_0 & 1 \end{pmatrix} \begin{pmatrix} A' & B' & D' \\ B' & C' & E' \\ D' & E' & F' \end{pmatrix} \begin{pmatrix} x/f_0 \\ y/f_0 \\ 1 \end{pmatrix} = 0. \tag{2.25}$$

Since the Equation (2.24) has scale indeterminacy, meaning that the same ellipse can be represented if A',B',C',D',E' and F' were multiplied by a non-zero constant, they are normalized to

$$A'^2 + B'^2 + C'^2 + D'^2 + E'^2 + F'^2 = 1 \tag{2.26}$$

Defining two 6D vectors as

$$\xi = \begin{pmatrix} x^2 \\ 2xy \\ y^2 \\ 2f_0x \\ 2f_0y \\ f_0^2 \end{pmatrix}, \qquad \theta = \begin{pmatrix} A' \\ B' \\ C' \\ D' \\ E' \\ F' \end{pmatrix} \tag{2.27}$$

it is possible to represent Equation (2.24) as $(\xi, \theta) = 0$, where (ξ, θ) represents the inner product of ξ and θ and the normalization of the Equation (2.26) is equivalent to the vector normalization $||\theta|| = 1$. The notation in (2.27) will be used for throughout of this work.

2.2.4 Noise Characterization

It also necessary to take into account the properties of image noise for accurate fitting. Supposing that x and y are disturbed from their true values x', y' by $\triangle x$ and $\triangle y$ as in

$$x = x' + \triangle x, \qquad y = y' + \triangle y. \tag{2.28}$$

Replacing this in ξ seen in Equation (2.27) its obtained

$$\xi = \xi' + \triangle_1 \xi + \triangle_2 \xi \tag{2.29}$$

where ξ' represents the value of ξ with $x = x'$, $y = y'$ and $\triangle_1 \xi$ and $\triangle_2 \xi$ being the first order noise and second order noise correspondingly.

$$\triangle_1 \xi = \begin{pmatrix} 2x' \triangle x \\ 2 \triangle xy' + 2x' \triangle y \\ 2y' \triangle y \\ 2f_0 \triangle x \\ 2f_0 \triangle y \\ 0 \end{pmatrix}, \qquad \triangle_2 \xi = \begin{pmatrix} \triangle x^2 \\ 2 \triangle x \triangle y \\ \triangle y^2 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{2.30}$$

Regarding $\triangle x$ and $\triangle y$ as random variables subjected to an independent Gaussian distribution with mean 0 and standard deviation σ we obtain the covariance matrix of ξ as

$$V[\xi] = E[\triangle_1 \xi \, \triangle_1 \xi^T] = \sigma^2 V_0[\xi], \qquad V_0[\xi] = 4 \begin{pmatrix} x'^2 & x'y' & 0 & f_0 x' & 0 & 0 \\ x'y' & x'^2 + y'^2 & x'y' & f_0 y' & f_0 x' & 0 \\ 0 & x'y' & y'^2 & 0 & f_0 y' & 0 \\ f_0 x' & f_0 y' & 0 & {f_0}^2 & 0 & 0 \\ 0 & f_0 x' & f_0 y' & 0 & {f_0}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.31}$$

where $E[.]$ represents the expectation over the noise distribution. The diagonal elements of the covariance matrix indicate the noise susceptibility of each component and the others measure the correlation between its components. Note that the expected value has only the first order noise term, that is because the second order noise will have little impact in the final results due to being very small in comparison with the the previous one [Kanatani et al., 2016].

3

State of Art

Contents

3.1 Related Work

This chapter will focus on providing a brief analysis on work that has already been developed in ball tracking. The papers can be separated into two categories, 2D and 3D tracking, were in the the latter will be more closely followed in our work.

3.1.1 2D tracking

There are several methods that can be used in order to track balls. The most simple one relies on tracking the ball in the image plane through the help of image processing.

[Chakraborty, 2013] proposed a physics-based algorithm to track balls in basketball videos. It consisted in a background subtraction and frame differencing methods in order to apply segmentation to moving objects. Furthermore, the ball candidates were processed through a shape and circularity filter, where the eccentricity and normalised circularity where the parameters evaluated. Finally, the trajectories were analysed separately in the x and y axis, with their lengths and prediction errors used to assess the performance of the method.

[Olufs et al., 2007] proposed a robust object tracking method using a sparse shape-based object model. It consisted in an algorithm that was divided into two segments, one with short term memory and other with long term memory. Firstly, an object model was created where the contour of the object was determined, then an image segmentation process was done with the help of histograms and their likelihood calculation with the Bhattacharyya metric. Secondly, a short term memory detect objects moving through subsequent frames and the long term memory helps finding objects lost through occlusions by giving feedback to the short memory segment.

[Chen et al., 2007] proposed a physics based algorithm to track balls in volleyball games. The segmentation was done by analysing the positive variations of intensity throughout frames and applying morphological operators to clear noise and fill potential ball candidates. Next, the potential candidates were filtered based on ball size, shape and fullness. Moreover, the trajectory was dissected into x and y axis, were the trajectories were a line and a parabola respectively. They were identified based on estimation error, length and ratio of isolated candidates.

[Setiawardhana et al., 2017] proposed a ball tracking method for soccer robots. It consisted in applying image segmentation through thresholds in the space color Hue-Saturation-Value (HSV) and then clearing noise with morphological operators. Furthermore, the center of the ball was then fed to a backpropagation neural network, and the result consisted in the output goal area and the ball area position.

Regarding two dimensional tracking, the full position information of the object is never obtained due to the lack of depth, which is a crucial parameter to track objects in the real world.

3.1.2 3D tracking

Tracking in 3D world represents a bigger challenge. Either the camera used has a depth sensor that is able to indicate how further the object is or depth must be estimated by some other means. We categorize the existing methods in two classes, Direct and Indirect.

3.1.2.A Direct Methods

The direct methods consist in spreading hypothesis in the 3D world. A reference model of the ball can then be used to project the hypothesis in the image and check if the image information is coherent with that projection. This will indicate how likely that hypothesis is representing the true position of the ball. This method is represented in Figure 3.1.

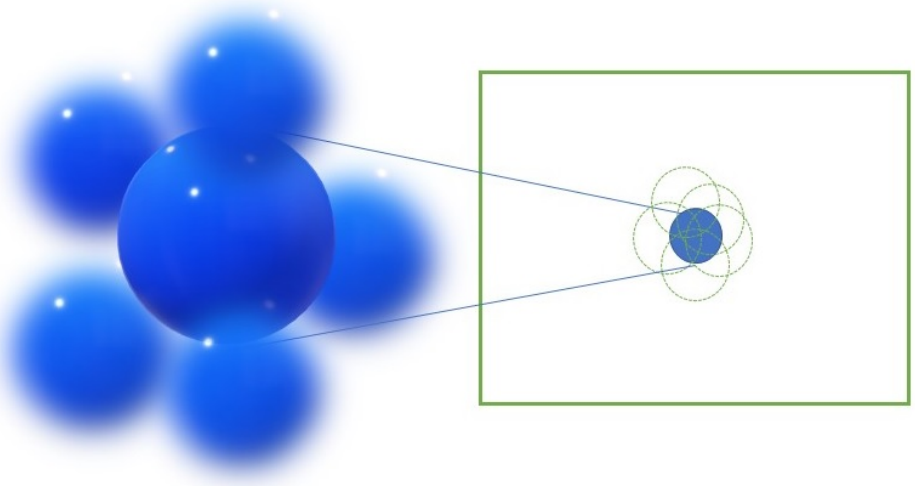


Figure 3.1: Schematic representation of the direct method

An example of a direct method is presented in [Taiana et al., 2008] that proposed a 3D tracking based on particle filters. The camera used was omnidirectional, therefore the projection model used was different from the one used in normal cameras. The Unified Project Model (UPM) was the one applied. The algorithm used to track the ball was a particle filter that consisted in three parts, the prediction where the particles were spread regarding a motion model, the update where each particle was given a probability based on the observation model, which used Bhattacharyya similarity metric and histograms to return a hypothesis, and the resampling where the particles with highest weight were replicated and the others were forgotten.

[Hou et al., 2017] proposed a method to 3D ball tracking with CPU-GPU acceleration. It consisted on a particle filter algorithm, where the several steps required to obtain a 3D position of the ball were separated and then reorganised, so that the process could be speed up, with the help of multi-command queue and stepped parallelism iteration.

[Wang et al., 2016] presented a method to track tennis balls. It consisted in a particle filter, with weighted particles, then reconstructed to 2D to display accurate tracking. The main contribution consisted in a noise characterization, it consisted in two parts, general and abrupt noise, and were classified according to the motion prejudgment result. The abrupt changes in the ball velocity were divided and characterized. A different model was used to characterize the noise based on which situation the data disclosed.

3.1.2.B Indirect Methods

The indirect methods correspond to the method of finding the position of the ball in the 3D world from the image plane, recurring to image projection as well. There are different ways to apply this process, the most common one is by gathering the information from more than one camera and triangulating the position of the ball, but it can also be done with one camera, for example recurring to monocular reconstruction mentioned in section 4.3.1.

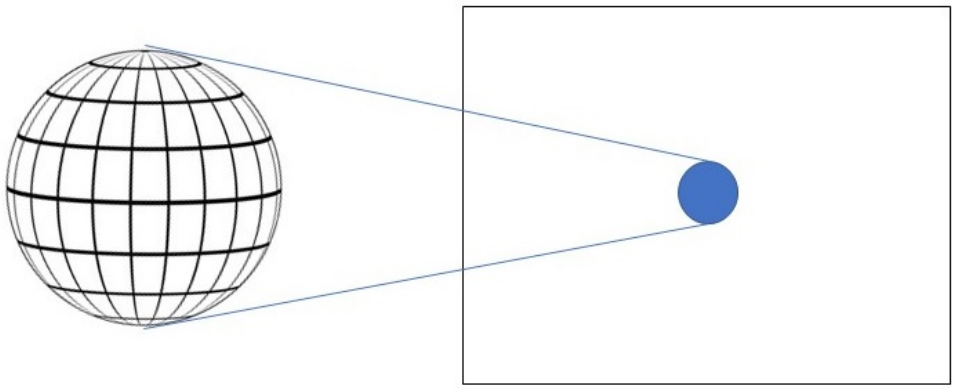


Figure 3.2: Schematic representation of the indirect method

[Ren et al., 2004] presented a method to track a ball using multiple cameras. It relied on kalman-based tracking and image differencing through a per-pixel Gaussian Model to detect, track and distinguish the moving objects from the background. It then attributed a likelihood to all moving objects of being a ball by analyzing several features, such as height, width, area, percentage of pixels, velocity and longevity, within specified thresholds. Finally, the strongest candidates were then reconstructed to 3D, through the triangulation of several inputs from different cameras and the trajectory was concluded from the previous result.

[Huang et al., 2012] studied ball tracking applied to volleyball games. Firstly, the single camera used, was calibrated with points known from the 3D space and the 2D image plane in order to obtain a projection matrix. Furthermore, ball candidates were detected in each frame by using constraints such

as size, shape and compactness, correlating each candidate over frames. Then, the possible trajectories in 2D were discovered by applying an algorithm that tracked parabolas in the Y axis and lines in the X axis. From those possibilities a point based system was applied, by verifying certain conditions, such as prediction error and trajectory length. Lastly, a linear system using the projection matrix was applied so that one may discover the 3D position and velocities of the ball to be tracked.

[Gómez-González et al., 2019] presented a system to track ping pong balls. It divided itself into two subsystems, one that returned the ball position in pixel space from each image, and the other produced an output of a single 3D ball position from the ball positions in pixel space obtained from multiple cameras. Firstly, semantic segmentation was used to find the pixel with highest probability corresponding to the ball, then the surrounding pixels were distinguished from ball or background through a threshold using Breath Search First algorithm. Formerly, the determination of the 3D position of the ball was attained through stereo vision where a Random Sample Consensus (RANSAC) similar based algorithm was applied, by selecting all possible camera pairs and then acquiring a 3D position that could be verified by the largest amount of camera pairs.

[Lippiello and Ruggiero, 2012] presented a method for 3D monocular robot ball catching. The image processing consisted in a segmentation using Hue-Saturation-Intensity (HSI) color space and histograms to identify the blobs in each image. Next, 2D information was collected and elaborated in order to get a first prediction of the ball trajectory through a rough linear estimation. This prediction was used as starting point for a more precise trajectory refinement through a nonlinear estimator.

3.1.3 Taxonomy

This section presents a summary of the papers briefly described in Chapter 3, which can be seen in Table 3.1.

Table 3.1: Characterizing papers that mention ball tracking

	2D tracking	Direct	Indirect	One Camera	Multiple Cameras
[Ren et al., 2004]			✓		✓
[Chakraborty, 2013]	✓			✓	
[Huang et al., 2012]			✓	✓	
[Olufs et al., 2007]	✓			✓	
[Chen et al., 2007]	✓			✓	
[Setiawardhana et al., 2017]	✓			✓	
[Taiana et al., 2008]		✓		✓	
[Hou et al., 2017]		✓		✓	
[Wang et al., 2016]		✓			✓
[Gómez-González et al., 2019]			✓		✓
[Lippiello and Ruggiero, 2012]			✓	✓	

The majority of the works mentioned don´t characterize the noise. The only exception is the one proposed by [Wang et al., 2016] where the noise characterization is specific to tennis balls, requires the use of multiple cameras and if severe occlusion occurs the tracking of the position of the ball hardly recovers. The scope of this work complies in addressing this problem, providing through the indirect method described above, a solution that increases the efficiency and accuracy of the tracker taken into consideration.

4

Methodologies

Contents

The proposed method in order to track the ball is represented by Figure 4.1.

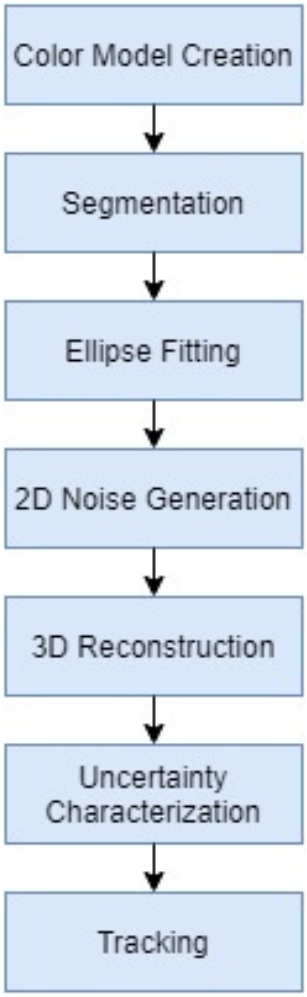


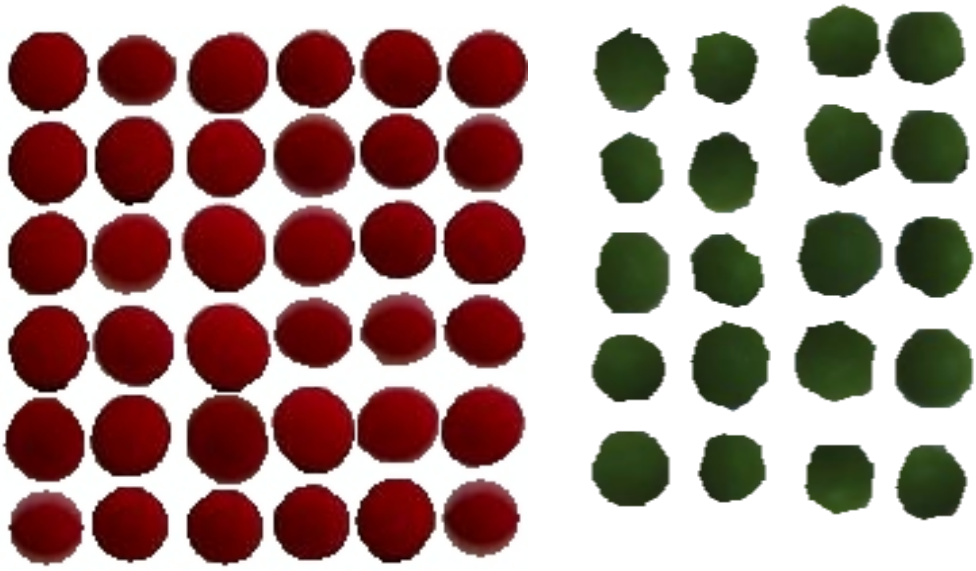
Figure 4.1: Main flowchart of the algorithm to be implemented

This method firstly consists in identifying where the ball is located on the image using an observation model. The proposed method starts by identifying where the ball is located on the image using an observation model based on color histograms to perform color segmentation in each frame. Then, an ellipse is fitted to the segmented regions in the image so a conic can be obtained from the image plane. Successively, the ellipse is reconstructed into the 3D world. We are left with a 3D measurement from a non linear transformation, therefore noise is spread on the 2D measurement in order to characterize the uncertainty from this process. Two methods are presented, one based on Monte Carlo and another on the Unscented Transform, where both require noisy 2D samples. Furthermore, these samples are all reconstructed into the 3D world and the covariance is obtained from them. Finally, this serves as an input to a tracker in order to test the validity and performance of this approach.

4.1 Image Segmentation

One of the main challenges with image segmentation is dealing with motion blur. This phenomenon occurs due to relatively high speed of the object being tracked in regard of the camera frame rate. Therefore, the contour can be very distorted and trackers that use this feature to know the location of the ball would have a poor efficiency and accuracy. Using color has been proved to be a method that is able to overcome this hurdle [Olufs et al., 2007] and [AlBasiouny et al., 2015], so color histograms are going to be used throughout this work.

