Real-Time 3D Tracking of Simple Objects with an RGB Camera

Lino Pereira
lino.pereira@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

October 2020

Abstract

Nowadays, algorithms that obtain 3D information of objects, have a high importance in applications in different areas, such as in sports, robotics, medicine, among others.

This work, intends to improve a monocular region-based tracking algorithm using an RGB camera. The algorithm to be improved, derives from a particle filter where each particle represents a hypothesis of the state of the object in 3D, which allows the use of realistic 3D motion models. However, the literature mentions that the particle filter (PF) uses a very limited importance distribution to propagate the particles, which easily leads to the filter degenerating and losing track of the object. Given the limitation of the PF, an unscented particle filter (UPF) is proposed, this one obtains an approximation to the optimal importance distribution, by adding a current observation of the state.

Since, only the importance distribution is different, the observation model remains identical, where a method based on the color distributions of the interior and exterior regions of the object's silhouette, is used to calculate the likelihood of each particle.

In order to compare the proposed algorithm with the previous one, both are implemented in this work and several real and simulated experiments with a simple object are performed. From the results obtained, is shown that the filters are successful, with the UPF being more robust against degeneration, different numbers of particles and different adjustments in the interior and exterior regions of the object's silhouette.

Keywords:3D object tracking, monocular tracking, region-based methods, particle filter, unscented particle filter

1. Introduction

To create an algorithm that detects and tracks objects in 3D, it is necessary to use devices capable of acquiring 3D information, or devices that obtain data that can be inferred to obtain 3D information. In this work, a camera is used to get chromatic data from the environment and from that data, information of an object in space can be obtained. The main of objective of this work, is to track and estimate the state of a homogeneous ball in 3D, using only an RGB camera. In order to accomplish this task, two algorithms are developed. The first one, uses a particle filter as the algorithm's core and reproduces the project developed by Taiana, M. et al [5], the second one, uses an unscented particle filter developed by Van der Merwe et al. [7], and it is proposed with the intent to improve the estimation of the state of the ball from the previous one. Towards the principal objective, it's assumed that the object's shape, size and homogeneous color is known, and in order to access the tracking performance of the filters, simulated and real experiments were done. The mains contributions of this thesis are, a theoretical review about methods of detection and tracking of objects in 3D, a simulator to create synthetic trajectories for a ball to follow, and two algorithms developed in C++ to track and estimate the state of a homogeneous ball.

1.1. Bayesian filtering

Bayesian filtering refers to the Bayesian way of formulating optimal filtering, and the term optimal in this context, refers to statistical optimality. In mathematical terms, optimal filtering is considered to be a statistical inversion problem, where the unknown quantity is a vector value time series $\{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, ...\}$ which is observed trough a set of noisy measurements $\{\mathbf{z}_1, \mathbf{z}_2, ...\}$. In Bayes perspective, this means to compute the joint posterior probability distribution of all the states given all the measurements. In principle this can be done by applying the Bayes rule:

$$p(\mathbf{x}_{0:T}|\mathbf{z}_{1:T}) = \frac{p(\mathbf{z}_{1:T}|\mathbf{x}_{0:T})p(\mathbf{x}_{0:T})}{p(\mathbf{z}_{1:T})}. \quad (1)$$

where $p(\mathbf{x}_{0:T}) = \{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T\}$ and $p(\mathbf{z}_{1:T}) = \{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_T\}$. Unfortunately, this full posterior formulation has the serious disadvantage that each time we obtain a new measurement, the full posterior dimensionality grows and needs to be recomputed, therefore as time increases, eventually the computations will become intractable. However, if one restricts the problem to probabilistic Markov sequences, it is only necessary to estimate the marginal posterior distribution of the states given all available measurements, that is, to compute $p(\mathbf{x}_t|\mathbf{z}_{1:t})$, and this can be done recursively as a new measurement becomes available. It's only necessary to specify an initial prior probability distribution $p(\mathbf{x}_0)$, a state transition probability $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ and a measurement probability $p(\mathbf{z}_t|\mathbf{x}_t)$, then $p(\mathbf{x}_t|\mathbf{z}_{1:t})$ can be computed by the previous one:

$$p(\mathbf{x}_t|\mathbf{z}_{1:t}) = \eta\, p(\mathbf{z}_t|\mathbf{x}_t)\times \int p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1})d\mathbf{x}_{t-1}. \tag{2}$$

where η is the normalization constant of the Bayes rule. Even though, the recursive formulation is not practical, in particular, one either need to be able to compute the formula above in closed form, or to restrict the solution to finite state spaces, so that the integral integral becomes a finite sum [8, 9].

1.2. Particle Filter

As already stated, the equation (2) is intractable, and for this reason, many kind of numerical approximations methods have been developed. One type of these methods are the sequential Monte Carlo methods, that represents the posterior distribution as a weighted set of Monte Carlo samples from which the particle filter (PF) is a specialization. In this kind of filters, the samples are commonly designated by particles and each one, has a factor associated, denominated by importance weights. Each particle can be described by:

$$\chi_t = \{\mathbf{x}_t, \mathbf{w}_t\} = \left\{\left\{x_t^{(1)}, w_t^{(1)}\right\}, ..., \left\{x_t^{(N)}, w_t^{(N)}\right\}\right\} \tag{3}$$

where N is the number of particles, each particle $\mathbf{x}_t^{(i)}$ represents a hypothesis about what the actual state of the system may be at time t, and $w_t^{(i)}$ are the importance weights [8]. The intuition of the filter is to approximate the posterior distribution by the set of the weighted particles χ_t:

$$p(\mathbf{x}_t|\mathbf{z}_{1:t}) \approx \sum_{i=1}^{N} w_t^{(i)} \delta\left(\mathbf{x}_t - \mathbf{x}_t^{(i)}\right) \tag{4}$$

where $\delta(\cdot)$ is the Dirac delta function [5] and by the law of the big numbers, the bigger the number of particles, the lower is the variance of the approximation error [7]. Based on this discrete approximation, the Monte Carlo approximation of expectation can be computed to get the state estimate:

$$E[\mathbf{x}_t|\mathbf{z}_{1:t}] \approx \frac{1}{N}\sum_{i=1}^{N} w_t^{(i)}\mathbf{x}_t^{(i)}. \tag{5}$$

