Monocular Ball Detection and Tracking with Noise Characterization

Bernardo Ferreira
bernardoppferreira@gmail.com
Instituto Superior Técnico

***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 into tracking algorithms to increase their robustness. This approach uses the known color and ball size information and through image segmentation its able to generate an estimation of its 3D location in space. Furthermore, two methods based on the Monte Carlo method or Unscented Transform, are used to estimate the 3D noise characteristics which can then be used as an input in a tracker to increase its accuracy.

The proposed method is compared to a method whose noise covariance is optimal for the trajectory but fixed, showing a decrease on the error up to 76% in some trajectories.

***Index Terms*—Projection, Covariance, Noise characterization, RANSAC**

I. Introduction

Ball tracking is a tool important in many industries. These industries are, for the most part, related to sports, such as football [Ren et al., 2004], basketball [Chakraborty, 2013], volleyball [Chen 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.

This work proposes a method for the detection of a ball with known size and color from a single camera with characterization of the uncertainty of its position. This is accomplished by analysing each frame of the corresponding video to segment the possible locations of the ball with a color based method, followed by an ellipse fitting procedure and a monocular 3D reconstruction algorithm. Then noise in the 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.

A. Outline

This paper starts by describing the state of the art ball tracking in II. Then, the proposed approach is described in III and its implementation in IV. Finally, the results obtained are presented in V and the conclusion highlighted in VI.

II. State of the Art

This section provides 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, where the latter will be more closely followed in our work.

A. 2D Tracking

[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. 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 the candidates were filtered based on ball size, shape and fullness.

[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 back propagation neural network, and the result consisted in the output goal area and the ball area position.

Regarding two dimensional tracking, the 3D position of the object is never obtained due to the lack of depth, which is a crucial parameter to track objects in the real world.

B. 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.

1) 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.

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, (i) the prediction where the particles are spread regarding a motion model, (ii) the update where each particle is given a probability based on the Bhattacharyya similarity metric and (iii) the resampling where the particles with highest weight are replicated and the remaining are discarded.

[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, which 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 for each situation to characterize the noise.

2) Indirect Methods: The indirect methods correspond to finding the position of the ball in the image plane, and then reconstructing its position to 3D by "inverting" the projection model.

[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.

[Chen et al., 2012] studied ball tracking applied to volleyball games. With a calibrated camera the 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 analyzing parabolas in the Y axis and lines in the X axis. From those possibilities a point based system was applied and the 3D positions and velocities were discovered.

[Gomez-Gonzalez 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.

[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.

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.

III. Methodologies

A. Background Theory

In order to relate a digital image to the 3D world and reconstruct an object's position, the connection between 3D points (X, Y, Z) and pixel coordinates (x, y) must be known. This is possible through the pinhole camera model, valid for perspective cameras (1) where the matrix K, which is the camera intrinsic matrix [Spong et al., 2006].

$$Z \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = K \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{1}$$

$$K = \begin{bmatrix} a_x & 0 & u_0 \\ 0 & a_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

Considering an object of radius R and positioned in the origin of the reference frame, any point on its surface can be represented in the projective space by the quadratic form

$$M^T Q_0 M = 0 \tag{3}$$

where

$$M = \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \quad , \quad Q_0 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -R^2 \end{pmatrix}. \tag{4}$$

Implementing a homogeneous coordinate transformation to the sphere Q_0 to an arbitrary location $t = [t_x t_y t_z]^T$ as $Q = T_0^Q T$ we get the representation of any point on the sphere surface at a location t. Therefore, applying perspective projection on the points of the sphere's surface, the projected figure in the image plane is represented by an ellipse A_k [Cross and Zisserman, 1998]

$$A_k^* \sim P Q^* P^T. \tag{5}$$

where P is the camera 3x4 projection matrix, $A_k^* = adj(A_k)$, $Q^* = adj(Q)$, adj represents the classical adjoint, which is the transpose of its cofactor matrix, and $\sim$ denotes equality through a scale factor. We are interested in the ellipse A_k and a general form for it is

$$A'x^2 + 2B'xy + C'y^2 + 2f_0(D'x + E'y) + f_0^2F' = 0 \quad (6)$$

The variable f_0 is a constant for adjusting the scale increasing accuracy and reducing the loss of significant digits [Kanatani et al., 2016]. Defining two 6D vectors as

$$\xi = \begin{pmatrix} x^2 \\ 2xy \\ y^2 \\ 2f_0x \\ 2f_0y \\ f_0^2 \end{pmatrix}, \qquad \theta = \begin{pmatrix} A' \\ B' \\ C' \\ D' \\ E' \\ F' \end{pmatrix} \quad (7)$$

it is possible to represent Equation (6) as $(\xi, \theta) = 0$, where (ξ, θ) represents the inner product of ξ and θ.