The first step taken into account is to create a reference image, which can be seen in Figure 4.2, that consists in several pictures of the cropped ball meant to be tracked. The reference image is used to create the color model of the ball through the use of histograms. These images comprehend the ball cropped in several different lighting conditions and there are used multiple representations in order to increase the robustness of the algorithm. This image is then converted from the default format Red-



(a) Reference image from crops of a red ball **(b)** Reference image from crops of a green ball

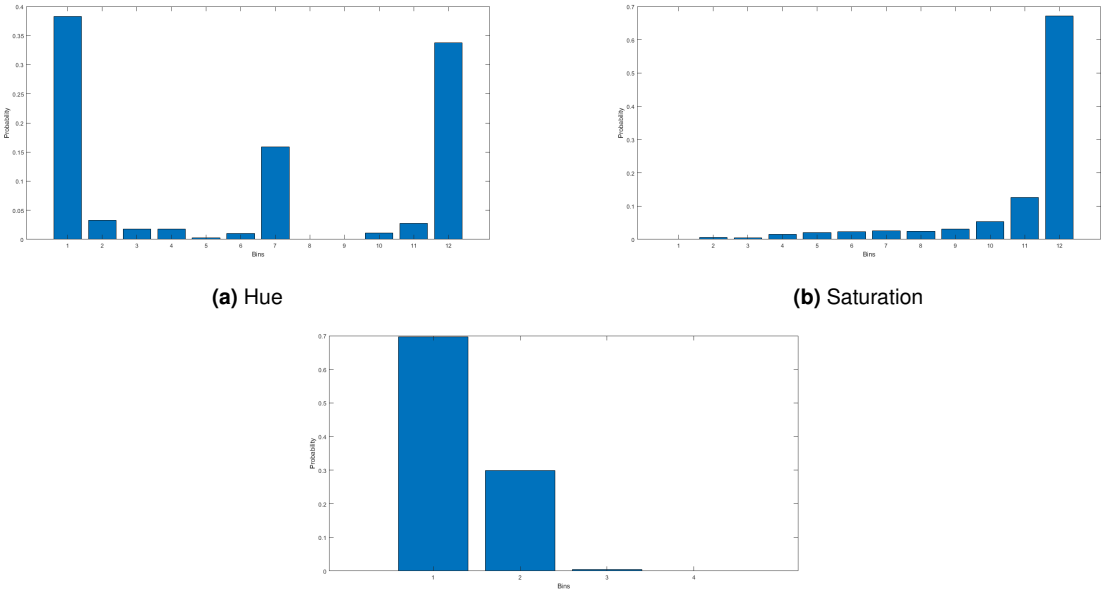
Figure 4.2: Images used to compute the reference histogram. The histogram is based on the non white pixels of the images

Green-Blue (RGB) to Hue-Saturation-Intensity (HSI) according to the methods in [Smith, 1978]. The main advantages in using the HSI are that the intensity is decoupled from the color information and both hue and saturation are closely related to how human vision perceives the color attributes. HSI has also a better performance in terms of true color image processing than RGB [Ledley et al., 1990]. The HSI format is also chosen instead of the most common Hue-Saturation-Value (HSV) color space, because the intensity takes all colors into account $I = avg(R + G + B)/3$ instead of the maximum value

$V = max(R, G, B)$. The equations to generate the values of a histogram are represented in (4.1), where m_i the total number of observations in the correspondent bin i, N the total number of observations and p_i the probability associated to each bin.

$$p_i = \frac{m_i}{N} \tag{4.1}$$

Typically, hue, saturation and intensity have values in the range of $[0, 255]$. To create the color histograms, the color values were divided into bins: 12 to hue and saturation and 4 for the intensity, which are represented in Figure 4.3. The number of bins were lower in the intensity so the algorithm would be more robust and less variant to lighting changes. After acquiring the histograms, the same process is



(a) Hue

(b) Saturation

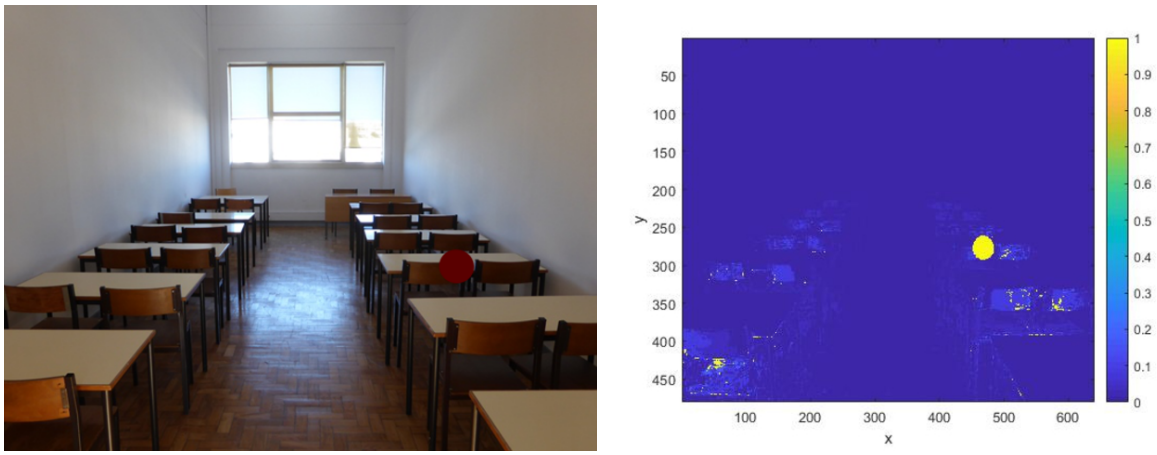
(c) Intensity

Figure 4.3: Histograms obtained from the reference image 4.2a

applied to each frame of the video being analysed. The frames are separated into hue, saturation and intensity, then these segments are divided into bins just like the reference image was, so that it is possible to identify blobs which resemble the ball that is going to be tracked. A probability is assigned to each pixel by multiplying the probabilities of the bins in which that pixel maps into the reference histograms.

$$p_i = h_i * s_i * i_i \tag{4.2}$$

We are then left with the pixel probabilities like it is represented in Figure 4.4



(a) Original image

(b) Pixel probability image

Figure 4.4: Assignment of pixel probability process

4.1.1 Otsu Method

After obtaining the image with the probabilities associated with each pixel, image binarization must be applied. This is done recurring to the Otsu's method. This algorithm focuses on finding the best threshold that minimizes the intra-class variance defined as a weighted sum of variances of two classes, being the classes object or background [Otsu, 1979]. It is represented by equation (4.3) where w_0 and w_1 represent each class probability, σ_0^2 and σ_1^2 are the variances of each class and they are separated by a a threshold t.

$$\sigma_w^2 = w_0(t)\sigma_0^2(t) + w_1(t)\sigma_1^2(t) \tag{4.3}$$

First an histogram is generated from the image and a threshold t is selected so that the image can be separated into two classes. The variance of the classes are calculated through

$$\sigma_{0,1}^2(t) = \frac{\sum_{i=0}^{N}(X_i - \mu)^2}{N} \tag{4.4}$$

where X_i is the pixel value, μ the mean and N the number of pixels in that class. The weights of each are given by

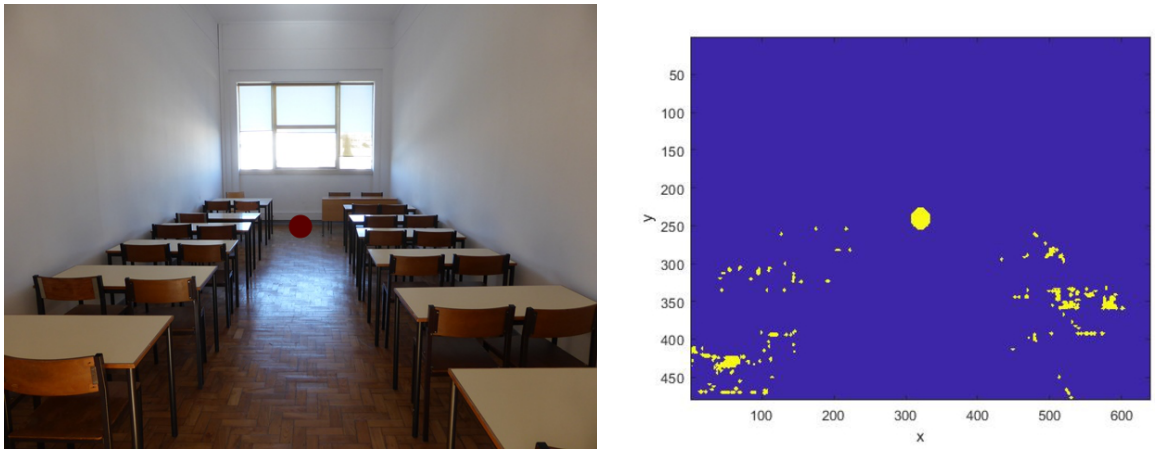
$$w_{0,1}(t) = \frac{N}{M} \tag{4.5}$$

where M is the total number of pixels from the image. Then, for every possible threshold t, the intra-class variance is calculated and the one with lowest is selected as the best threshold. After obtaining the best threshold, each pixel is then submitted to a condition which decides what pixel gets the value

one or zero depending if its value v is above or below the calculated threshold.

$$B(v,t) = \begin{cases} v > t, & p = 1 \\ else, & p = 0 \end{cases} \tag{4.6}$$

The final result is represented in Figure 4.5.



(a) Original image **(b)** Otsu threshold binarization

Figure 4.5: Binarization process

4.1.2 Morphological operators

After applying the method in section 4.1.1, corrections to the binary images must be made in order fill holes in possible ball candidates and remove noise from the background that was above the threshold. The most common binary image operations are called morphological operators. To perform such operations, a structuring element must be convolved with the image. For our problem we chose, as a structuring element, a disk since the objective is to track balls. The structuring element will be positioned in every possible position comparing each pixel with its neighbors [Szeliski, 2010].The morphological operation used was $opening$, which consists in a $erosion$ followed by a $dilate$. $Erosion$ consists in probing and reducing the image. It is used first so that noise that survived the Otsu's threshold will be eliminated and the larger blobs remain intact. Next, we have $dilation$ which is used to probe and expand the shapes contained in the image [Serra, 1986]. It is the last operator used in order to fill possible holes in the blobs that survive the previous filters. If we define A as a binary image and S as a structuring element with its reference pixel at (i, j) represented in 4.7,

$$S_{(i,j)} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{4.7}$$

erosion and dilate can be defined by 4.8a and 4.8b respectively.

$$A \ominus S = \{(i,j) : S_{(i,j)} \subset A\} \tag{4.8a}$$

$$A \oplus S = \bigcup_{(i,j) \in A} S_{(i,j)} \tag{4.8b}$$

After applying the morphological operators we are then left with binary images as the one represented in Figure 4.6.

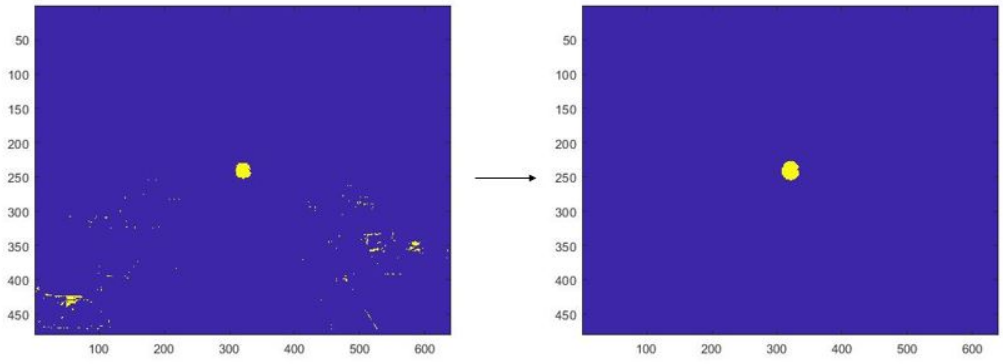
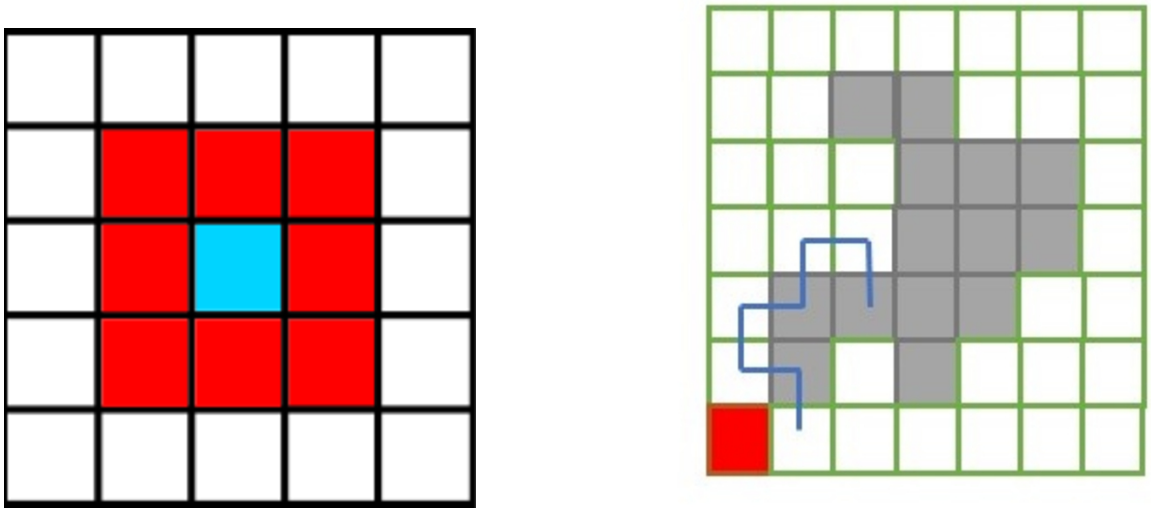


Figure 4.6: Performing an opening in a binary image

4.1.3 Contour Extraction

It is now necessary to fit an ellipse to the blobs that survived the image segmentation and morphological operations. First, it is necessary to find the contour, in order to fit the best possible ellipse. In order to do this it is necessary to locate the pixels with value "1" that are $connected$ where the Moore Neighbor method is applied. The Moore Neighborhood of a pixel consists in a set of 8 pixels that share a vertex or a edge with it, which is represented in Figure 4.7a.

The Moore Neighborhood tracing consists in selecting a starting pixel in the most bottom left pixel, then going up the column and left to right, moving along a clockwise direction in the neighborhood represented in Figure 4.7b. Every time a pixel with value one is found, the algorithm backtracks, goes around that specific pixel and repeats this process until the first pixel with value $"1"$ is reached again. In the end, the pixels with value one will be the contour of the pattern. This algorithm has also an adaptation regarding the stopping criteria, which is, the algorithm stops after entering the starting pixel with value



(a) Moore Neighborhood, from [Wikipedia, the free encyclopedia, 2013]

(b) Example of contour finding

Figure 4.7: Binarization process

"1" the same way it entered it initially [Reddy et al., 2012]. Through this method, in which called Jacob's stopping criteria, the performance of the algorithm is increased making it the best to extract the contour of any pattern, no matter the connectivity.

4.2 Ellipse Fitting

The contour of the noisy data is now available, therefore the ellipse fit can be applied in order to describe as best as possible a ball in the 2D image. The equation (2.26) does not entirely describe only ellipses, as it was described in Chapter 2, it can represent an parabola or even an hyperbola. Therefore, a method that forces the fit to be an ellipse is going to be implemented.

4.2.1 Least Squares Approach

The most simple method consists in using the Least Squares. The first step is to convert each point in the contour (x_a, y_a) into ξ_a from (2.27) and calculate the matrix M represented in (4.9), where N is the total number of points selected from the contour.

$$M = \frac{1}{N} \sum_{a=1}^{N} \xi_a \xi_a^T \tag{4.9}$$

If a point belongs to the ellipse then the condition $(\xi_a, \theta) = 0$ must be verified. Since this process has noise associated with, it focuses on minimizing the sum of squares in (4.10), which consists on selecting

the eigenvector with the lowest eigenvalue associated as in (4.11).

$$J = \frac{1}{N}\sum_{a=1}^{N}(\xi_a, \theta)^2 = \frac{1}{N}\sum_{a=1}^{N}\theta^T\xi_a\xi_a^T\theta = (\theta, M\theta) = \theta^T M\theta \tag{4.10}$$

Therefore, to find the solution the eigenvalue problem must be solved

$$M\theta = \lambda\theta \tag{4.11}$$

and the eigenvector θ for the smallest eigenvalue λ is the best ellipse. This approach is characterized by easy computation and the result can be obtained immediately but it is usually accompanied by poor performance because it does not take into account the possibility of outliers.

4.2.2 RANSAC

The random sampling methods is an iterative method to estimate parameters. It consists in selecting from the entire "population" a particular sample, where every "member" has the exactly same probability of being selected. The method of random sampling consists of sampling random points on the contour and in fitting an ellipse many times in order to find the best one. One of these methods is Random Sample Consensus (RANSAC), which is a method that is robust to the existence of $outliers$. Outliers are points that don't belong to the ellipse that fits the blob pretended, either by belonging to of other object boundaries or dispersion of the edge pixels due to image processing inaccuracy. The points inside the ellipse taken into consideration are named $inliers$. This algorithm, by being able to deal with a considerable amount of $outliers$, is said to be robust and focuses on using a small and feasible initial data set, enlarging it with consistent data when possible.