Unfortunately, it's often impossible to sample directly from the posterior distribution, but this can be surpassed by sampling from a known and easy-to-sample distribution $q(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)}, \mathbf{z}_{1:t})$, called importance distribution, and that is where the importance weights come from. By drawing the samples from this distribution, a recursive estimate for the importance weights can be derived as follows [7]:

$$w_t^{(i)} \propto \frac{p(\mathbf{z}_t|\mathbf{x}_t^{(i)})p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{q(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)}, \mathbf{z}_{1:t})} w_{t-1}^{(i)}. \tag{6}$$

where $\propto$ means up to scale. The specialization made by the PF is to use the state transition probability as the importance distribution, which simplifies the equation (6) to $w_t^{(i)} \propto p(\mathbf{z}_t|\mathbf{x}_t^{(i)})w_{t-1}^{(i)}$. Thus, the algorithm to compute each particle of the particle filter is composed by the following steps:

Algorithm 1: PF

```
  For $t = 0$:
    For $i = 1, ..., N$:
1     $\mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0)$,    $w_0^{(i)} = \frac{1}{N}$
  For $t \geq 1$:
    For $i = 1, ..., N$:
2     $\mathbf{x}_t^{(i)} \sim p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)})$
3     $w_t^{(i)} \propto p(\mathbf{z}_t|\mathbf{x}_t^{(i)})w_{t-1}^{(i)}$
4     $\tilde{w}_t^{(i)} = \frac{w_t^{(i)}}{\sum_{j=1}^{N} w_t^{(j)}}$
5     RESAMPLING($\mathbf{x}_t^{(i)}, \tilde{w}_t^{(i)}$),    $w_t^{(i)} = \frac{1}{N}$
```

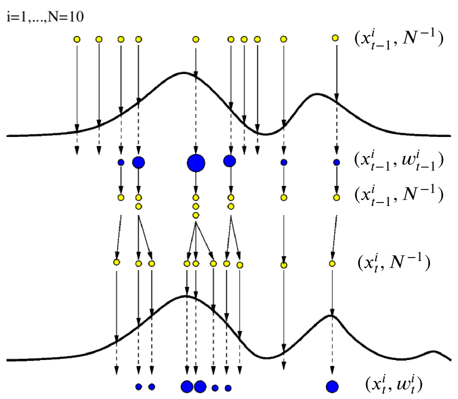

Figure 1: Illustration of the particle filter. Where the balls are the particles and their sizes, the weights [7].

This type of filters exhibit a phenomenon called degeneration. This happens when some particles get all the weight and a lot of them get insignificant. In order to prevent this, a process of resampling is implemented to replicate the particles with high weights and discard the ones with low weight [7, 9]. Doing this, brings more particles to regions of high likelihood, which not only contributes to get better estimates, but also to avoid the particles from moving wrongly in the state space. In the figure 1, one can see that the likelihood of each particle is evaluated by the measurement probability $p(\mathbf{z}_t|\mathbf{x}_t^{(i)})$ which results in the blue balls, after that, resampling is done, which aim the particles to the desired regions. Afterwards, a new iteration arises.

1.3. Unscented Particle Filter

The performance of the particle filter depends not only on the number of the particles but also on the quality of the importance distribution. The better the importance distribution resemble to the posterior distribution, better will be the filter performance. In a Markov process, the optimal importance distribution in terms of minimizing the variance of the weights is given by:

$$q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{z}_{1:t}) = p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{z}_t). \quad (7)$$

An optimal importance distribution, moves the particles from the prior distribution to the regions of high likelihood in a better way. This is extremely important if the likelihood is too narrow, or if it lies in one of the tails of the prior distribution. Despite of the transition model being easy to sample, in general, sampling from the optimal distribution is nontrivial, because of the dependence on the actual observation $\mathbf{z}_t$. Thus with the purpose of getting an approximation to the optimal distribution, the unscented particle filter (UPF) was developed. This one approximates the optimal distribution, by introducing the current observation together with the Gaussian estimate of the state. The filtering mechanism of the UPF consists in a particle filter that uses the unscented Kalman filter (UKF), as the importance distribution to propagate each particle:

$$q(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)}, \mathbf{z}_{1:t}) \approx \mathcal{N}(\boldsymbol{\mu}_t^{(i)}, \mathbf{P}_t^{(i)}),\ i = 1, ..., N. \quad (8)$$

The UKF is a filter based on the Kalman filter (which is a closed form solution of the Bayesian filtering equations, if the problem is linear and the noise is additive and Gaussian [9]), but developed to cope with nonlinear state transition and measurement models. The essence of the UKF is to use the scaled unscented transform, that propagate through the true nonlinear models a set of selected deterministic points, and from these points, the random Gaussian variables are computed, then the Kalman filter equations are applied where the current observation $\mathbf{z}_t$ is added [7]. The UPF algorithm is:

Algorithm 2: UPF

```
  For $t = 0$:
    For $i = 1, ..., N$:
1     $\mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0), \qquad w_0^{(i)} = \frac{1}{N}$
2     $\mathbf{P}_0^{(i)} = E[(\mathbf{x}_0 - \bar{\mathbf{x}}_0)(\mathbf{x}_0 - \bar{\mathbf{x}}_0)^T]$
  For $t \geq 1$:
    For $i = 1, ..., N$:
3     $(\boldsymbol{\mu}_t^{(i)}, \mathbf{P}_t^{(i)}) = \text{UKF}(\mathbf{x}_{t-1}^{(i)}, \mathbf{P}_{t-1}^{(i)}, \mathbf{z}_t)$
4     $\mathbf{x}_t^{(i)} \sim \mathcal{N}(\boldsymbol{\mu}_t^{(i)}, \mathbf{P}_t^{(i)})$
5     $w_t^{(i)} \propto p(\mathbf{z}_t|\mathbf{x}_t^{(i)}) w_{t-1}^{(i)}$
6     $w_t^{(i)} \propto \frac{p(\mathbf{z}_t|\mathbf{x}_t^{(i)}) p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{q(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)}, \mathbf{z}_{1:t})} w_{t-1}^{(i)}$
7     $\tilde{w}_t^{(i)} = \frac{w_t^{(i)}}{\sum_{j=1}^{N} w_t^{(j)}}$
8     RESAMPLING$(\mathbf{x}_t^{(i)}, \mathbf{P}_t^{(i)}, w_t^{(i)})$
9     $w_t^{(i)} = \frac{1}{N}$
```

Both the UPF and the PF are pretty similar, the principal difference is the importance distribution from where the particles are sampled. Using the UKF instead of the simple state transition probability, yields superior results, not only against degeneration, but with more particles in high likelihood regions, the estimates will be better too. The main drawback is the bigger computational complexity of the UPF, primarily because of the need to compute the UKF for each particle.

1.4. Localization and Tracking Methods

A big number of methods of localization and tracking 3D of objects have been proposed in the last decades. This methods have advantages and drawbacks and can be classified in four categories: feature-based methods, edge-based methods, region-based methods and direct methods. Feature-based methods require sufficient texture on the object surfaces, in order to establish feature correspondences between the object and the object's model, thus they do not cope well with poor-textured objects. Edge-based methods rely on strong edges, and so they often struggle with messy backgrounds and motion blur. Region-based methods focus on the statistical properties of different regions in an image, in general, an outer and inner regions of an object in an image, where the objective is to maximize the difference between the regions and to minimize the difference between the inner region and the object's model. However, for objects or backgrounds with a lot of texture the statistical difference between the regions degrades due to the variation of the model and the background statistics, therefore they should be used to track homogeneous objects. The objective of the direct methods is to minimize the photometric error through the direct alignment of object's model in an object in the image, the biggest drawback of this method is that it is very sensitive to illumination changes [4].

Given the disadvantages of the region-based method, the algorithms developed in this work, only considered a homogeneous ball.

2. Methodology

Based on the algorithm developed by Taiana et al. [5] and with the intent of getting a better performance of the algorithm, the UPF was proposed in this work. As already declared, the biggest advantage of the UPF is the fact that, it introduces a current observation with the purpose to steer the particles to high likelihood regions. For instance, if an object hits an obstacle, that might cause the object to completely change its trajectory, but if the model does not account that, the prediction will be completely wrong. Having a current observation will correct the prediction and steer the particles away from the prediction. Before explaining the algorithm proposed, the one developed by Taiana et al. will be enlightened.

The architecture of the system developed by Taiana et al. is based on a particle filter, where each particle represents an hypothesis about what the actual state of the system may be, rather than the state in the image. This allows one to overcome the inversion of the nonlinearity caused by the camera projection model and enables the use of realistic 3D motion models. Doing this concentrates all nonlinearities in the observation model, in which, each particle project a few tens of points onto the image, one set inside and the other outside the 3D object's contour model. From these points the color distribution of the inner and outer regions are constructed and with the color distribution of the object's color model, a metric is applied to compute the likelihood of each particle [5].

Having said that, the algorithm can be explained in two steps, the prediction model, also known as state transition probability and the observation model, also known as measurement probability. The state of the object in this work corresponds to the 3D position and velocities of the ball:

$$\mathbf{x}_t = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T. \tag{9}$$

2.1. Prediction Model

The algorithm has been conceived to work with any object and to follow any arbitrary trajectory that an object might take. Given the high diversity of objects and trajectories, a simple model is adopted to cope with the evolution of the state uncertainty. That model is the constant velocity model, which is applied to each particle, and in discrete time is expressed as:

$$\mathbf{x}_t^{(i)} = \mathbf{F}\mathbf{x}_{t-1}^{(i)} + \mathbf{B}\mathbf{a}_t, \qquad \mathbf{a}_t = \mathcal{N}(\mathbf{0}, \mathbf{\Lambda})$$
$$\mathbf{F} = \begin{bmatrix} \mathbf{I}_3 & \Delta_t \mathbf{I}_3 \\ \mathbf{0} & \mathbf{I}_3 \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} \frac{\Delta_t^2}{2}\mathbf{I}_3 \\ \Delta_t \mathbf{I}_3 \end{bmatrix} \tag{10}$$

where $\mathbf{I}_3$ represent the 3×3 identity matrix, Δ_t the sampling time and the process noise $\mathbf{B}\mathbf{a}_t$ corresponds to an acceleration disturbance, where $\mathbf{a}_t$ add random Gaussian noise to the state variables. $\mathbf{\Lambda}$ represents the diagonal matrix that contains the variances of the random noise [5].