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

$$x = x' + \triangle x, \qquad y = y' + \triangle y. \quad (8)$$

Replacing this in ξ seen in Equation (7) and assuming that $\triangle x$ and $\triangle y$ are random variables subjected to an independent Gaussian distribution with mean 0 and standard deviation σ we obtain the covariance matrix of ξ as

$$V_0[\xi] = 4 \begin{pmatrix} x'^2 & x'y' & 0 & f_0x' & 0 & 0 \\ x'y' & x'^2 + y'^2 & x'y' & f_0y' & f_0x' & 0 \\ 0 & x'y' & y'^2 & 0 & f_0y' & 0 \\ f_0x' & f_0y' & 0 & {f_0}^2 & 0 & 0 \\ 0 & f_0x' & f_0y' & 0 & {f_0}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad (9)$$

B. Pipeline

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 into a tracker in order to test the validity and performance of this approach.

C. 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 respect to the camera frame rate. 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], therefore the first step taken into account is to create a reference image as in Figure 1 that consists in several pictures of the cropped ball meant to be tracked.

Fig. 1: Example of reference image

This image is then converted from the default format Red-Green-Blue to Hue-Saturation-Intensity and reference histograms are created for each parameter. To create the color histograms, the color values were divided into bins: 12 to hue and saturation and 4 for the intensity. After acquiring the histograms, the same process is 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.

D. 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 $\sigma_w^2 = w_0(t)\sigma_0^2(t) + w_1(t)\sigma_1^2(t)$ 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 threshold t.

E. Morphological operators

After applying the method mentioned in III-D, 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 morphological

operation used was $opening$, which consists in a $erosion$ followed by a $dilate$.

After applying the morphological operators we are then left with binary images as the one represented in Figure 2.

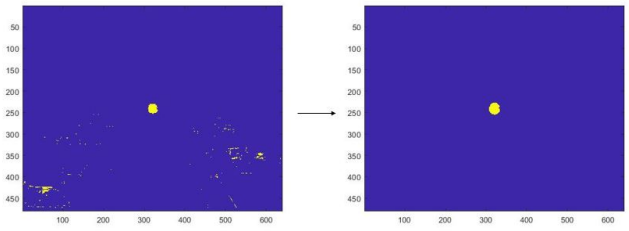


Fig. 2: Performing an $opening$ in a binary image

F. Contour Extraction

It is now necessary to fit an ellipse to the blobs that survived the image segmentation and morphological operations. 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 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. 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.

G. 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 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 (7) and calculate the matrix M represented below, where N is the total number of points selected from the contour.

$$M = \frac{1}{N}\sum_{a=1}^{N} \xi_a \xi_a^T \tag{10}$$

Next, the eigenvalue problem in (11) must be solved

$$M\theta = \lambda\theta \tag{11}$$

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.

H. RANSAC

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 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 (12) and the unit eigenvector with the smallest eigenvalue associated is stored as a candidate for the ellipse parameters θ.

$$M = \sum_{a=1}^{5} \xi_a \xi_a^T \tag{12}$$

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

$$S = \frac{1}{N}\sum_{a=1}^{N} \left((x_a - \overline{x_a})^2 + (y_a - \overline{y_a})^2\right) = d_a^2. \tag{13}$$

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 (14) [Kanatani et al., 2016], where $V_0[\xi]$ is the covariance matrix in (9).

$$d^2 = (x_a - \overline{x_a})^2 + (y_a - \overline{y_a})^2 \approx \frac{(\xi, \theta)^2}{(\theta, V_0[\xi]\theta)} \tag{14}$$

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.

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 (15) where s represents the size of the samples and ϵ the proportion of outliers.

$$N = log(1-p)/log(1-(1-\epsilon)^s) \tag{15}$$

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.