The first step in applying the RANSAC algorithm, when applied to ellipse fitting, is to select five points from the contour obtained in the previous section. There are selected five points because that is the minimum amount of points needed to describe a unique conic section. Next, the matrix M is computed using the vectors from the five randomly chosen points (4.12) and the unit eigenvector with the smallest eigenvalue associated in stored as a candidate for the ellipse parameters θ.

$$M = \sum_{a=1}^{5}\xi_a\xi_a^T \tag{4.12}$$

The next step classifies the ellipse as being the best one based on measuring the distance from the edge points from the ellipse that was generated. Geometric fitting refers to minimizing the sum of squares of the distance of observed points to the ellipse and is given by

$$S = \frac{1}{N}\sum_{a=1}^{N}\left((x_a - \overline{x_a})^2 + (y_a - \overline{y_a})^2\right) = d_a^2. \tag{4.13}$$

The observed (x_a, y_a) is a point from the blob's edge and $(\overline{x}_a, \overline{y}_a)$ is the closest point from the ellipse to the previous one. When the observed point is close to the ellipse the geometric distance can be approximated by (4.14) [Kanatani et al., 2016], where $V_0[\xi]$ is the covariance matrix described in section 2.2.3.

$$d^2 = (x_a - \overline{x_a})^2 + (y_a - \overline{y_a})^2 \approx \frac{(\xi, \theta)^2}{(\theta, V_0[\xi]\theta)} \tag{4.14}$$

This distance, also known as Sampson distance, is then submitted to a certain threshold for an admissible deviation from the fitted ellipse, which in this work was chosen empirically to be one pixel. The number of points from the edge that respect this threshold are then stored and the ellipse with the highest number is considered the one that fits the blob the best.

Finally, a new set of five points is then selected from the input sequence and the previous steps are repeated as can be seen in Figure 4.8.

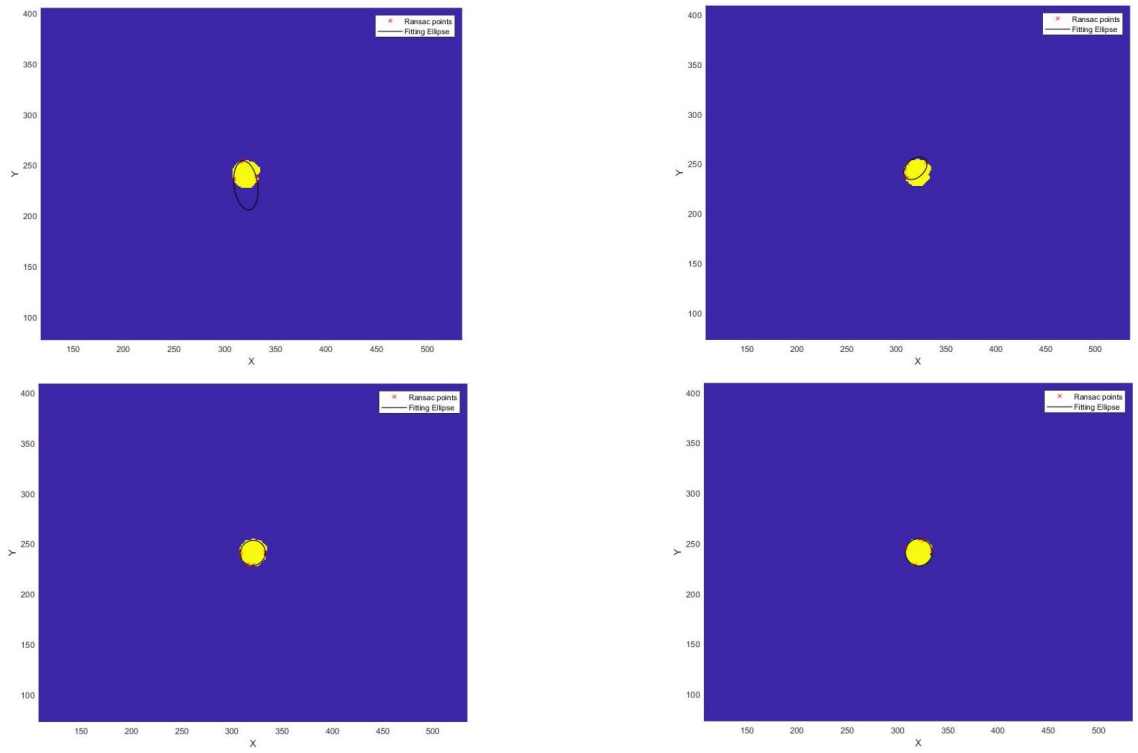


Figure 4.8: Representation of RANSAC iterations

It is computationally infeasible to test every possible sample. Therefore, a number of samples N is chosen to ensure that a probability p that one sample has no outliers is 0.99 [Hartley and Zisserman, 2003]. This number is chosen through equation below where s represents the size of the samples and ϵ

the proportion of outliers.

$$N = log(1-p)/log(1-(1-\epsilon)^s) \tag{4.15}$$

Since the proportion of outliers is unknown the worst case estimate must be assumed, meaning a high proportion of outliers must be chosen. Therefore, needing sets of five points $(s = 5)$ and a proportion of outliers of about 45% chosen, we get sets of $N = 100$ samples to be used in this work. The best ellipse will be the one with the highest number of inliers and consequently the one selected.

4.3 3D Reconstruction

The next step consists in applying the non linear transformation from the 2D to the 3D world. The method which is going to be used is monocular reconstruction, which takes advantage of the $apriori$ knowledge of the ball size and the intrinsic camera parameters to obtain a ball location from the ellipse fitted.

4.3.1 Monocular Reconstruction

Noting that the matrices Q (quadric representation of the ball) and A_k (conic representation of an ellipse in 2D) in equations (2.17) and (2.25) are symmetric, their adjoint matrices coincide with their inverses [Hartley and Zisserman, 2003, Chapter 2.2.3]. Therefore, the projection referred in (2.19) can be written as

$$A_k^{-1} \sim PQ^{-1}P^T \quad , \quad Q^{-1} = \begin{pmatrix} 1-x^2/R^2 & -xy/R^2 & -xz/R^2 & -t_x/R^2 \\ -xy/R^2 & 1-y^2/R^2 & -yz/R^2 & -t_y/R^2 \\ -xz/R^2 & -yz/R^2 & 1-z^2/R^2 & -t_z/R^2 \\ -t_x/R^2 & -t_y/R^2 & -t_z/R^2 & -1/R^2 \end{pmatrix}. \tag{4.16}$$

With the assumptions that the camera and world coordinate frame coincide, $P = K[I_3 \quad 0_3]$ with K being an invertible matrix from (2.6), the projection in (4.16) can be rewritten as

$$\begin{pmatrix} 1-x^2/R^2 & -xy/R^2 & -xz/R^2 \\ -xy/R^2 & 1-y^2/R^2 & -yz/R^2 \\ -xz/R^2 & -yz/R^2 & 1-z^2/R^2 \end{pmatrix} \sim K^{-1}A_k^{-1}K^{-T}. \tag{4.17}$$

Computing the eigenvalues from the left hand side matrix, its obtained $\lambda_1 = 1$,$\lambda_2 = 1$ and $\lambda_3 = 1 - \frac{||t||^2}{R^2}$. The two unitary eigenvalues imply that the right hand side matrix has also two equal eigenvalues but not unitary because the equality is only up to a scale factor. The next step consists on finding the scale that makes both sides equal and it only has to impose that the equal eigenvalues of the right hand side are scaled to become unitary, which can be done by sorting the three eigenvalues and selecting the middle one. Therefore, defining H as the normalized conic, $H = K^{-1}A_k^{-1}K^{-T}$, it is possible to remove the

scale ambiguity by computing the median of the eigenvalues

$$I_3 - tt^T/R^2 = H/median(eig(H)), \tag{4.18}$$

where

$$I_3 - tt^T/R^2 = \begin{pmatrix} 1-x^2/R^2 & -xy/R^2 & -xz/R^2 \\ -xy/R^2 & 1-y^2/R^2 & -yz/R^2 \\ -xz/R^2 & -yz/R^2 & 1-z^2/R^2 \end{pmatrix} \tag{4.19}$$

Next, from (4.18) we get $tt^T/R^2 = I_3 - H/median(eig(H))$ and our objective is to find the location t. If we define $H_2 = I_3 - H/median(eig(H))$ we get

$$tt^T/R^2 = H_2. \tag{4.20}$$

From (4.20) its possible to define $v = t/R$ and see that the relation $vv^T = H_2$ is true. Therefore, if we expand this last relation into matrices we obtain

$$\begin{pmatrix} v_1^2 & v_1 * v_2 & v_1 * v_3 \\ v_2 * v_1 & v_2^2 & v_2 * v_3 \\ v_3 * v_1 & v_3 * v_2 & v_3^2 \end{pmatrix} = \begin{pmatrix} H_{2(1,1)} & H_{2(1,2)} & H_{2(1,3)} \\ H_{2(2,1)} & H_{2(2,2)} & H_{2(2,3)} \\ H_{2(3,1)} & H_{2(3,2)} & H_{2(3,3)} \end{pmatrix} \tag{4.21}$$

we are interested in the diagonal that represents the vector v, consequently we get to the relation given by (4.22) where s_1, s_2 and s_3 represent the signs of the components v, extracted from H_2 as $s_1 = sign(H_2(1,3))$, $s_2 = sign(H_2(2,3))$ and $s_3 = 1$ because the z axis is pointing forward [Greggio. et al., 2011]. Note that $sign()$ returns $+1$,0 or -1 for positive, null or negative arguments respectively.

$$v = \begin{pmatrix} s_1\sqrt{H_{2(1,1)}} \\ s_2\sqrt{H_{2(2,2)}} \\ s_3\sqrt{H_{2(3,3)}} \end{pmatrix} \tag{4.22}$$

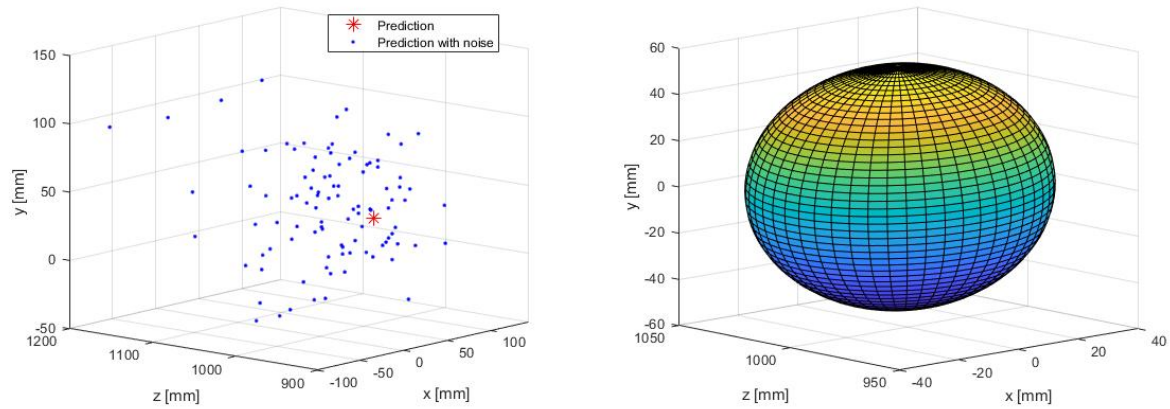
Finally, given the ball radius R, the location of the ball in 3D is a scaling of the component v given by $t = vR$.

4.4 Noise Propagation

A good fit of an ellipse is obtained after the method discussed in section 4.2.2, but there is still some uncertainty due to noise. This uncertainly will be reflected in the 3D reconstruction of the ball in 3D. Therefore, we develop a method to propagate the uncertainty from the 2D image to the 3D world. Our goal is to estimate the probability density of the position of the ball in 3D, to be used in probabilistic 3D tracking methods (e.g. Kalman Filter (KF) or Particle Filter). We use a Gaussian approximation for the probability density, thus our goal is to estimate the expected value and the covariance of the ball.

4.4.1 Monte Carlo

The first method applied is Monte Carlo (MC) estimation. The underlying concept relies on using randomness to solve a problem, which in this case is the characterization of the uncertainty associated with a 2D to 3D reconstruction. This method consists in selecting an input variable, which in this case consists in the parameters of the affine transformation of the ellipse fitted to the blob. Next, inputs are generated randomly from the probabilistic distribution of that variable. In this work we consider a independent Gaussian distribution with mean 0 and a standard deviation σ for the parameters of affine transformations of the ellipse. The inputs would then be put through the reconstruction mentioned in section 4.3 and the results would be aggregated in the 3D world as is represented in Figure 4.9. This method can be used to solve any problem that requires a probabilistic interpretation, although being expensive in terms of computation, and it can be applied in the estimation of covariance from an unknown transformation. This is made with the assumption that the true values (Ground Truth (GT)) are sufficiently close to the estimated values so that the covariance matrix does not present a bias and represents as best as possible the transformation [Hartley and Zisserman, 2003].



(a) Predictions with noise

(b) Covariance obtained from predictions

Figure 4.9: Example of the application of the MC

4.4.2 Unscented Transform

Another way to achieve a similar result with the method described in section 4.4.1 is through the Unscented Transform (UT). This method relies in approximating a probability distribution using carefully selected test points. These test points denominated $sigma\ points$, are propagated through a non linear transformation and allow the estimation of the mean and covariance.

A set of $2n + 1$ sigma points, where n is the dimension of the random variable to be propagated, are generated and each one is represented by $S_i = \{X_i, W_i\}$ in which X_i are points coordinates given

by equations (4.23) and W_i are weights given by equations (4.24), where P^x is the covariance of the variable.

$$\begin{aligned} X_0 &= \bar{x} \\ X_i &= \bar{x} + (\sqrt{(n+\lambda)P^x})_i \qquad i = 1, ..., n \\ X_i &= \bar{x} - (\sqrt{(n+\lambda)P^x})_i \qquad i = 1+n, ..., 2n \end{aligned} \tag{4.23}$$

$$\begin{aligned} W_0^{(m)} &= \lambda/(n+\lambda) \\ W_0^{(c)} &= \lambda/(n+\lambda) + (1-\alpha^2+\beta) \\ W_i^{(m)} &= 1/(2(n+\lambda)) \qquad i = 1, ..., 2n \\ W_i^{(c)} &= 1/(2(n+\lambda)) \qquad i = 1, ..., 2n \end{aligned} \tag{4.24}$$

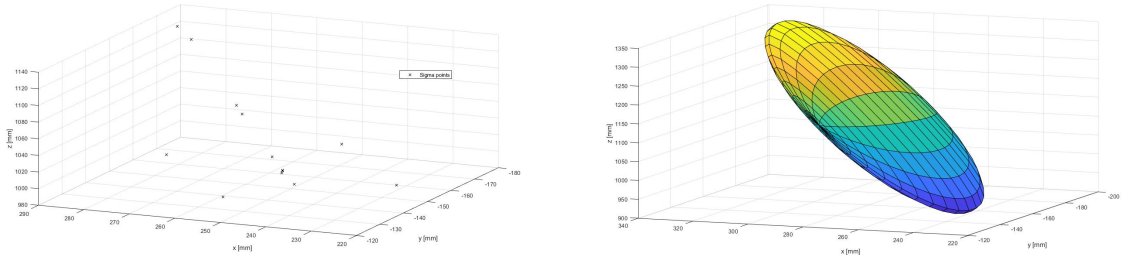
In the previous equations λ is chosen as $\lambda = \alpha^2(n+\kappa) - n$ [Wan and Van Der Merwe, 2000]. The κ is a parameter for scaling the sigma points towards or away from the mean $\bar{x}$ and can be any value as long it respects the condition $(n + \kappa \neq 0)$ [Julier et al., 2000], therefore, in our case, it is set to 20 so that there are no negative weights regarding the mean or covariance which are proven to be susceptible to a variety of numerical errors. The parameter α determines the spread of the sigma points around $\bar{x}$ and it is set to 0.5 so that we can have well spread sigma points in order to be able to properly estimate the covariance. Finally, β is a weighting term used to incorporate prior knowledge of the distribution, being set to 2 for an optimal Gaussian prior. Each sigma points is then propagated through the non linear function, which in our case is the monocular reconstruction $g()$,

$$\gamma_i = g(X_i) \qquad i = 0, ..., 2n_x \tag{4.25}$$

such that the mean and covariance can be given by

$$\begin{aligned} \bar{y} &= \sum_{i=0}^{2n} W_i^{(m)} \gamma_i \\ P^\gamma &= \sum_{i=0}^{2n} W_i^{(c)} (\gamma_i - \bar{y})(\gamma_i - \bar{y})^T \end{aligned} \tag{4.26}$$

where γ_i are the sigma vectors propagated through the non linear function and $\bar{y}$ is the weighted mean. This method, which is represented in Figure 4.10, differs substantially from the MC which on one hand requires more sample points in an attempt to propagate an accurate distribution but is also able to represent any distribution whether this method is more adequate for unimodal distributions. It consists in a trade off between computation power and accuracy that will be represented in the results.



(a) Sigma points

(b) Covariance obtained from the sigma points

Figure 4.10: Example of the application of the UT

4.5 Tracking

Finally, we are left with estimates of the 3D ball position and their respective covariances, therefore we are able to apply into a tracker and the one selected for this work was the KF.

4.5.1 Kalman Filter

The KF uses a system dynamic model, known controls inputs of that system and multiple measurements in order to form an estimate that can not be measured directly. Noisy sensor data, approximations in the equations that characterize the system evolution and external factors, like occlusions or collisions, limit how well it is possible to determine the next state. It produces an estimate of the state based on a weighed average between the system previous predicted state and the current measurement. The weights are calculated from the covariance, which in this work comes from the noise propagation through the non linear transformation from 2D to 3D.

