Received March 8, 2021, accepted March 22, 2021, date of publication March 31, 2021, date of current version April 12, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3070180

Multi-Robot Sensor Fusion Target Tracking With Observation Constraints

THULIO G. S. AMORIM[1], LEONARDO A. SOUTO[1], TIAGO P. DO NASCIMENTO[1,2], (Member, IEEE), AND MARTIN SASKA[2], (Member, IEEE)

[1]Laboratory of Systems Engineering and Robotics, Universidade Federal da Paraiba, João Pessoa 580480-180, Brazil
[2]Department of Cybernetics, Czech Technical University in Prague, 166 36 Prague, Czech Republic

Corresponding author: Tiago P. Do Nascimento (tiagopn@ci.ufpb.br)

This work was supported by the Czech Science Foundation (GAČR) under Project 20-10280S.

ABSTRACT In Mobile Robotics, visual tracking is an extremely important sub-problem. Some solutions found to reduce the problems arising from partial and total occlusion are the use of multiple robots. In this work, we propose a three-dimensional space target tracking based on a constrained multi-robot visual data fusion on the occurrence of partial and total occlusion. To validate our approach we first implemented a non-cooperative visual tracking where only the data from a single robot is used. Then, a cooperative visual tracking was tested, where the data from a team of robots is fused using a particle filter. To evaluate both approaches, a visual tracking environment with partial and total occlusions was created where the tracking was performed by a team of robots. The result of the experiment shows that the non-cooperative approach presented a lower computational cost than the cooperative approach but the inferred trajectory was impaired by the occlusions, a fact that did not occur in the cooperative approach due to the data fusion.

INDEX TERMS Visual tracking, multi-robot systems, data fusion, particle filter.

I. INTRODUCTION

Object detection is a well-known research area in Computer Vision, while object tracking and its various applications are well researched in Mobile Robotics as well. Given the initial state (for example, position and dimensions) of a targeted object in a first image frame, the detected object is tracked by the target states in the following image frames [1]. According to [2], efforts have been directed towards the deployment of groups of networked autonomous mobile robots that interact autonomously with one another and with the environment to significantly improve the efficiency, performance, reconfigurability, and robustness that individual vehicles currently do not have.

When there is an interaction between robots, the communication between them allows the possibility of merging the data collected from the environment. Data fusion is an advanced technique for combining information from various sources in order to obtain more accurate results [3]. Data fusion systems are now widely used in several areas, such as sensor networks, robotics, video and image processing, and intelligent system design [4].

Although much effort has already been put into the tracking community, robust long-term tracking is still a challenging problem and an active research topic due to changes in lighting and posture, objects that have complex movements, total or partial occlusion and real-world scenarios [5]. Tracking objects or people moving in large spaces, where the field of view of common sensors is relatively small when compared to the monitored area, is best achieved using cooperative, static, or dynamic sensor teams, for example, mounted on mobile robots [6]. Due to these facts, recent work on tracking seeks to develop methods that allow the use of multiple sensors. In this work, however, we propose a method of fusing data captured from the environment using different robot sensors, making the visual tracking of an object in three-dimensional ($3D$) space robust to partial and total occlusions. Thus, our contributions can be summarized as follows:

1) A multi-robot tracking sensor fusion system using pinhole-type RGB cameras with a constrained field of view;
2) A modified particle filter to track an object subject to occlusions and constrained field of view;

The associate editor coordinating the review of this manuscript and approving it for publication was Joewono Widjaja.

3) The application of the particle filter only in the viewing space provided by the camera, avoiding the need for a global map.

Finally, our work is structured as follows. The following section we briefly discuss about the related works. In sec. III we explain our proposed approach in a detailed manner. The following section (sec. IV) shows our experiments that validated our approach. Finally we conclude our work on sec. V.

II. RELATED WORKS

Estimating the position of objects of interest is an important subproblem in many real-world robotic applications. The challenge, however, is to deal with the limitations of the sensors used to detect targets [5]. Usually, several static wireless sensors are employed to improve tracking accuracy and increase the size of the surveillance area. Unlike static sensors, whose density and detection range are fixed, mobile sensors (robots) can cover larger areas over time without the need to increase their number [7]. In this case, the task becomes to move the sensor so that the target falls within its field of view. The use of multiple robots can increase the effective field of view of the sensor and therefore should, in theory, allow for superior performance.