2.2. Observation Model

To compute the likelihood of a particle, multiple points are projected in an image. The likelihood is considered high, if a lot of points in the inner region and few points in the outer region, have colors similar to the object's color model. To express this mathematically, with the chromatic information of the points, the inner and outer color normalized histograms $\mathbf{H}_{IN}$ and $\mathbf{H}_{OU}$ are computed together with the object's color model normalized histogram $\mathbf{H}_{MO}$. The color histograms are computed using the Hue-Saturation-Intensity (HSI) color model, which represents the red, blue and green components in the chromatic components hue and saturation and the achromatic component intensity. The advantage of using this model is the robustness against illumination changes [12]. Thus, a histogram with the number of bins by component hue, saturation and intensity given respectively by B_H, B_S ad B_I can be computed as:

$$\mathbf{H}(i,j,k) = \beta \sum_{n=1}^{N} \delta_{i,b(\mathbf{p}_n^H)} \delta_{j,b(\mathbf{p}_n^S)} \delta_{k,b(\mathbf{p}_n^I)} \tag{11}$$

where N is the number of used points to compute the histogram, δ represent the Kronecker delta function, the terms $b(\mathbf{p}_n^H)$, $b(\mathbf{p}_n^S)$ and $b(\mathbf{p}_n^I)$ indicate the index of the bin of the histogram associated to the H, S and I components of the pixel color $\mathbf{p}_n$ and β is the constant used to normalized the histogram [6]. The histograms bins were set to $B_H = B_S = 12$ and $B_I = 4$, with a lower number of bins for the intensity, to achieve more robustness against illumination changes [11]. Next, to quantify the similarity between histograms, a metric based on the Bhattacharyya coefficient is defined:

$$\mathcal{D} = (1 - S_{IM} + KS_{IO})/(1+K) \tag{12}$$

where S_{IM} is the similarity between the inner and the model histograms, S_{IO} corresponds to the similarity between the inner and outer histograms and the parameter K balances the influence of the contributions of the similarities and was adjusted to $K = 1.5$. Both are calculated by the Bhattacharyya coefficient:

$$S_{IM} = \sum_{i=1}^{B_H} \sum_{j=1}^{B_S} \sum_{k=1}^{B_I} \sqrt{\mathbf{H}_{IN}(i,j,k)\mathbf{H}_{MO}(i,j,k)}$$
$$S_{IE} = \sum_{i=1}^{B_H} \sum_{j=1}^{B_S} \sum_{k=1}^{B_I} \sqrt{\mathbf{H}_{IN}(i,j,k)\mathbf{H}_{OU}(i,j,k)}. \tag{13}$$

This metric must be near zero, when the inner histogram is similar to the model histogram and at the same time, different from the outer histogram. At last, the observation model is modeled by a Laplacian distribution over the distance $\mathcal{D}$:

$$p(\mathbf{z}_t|\mathbf{x}_t^{(i)}) \propto e^{-\frac{\mathcal{D}}{\epsilon}} \tag{14}$$

where ϵ was set to $\epsilon = 1/30$ [5].

2.3. Proposed algorithm

As it can be perceived in the section 2.1, using the constant velocity model as the prediction model, might not be adequate in cases of high variability or spaces with a lot of obstacles. For theses cases, adding a current observation to correct the prediction can make a big difference, impeding the degeneration of the filter. For that reason, in this project the UPF was proposed, which uses the UKF as the importance distribution. Since the rest stays almost identical to the PF, only the UKF will be addressed.

The UKF main tool is the scaled unscented transform (SUT), this one is a method for calculating the statistics of a random variable $\mathbf{x}$, after suffering a nonlinear transformation $\mathbf{y} = f(\mathbf{x})$. To estimate $\mathbf{y}$, first a set of $2n+1$ weighted samples also know as sigma points $\mathcal{S} = \{\boldsymbol{\chi}, \mathbf{w}\}$, where n is the dimension of the random variable $\mathbf{x}$ are deterministically set as follows:

$$\begin{aligned}
\boldsymbol{\chi}_0 &= \bar{\mathbf{x}} \\
\boldsymbol{\chi}_i &= \bar{\mathbf{x}} \pm \left(\sqrt{(n+\lambda)\mathbf{P}}\right)_i, \quad i = 1, ..., 2n \\
w_0^{(m)} &= \lambda/(n+\lambda) \\
w_0^{(c)} &= \lambda/(n+\lambda) + 1 - \alpha^2 + \beta \\
w_i^{(m)} &= w_i^{(c)} = 1/(2(n+\lambda)), \quad i = 1, ..., 2n
\end{aligned} \tag{15}$$

where λ, α and β are parameters for adjusting the dispersion of the sigma points, $w^{(m)}$ and $w^{(c)}$ are the weights to compute the mean and covariance of $\mathbf{y}$ and $\left(\sqrt{(n+\lambda)\mathbf{P}}\right)_i$ represents the ith row or column of the matrix square root of $(n+\lambda)\mathbf{P}$. Then, each sigma point is propagated through the nonlinear function $\boldsymbol{\mathcal{Y}} = f(\boldsymbol{\chi})$, and finally the estimated mean and covariance of $\mathbf{y}$ are computed as:

$$\bar{\mathbf{y}} = \sum_{i=0}^{2n} w_i^{(m)} \mathcal{Y}_i \quad \mathbf{P}_y = \sum_{i=0}^{2n} w_i^{(c)} (\mathcal{Y}_i - \bar{\mathbf{y}})(\mathcal{Y}_i - \bar{\mathbf{y}})^T. \tag{16}$$

The UKF, has a prediction step and an update step. The prediction is composed by a process model and a process noise and the update is composed by an observation model and an observation noise. The noises serve to model the uncertainties of the process and the observation. The chosen process model is the constant velocity model but this time, no noise is added directly, the UKF takes noise into account by using the covariance matrix $\mathbf{Q}_t$ set by the user. So, for each particle the sigma points are generated and propagated trough the process model $\boldsymbol{\mathcal{Y}} = f(\boldsymbol{\chi})$ and the prediction distribution is estimated. Then it's time to insert a current observation into the estimate. The observation $\mathbf{z}_t$ was acquired, by performing an edge-based method where first a process of color segmentation is applied to the image, thus obtaining various candidates to the ball position. Then, some other methods are applied to get the remaining contours of the ball in the image, and a reconstruction process from 2D to 3D is performed to return the estimate of ball's position. The observation noise is also a covariance matrix $\mathbf{R}_t$ that is set by the user and expresses the uncertainty of the observation estimate. For the filter to perform the update, that is to incorporate the information of the observation in the estimate, the prediction must be in the observation space so these two can be compared. To accomplish that, the observation model g is used to transform the sigma points $\boldsymbol{\mathcal{Z}} = g(\boldsymbol{\mathcal{Y}})$. Since $\mathbf{z}_t$ measures directly the position components of the state, then for each sigma point:

$$g = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_3 \end{bmatrix}. \tag{17}$$