In order to understand how the KF works, it is necessary to specify the motion and observation models that are going to be used. Firstly, the KF assumes an evolution from the state at an instant k from $k-1$ according to

$$x_{k+1} = F_k x_k + B_k u_k + w_k, \tag{4.27}$$

where F_k is the state temporal transition model applied to the current state x_k, B_k the control applied to the entrance of the system u_k, and w_k the process noise, assumed to be drawn from a multivariate normal distribution with covariance Q_k as $w_k \sim \mathcal{N}(0, Q_k)$. It is also assumed that is possible to obtain a measurement of the system, given by

$$z_k = H_k x_k + v_k \tag{4.28}$$

where H_k is the observation model representing the used sensor and v_k is the observation noise assumed to be drawn from a Gaussian distribution with mean zero and covariance R_k, which in our case is the one calculated through noise propagation, given as $v_k \sim \mathcal{N}(0, R_k)$.

4.5.1.A Linear Kalman Filter

The linear KF is the simplest form and it is a recursive estimator, meaning that the previous estimated state and the current measurement are required to obtain the estimate to the current state. It divides itself into two parts, in order to propagate the mean of the state and its covariance: the predict and update steps.

The predict step uses the state estimate from the previous time $\hat{x}_{k-1|k-1}$ step to produce an estimate *a priori* of the current state $\hat{x}_{k|k-1}$. It is denominated $a\ priori$ because although it is an estimate of the current state it does not withhold the information about the observation of the current state. This section is characterized by the following equations (4.29), where $P_{k|k-1}$ is the state $a\ priori$ covariance and $P_{k-1|k-1}$ is the state covariance of the previous time step.

$$\hat{x}_{k|k-1} = F_k\hat{x}_{k-1|k-1} + B_k u_k \tag{4.29a}$$

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k \tag{4.29b}$$

The next part is the update step where the current estimate from (4.29a) is combined with the current measurement in order to produce a refined current state estimate. With the measurement z_k available the update is done via the following equations

$$K_k = P_{k|k-1}H_k^T(H_k P_{k-1|k-1} H_k^T + R_k)^{-1} \tag{4.30a}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(z_k - H_k\hat{x}_{k|k-1}) \tag{4.30b}$$

$$P_{k|k} = (I - K_k H_k)P_{k|k-1} \tag{4.30c}$$

where K_k is the kalman gain and $\hat{x}_{k|k}$ and $P_{k|k}$ are the current state estimate and covariance with the measurement included [Li et al., 2015]. The kalman gain weights the estimate and the measurement. If the gain is high, the filter gives more weight to the recent measurement, if it is low the model prediction is followed more closely. The estimate and covariance are then used on the next state in order to attain a new estimate and a new covariance which can be denominated $x_{k+1|k}$ and $P_{k+1|k}$. Concluding, assuming good knowledge of the system and noise models, the KF will converge for the best possible state estimate.

5

Implementation

Contents

This chapter will focus on explaining on how the approaches considered in Chapter 4 were applied. It will also mention how the experimental setup was designed and implemented, in order to test the algorithm before applying it to real scenarios.

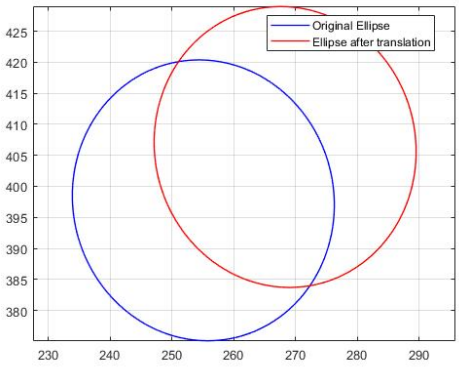
5.1 Noise propagation

The propagation of noise is a necessary tool in order to propagate uncertainty from 2D to 3D using the methods mentioned in section 4.4. There are several ways to implement noise. Firstly, noise could be implemented in the points of the ellipse, but since the intended method would be to use a normal Gaussian distribution with mean zero, the final ellipse would be too similar to the one without noise. Furthermore, noise could be implemented directly into the ellipse parameters (center, axes and angle). This method would involve the conversion of all ellipses into these parameters, since they were represented in matrices, which would just require unnecessary computation, where the same result could be obtained on applying noise in the parameters of an affine transformation of the ellipse, which was the one used in this work. The four transformations that were taken into consideration were translation, rotation, scaling and shear. The order of transformations matters, because the result might vary if the same geometric transformations are applied in different order. Therefore, the order selected was (i) translation, which corresponds to moving every point of the ellipse the same direction and distance, (ii) scaling where an enlargement or shrinking occurs based on a scalar, (iii) rotation, where the ellipse is rotated based on the origin and finally (iv) shear, where the shape of the ellipse is slanted in a specific coordinate. The translation and rotation are transformations that belong in the isometries transformations class and preserve the Euclidean distance. These type of transformations are the most specialized and model the motion of a rigid object. An isometry preserves the length, angle and area of the object applied to and can be seen in Figures 5.1a and 5.1c. Next, the scaling belongs to the similarity class. This class gets its name because it preserves the ”shape” of the object, therefore it preserves the angles and the ratios between lenghts and areas. An example can be seen in Figure 5.1b. Finally, we have the affine transformations class, where the shear fits in. This class is characterized by preserving parallel lines, their ratios and ratios of areas and is portrayed in Figure 5.1d. The matrix representation of these transformations are

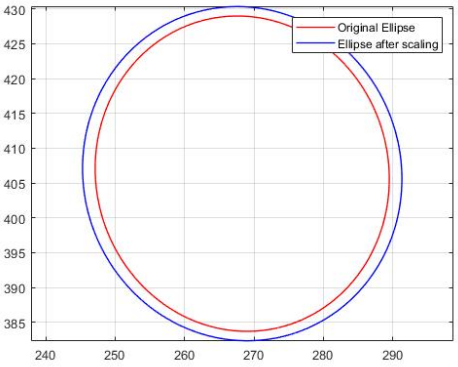
$$H_{tra} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix}, H_{sca} = \begin{pmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{pmatrix}, H_{rot} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}, H_{she} = \begin{pmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \quad (5.1)$$

Note that shear, in this work, was selected to be applied in the x coordinate. The translation was based on a normal Gaussian distribution as $\mathcal{N}(0, 5^2)$ so that 99% of the values are contained in the

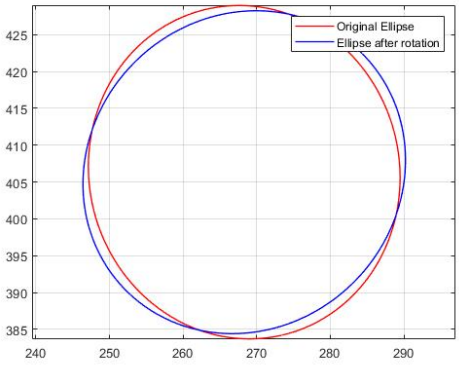
interval $[-20, 20]$. In order to distribute noise in the scaling parameter using a Gaussian distribution, the logarithm of the scale was used as $\mathcal{N}(0, \log_e(1.1)^2)$, where the value were then extracted using the exponential function. Next, the rotation angle used to introduce noise was generated by $\mathcal{N}(0, 5^2)$ so 99% of the interval values would fall between $[-15, 15]$. Finally, the value for shear was obtained from $a = \mathcal{N}(0, 0.01^2)$.



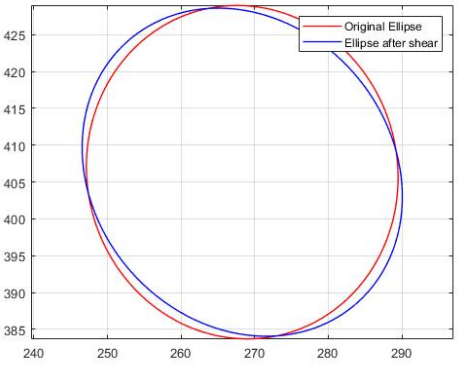
(a) Translation



(b) Scaling



(c) Rotation



(d) Shear

Figure 5.1: Transformations applied to the ellipses

5.1.1 Composite transformation

Using all the transformations described above and combining them, it is possible to create a single transformation which is going to be used to generate noisy samples.

$$H_{noise} = H_{tra}H_{sca}H_{rot}H_{she} = \begin{pmatrix} s\cos\theta & s\,a\cos\theta - s\sin\theta & t_x \\ s\sin\theta & s\,a\sin\theta + s\cos\theta & t_y \\ 0 & 0 & 1 \end{pmatrix} \tag{5.2}$$

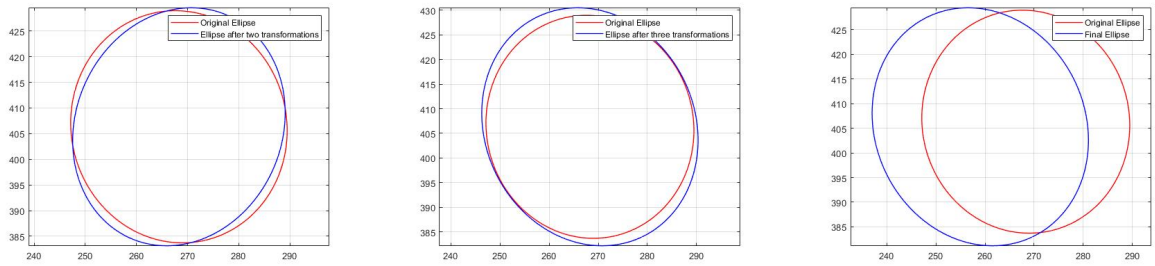
Since the rotation is based on the center of the coordinate system there is also the necessity to translate the ellipse to the origin so that the ellipse can rotate on its center. Therefore, we have

$$H_0 = \begin{pmatrix} 1 & 0 & -q_x \\ 0 & 1 & -q_y \\ 0 & 0 & 1 \end{pmatrix} \tag{5.3}$$

where $[q_x \quad q_y]$ represent the center of the ellipse. The goal was to apply a variable transformation in the center of the object by changing the space and then return it to the original space. A step by step example is represented in Figure 5.2 . Therefore using the equation defined in (2.18), the desired transformation for noise propagation is represented through the homogeneous equation in

$$m^T H_0^T H_{noise}^T H_0^{-T} A_k H_0^{-1} H_{noise}^{-1} H_0 m = 0. \tag{5.4}$$

In Appendix A it is possible to find a more detailed explanation on how to the equation 5.4 is reached.



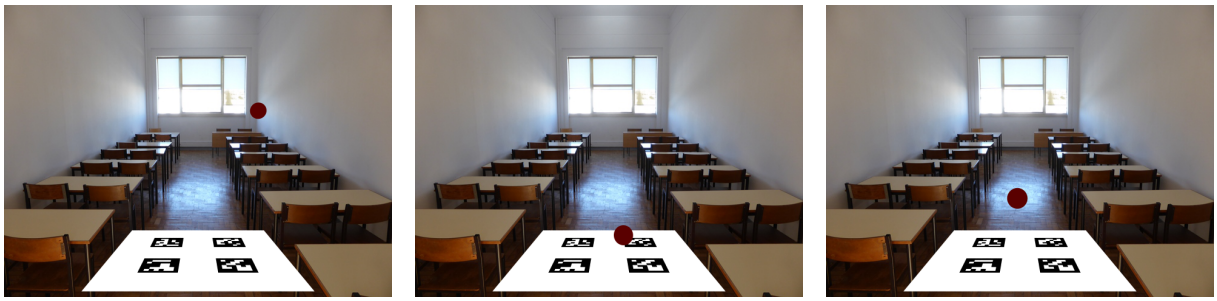
(a) **(b)** **(c)**

Figure 5.2: Complete affine transformation divided by steps $(a) \rightarrow (b) \rightarrow (c)$

5.2 Simulation

In order to validate the algorithm, ground truth trajectories from a simulator were made. These trajectories were composed of a sequence of images showing the different states of the ball. These frames were created by a simulator which was developed by [Pereira, 2020] and implemented in C++ with the aid of the functionalities of the the library $openCV$. This simulator took into consideration the equation of

a sphere and the projection of itself in the image plane. Through this method, it colored only the pixels that corresponded correctly to the projection of a ball moving in the world frame. The simulator also had a projection of a plane in order to simulate scenarios with ball collisions, as in Figure 5.3. The markers have the objective to identify the obstacle in the ball trajectory.



(a) (b) (c)

Figure 5.3: Example of frames generated by the simulator corresponding to a free fall trajectory

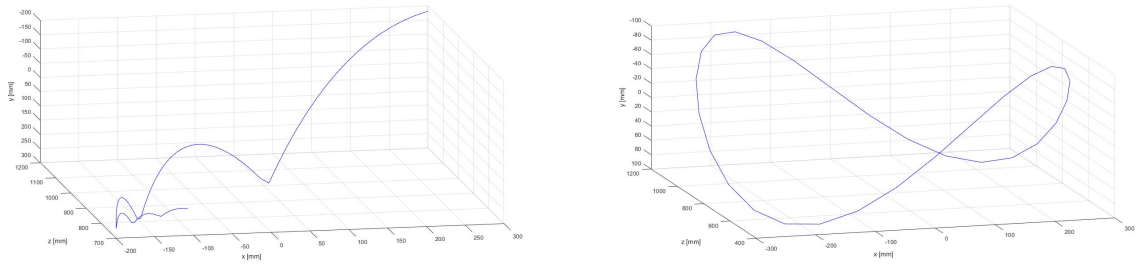
Since these frames were generated artificially, it is possible to know the exact position of the ball. The ground truth trajectories were generated in order to test the efficiency of the algorithm developed. With the known location of the ball it is possible to extract the accuracy of the tracking method implemented. The trajectories were based on motion models, such as the ones represented in (5.5). Regarding (5.5a) the parameters p_i and v_i correspond to the position and velocity in the instant $t = t_i$. Also $\dot{x}(t_{i+1})$ and V_{i+1} are the ball velocity before and after the impact in the instant $t = t_{i+1}$. Finally a refers to acceleration, e the coefficient of restitution and n the normalized vector in the direction of the line of impact. Concerning (5.5b), $[x_0, y_0, z_0]$ corresponds to the initial position, A_x, A_y and A_z are off set variables and f is the frequency.

$$k(t) = \begin{cases} p_i + v_i(t - t_i) + \frac{a}{2}(t - t_i)^2, & t_i < t < t_{i+1} \\ p_{i+1} = x(t_{i+1}), \quad v_{i+1} = \dot{x}(t_{i+1}) - (1 + e)(\dot{x}(t_{i+1})^T n)n, & t = t_{i+1} \end{cases}, \quad i = 0, 1, 2, ... \tag{5.5a}$$

$$k(t) = [x(t) = x_0 + A_x \sin(2\pi f t), \quad y(t) = y_0 + A_y \cos(2\pi f t), \quad z(t) = z_0 + A_z \cos(2\pi f t)] \tag{5.5b}$$

The first trajectory simulates a ball in free fall, colliding with an obstacle and with constant lateral wind. It was used to analyze the behaviour regarding collisions of a ball and it is described by the Equations (5.5a), which consists in the free fall equation and the change in velocity after an impact, in an instant t_{i+1}, with e as a coefficient of restitution. The trajectory consisted in 75 frames and had for parameters $p_0 = (300, -180, 1200)mm$, $v_0 = (-1000, 0, -1000)mm/s$, $a = (1000, 9810, 1000)mm/s^2$ and $e = 0.7$. The second trajectory consisted in 100 frames of a curved trajectory described by (5.5b), with parameters f=1Hz, x_0=y_0=$0mm$, z_0=$800mm$, A_x=$300mm$, A_y=$100mm$, A_z=$400mm$ and was used to evaluate the propagation of the error in every coordinate associated with the monocular reconstruction method. In

both of them a ball with $30\,mm$ radius was used and their ground truth trajectories can be seen in Figure 5.4.



(a) **(b)**

Figure 5.4: Ground truth trajectories generated by the simulator

5.3 Algorithm Description

The system was implemented in Matlab and this section will provide a brief description of the main algorithms implemented. First, the image processing was done using the Image Processing Toolbox in Matlab. The main functions used to perform the image segmentation, binarization and contour extraction were imbinarize(), imdilate(), imopen() and bwboundaries(). Furthermore, a function bwlargestblob() was also used to select only the largest blob on a binarized image to remove any possible noise that may have survived the morphological operators.

Next, comes the ellipse fitting, where the Random Sample Consensus (RANSAC) algorithm returns the parameter θ of the best possible fitted ellipse and the number of contour points n from the blob that pass the condition described in (4.2.2), using the coordinates returned by the bwboundaries() function.

Algorithm 5.1 RANSAC

Result: $(\theta, n) = RANSAC(\xi, d, V_0[\xi])$
ξ_{rand}=randperm(ξ)
$M = \sum_{a=1}^{5} \xi_{rand}(a)\xi_{rand}^{T}(a)$
$\theta = eigs(M, 1,' SM')$
$n = 0$
if $\theta(2)^2 - 4 * \theta(1) * \theta(3) \geqslant 0$ **then**
  $\theta = [0; 0; 0; 0; 0; 0]$
else
  for i=1:1:length(ξ) **do**
    A = $\frac{(\xi,\theta)^2}{(\theta,V_0[\xi]\theta)}$
    if $A < d^2$ **then**
      $n = n + 1$
    else
      $n = n$
    end
  end
end

RANSAC is a new function that uses randperm(), which selects a random permutation without repeating elements, to select five random points from the contour points and eigs() to return the eigenvector with the smallest eigenvalue. Note that length() returns a number corresponding to the length of the variable. After obtaining the ellipse, the following step consists in spreading noise through an affine transformation. A function was created to accomplish that and takes as arguments the ellipse parameters θ and the matrices of the ellipse transformations, returning the ellipse parameters with the associated noise θ_noise with the corresponding matrix of the ellipse parameters A_k. The $Ellipse\_param()$ uses Equations from (2.23) to return the ellipse center based on its parameters (A,B,C,D,E,F).