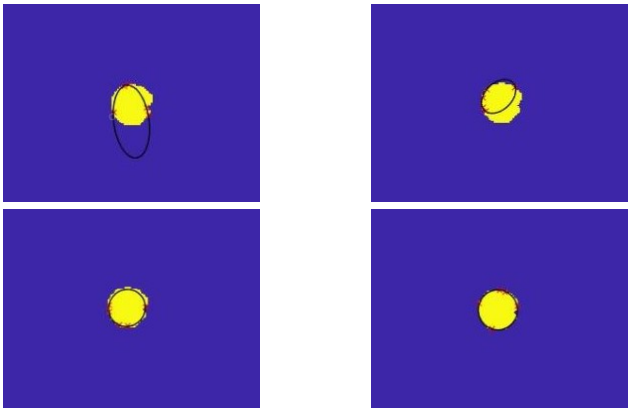


Fig. 3: Representation of RANSAC iterations

I. 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.

Noting that the matrices Q (quadric representation of the ball) and A_k (conic representation of an ellipse in 2D) are symmetric, their adjoint matrices coincide with their inverses [Hartley and Zisserman, 2003, Chapter 2.2.3]. Therefore, the projection referred in (5) can be written as

$$A_k^{-1} \sim PQ^{-1}P^T \tag{16}$$

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), the projection in (16) can be rewritten as

$$\begin{pmatrix} 1-x^2/R^2 & -xy/R^2 & -xz/R^2 \\ -xy/R^2 & 1-y^2/R^2 & -yz/R^2 \\ -xz/R^2 & -yz/R^2 & 1-z/R^2 \end{pmatrix} \sim K^{-1}A_k^{-1}K^{-T} \tag{17}$$

Computing the eigenvalues from the left hand side matrix, it is 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. Note that $I_3 - tt^T/R^2$ is Q^{-1} represented by the left hand side in 17.

$$I_3 - tt^T/R^2 = H/median(eig(H)). \tag{18}$$

Next, from (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

$$tt^T/R^2 = H_2. \tag{19}$$

From (19) its possible to define $v = t/R$ and see that the relation $vv^T = H_2$ is true. We are interested in the diagonal that represents the vector v, consequently we get to the relation given by (20) 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.

$$v = \begin{pmatrix} s_1\sqrt{H_{2(1,1)}} \\ s_2\sqrt{H_{2(2,2)}} \\ s_3\sqrt{H_{2(3,3)}} \end{pmatrix} \tag{20}$$

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

J. Noise propagation

A good fit of an ellipse is obtained after the method discussed in section III-H, but there is still some uncertainty due to noise in the process. This uncertainly will be reflected in the 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).

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 affine transformation of the ellipse fitted to the blob that affect the corresponding (A', B', C', D', E', F') parameters. 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 3D reconstruction and the results would be aggregated in the 3D world

Another way to achieve a similar result with the method described previously 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 the same 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 (21) and W_i are weights given by (22), where P^x is the covariance of the variable.

$$\begin{aligned} X_0 &= \bar{x} \\ X_i &= \bar{x} + (\sqrt{(n+\lambda)P^x})_i \qquad i = 1, ..., n \\ X_i &= \bar{x} - (\sqrt{(n+\lambda)P^x})_i \qquad i = 1+n, ..., 2n \end{aligned} \tag{21}$$

$$
\begin{aligned}
W_0^{(m)} &= \lambda/(n+\lambda) \\
W_0^{(c)} &= \lambda/(n+\lambda) + (1-\alpha^2+\beta) \\
W_i^{(m)} &= 1/(2(n+\lambda)) \qquad i = 1, ..., 2n \\
W_i^{(c)} &= 1/(2(n+\lambda)) \qquad i = 1, ..., 2n
\end{aligned} \tag{22}
$$

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 that scales 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 it is set to 20 so that there are no negative weights in the calculation of the mean and 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 from the non linear transformation. 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 are then propagated through the non linear function, which in our case is the monocular reconstruction $g()$,

$$
\gamma_i = g(X_i) \qquad i = 0, ..., 2n_x \tag{23}
$$

such that the mean and covariance can be given by

$$
\begin{aligned}
\bar{y} &= \sum_{i=0}^{2n} W_i^{(m)} \gamma_i \\
P^{\gamma} &= \sum_{i=0}^{2n} W_i^{(c)} (\gamma_i - \bar{y})(\gamma_i - \bar{y})^T
\end{aligned} \tag{24}
$$

where γ_i are the sigma vectors propagated through the non linear function and $\bar{y}$ is the weighted mean.