2.4. Resampling method

For both methods the resampling method used was the systematic resampling. The basic idea of this method is to generate one random number like $u_t^{(1)} \sim \mathcal{U}(0, 1/N)$ and set other $N-1$ numbers as $u_t^{(n)} = u_t^{(1)} + (n-1)/N, \quad n = 2, ..., N$ where N is the number of particles. Then the particle $\mathbf{x}_t^{(m)}$ will be resampled if:

$$Q_t^{(m-1)} < u_t^{(n)} \leq Q_t^{(m)} \tag{18}$$

where $Q_t^{(0)} = 0$ and $Q_t^{(m)} = \sum_{k=1}^{m} \tilde{w}_t^{(k)}$ [10].

2.5. Getting the inner and outer points

A method that projects on the image, a state hypothesis of the ball is essential. It's with this projection, that the color histograms are computed to evaluate the likelihood of each particle in the observation model of both filters, thus this method should be efficient because it will be used numerous times.

As a ball is a spherical surface and a spherical surface is a particular case of a quadric, in the work developed by Cross and Zisserman [1], one finds that the projection of the silhouette of a quadric in a image, results in a conic, and for the particular case of a spherical surface the conic is an ellipse, as it can be seen in figure 2 [1].

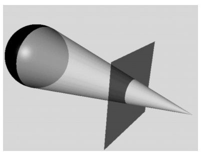
Figure 2: Projection of a spherical surface to an image [1].

Well, admitting that the camera coordinate system coincides with the world coordinate system then there is an expression that relates the ellipse **C** with an arbitrary spherical surface and it's given by [3]:

$$\mathbf{C} = \mathbf{K}^{-T} \underbrace{\left(\mathbf{I}_3 - \frac{\mathbf{x}_0\mathbf{x}_0^T}{||\mathbf{x}_0||^2 - R^2}\right)}_{=\mathbf{H}} \mathbf{K}^{-1}. \tag{19}$$

where **K** is the camera intrinsic matrix, $\mathbf{x}_0 = (x, y, z)$ is the ball location and R is radius of the ball. Thus, for a point in the image given in homogeneous coordinates $\tilde{\mathbf{m}} = \begin{bmatrix} u & v & 1 \end{bmatrix}^T$ to belong to the projection of the spherical surface silhouette, it needs to satisfy the following equation:

$$\tilde{\mathbf{m}}^T \mathbf{K}^{-T} \mathbf{H} \mathbf{K}^{-1} \tilde{\mathbf{m}}^T = 0. \tag{20}$$

With this equation, a simple and efficient way to project the points can be achieved by starting from a circle centered in the origin and transforming it to the ellipse **C**, by performing the eigenvalue decomposition of the **H** matrix, thus $\mathbf{H} = \mathbf{VSV}^T$, and making the matrix that contains the eigenvectors **V** to be orthonormal. The obtained matrix of the eigenvalues **S**, as can be seen in equation (21):

$$\mathbf{S} = diag\left(1, 1, \frac{1}{1 - \left|\left|\frac{\mathbf{x}_0}{R}\right|\right|^2}\right) \tag{21}$$

has two unitary eigenvalues and so, it's possible to map the ellipse **C** in a circumference, by making the transformation $\tilde{\mathbf{m}}_p = \begin{bmatrix} u_p & v_p & 1 \end{bmatrix}^T = \mathbf{V}^T\mathbf{K}^{-1}\tilde{\mathbf{m}}$:

$$\begin{aligned} \tilde{\mathbf{m}}_p^T \mathbf{S} \tilde{\mathbf{m}}_p &= 0 \\ u_p^2 + v_p^2 &= \frac{1}{\left|\left|\frac{\mathbf{x}_0}{R}\right|\right|^2 - 1}. \end{aligned} \tag{22}$$

Finally, to obtain a set of projected points $\tilde{\mathbf{m}}'$ in the ellipse **C**, one just needs to sample some points **m** from the circumference above (22) and apply the inversion of the transformation made above ($\tilde{\mathbf{m}}' = \mathbf{V}^T\mathbf{K}^{-1}\tilde{\mathbf{m}} = \mathbf{KV}\tilde{\mathbf{m}}$). However, as said earlier, the points must be put around the contours of the ball, so to project the set of inner points, it's used a radius below the real one, for instance $0.95R$ and for the set of outer points, it's used a bigger radius, like $1.05R$.

2.6. Impact Model

In this work, one wants to expand the constant velocity model with the purpose to deal with collision against obstacles. In an impact, the colliding bodies exert big forces in each other, and so, the resulting velocities of the bodies after the impact, can be really different from the ones before the impact. If only the constant velocity model is applied in such a case like this, the filters will return poor estimates of the state, specially in the velocity components. For this reason, an impact model will be incorporated to the constant velocity model.