Algorithm 5.2 Noise propagation

Result: $(\theta\_noise, A_k) = Homografia\_noise(\theta, H_r, H_s, H_t, H_{she})$
$[x_0.y_0] = Ellipse\_param(\theta)$
$Define\ translation\ to\ origin\ [x_0, y_0]$ (5.3)
$Composite\_Trans = H_t H_s H_r H_{sh}$
$Obtain\ A_k\ after\ complete\ transformation\ through\ Equation$ (5.4)
$Obtain\ \theta\_noise\ from\ A_k$

The next step relies on applying the monocular reconstruction to all the ellipses, the original one and the ones with noise applied to. It returns the 3D position of the ball t and has as inputs the ellipse parameters θ, the intrinsic camera parameters K and the ball radius R. $Theta\_to\_C()$ is a function that converts the vector with the ellipse parameters into the matrix representation that is represented in Equation (2.25).

Algorithm 5.3 Monocular reconstruction

Result: $(t) = Monocular\_recons(\theta, K, R)$

if $\theta == [0; 0; 0; 0; 0; 0]$ **then**

  $t = [0, 0, 0]$

else

  $A_k = theta\_to\_C(\theta)$

  $H = K^{-1}C^{-1}K^{-T}$

  $H_2 = eye(3) - (H/median(eig(H)))$

  $v = [sign(H_2(1,3) * sqrt(H_2(1,1)))\ ;\ sign(H_2(2,3) * sqrt(H_2(2,2)))\ ;\ sign(H_2(3,3) * sqrt(H_2(3,3)))]$

  $t = R * v;$

Finally, in the Monte Carlo (MC) case, the covariances are obtained from a function that takes in as parameters the 3D locations of the ball and returns the covariance matrix, where $\bar{t}$ represents the average of the outputs returned by monocular reconstruction.

$$Cov_{matrix} = \sum_{i=0}^{N} \frac{(t_i - \bar{t})(t_i - \bar{t})^T}{N} \tag{5.6}$$

In the Unscented Transform (UT) case, the resulting covariances are calculated through a function that takes in the ellipse parameters θ, the camera intrinsic parameters K and the ball radius R. It returns the covariance associated with the ellipse used as input. The noise is applied the same way as in the MC method so that all the noise applied can have a Gaussian distribution, which is a requirement for the UT, and both methods use as an input the same variables with similar variance. The noise is applied in translation, rotation, scaling and shear, existing therefore five parameters. The average $\hat{x}$ consists on the values without noise $[t_x, t_y, s, angle, shear]$. The variables used were discussed in section 4.4.2.

Algorithm 5.4 Unscented Transform

Result: $(Covariance) = UT(\theta, K, R, P_x)$
$Mean = 0,\ L = 6,\ k = 0,\ \alpha = 0.5,\ \beta = 2$
$\lambda = \alpha^2(n + k) - n$
$\hat{x} = [0, 0, 1, 1, 0, 0]$
if $k = 0$ **then**
  $X_k = \hat{x}$
  ${W_k}^{(m)} = \lambda/(n + \lambda)$
  ${W_k}^{(c)} = \lambda/(n + \lambda) + (1 - \alpha^2 + \beta)$
else
  if $k > 0$ && $k \leqslant n$ **then**
    $X_k = \hat{x} + \sqrt{(n + \lambda)P_x}$
    ${W_k}^{(m)} = {W_k}^{(c)} = 1/\big(2(n + \lambda)\big)$
  else
    $X_k = \hat{x} - \sqrt{(L + \lambda)P_x}$
    ${W_k}^{(m)} = {W_k}^{(c)} = 1/\big(2(n + \lambda)\big)$
  end
end
$Get\ transformation\ matrices(H_r, H_s, H_t, H_{she})\ from\ X_k$ (5.1)
$\theta_{noise} = Homografia\_noise(\theta, H_r, H_s, H_t, H_{she})$
$t_i = Monocular\_recons(\theta_{noise}, K, R)$
for $i = 1 : 1 : length(k)$ **do**
  $Mean = Mean + t_i * W_i^{(m)}$
end
for $i = 1 : 1 : length(t)$ **do**
  $Covariance = Covariance + W_i^{(c)} * (t_i - Mean) * (t_i - Mean)'$
end

These covariances are then applied to a function that implements a kalman filter to predict the location of the ball. It takes in as arguments the process noise covariance Q_k, the observation noise covariance R_k, which is replaced by the covariances mentioned early or a fixed covariance with the same order of magnitude, the measurement z_k, the covariance of the first state P_0 and the same with the position x_0. It returns the predicted location of the ball and the covariance associated with that state.

Algorithm 5.5 Kalman Filter

Result: $(x_{1:k}, P_{1:k}) = KF(Q_{1:k}, R_{1:k}, z_{1:k}, x_0, P_0)$
for $k = 1 : 1 : length(z_{1:k})$ **do**
  if k=1 **then**
    $x_k = x_0 \quad P_k = P_0$
  else
    $x_k = F_k x_{k-1}$
    $P_k = F_k P_{k-1} F_k^T + Q_k$
    $S_k = H_k P_k H_k^T + R_k$
    $K_k = P_k H_k^T S_k^{-1}$
    $x_k = x_k + K_k(z_k - H_k x_k)$
    $P_k = (I - K_k H_k) P_k$
  end
end

6

Results

Contents

In order to validate the algorithm a particular set of tests were performed. First the trajectories provided by the simulator were analyzed in order to obtain the measurement and the error between it and the ground truth positions. This was done also with the real scenarios to prove its performance in the real world. After obtaining the measurements of the ball location, noise was then spread around the correspondent 2D estimation of those measurements and the correspondent covariance matrices were obtained from the methods described earlier. Both of these parameters, measurements and covariance matrices, served as input to track the ball along the frames of the videos used. In the tracking environment the covariances were compared with a fixed covariance, that was selected based on the Monte Carlo (MC), which was though to be optimal. This was also done in simulated and real scenarios to see the disparities provoked by this change. The real experiments were executed with two different balls: a red ball with $30mm$ radius and a green ball with $35.85mm$ radius. The metric used to evaluate the performance of the system was the Mean Squared Error (MSE) where Y_i is the Ground Truth (GT) and $\hat{Y_i}$ is the predicted location. Note that this is done independently for each coordinate and with the norm as $||(Y_i - \hat{Y_i})||^2$ for the total MSE.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y_i})^2 \tag{6.1}$$

This metric accesses the quality of an estimator by returning the average of the squared error between the true position of the ball and its estimated location. It will also be calculated the average error and the variance of the estimation error. This will be done through

$$Avg\_error = mean(Y_i - \hat{Y_i}), \quad Variance = mean(Error - Avg\_error)^2 \tag{6.2}$$

in order to compare it with the variances obtained from the MC and Unscented Transform (UT) methods.

6.1 Simulator

The 3D measurements, regarding the simulated trajectories, which were obtained from the algorithm 5.3 had as intrinsic camera parameters

$$K = \begin{pmatrix} 500 & 0 & 320 \\ 0 & 500 & 240 \\ 0 & 0 & 1 \end{pmatrix} \tag{6.3}$$

and are represented in Figures 6.1 and 6.3. Their respective errors associated with each coordinate are in Figures 6.2 and 6.4. The inversion of the y axis is for visual representation.

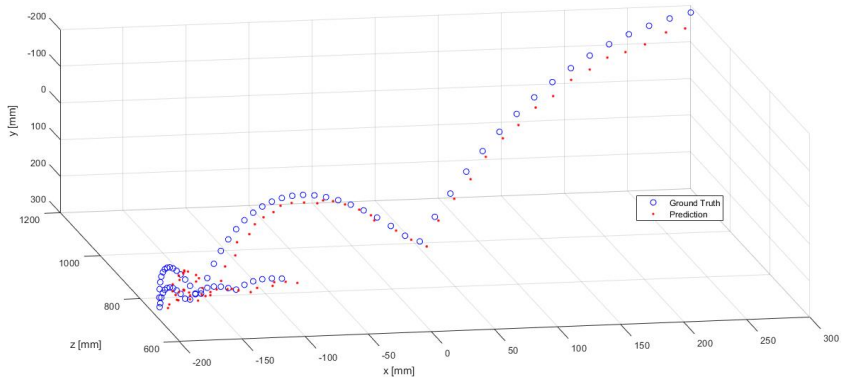
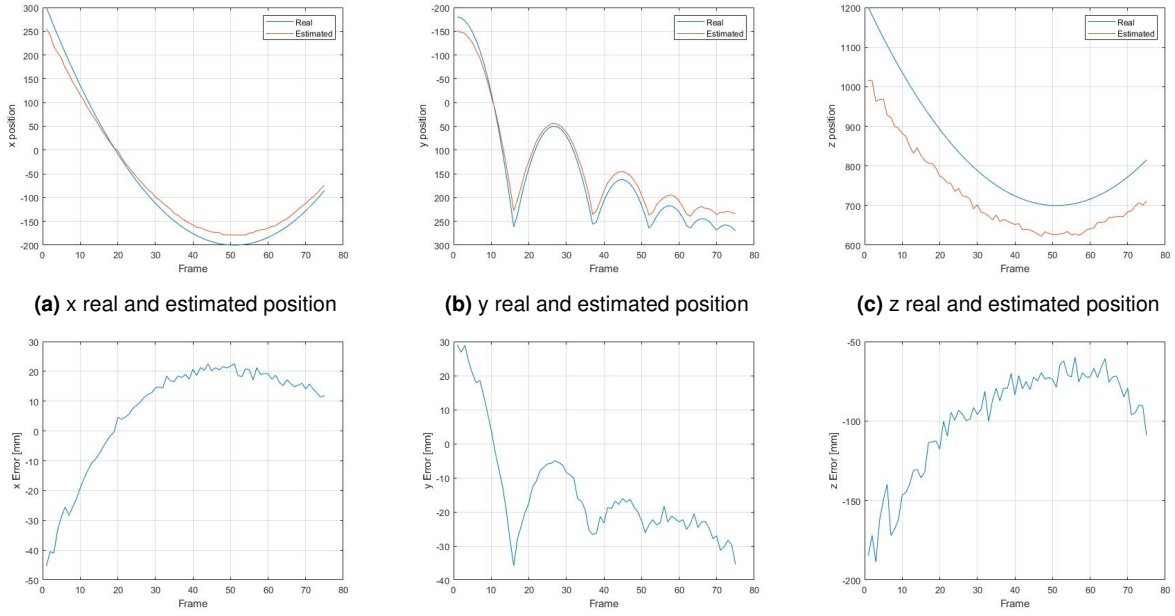


Figure 6.1: 3D Reconstruction measurements of the red ball location on the first trajectory



(a) x real and estimated position **(b)** y real and estimated position **(c)** z real and estimated position

(d) x error **(e)** y error **(f)** z error

Figure 6.2: 3D Reconstruction measurement errors of the red ball throughout the frames in the first simulated trajectory

On the first trajectory the errors associated with x and y are pretty similar. This does not happen in the second trajectory, because there is a greater variation in the x coordinate comparing to the y coordinate regarding the optic center. This leads to an increase in error, because the further away from the optic center the ball is, the worst its representation by the ellipse is. It is also possible to observe that the z coordinate, in both trajectories, is the one with the highest error associated, as expected, due to the use of a single camera.

There is a bias associated with this coordinate, represented in Table 6.1,

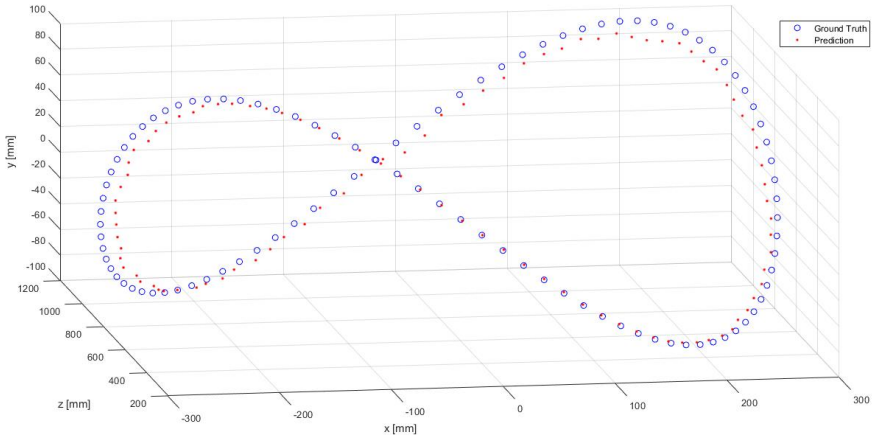
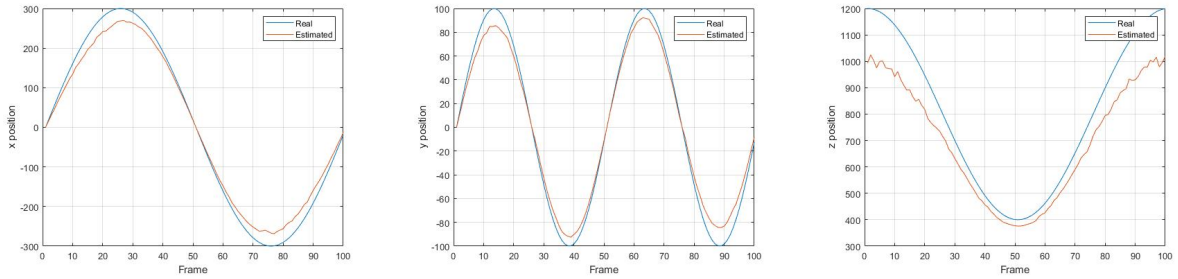
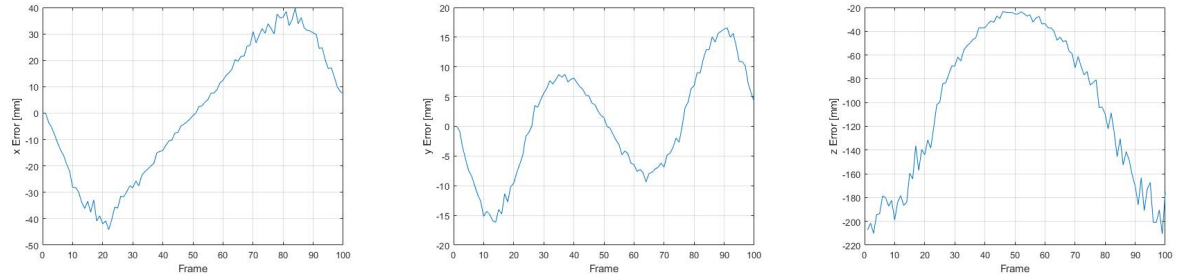


Figure 6.3: 3D Reconstruction measurements of the red ball location on the second trajectory



(a) x real and estimated position **(b)** y real and estimated position **(c)** z real and estimated position



(d) x error **(e)** y error **(f)** z error

Figure 6.4: 3D Reconstruction measurement errors of the red ball throughout the frames in the second simulated trajectory

Table 6.1: Average error and variance of the estimation error of the simulated scenarios regarding the red ball

	Average Error $[mm]$			Variance $[mm]^2$		
	x	y	z	x	y	z
6.1	0,35	0,42	-100,97	618,25	76,54	3847,9
6.3	6,95	-14,39	-100,86	337,03	253,67	1198,8

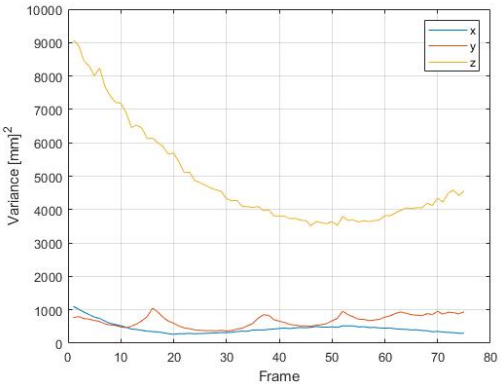
and a reason for this could be the loss of pixel information regarding the contour of the ball in the image segmentation. This leads to a different blob size than the real one, where the fitted ellipse doesn't

represent well the true size of the ball, leading to a worst prediction of the true location of the ball. The corresponding MSE for each coordinate is represented in Table 6.2.

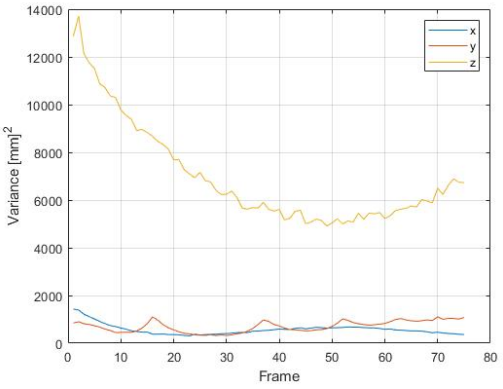
Table 6.2: 3D Reconstruction MSE for the simulated trajectories

MSE	x $[mm]^2$	y $[mm]^2$	z $[mm]^2$
6.1	379,3	466,8	11295,1
6.3	615,12	76,1	14111,7

On Figure 6.5 and 6.6 it is possible to observe the variances obtained from the MC and UT methods. Their values are similar throughout the frames with a slight increase on the z variance from the UT. By comparing them with the Table 6.1 is possible to conclude that their values are higher than the variance of the estimation error. This is due to the good image segmentation obtained from the simulated images, which leads to a lower error between the GT and prediction and therefore leading to a lower variance.