With respect to a single robot, teams of robots are obviously less prone to failure, able to operate simultaneously in larger areas. The exchange of information between robots allows data to merge [8], [9]. Furthermore, since the appearance of a target and the environment changes dynamically, the majority of existed visual tracking algorithms tend to drift away from targets. To address this issue, Dou *et al.* [10] proposed a robust tracking algorithm by integrating the generative and discriminative model. The object appearance model is made up of a general target model and a discriminative classifier. For the generative target model, the authors adopt the weighted structural local sparse appearance model combining patch-based gray value and Histogram of Oriented Gradients feature as the patch dictionary. By sampling positives and negatives, alignment-pooling features were obtained based on the patch dictionary through local sparse coding. Then they used a support vector machine to train the discriminative classifier. The proposed method was also embedded into a Bayesian inference framework for visual tracking.

In mobile robot applications, fusion refers to any stage of the integration process where a real combination of different sources of information occurs. The main advantage of data fusion is improving data authenticity or availability [4]. Although a wide variety of methods have been proposed [11], [12], long-term object tracking is still a challenging problem when dealing with occlusion, out-of-view, scale, and illumination variation. To address these challenges, Hu *et al.* [5] proposed a robust visual object tracking method based on binocular vision. Their method formulated the object tracking problem in a multi-cue fusion framework which allowed the system to recover from tracking drift and occlusion.

In contrast, applications that use data fusion should take precautions in relation to problems that may arise from the combination of the data [13]. Among these problems, we have imperfect data. According to Khaleghi *et al.* [4], imperfect data appears because the data provided by the sensors is always affected by some level of inaccuracy, as well as uncertainty in the measurements. Data fusion algorithms must be able to express these imperfections effectively and exploit data redundancy to reduce their effects. In this same line of research, the work of Leang *et al.* [14] presented a description of a generic framework for combining and/or selecting on-line the different components of the processing chain of a set of trackers, and examines the impact of various fusion strategies.

Uncertainty arises if the robot lacks critical information to perform its task. The ability to deal with uncertainty is critical to building successful robots [3], [7]. Probabilistic models have become popular due to their robustness in the presence of uncertainty, which makes them capable of handling large amounts of sensor noise and occlusion [6], [15]. One of the probabilistic models that are widely used to perform visual tracking within Robotics is the Particle Filter.

Particle filters have become popular tools for solving the tracking problem [16]. Its popularity stems from its simplicity, flexibility, ease of implementation, and modeling success in a wide range of challenging applications and has been used extensively in recent years [6], [17]–[19]. The work of Truong *et al.* [19] is an example. Their work presented an algorithm for single object tracking using a particle filter framework and color histograms. Color histograms were embedded in the particles, and the distances between histograms were used to measure the likelihood between targets and observations. One downside of color histograms is that they ignore spatial information, which may produce tracking failure when objects appear that are similar in color. To overcome this disadvantage, the authors proposed a salience-based weighting scheme for histogram calculation. Given an image region, first, its salience map is generated. Next, its histogram is calculated based on the generated salience map.

III. MODIFIED PARTICLE FILTER APPROACH COOPERATIVE TARGET TRACKING

In this work, we propose a multi-robot sensor fusion based on a modified particle filter for object tracking. Although there exists a similarity to the work of Ahmad and Lima [6], our approach differs on the existence of the constraint of the field-of-view and the fact that the use of a global map becomes needless. Furthermore, we demonstrate that our approach is robust to occlusion.

It is well known in the literature that the particle filter is an approximation technique for calculating posteriors [20], assuming that an x state evolves over a discrete-time in a stochastic dynamic system. The idea is to represent the later state through a hypothesis set X_t in which each hypothesis is represented by a particle. A particle x_t consists of a concrete instantiation of the state x at time instant t together with the weight w_i that determines how close to the true state the instance is.

A. THE MODIFIED PARTICLE FILTER

As an object of interest, a spherical object was used with its surface composed almost entirely of a single color. This object was selected because the image processing in order to detect the object tends to have a lower computational cost due to its simplistic characteristics. However, the implementation is not attached to the object of interest that was selected, so much so that the object of interest can be replaced only by changing the detection step according to the characteristics of the new object and making little or no change on the rest of the implementation.

The representation of the object of interest consists of a point in 3D space that is the center of mass of the spherical object. Regarding the appearance model, global visual representations of color and outline of the spherical object were used. Generative methods were used as a search strategy, that is, it was sought to detect the object of interest only by determining the degree of similarity between it and the candidates found in the environment.