K. Tracking

Finally, we are left with estimates of the 3D ball position and respective covariances, We apply the measurements and time dependent covariance matrix into a tracker and the one selected for this work was the 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.

IV. Implementation

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

A. Noise Propagation

The propagation of noise is a necessary tool in order to propagate uncertainty from 2D to 3D. We apply 2D Gaussian noise to the parameters of an affine transformation of the ellipse was the one selected. 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 about the center of the ellipse occurs based on a scalar, (iii) rotation, where the ellipse is rotated around its center and finally (iv) shear, where the shape of the ellipse is slanted in a specific coordinate.

The matrix representation of these transformations are

(25)

B. Simulation

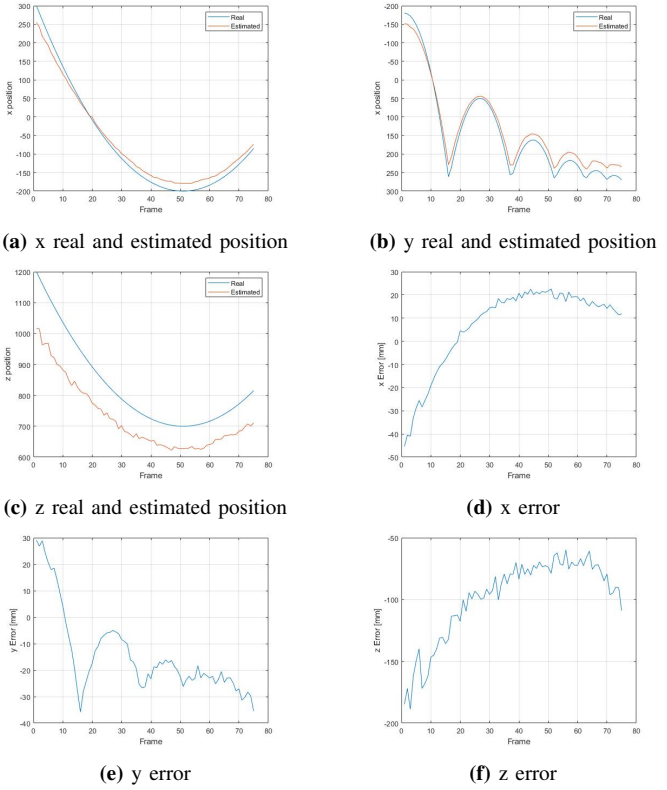
In order to validate the algorithm, ground truth trajectories from a simulator were made. These trajectories were composed of a sequence of images showing the different states of the ball. These frames were created by a simulator which was developed by [Pereira, 2020] and implemented in C++ with the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

V. Results

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 the estimates 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, which was though to be optimal. This was also done in simulated and real scenarios to see the disparities provoked by this change. The metric to evaluated and compare the results was the Mean Squared Error (MSE). The real experiments were executed with two different balls: a red ball with $30mm$ radius and a green ball with $35.85mm$ radius.

A. Simulator

The 3D measurements of the simulated trajectories had as intrinsic camera parameters $K = [500\ 0\ 320;\ 0\ 500\ 240;\ 0\ 0\ 1]$. Their respective errors associated with each coordinate are in Figures 4 and 5 and their global representation is in Table I. The inversion of the y axis is for visual representation.



(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

Fig. 4: 3D Reconstruction measurement errors of the red ball throughout the frames in the first simulated trajectory