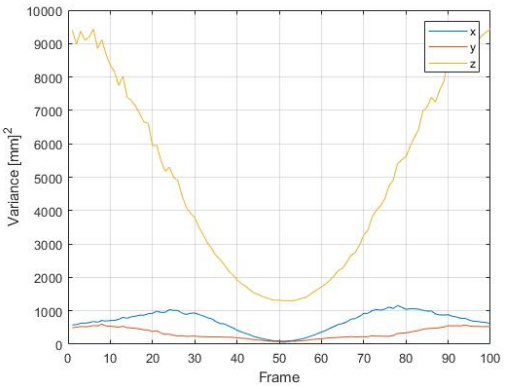


(a) Variance from the covariance matrices obtained from MC

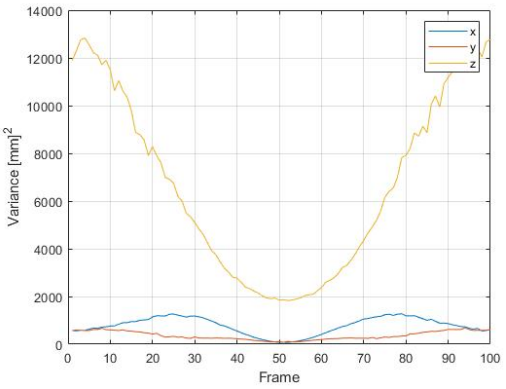


(b) Variance from the covariance matrices obtained from UT

Figure 6.5: Variances of the estimation error for the 6.1 trajectory



(a) Variance from the covariance matrices obtained from MC



(b) Variance from the covariance matrices obtained from UT

Figure 6.6: Variances of the estimation error for the 6.3 trajectory

6.2 Real Scenarios

Two situations were analyzed in the real scenarios. First with a red ball to be as close as possible with the scenarios simulated in the previous section. Next, with a green ball to see if there would be any significant changes in using a ball with a different color. It is important to keep in mind that the ground truth trajectories for the real tests were obtained manually, this means that the coordinates from the 3D position were tuned so that the ball would coincide its projection in the image frame. This implies that the values obtained can only give an indication of the accuracy due to the lack of an absolute GT. The intrinsic camera parameters used for the real scenarios were

$$K_f = \begin{pmatrix} 497,01 & 0 & 372,588 \\ 0 & 496,392 & 231,468 \\ 0 & 0 & 1 \end{pmatrix} \quad K_p = \begin{pmatrix} 647,099 & 0 & 375.745 \\ 0 & 649.231 & 233.407 \\ 0 & 0 & 1 \end{pmatrix} \tag{6.4}$$

for the red ball regarding the free fall and pendulum trajectory respectively. For the green ball they were

$$K_f = \begin{pmatrix} 475.316 & 0 & 357.815 \\ 0 & 478.232 & 223.064 \\ 0 & 0 & 1 \end{pmatrix} \quad K_p = \begin{pmatrix} 501.698 & 0 & 373.208 \\ 0 & 502.41 & 246.066 \\ 0 & 0 & 1 \end{pmatrix} \tag{6.5}$$

for the free fall and pendulum trajectories.

6.2.1 Red Ball

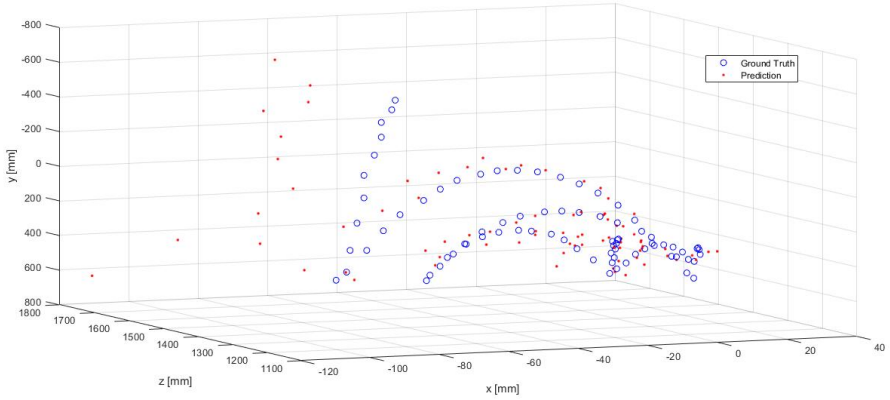
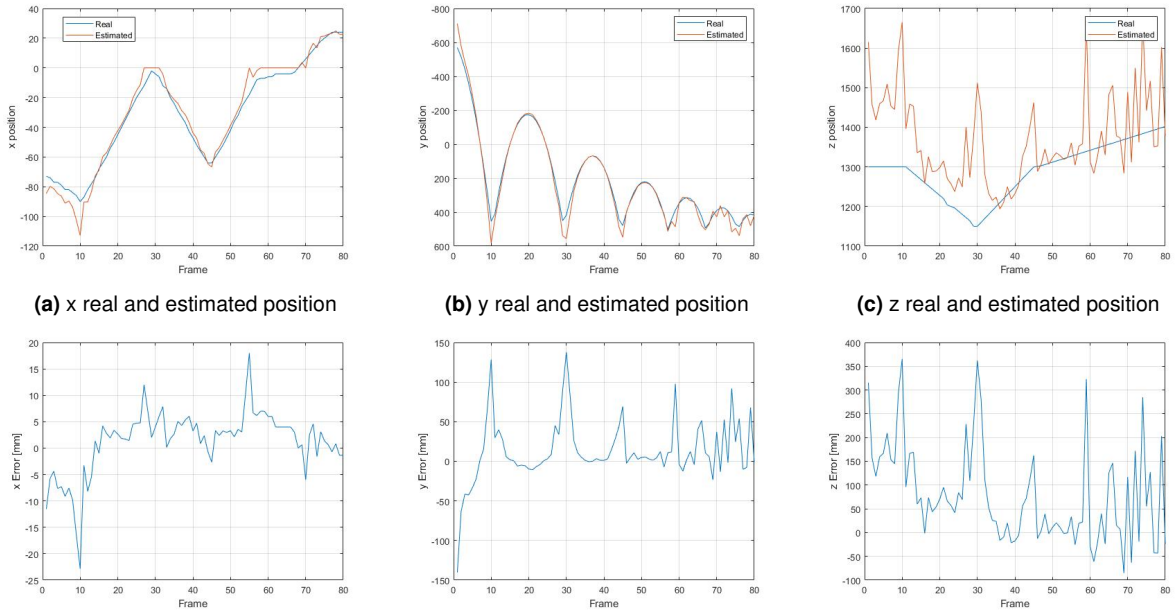


Figure 6.7: 3D Reconstruction measurements of the red ball location on the free fall trajectory



(a) x real and estimated position **(b)** y real and estimated position **(c)** z real and estimated position

(d) x error **(e)** y error **(f)** z error

Figure 6.8: 3D Reconstruction measurement errors of the red ball throughout the frames throughout the free fall trajectory

On the previous figures it is possible to see how the error is correlated with depth z. It is also possible to conclude that the spikes in the error y are due to the collisions of the ball with the floor, which leads to a deformation and consequently to a not so accurate representation of the ball through an ellipse. The initial error in the y coordinate is a consequence of a partial occlusion of the ball by a hand.

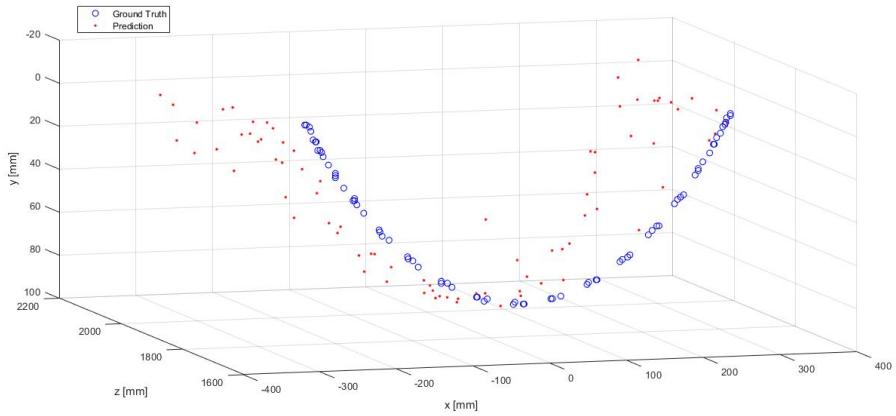
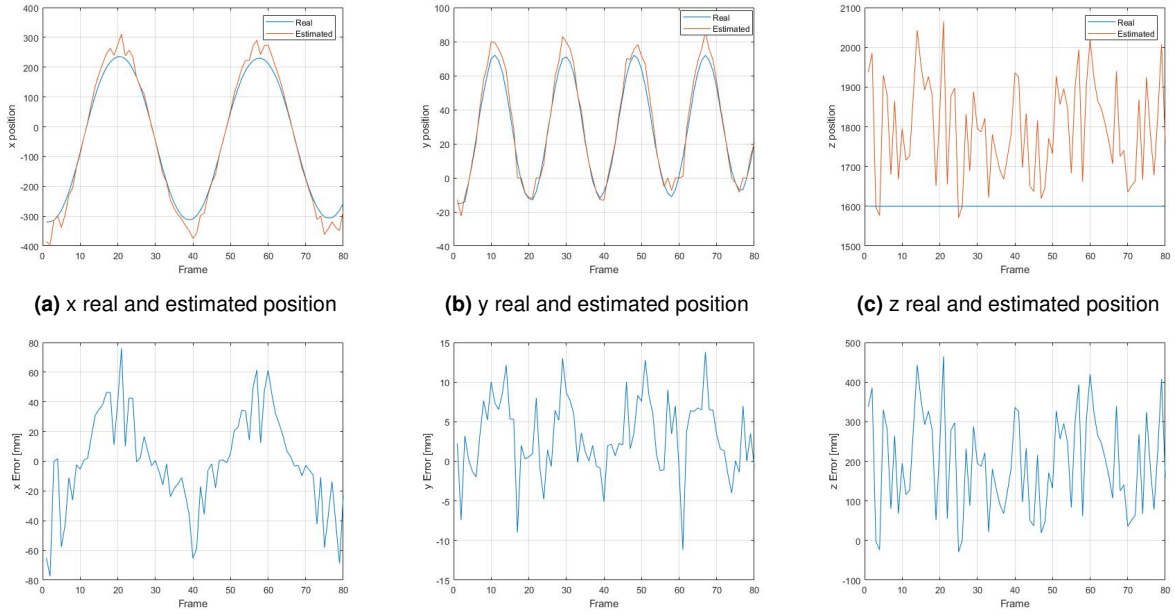


Figure 6.9: 3D Reconstruction measurement of the red ball location on the pendulum trajectory



(a) x real and estimated position

(b) y real and estimated position

(c) z real and estimated position

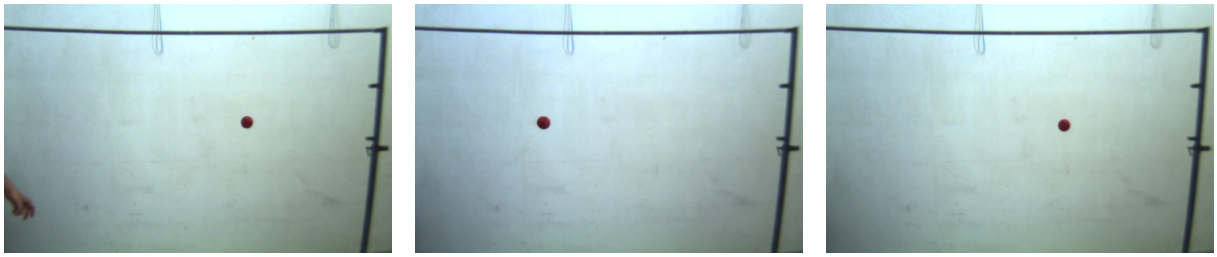
(d) x error

(e) y error

(f) z error

Figure 6.10: 3D Reconstruction measurement errors of the red ball throughout the frames throughout the pendulum trajectory

The variation that can be seen in the position error in first 10 frames is due to the partial occlusion of the ball from a hand. It is possible to see the same effect that was obtained in the simulation, the closest the ball projection is to the optical center the less the error its 3D reconstruction presents. This happens because the furthest the ball is from the optical center the less the optical rays pass through it, resulting in an increased distance between them and in a more elongated ellipse. The frames where the ball was furthest from the center where the 20th, 40th and 60th, where the highest error appears in the x coordinate as in Figure 6.11.



(a) 20th frame **(b)** 40th frame **(c)** 60th frame

Figure 6.11: Video frames of the red ball pendulum trajectory

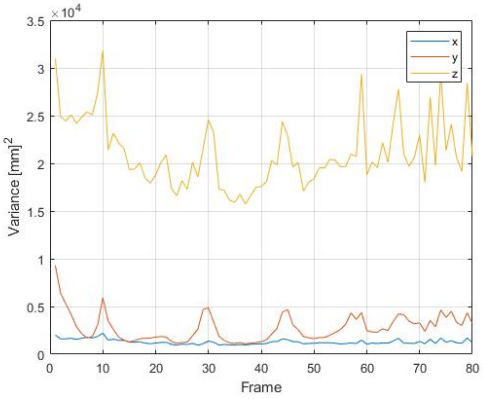
Table 6.3: Average error and variance of the estimation error of the real scenarios regarding the red ball

	Average Error $[mm]$			Variance $[mm]^2$		
	x	y	z	x	y	z
6.7	1,29	13,34	79,94	35,43	1520,7	10660,9
6.9	-0,17	3,12	195,6	1037,4	23,11	15053,1

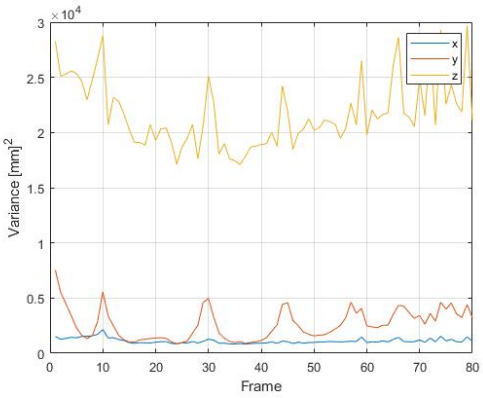
These results present a higher variation than the ones simulated due to the less effective image segmentation, showing consequently more error in the estimation of the location of the ball. The MSE is in Table 6.4.

Table 6.4: 3D Reconstruction MSE for the real scenarios of the red ball trajectories

MSE	x $[mm]^2$	y $[mm]^2$	z $[mm]^2$
6.7	39,7	1706,5	18045,8
6.9	961,6	29,7	49323,1

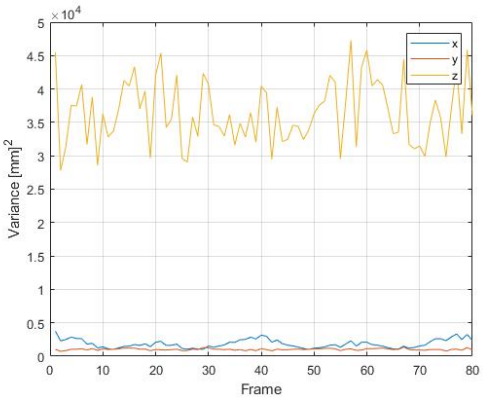


(a) Variance from the covariance matrices obtained from MC

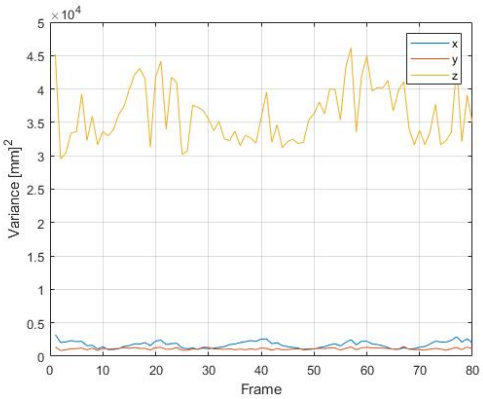


(b) Variance from the covariance matrices obtained from UT

Figure 6.12: Variances of the estimation error for the 6.7 trajectory



(a) Variance from the covariance matrices obtained from MC



(b) Variance from the covariance matrices obtained from UT

Figure 6.13: Variances of the estimation error for the 6.9 trajectory

On Figures 6.12 and 6.13 it is possible to see the variances obtained from the MC and UT methods, where they present similar values throughout the frames. The variances are higher than the ones obtained from the estimation error which results in a higher uncertainty associated. The higher variances in the free fall trajectory would be considered better due to the higher variability of the trajectory, but regarding trajectories with less fluctuation, such as the pendulum, the variances from the estimation error would suffice.

The values are in accordance because for the trajectory 6.7 the y coordinate presets higher variance due to the bounces with the floor, resulting in the spikes as in Figure 6.12, and in the 6.9 trajectory the x coordinate is the one that moves further away from the optical center resulting in an increase in the error and therefore a higher variance than the y coordinate. The variance is higher for the z coordinate in the trajectory 6.9 due to the pendulum trajectory being at a bigger distance from the camera comparing to

the free fall.