The most known and simple formulas to calculate the velocities of two colliding bodies after an impact, are the next ones:

$$e = -\frac{(\mathbf{v}_A^{'})_n - (\mathbf{v}_B^{'})_n}{(\mathbf{v}_A)_n - (\mathbf{v}_B)_n} \tag{23}$$

$$(\mathbf{v}'_A)_t = (\mathbf{v}_A)_t, \quad (\mathbf{v}'_B)_t = (\mathbf{v}_B)_t \tag{24}$$

where e is the coefficient of restitution that models the loss of energy in an impact, n and t corresponds respectively to the normal and tangential components to the impact surface, $\mathbf{v}_A$ and $\mathbf{v}_B$ and $\mathbf{v}'_A$ and $\mathbf{v}'_B$ represent respectively the velocities of the bodies A and B before and after an impact [2]. Assuming that the collisions are always central and that the obstacle remains static, combining both the normal and tangential equations of the velocity, then the velocity after the impact of the ball $\mathbf{v}'_{Ball}$ is given by:

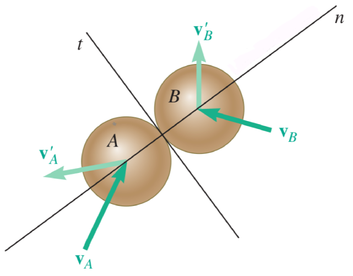

Figure 3: Central impact between two bodies A e B where $\mathbf{v}_A^{'}$ e $\mathbf{v}_B^{'}$, represent the velocities after the impact [2].

$$\mathbf{v}'_{Ball} = \mathbf{v}_{Ball} - (1+e)\big(\mathbf{v}_{Ball}{}^T\mathbf{n}\big)\mathbf{n} \tag{25}$$

where $\mathbf{v}_{Ball}$ is the velocity before the impact. Having presented this formula, this one is incorporated with the process model as follows. First, each particle is propagated normally with the process model:

$$\mathbf{x}_t^{(i)} = f(\mathbf{x}_{t-1}^{(i)}, \Delta_t) \tag{26}$$

where f is the process model and Δ_t the sampling time. If the particle don't surpass an obstacle, then the impact model is not applied. However if it surpasses an obstacle, it indicates that there was a collision and in that case, the time of collision t_{OC} is calculated and instead of propagating the particle for Δ_t, it's propagated as $\mathbf{x}_t^{'(i)} = f(\mathbf{x}_{t-1}^{(i)}, t_{OC})$. At the time of collision, the impact model (25) is applied to get the velocity after the impact and finally, the process model is repeated again, but using the rest of the time till the sampling time $\mathbf{x}_t^{(i)} = f(\mathbf{x}_t^{'(i)}, \Delta_t - t_{OC})$.

3. Results and discussion

In order to access the performance of the PF and UPF, several tests over simulated trajectories and real experiments were made. Results of those tests for a simulated circular trajectory and for a real free fall trajectory, where the impact model will be applied are shown in this section.

The main advantage of a simulator is that, the exact state of the ball is previously known, instead, for a real experience the state is not accessible and needs to be measured. Not having access to a precise sensor, to obtain the ground truth position of the ball, a trail and error method was used. This method basically consists in manually defining a 3D point $\mathbf{p} = (x, y, z)$, which represents the ball position in space, project the silhouette of the ball in that point to the image and iteratively, adjust the x, y and z components, in a way that the silhouette coincides perfectly with the true ball position. This process is not rigorous, it depends on the real camera calibration process and because it is really difficult to get the silhouette to coincide perfectly with the ball, and for that reason it was decided not to acquire the velocity ground truth. Thus for the real trajectory only the ball's position estimates will be shown.

For both filters, there are adjustments parameters that affects the performance of the filters. The principal and only parameters tested are: the number of particles (the higher the number the better are the estimates [7] but the lower is the efficiency), the sizes of the inner and outer silhouettes, (that controls the measurement error which can be seen in figure 4) and the process model noise (that regulates the dispersion of the particles in the state space).

To analyze the influence of the number of particles it will be used the root mean square error:

$$\text{RMSE} = \sqrt{1/N \sum_{i=0}^{N} ||\mathbf{x}_i - \hat{\mathbf{x}}_i||^2} \tag{27}$$

where N represents the number of estimates that belong to the experience, and $\mathbf{x}_i$ and $\hat{\mathbf{x}}_i$ corresponds, respectively, to the exact state and the state estimate i. To examine the influence of the silhouette sizes and process noises, precision plots will be used. This plots instead of the RMSE, can catch if a filter loses track of an object, and for filters like this, this scenario often happens. Precision plots express the percentage of estimates that possess an error below a given error threshold, as this error threshold increases [13]. The error threshold that will be used for the results is the relative error δ. Therefore, the following equation is used to compute the percentage of the estimates F, that possess an error below an arbitrary relative error δ:

$$F = \frac{100}{N} \sum_{i=1}^{N} H\left(\delta - \frac{||\mathbf{x}_i - \hat{\mathbf{x}}_i||}{||\mathbf{x}_i||}\right) \quad [\%] \tag{28}$$

where N is the number of estimates, and $\mathbf{x}_i$ and $\hat{\mathbf{x}}_i$ represent respectively, the exact state and the state estimate i and the function H, is the Heaviside function. Both filters deal with random variables, thus, making tests with the same parameters originates different results, for this motive each test is repeated 100 times. For all the plots, the tests were made using $N = 1024$ particles, where the red lines represent the tests with an inner silhouette of $0.9R$ and outer silhouette of $1.1R$, the green corresponds to $0.8R$ and $1.2R$ and the blue $0.7R$ and $1.3R$. This means that the red lines use closer silhouettes, as shown in the figure 4.

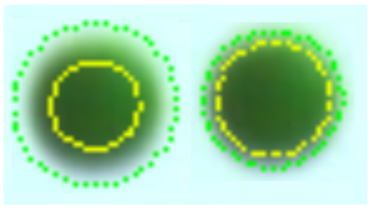
Figure 4: Projection of different silhouette sizes. On the left, is used as inner and outer silhouettes a radius of $0.7R$ and $1.3R$ and on the right a radius of $0.9R$ and $1.1R$.

The solid, dashed and dotted lines, represent three different process noise configurations. This parameter is really difficult to adjust and depends on the trajectory. For the PF the solid lines represents the results when using an experimental adjusted process noise, and the dashed and dotted lines, corresponds respectively to a configuration of noise one order of magnitude above and below the one used for the solid lines. For the UPF, also three different configurations of the process noise using different orders of magnitudes are set for both trajectories, only the noise component of the velocity in the z axis, is adjusted for the trajectory and the observation noise for the simulated trajectory was set to a covariance of $\mathbf{R}_t = diag(10^2, 10^2, 10^2)$ mm^2 and for the real trajectory $\mathbf{R}_t = diag(50^2, 50^2, 50^2)$ mm^2.