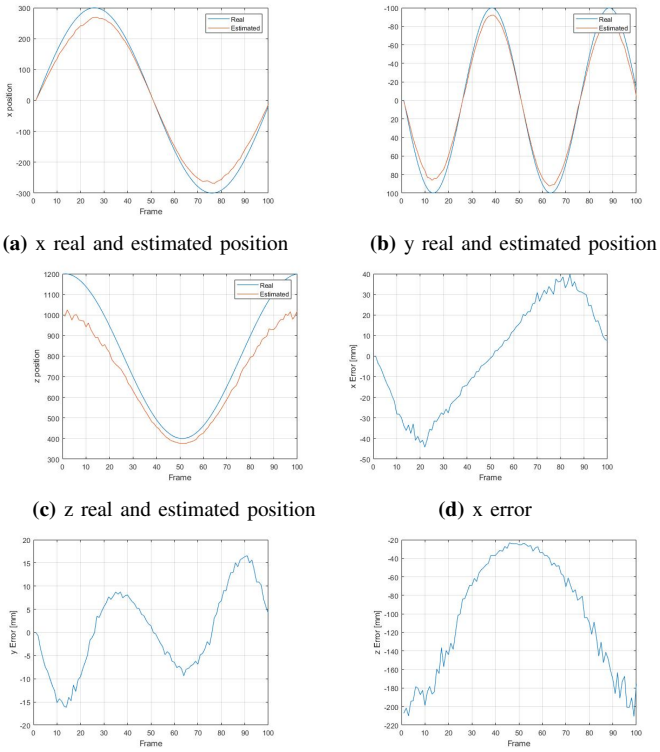
On the first trajectory the errors associated with x and y are pretty similar. This does not happen in the second trajectory, because there is a greater variation in the x coordinate comparing to the y coordinate regarding the optic center. This leads to an increase in error as expected, because the further away from the optic center the ball is, the worst its representation by the ellipse is.

TABLE I: 3D Reconstruction MSE for the simulated trajectories

MSE	x $[mm]^2$	y $[mm]^2$	z $[mm]^2$
Fig 4	379,3	466,8	11295,1
Fig 5	615,12	76,1	14111,7

B. 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. The Ground Truth for the real trajectories was estimated manually, therefore the



(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

Fig. 5: 3D Reconstruction measurement errors of the red ball throughout the frames in the second simulated trajectory

The intrinsic camera parameters used for the real scenarios were

$$\mathrm{K}_f = \begin{pmatrix} 497,01 & 0 & 372,588 \\ 0 & 496,392 & 231,468 \\ 0 & 0 & 1 \end{pmatrix} \quad K_p = \begin{pmatrix} 647,099 & 0 & 375.745 \\ 0 & 649.231 & 233.407 \\ 0 & 0 & 1 \end{pmatrix} \tag{26}$$

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

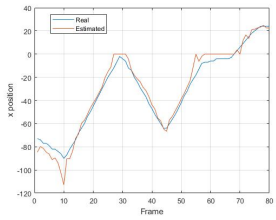
$$\mathrm{K}_f = \begin{pmatrix} 475.316 & 0 & 357.815 \\ 0 & 478.232 & 223.064 \\ 0 & 0 & 1 \end{pmatrix} \quad K_p = \begin{pmatrix} 501.698 & 0 & 373.208 \\ 0 & 502.41 & 246.066 \\ 0 & 0 & 1 \end{pmatrix} \tag{27}$$

for the free fall and pendulum trajectories.

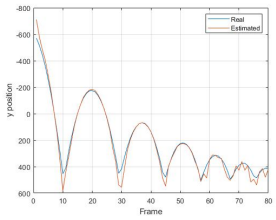
1) Red Ball: In the following Figure 6 and 7 the trajectories corresponding to the free fall and pendulum in the real scenario for the red ball are represented and their global errors in Table II.

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.

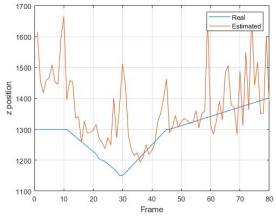
The variation that can be seen in the position error in first 10 frames is due to the partial occlusion of the ball from a hand. It is possible to see the same effect that was obtained in the simulation, the closest to the optical center the less the error. This happens because the furthest the ball 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.



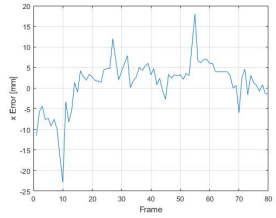
(a) x real and estimated position



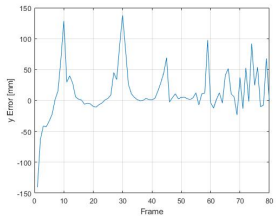
(b) y real and estimated position



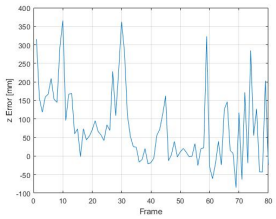
(c) z real and estimated position



(d) x error

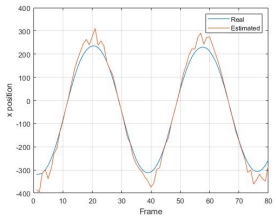


(e) y error

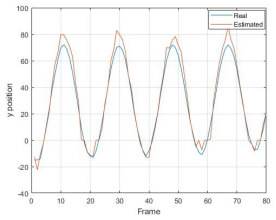


(f) z error

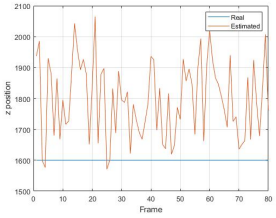
Fig. 6: 3D Reconstruction measurement errors of the red ball throughout the frames throughout the free fall trajectory