6.2.2 Green Ball

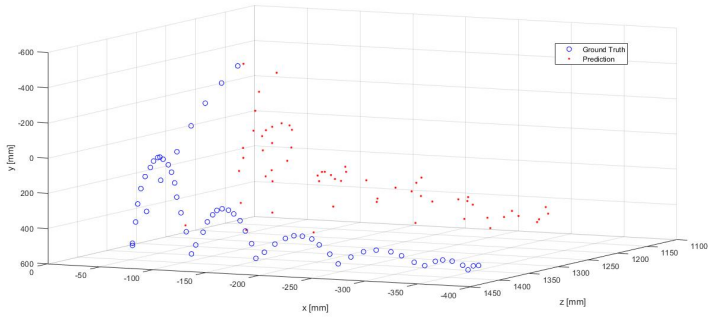


Figure 6.14: 3D Reconstruction measurement of the green ball location on the free fall trajectory

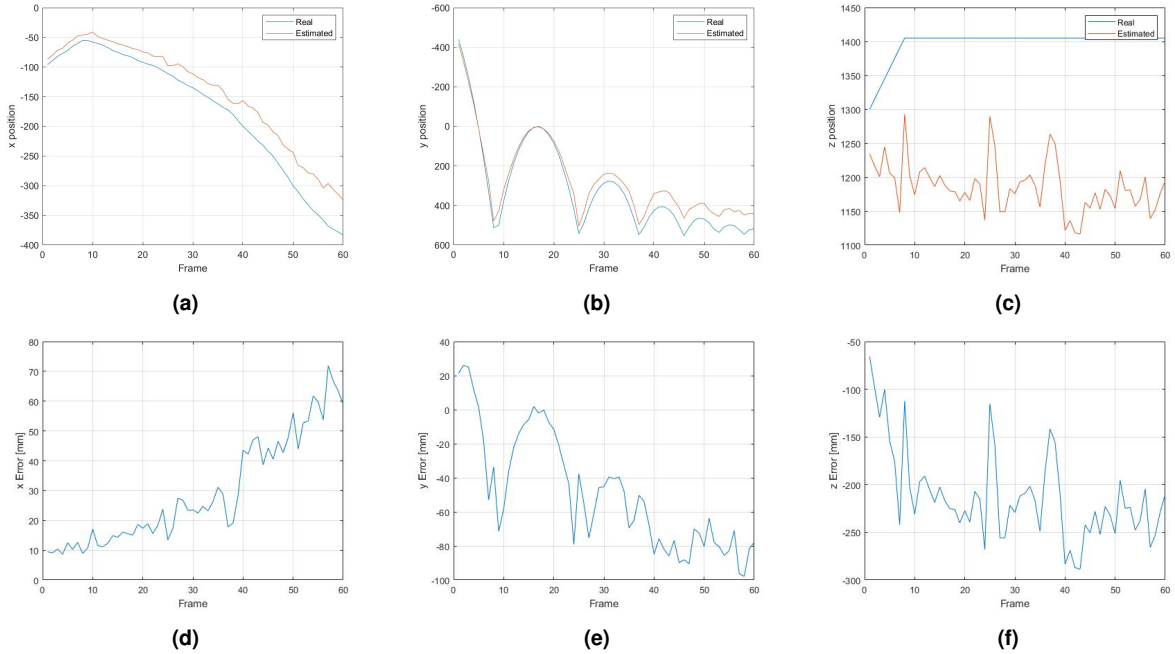


Figure 6.15: Measurement error of the green ball throughout the frames in the free fall trajectory

The green ball, on the free fall trajectory, presents similar results to the red ball with the only difference being the increment of the MSE in the x coordinate, because as the ball tends to move away from the center of the coordinate system.

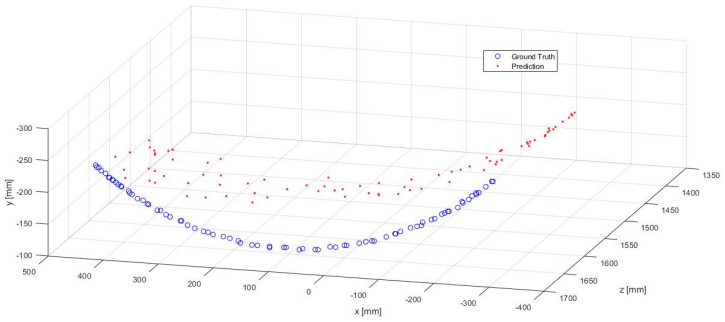


Figure 6.16: 3D Reconstruction measurement of the green ball location on the pendulum trajectory

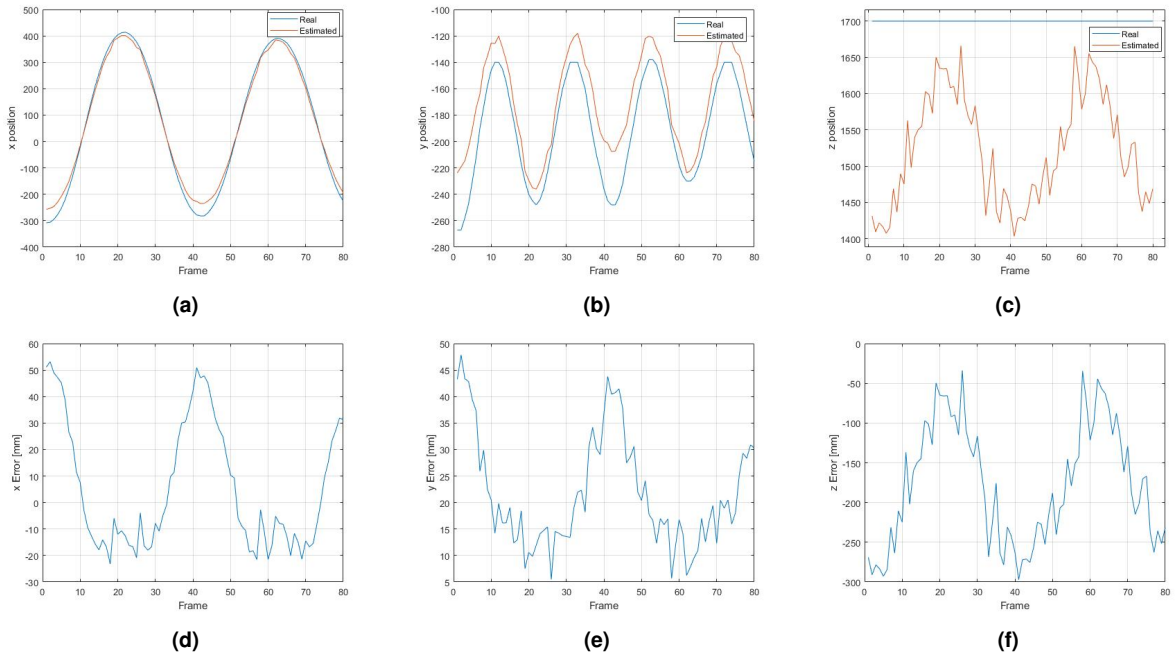


Figure 6.17: 3D Reconstruction measurement errors of the green ball throughout the frames in the pendulum trajectory

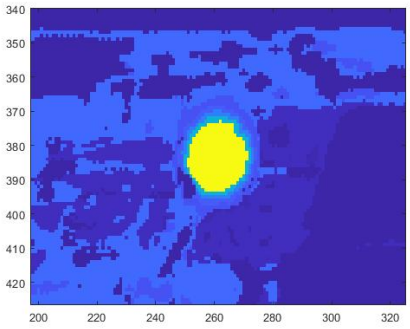
It is possible to see that the results with the green ball are worse than with the red ball. This is due to the fact that the green ball had less of a contrast with the background than the red ball as in Figure 6.18.

(a) Frame of the video of the green ball

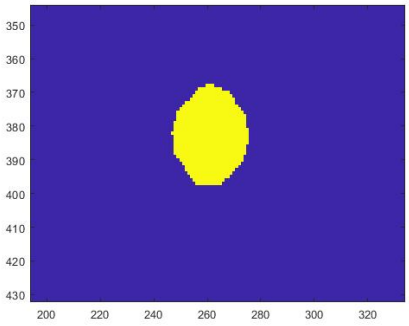
(b) Frame from the video of the red ball

Figure 6.18: Difference between the contrast of the red and green ball from the background

Consequently, this leads to a slightly worse segmentation, particularly in the binarization segment which can be seen in Figure 6.19 where the resulting blob is elongated due to the presence of some green colour on the floor. This faulty segmentation reflects itself on the estimation error, as in Table 6.6, increasing it comparing to the red ball.



(a) Pixel probability image



(b) Binarizated image

Figure 6.19: Example of a bad segmentation

Table 6.5: Average error and variance of the estimation error of the real scenarios regarding the green ball

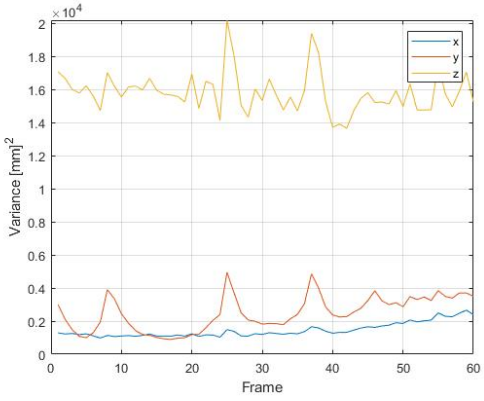
	Average Error $[mm]$			Variance $[mm]^2$		
	x	y	z	x	y	z
6.14	-29,5	48,9	211,76	317,5	1132,5	2268,6
6.16	-5,54	-21,69	177,64	551,4	110,1	5688,1

Table 6.6: 3D Reconstruction MSE for the real scenarios of the green ball trajectories

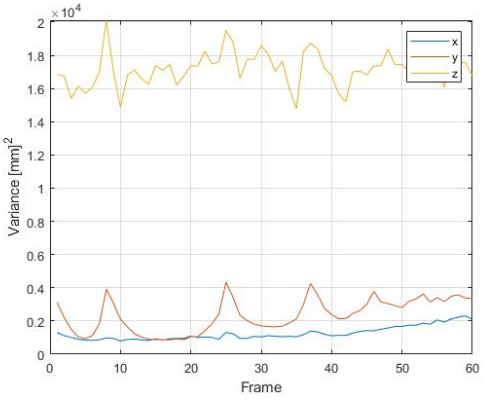
MSE	x $[mm]^2$	y $[mm]^2$	z $[mm]^2$
6.14	442,5	12521,2	27420,9
6.16	545.429	552.277	36079.059

The Figures 6.20 and 6.21 represent the variance obtained to each coordinate from the methods MC and UT. It is possible to see the higher variance in y coordinate, consequence of the impacts with the floor and deformation of the ball as it happened with the red ball, resulting in a worse prediction. It is also interesting to note that despite the results from the green ball presenting higher errors than the ones with the red ball, they present an overall lower variance in the z coordinate, displaying therefore a lower accuracy but a higher precision. This is directly correlated to the influence of the contrast existing between the background and the ball.

The 6.9 and 6.16 trajectories present the worst results from all the trajectories simulated and a higher variance for the z coordinate.. This is directly correlated with the distance at which the ball is from the camera, which in these scenarios are the highest from all the ones tested. This indicates a decrease in accuracy of the method implemented with the increase of the depth, resulting in higher variances due to the higher uncertainty of the location of the ball.

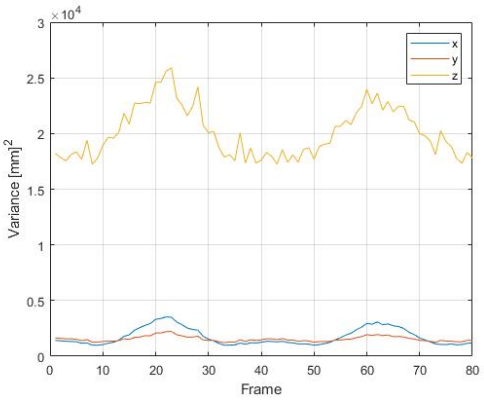


(a) Variance from the covariance matrices obtained from MC

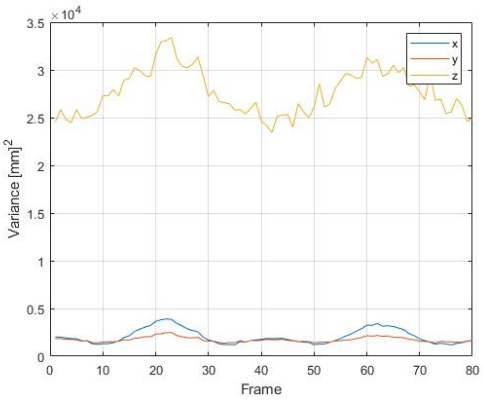


(b) Variance from the covariance matrices obtained from UT

Figure 6.20: Variances of the estimation error for the 6.14 trajectory



(a) Variance from the covariance matrices obtained from MC



(b) Variance from the covariance matrices obtained from UT

Figure 6.21: Variances of the estimation error for the 6.16 trajectory

6.3 Tracking

In this section we apply the obtained measurements and corresponding estimated measurement error covariances to perform the 3D tracking of the object. We use a Kalman Filter (KF) for this purpose. In order to properly estimate the MSE, from the tracking trajectories without the measurements bias interfering with the random noise, the average error was extracted and subtracted to the bias estimations. Note that the parameter MSE in this section is the same as the variance of the estimation error in the previous section. The process noise used in the KF for the realization of this tests was

$$Q_k = \begin{pmatrix} 1000 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1000 & 0 & 0 & 0 & 0 \\ 0 & 0 & 50000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 \end{pmatrix} \tag{6.6}$$

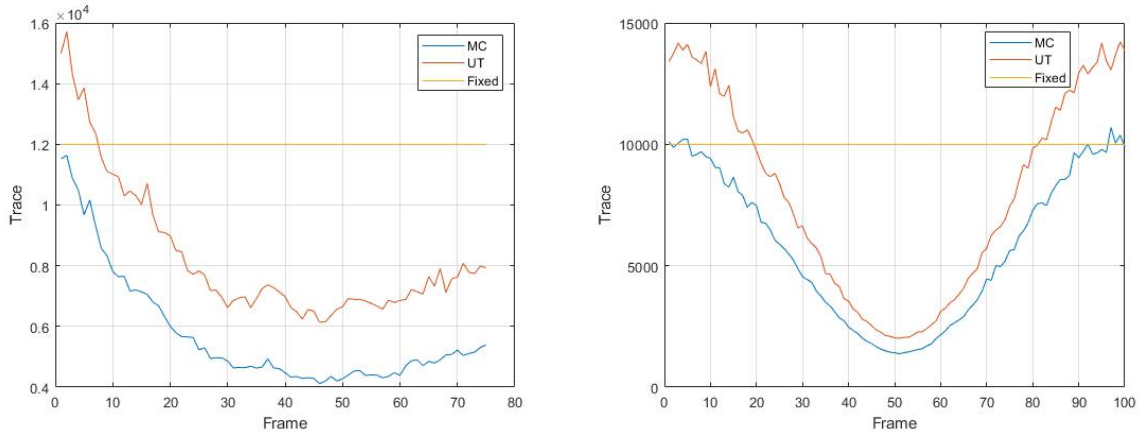
and these values were empirically selected based on the deviations of the trajectory from the motion model assumed, which was one with constant velocity. The variances used to spread noise in the parameters described in Chapter 5 were

$$\sigma_{tra}^2 = 12^2, \quad \sigma_{sca}^2 = \log_e(1,1)^2, \quad \sigma_{she}^2 = 0.02^2, \quad \sigma_{rot}^2 = \frac{5 * pi}{180}^2. \tag{6.7}$$

On Tables 6.7 and 6.8 we can see the difference in the MSE using a fixed (F), generated by MC or obtained from the UT covariances. The traces, which correspond to the sum of the diagonal values of the matrix, of the covariance matrices are represented in Figure 6.22. The fixed covariance was chosen to be as close as possible to the first state generated by the MC which was thought to be optimal.

Regarding the simulated trajectories of the red ball, it is possible to see a decrease in the MSE

in Table 6.7 when using the covariances obtained from the methods that propagated error from the 2D image into the 3D world. On the trajectory 6.3, all coordinates present a similar MSE for every covariance used, being the most significant values an increase of 4% and a decrease of 6%, for the MC and UT respectively, in the x coordinate comparing with the fixed covariance. For the 6.1 trajectory, on the one hand, in the y coordinate, there is a decrease in the MSE of $\approx 16\%$ and $\approx 11\%$ for the MC and UT respectively, representing one of the most affected coordinates on a free fall trajectory. On the other hand there is an increase of $\approx 7\%$ and $\approx 11\%$ of the error in the z coordinate. This may be explained due to the fact that this coordinate is the hardest to predict. Therefore, a covariance matrix with a higher variance throughout the frames, as the fixed covariance is in this case, presents the best approach resulting in slighty better results in this coordinate.