The tracking system that implements the two approaches is described by means of the flowchart presented in Fig. 1. The visual tracking of the object of interest occurs through the use of a particle filter. The following definitions were used:

- The state to be estimated consists of the global position of the center of mass of the object of interest in 3D space;
- A particle x_t consists of instantiating the global position of the center of mass of the object of interest in 3D space at time t;
- An observation z_t represents the global position of the center of mass of the object of interest in 3D space estimated at time t;

The first step is to initialize the particles and the weights associated with each one. In particle filter visual tracking, the plausible positions of the object are represented by particles that are estimated through observations. Each observation is generated by processing the data collected by the sensors. Therefore, the settings of the sensors that are used to obtain information about the environment are decisive in the creation and distribution of the particles.

As shown in Fig. 2, the field of view of a Pinhole camera is determined by the horizontal opening angles a_h and vertical a_v and the maximum d_{max} and minimum d_{min} capture distances. The maximum and minimum capture distances consist of the maximum and minimum distance, respectively, in which the object of interest can be from the camera in relation to the optical axis.

In Fig. 3, some information about the field of view was highlighted. The main one is that given a point $P = (x, y, z)$ in 3D space, the value that variable z assumes delimits the possible values that variables x and y can assume that make point P to still be found within the field of view. Based on

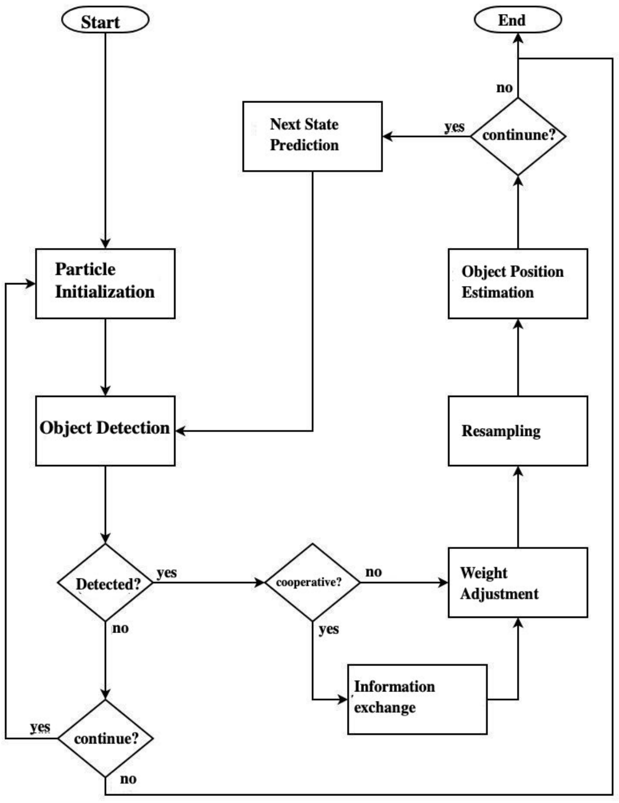


FIGURE 1. Tracking system in a 3*D* space. The contributions are within the Particle initialization block and the cooperative tracking blocks (except the Resampling block).

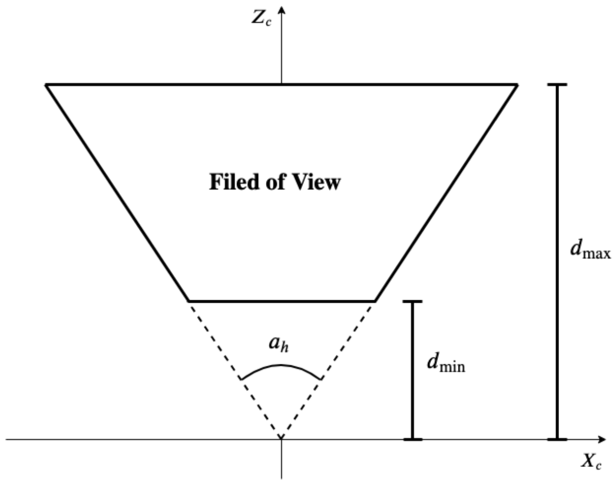


FIGURE 2. Field of view configuration.

trigonometric relationships, equations 1 and 2 were obtained, which were used to generate particles within the camera's field of view.