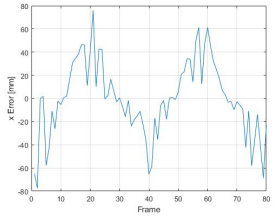
(a) x real and estimated position



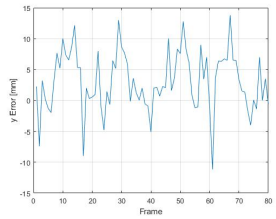
(b) y real and estimated position



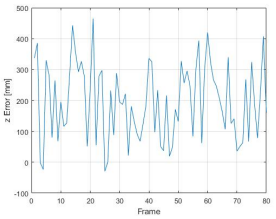
(c) z real and estimated position



(d) x error



(e) y error



(f) z error

Fig. 7: 3D Reconstruction measurements of the red ball throughout the frames throughout the pendulum trajectory

TABLE II: 3D Reconstruction MSE for the real scenarios of the red ball trajectories

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

2) Green Ball: The green ball trajectories were used mainly to see the differences in measurements provoked by changing the color, where in Figures 8 and 9 the trajectories are represented and in Table III the global error is demonstrated.

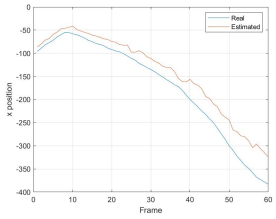
The green ball, on the free fall trajectory, presents similar results to the red ball with the only difference being the increment of the MSE (Mean Squared Error) in the x coordinate, because the ball tends to move away from the center of the coordinate system.

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 10.

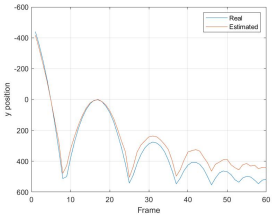
Consequently, this leads to a slightly worse segmentation, particularly in the binarization segment where the blob of the ball is deformed due to the presence of green in the floor. This faulty segmentation reflects itself on the estimation error, as in Table III, increasing it comparing to the red ball.

C. Tracking

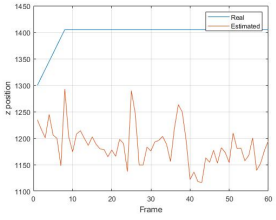
In this section we apply the obtained measurements and corresponding estimated measurement error covariances to