3.1. Simulated circular trajectory

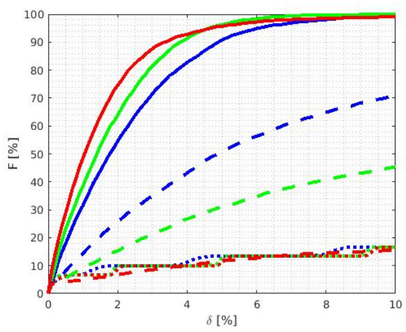

Figure 5: PF Precision plot for the estimate of the position.

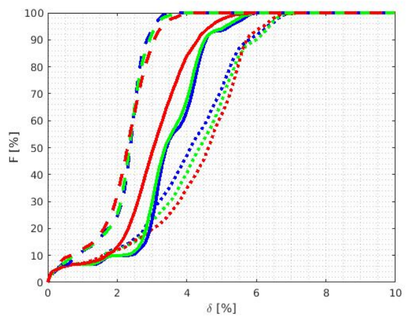

Figure 6: UPF Precision plot for the estimate of the position.

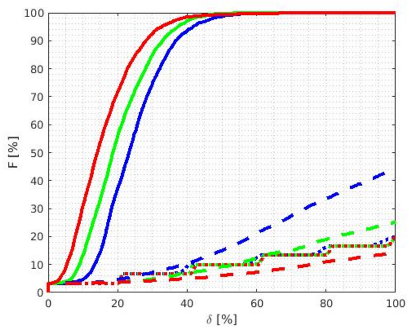

Figure 7: PF Precision plot for the estimate of the velocity.

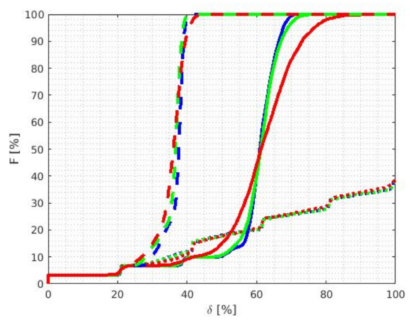

Figure 8: UPF Precision plot for the estimate of the velocity.

From the obtained results, one can conclude that for both filters the estimates of the position are way better than the velocity ones. This is because the observation model of the PF and UPF only considers the position of the particles, ignoring the velocities. For instance, a particle that possess an adequate position value but a wrong velocity, will be favored by the observation model, where this one will assign a high importance weight to the particle, this affects and deviates the velocity estimations of the ball.

Other aspect is the high sensitivity that the PF exhibits for different process noises (see figures 5 and 7). For this filter, the process noise is directly related to the acceleration of the object as it can be seen in equation (10) and for an order of magnitude below or above the process noise used by the solid lines, the filter loses track of the ball and degenerates, which leads to wrong estimates. For the dotted lines the problem is the low scattering of the particles, and so, the filter cannot keep up with the object's movement. For the dashed lines, the particles get scattered too much and deviate from the ball which degenerates the filter. For the UPF the estimates of the position are all adequate for different process models, however the same doesn't happen for the velocity (see figures 6 and 8). One possible explanation is that, the process noise might not be adequate to this trajectory, thus, if true, for this process model and observation model, the velocity estimates depends on the process noise covariance.

On the PF the solid lines that use an adequate process noise, exhibit good results in both position and velocity as it can be verified in figure 5 and 7. The difference resides in the silhouette sizes. The red line that correspond to the closer silhouettes, leads to a lower measurement error and for a relative error of 2% in position and 20% the percentage of frames below that errors are above the other two cases of furthest silhouettes. But as one considers bigger errors, the green and blue lines reach 100% first than the red line. This happens because from the 100 simulations, there was cases where, the filter degenerated. That's because, using a lower measurement error makes the likelihood function to narrow and that increases the risk of degeneration as it was said in section 1.3.

In the table 1 are exposed results of the root mean square error for both filters, using identical parameters varying just the number of particles. One can verify that the results coincides with the literature, once as the number of particle increases, better are the estimates.

N	128	256	1024
Position PF	6.32×10^8	392.35	24.13
Position UPF	27.05	26.04	23.98
Velocity PF	6.32×10^8	1319.17	415.82
Velocity UPF	673.49	666.71	660.97

Table 1: RMSE error in mm for different number of particles.

3.2. Real free fall trajectory

For this trajectory, both filters were tested with and without the impact model. One difference quite visible between these plots and the simulated ones, is that for the real experiences, the relative error is bigger, at least the double, which make sense, because the simulator does not take into account the real phenomena of the world. Other conclusion that one can take is that using the impact model increases the precision of the estimates, mainly for the particle filter, because the prediction of this one does not incorporate a current observation like the UPF. Making a more detailed examination, one can see that using the furthest silhouettes gives better results for both filters. One possible explanation for this is that factors like noise or motion blur, can augment the ball contour and because of that, using the closest silhouettes is not appropriate. With this results one can prove the practical utility of both filters, even though the error doubles, this values continue to be really small.

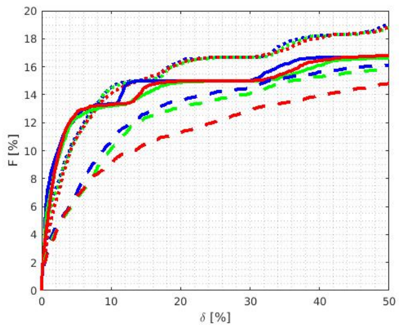

Figure 9: PF Precision plot for the estimate of the position without impact model.