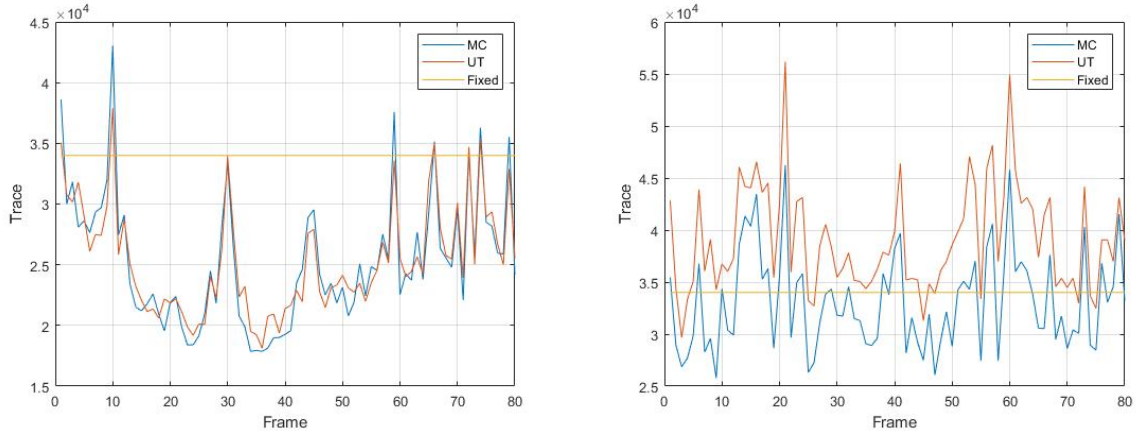
(a) 6.1 **(b)** 6.3

Figure 6.22: Trace of the covariance matrices used in the simulated trajectories of the red ball

Regarding the real trajectories of the red ball, the most significant results are the x coordinate in the pendulum and y coordinate in the free fall trajectories as in Table 6.7, because these are the variables that vary the most from the optical center and are more relevant to analyze in the tracking environment. On the first trajectory there is a decrease of $\approx 65\%$ and $\approx 62\%$ for the MC and UT methods and on the second one there is a decrease of $\approx 76\%$ and $\approx 71\%$ for the MC and UT respectively when compared with the results from the fixed covariance.

Table 6.7: Tracking MSE values for the red ball sequences

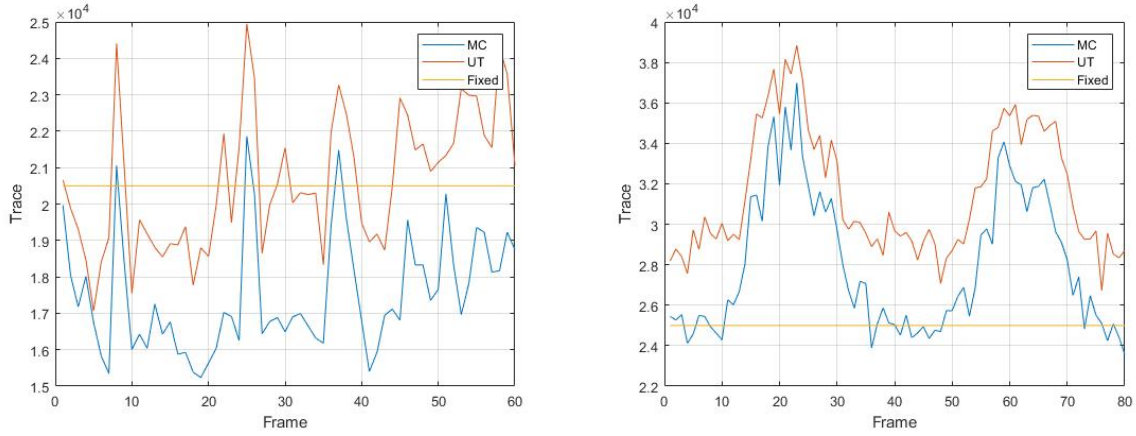
	Simulation						Real Test					
	6.3			6.1			Pendulum			Free Fall		
	F	MC	UT	F	MC	UT	F	MC	UT	F	MC	UT
x $[mm^2]$	499,71	523,59	473,05	278,62	285,73	288,36	2476,6	882,43	944,72	39,38	22,69	27,97
y $[mm^2]$	53,15	53,72	52,39	274,54	230,5	245,91	47,42	45,17	43,93	15975,1	3668,7	4432,38
z $[mm^2]$	3987,93	3857,52	3812,85	901,63	965,93	1090,4	10026,75	8758.58	8506,11	10205,98	15048,73	15588,38
Total $[mm^2]$	11366,55	12318,69	12639,43	13010,19	14399,77	15400,98	14565,58	11175,84	12217,46	54802,53	21292,05	22520,41



(a) Free fall **(b)** Pendulum

Figure 6.23: Trace of the covariance matrixes used in the real trajectories of the red ball

Regarding the green ball the most significant results are the x for the pendulum trajectory, which has a decrease in the MSE of $\approx 57\%$ and $\approx 54\%$ for the MC and UT respectively. Concerning the free fall it is possible to observe a decrease of $\approx 75\%$ and $\approx 90\%$ for x coordinate and $\approx 72\%$ and $\approx 69\%$ for the y coordinate. The respective traces of the matrices are represented in Figure 6.24.



(a) Free fall **(b)** Pendulum

Figure 6.24: Trace of the covariance matrixes used in the real trajectories of the green ball

Table 6.8: Tracking MSE values for the green ball real sequences

	Free Fall			Pendulum		
	Fixed	MC	UT	Fixed	MC	UT
x $[mm^2]$	383,15	84,48	34,51	3605,8	1572,4	1690,77
y $[mm^2]$	5203,82	1467,91	1658,8	123,7	176,7	202,54
z $[mm^2]$	2261,2	3595,8	7103,3	5285,8	5482,5	6656,7
Total $[mm^2]$	22564,17	4439,42	4664,88	7260,03	5714,05	5569,19

It is possible to see that the use of a covariance matrix obtained from the uncertainty characterization methods display generally better results than a fixed one in the x and y coordinates. It is also possible to see that these methods perform worse on the z coordinate, especially on the free fall trajectories. This coordinate was the hardest to obtain the ground truth information from the image data, therefore these results present the lowest credibility of them all.

7

Conclusion

Contents

The work described in this thesis presents a method that is able to provide an estimation of the location of the ball and also a covariance from the uncertainty characterization. The algorithm focuses on extracting the blob, that corresponds to the ball, recurring to image segmentation from frames of a video and morphological operators to smooth the image data. Then, it fits an ellipse to the blob so that a 3D position can be estimated through the use of monocular reconstruction by using the ball radius and the intrinsic camera parameters. Finally, the uncertainty behind this non linear transformation was characterized using two methods, which were Monte Carlo (MC) and Unscented Transform (UT). These measurements and covariances served then as an input to a kalman filter together with a fixed covariance to serve as a reference.

From the results it was possible to conclude that the inputs from the methods utilized were able to decrease the Mean Squared Error (MSE). The results from the simulated trajectories were quite similar because due to the better image segmentation, comparing the real scenarios, the measurements were already close to the the ground truth which resulted in almost no improvement using covariances from the methods proposed. It was in the real scenarios, where the uncertainty increased, due to more noise in the frames and not so optimal image segmentation that is possible to see the improvements in using the characterization of the uncertainty. The covariance presented by the MC method showed the best results, on average on all coordinates, as it was expected. The results from the UT also decreased the MSE and were more than enough in describing the non linear transformation for the x and y coordinate. Although it presented a higher error for the z coordinate, it still consisted on an approach that focused on saving computation power on the cost of less accuracy, which was more than enough for the result pretended.

7.1 Future Work

This section will provide possible ideas for future work in order to improve the efficiency and accuracy of the algorithm proposed.

- One of the most important part on these type of algorithms in the implementation in real time. Computation time was not taken into consideration and the frames were analyzed offline. In order to correct this, the algorithm should be implemented in C++ or a similar programming language to increase its speed and verify if its compatible with a real time solution.

- The objective was the prove that a covariance obtained from uncertainty characterization provides better results that one that remains fixed throughout the tracking process. In order to improve the results, a more advanced tracker could be implemented to increase the disparity between the errors obtained.

- Further improvements should be made to the ball identification. The color segmentation method applied is sensitive to disruptions and in presence of noise might misidentify the ball and it is only able to identify one ball per frame. A more robust method could be implemented which would cope with wrong identifications and expanded to identify and track multiple balls.
- The experimental setup was validated using ground truth information. However, in the real scenarios this information was obtained manually from the image data, therefore bringing the accuracy into doubt. A different method could be used to extract the ground truth with more precision.

Bibliography

[Agren, 2017] Agren, S. (2017). Object tracking methods and their areas of application : A meta-analysis. Master's thesis, Umeå universitet.

[AlBasiouny et al., 2015] AlBasiouny, E. R., Sarhan, A., and Medhat, T. (2015). Mean-shift-fast algorithm to handle motion-blur with tracking fiducial markers. In *2015 Tenth International Conference on Computer Engineering Systems (ICCES)*, pages 286–292.

[Cagnoni and Zhang, 2016] Cagnoni, S. and Zhang, M. (2016). Evolutionary computer vision and image processing: Some faqs, current challenges and future perspectives. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 1267–1271.

[Chakraborty, 2013] Chakraborty, B. (2013). A Trajectory-Based Ball Detection and Tracking System with Applications to Shooting Angle and Velocity Estimation in Basketball Videos. *2013 Annual IEEE India Conference (INDICON)*, pages 1–6.

[Chen et al., 2007] Chen, H., Chen, H., and Lee, S. (2007). Physics-based ball tracking in volleyball videos with its applications to set type recognition and action detection. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, volume 1, pages 1097–1100.

[Cross and Zisserman, 1998] Cross, G. and Zisserman, A. (1998). Quadric reconstruction from dual-space geometry. In *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, pages 25–31.

[Gómez-González et al., 2019] Gómez-González, S., Nemmour, Y., Schölkopf, B., and Peters, J. (2019). Reliable real time ball tracking for robot table tennis. *CoRR*, abs/1908.07332.

[Greggio. et al., 2011] Greggio., N., Gaspar., J., Bernardino., A., and Santos-Victor., J. (2011). Monocular vs binocular 3d real-time ball tracking from 2d ellipses. In *Proceedings of the 8th International Conference on Informatics in Control, Automation and Robotics - Volume 1: ICINCO,*, pages 67–73. INSTICC, SciTePress.

[Hartley and Zisserman, 2003] Hartley, R. and Zisserman, A. (2003). *Multiple View Geometry in Computer Vision*. Cambridge University Press, USA, 2 edition.

[Hou et al., 2017] Hou, Y., Cheng, X., and Ikenaga, T. (2017). Real-time 3d ball tracking with cpu-gpu acceleration using particle filter with multi-command queues and stepped parallelism iteration. In *2017 2nd International Conference on Multimedia and Image Processing (ICMIP)*, pages 235–239.

[https://www.zivid.com/, 2018] https://www.zivid.com/ (2018). Ball projection. `https://blog.zivid.com/why-is-this-3d-camera-an-award-winner`. [Online; Accessed 2020-10-12].

[Huang et al., 2012] Huang, C., Tsai, W.-J., Lee, S.-Y., and Yu, J.-Y. (2012). Ball tracking and 3d trajectory approximation with applications to tactics analysis from single-camera volleyball sequences. *Multimedia Tools and Applications - MTA*, 60.

[Huang, 2019] Huang, K. T. (2019). Integrating computer vision and non-linear optimization for automated deformable registration of 3d medical images. *Physics in Medicine & Biology*, 64(13):135014.

[Julier et al., 2000] Julier, S., Uhlmann, J., and Durrant-Whyte, H. F. (2000). A new method for the nonlinear transformation of means and covariances in filters and estimators. *IEEE Transactions on Automatic Control*, 45(3):477–482.

[Kanatani et al., 2016] Kanatani, K., Sugaya, Y., and Kanazawa, Y. (2016). Ellipse fitting for computer vision: Implementation and applications. In *Synthesis Lectures on Computer Vision*, pages 11–32. Morgan Claypool Publishers.

[Lawrence, 1972] Lawrence, J. D. (1972). *A Catalog of Special Plane Curves*, page 63. Dover Publications.

[Ledley et al., 1990] Ledley, R. S., Buas, M., and Golab, T. J. (1990). Fundamentals of true-color image processing. In *1990 10th International Conference on Pattern Recognition*, volume i, pages 791–795.

[Li et al., 2015] Li, Q., Li, R., Ji, K., and Dai, W. (2015). Kalman filter and its application. In *2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*, pages 74–77.

[Lippiello and Ruggiero, 2012] Lippiello, V. and Ruggiero, F. (2012). 3d monocular robotic ball catching with an iterative trajectory estimation refinement. In *2012 IEEE International Conference on Robotics and Automation*, pages 3950–3955.

[Olufs et al., 2007] Olufs, S., Adolf, F., Hartanto, R., and Plöger, P. (2007). Towards probabilistic shape vision in robocup: A practical approach. In Lakemeyer, G., Sklar, E., Sorrenti, D. G., and Takahashi, T., editors, *RoboCup 2006: Robot Soccer World Cup X*, pages 171–182, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Otsu, 1979] Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66.

[Pereira, 2020] Pereira, L. (2020). Seguimento 3D em Tempo Real de Objetos Simples com uma Câmara RGB. Master's thesis, Instituto Superior Técnico.

[Pettofrezzo, 1978] Pettofrezzo, A. J. (1978). *Matrices and transformations*. Dover books on mathematics. Dover Publications, New York, NY.

[Reddy et al., 2012] Reddy, P. R., Amarnadh, V., and Bhaskar, M. (2012). Evaluation of Stopping Criterion in Contour Tracing Algorithms. *International Journal of Computer Science and Information Technologies*, 3(3):3888–3894.

[Ren et al., 2004] Ren, J., Orwell, J., Jones, G., and Xu, M. (2004). Real-time 3d football ball tracking from multiple cameras. In *Proceedings of the British Machine Vision Conference*, pages 85.1–85.10. BMVA Press.

[Serra, 1986] Serra, J. (1986). Introduction to mathematical morphology. *Computer Vision, Graphics, and Image Processing*, 35(3):283 – 305.

[Setiawardhana et al., 2017] Setiawardhana, Dikairono, R., Sardjono, T. A., and Purwanto, D. (2017). Visual ball tracking and prediction with unique segmented area on soccer robot. In *2017 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, pages 362–367.

[Smith, 1978] Smith, A. (1978). Color gamut transform pairs. *ACM Siggraph Computer Graphics*, 12:12–19.

[Spain, 2007] Spain, B. (2007). *Analytical Conics*. Dover Publications, New York.

[Spong et al., 2006] Spong, M. W., Hutchinson, S., and Vidyasagar, M. (2006). Robot modeling and control. *IEEE Control Systems*, 26(6):331–352.

[Szeliski, 2010] Szeliski, R. (2010). *Computer Vision: Algorithms and Applications*, pages 87–180. Springer-Verlag, Berlin, Heidelberg, 1st edition.

[Taiana et al., 2008] Taiana, M., Gaspar, J., Nascimento, J., Bernardino, A., and Lima, P. (2008). 3d tracking by catadioptric vision based on particle filters. In Visser, U., Ribeiro, F., Ohashi, T., and Dellaert, F., editors, *RoboCup 2007: Robot Soccer World Cup XI*, pages 77–88, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Verma et al., 2015] Verma, K. K., Kumar, P., and Tomar, A. (2015). Analysis of moving object detection and tracking in video surveillance system. In *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 1758–1762.

[Wan and Van Der Merwe, 2000] Wan, E. A. and Van Der Merwe, R. (2000). The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, pages 153–158.

[Wang et al., 2016] Wang, Y., Cheng, X., Ikoma, N., Honda, M., and Ikenaga, T. (2016). Motion prejudgment dependent mixture system noise in system model for tennis ball 3d position tracking by particle filter. In *2016 Joint 8th International Conference on Soft Computing and Intelligent Systems (SCIS) and 17th International Symposium on Advanced Intelligent Systems (ISIS)*, pages 124–129.

[Wikipedia, the free encyclopedia, 2013] Wikipedia, the free encyclopedia (2013). Moore neighborhood. `https://commons.wikimedia.org/wiki/File:CA-Moore-Neighborhood.svg`. [Online; Accessed 2020-06-02].

[Wikipedia, the free encyclopedia, 2020] Wikipedia, the free encyclopedia (2020). General ellipse. `https://en.wikipedia.org/wiki/Ellipse`. Online; Accessed: 2020-06-15.

[Woodward and Delmas, 2005] Woodward, A. and Delmas, P. (2005). Computer vision for low cost 3-d golf ball and club tracking. In *Image and Vision Computing*, New Zealand.

[Wu et al., 2015] Wu, Y., Lim, J., and Yang, M. H. (2015). Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848.

[Zhang et al., 2009] Zhang, Y., Dhua, A., Kiselewich, S., and Bauson, W. (2009). Challenges of embedded computer vision in automotive safety systems. In *Embedded Computer Vision*, pages 257 – 279. Springer, London.

A

A1

From the equation (2.18) we have

$$m^T A_k m = 0 \tag{A.1}$$

and shifting the ellipse to the origin of the coordinate center, as A.2

$$m' = H_0 m \simeq m = H_0^{-1} m' \tag{A.2}$$

and applying it to (2.18) it is the same as A.3.

$$m'^T H_0^{-T} A_k H_0^{-1} m' = 0 \tag{A.3}$$

The next step relies on applying the composite transformation in the form of A.4

$$m'' = H_{noise} m' \simeq m' = H_{noise}^{-1} m'' \tag{A.4}$$

Knowing that $m'^T = m''^T H_{noise}^{-T}$ we obtain A.5

$$m''^T H_{noise}^{-T} H_0^{-T} A_k H_0^{-1} H_{noise}^{-1} m'' = 0 \tag{A.5}$$

Finally, replacing m'' with the equations A.2 and A.4 we are left with the final equation A.6.

$$m^T H_0^T H_{noise}^T H_0^{-T} A_k H_0^{-1} H_{noise}^{-1} H_0 m = 0 \tag{A.6}$$