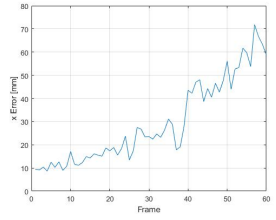
(a) x real and estimated position



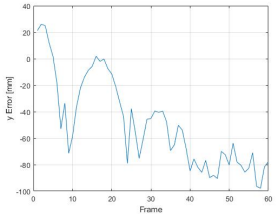
(b) y real and estimated position



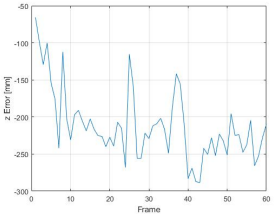
(c) z real and estimated position



(d) x error

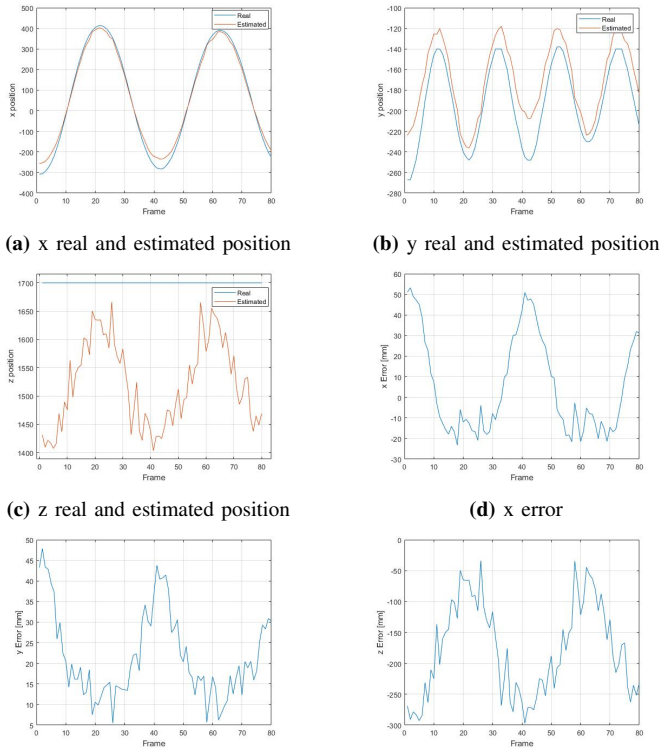


(e) y error



(f) z error

Fig. 8: 3D Reconstruction measurement error of the green ball throughout the frames in 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

Fig. 9: 3D Reconstruction measurement errors of the green ball throughout the frames in the pendulum trajectory

(a) Frame of the video of the green ball

(b) Frame from the video of the red ball

Fig. 10: Difference between the contrast of the red and green ball from the background

TABLE III: 3D Reconstrucion MSE for the real scenarios of the green ball trajectories

MSE	x $[mm]^2$	y $[mm]^2$	z $[mm]^2$
Fig 8	442,5	12521,2	27420,9
Fig 9	545.429	552.277	36079.059

perform the 3D tracking of the object. 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. The process noise used in the KF for the realization of these tests was $Q_k = diag(1000, 1000, 50000, 10, 10, 10)$.

Regarding the simulated trajectories of the red ball, it is possible to see a decrease in the MSE in Table IV when using the covariances obtained from the methods that propagated error from the 2D image into the 3D world. On the trajectory of Fig 5 the most significant values are an increase of 4% and a decrease of 6%, for the MC and UT respectively, in the x coordinate comparing with the fixed covariance, which was selected based on the covariance matrix provided in the first frame through the Monte Carlo method, which was though to be optimal. For the trajectory of Fig 4 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.

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 V, 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.

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.

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.

TABLE IV: Tracking MSE values for the red ball simulated sequences

	Fig 5			Fig 4		
	Fixed	MC	UT	Fixed	MC	UT
x $[mm^2]$	499,71	523,59	473,05	278,62	285,73	288,36
y $[mm^2]$	53,15	53,72	52,39	274,54	230,5	245,91
z $[mm^2]$	3987,93	3857,52	3812,85	901,63	965,93	1090,4
Total $[mm^2]$	11366,55	12318,69	12639,43	13010,19	14399,77	15400,98

TABLE V: Tracking MSE values for the red ball real sequences

	Free Fall			Pendulum		
	Fixed	MC	UT	Fixed	MC	UT
x $[mm^2]$	2476,6	882,43	944,72	39,38	22,69	27,97
y $[mm^2]$	47,42	45,17	43,93	15975,1	3668,7	4432,38
z $[mm^2]$	10026,75	8758.58	8206,11	10205,98	15048,73	15588,38
Total $[mm^2]$	14565,58	11175,84	12217,46	54802,53	21292,05	22520,41

TABLE VI: 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

VI. Conclusions

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 and Unscented Transform. 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 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 image 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 Monte Carlo method showed the best results, on average on all coordinates, as it was expected. The results from the Unscented Transform 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.

REFERENCES

[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.

[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 I–1097–I–1100.

[Chen et al., 2012] Chen, H.-t., Tsai, W.-j., and Lee, S.-y. (2012). Ball tracking and 3D trajectory approximation with applications to tactics analysis from single-camera volleyball sequences. pages 641–667.

[Cross and Zisserman, 1998] Cross, G. and Zisserman, A. (1998). Quadric surface reconstruction from dual-space geometry. In *IEEE International Conference on Computer Vision*, pages 25–31.

[Gomez-Gonzalez et al., 2019] Gomez-Gonzalez, S., Nemmour, Y., Schölkopf, B., and Peters, J. (2019). Reliable real-time ball tracking for robot table tennis. *Robotics*, 8(4):1–13.

[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.

[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.

[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.

[Ren et al., 2004] Ren, J., Orwell, J., Jones, G., and Xu, M. (2004). Real-time 3d soccer ball tracking from multiple cameras. pages 1–10.

[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.

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

[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.

[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.

[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.