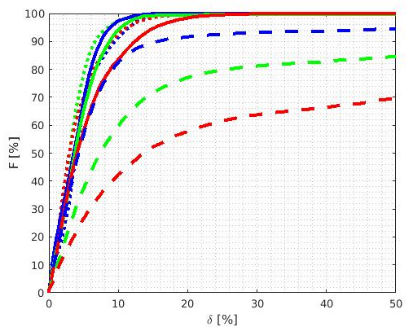

Figure 10: PF Precision plot for the estimate of the position with impact model.

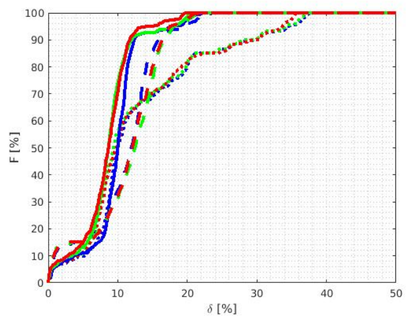

Figure 11: UPF Precision plot for the estimate of the position without impact model.

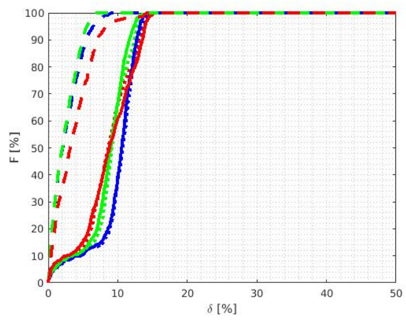

Figure 12: PF Precision plot for the estimate of the position with impact model.

4. Conclusions

The results obtained in this work, allows one to conclude that the implemented filters function with success, if the filters initial parameters are adjusted accordingly to the object's trajectory. It can be seen that the particle filter is really sensitive to the initialization parameters, whereas the unscented particle filter is not, once in the PF with a suitable process noise, using a low number of particles often resulted in the degeneration, on the other hand, with the UPF that did not happen. Also with this results, one can verify that the UPF is very robust against different silhouette sizes, as well as for different process noises. However, for the velocity estimation, the results were not as accurate as it was expected, as the results prove to be more dependent on the initialization parameters. Another information that can be perceived is that the impact model improves the accuracy of the estimated state, specially for the PF. About the efficiency, the PF is way faster than the UPF, so for an application that requires fast estimates, the UPF may not be appropriate.

The work present in this thesis exhibit some limitations. One of them, is the fact that the velocity estimations have not been made for the real trajectories, also, the method used to obtain the exact position of the real trajectories is not rigorous, and so, the position estimates for the real experiences may not be exact, and given the infinity possibilities of different tests, a small number of tests were made. Other limitation, was proprieties like re-tracking after a filter loses track of the ball, which happens frequently in environments with a lot of obstacle that may occlude the ball, was not considered.

As future work, and given the drawbacks of the region-based methods referred in section 1.4, in which states, that they can only function properly for homogeneous objects, one project to be considered is the one developed by Leisheng Zong et. al [4], which combines region-based methods with direct-methods, creating a hybrid model, on the attempt to join the advantages of both methods. This work can be adapted in this project in order to make it usable for more complex objects. From the efficiency point of view, one propriety quite interesting, is that the particles in these algorithms are all computed in the same way and independently. This two characteristics make them fit to be computed in parallel, and that would speedup the estimation time of the filters. Both algorithms were implemented in serial, so, if one wants to track other type of objects more complicated, that requires a more complex particle propagation or observation models, a serial code would not function in real-time. For this reason, developing the code using the graphics processing Unit (GPU) can be considered, this processor is composed by hundreds or thousands of small and simple specialized cores, optimized to perform instructions in parallel.

References

[1] G. Cross and A. Zisserman. Quadric Reconstruction from Dual-Space Geometry. In *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, pages 25–31. IEEE, January 1998.

[2] E. Johnston, F. Beer and E. Eisenberg. *Vector Mechanics for Engineers: Statics and Dynamic.* McGraw-Hill, New York, 12th edition, 2009.

[3] N. Greggio, J. Gaspar, A. Bernardino, and J. Santos-Victor. 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, 2011.

[4] L. Zhong and L. Zhang. A Robust Monocular 3D Object Tracking Method Combining Statistical and Photometric Constraints. *International Journal of Computer Vision*, 127:973–992, August 2019.

[5] M. Taiana, J. Santos, J. Gaspar, J. Nascimento, A. Bernardino and P. Lima. Tracking objects with generic calibrated sensors: An algorithm based on color and 3d shape features. *Robotics and Autonomous Systems*, 58(6):784–795, June 2010.

[6] P. Perez, J. Vermaak and A. Blake. Data Fusion for Visual Tracking with Particles. *Proceedings of the IEEE*, 92(3):495–513, November 2004.

[7] R. Van Der Merwe, A. Doucet, N. de Freitas and E. Wan. The unscented particle filter. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, NIPS'00, pages 563–569, Cambridge, MA, USA, 2000. MIT Press.

[8] S. Thrun, W. Burgard and D. Fox. *Probalistic robotics*. The MIT Press, 1st edition, 2005.

[9] S. Särkkä. *Bayesian Filtering and Smoothing*. Cambridge University Press, 1st edition, 2013.

[10] T. Li, M. Bolic and P. Djuric. Resampling methods for particle filtering: Classification, implementation, and strategies. *IEEE Signal Processing Magazine*, 32(3):70–86, 2015.

[11] M. Taiana. 3D model-based tracking with one omnidirectional camera and particle filters. Thesis, Electrical and Computer Engineering, IST and Politecnico di Milano. October 2007.

[12] U. Jau, C. Teh and G. Ng. A Comparison of RGB and HSI Color Segmentation In Real - Time Video Images: A Preliminary Study On Road Sign Detection. In *2008 International Symposium on Information Technology*, volume 4, pages 1–6. IEEE, August 2008.

[13] Y. Wu, J. Lim and M. Yang. Online Object Tracking: A Benchmark. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2411–2418. IEEE, June 2013.