$$\text{mín. } x_i = -tan(\frac{a_h}{2}) \cdot z_i, \tag{1}$$

$$\text{máx. } x_i = tan(\frac{a_h}{2}) \cdot z_i. \tag{2}$$

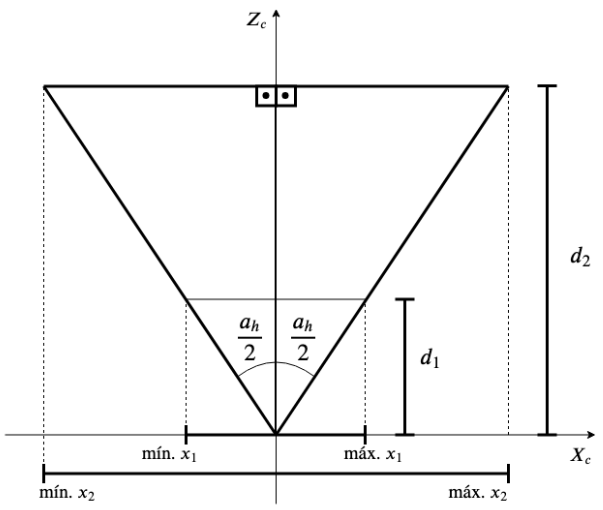


FIGURE 3. Detailed information regarding the field of view.

From the opening angles of the camera and the maximum and minimum capture distances it is possible to determine if a point in $3D$ space is within the field of view. Having V as the set of all points that are within the field of view of a camera and $P = (x, y, z)$ as any point belonging to $3D$ space, point P belongs to set V if the following statements are true:

$$d_{min} \leq z \leq d_{max}, \tag{3}$$
$$-tan(\frac{a_h}{2}) \cdot z \leq x \leq tan(\frac{a_h}{2}) \cdot z, \tag{4}$$
$$-tan(\frac{a_v}{2}) \cdot z \leq y \leq tan(\frac{a_v}{2}) \cdot z. \tag{5}$$

Using the previous statements it was possible to generate particles within the camera's field of view. Note that the initialization algorithm can generate particles below the ground line, that is, points below the positive part of the Y-axis of the camera within the randomness of the algorithm. However, the most important thing in our approach is that the particles are initially sparse. After generating the particles within the camera's coordinate system, they are taken to the global coordinate system. Algorithm 1 describes the particle initialization algorithm. The horizontal opening angles a_h and vertical a_v of the Kinect version 1.0 color sensor are 57° and 43°, respectively. As minimum distance d_{min} and maximum d_{max} the values of 0.8 and 4.0 meters were used. The camera translation variables t_{cx}, t_{cy} and t_{cz} represent the difference between the position of the camera's center of mass and the position of the robot's center of mass in which the camera is fixed in the global coordinate system. The values of the translation variables of the robot t_{rx}, t_{ry} and θ represent the pose in the global coordinate system. The $rand(inf, sup)$ procedure consists of a function that generates a random number from a uniform distribution with a lower limit equal to inf and an upper limit equal to sup. The variable M represents the number of particles.

In order for the data obtained through the sensors to be used by the particle filter, processing must take place so that information about the state that seeks to be observed can be generated. This processing is carried out within the detection stage of the object of interest and results in the global position of the object of interest's center of mass in $3D$ space, that is, a z observation. The data may or may not generate a good observation, since there will be measurements from which it is not possible to extract any information about the state being observed. Observations are differentiated according to criteria that measure this information and these criteria are called the degree of confidence.

Algorithm 1 Particle Initialization Algorithm

```
1:  Input: $(d_{min}, d_{max}, a_h, a_v, t_{cx}, t_{cy}, t_{cz}, t_{rx}, t_{ry}, \theta, M)$
2:  $X_0 = \emptyset$
3:  for $m = 1$ to $M$ do
4:      {Particles are created in the coordinate system of the camera}
5:      $cam_z = rand(d_{min}; d_{max})$
6:      $cam_x = rand(-1; 1) \cdot \tan \frac{a_h}{2} \cdot cam_z$
7:      $cam_y = rand(-1; 1) \cdot \tan \frac{a_v}{2} \cdot cam_z$
8:      {Passes the particles to the coordinate system of the robot (Eq. 22)}
9:      $robot_x = cam_z + t_{cx}$
10:     $robot_y = cam_x + t_{cy}$
11:     $robot_z = cam_y + t_{cz}$
12:     {Passes the particles to the global coordinate system (Eq. 24)}
13:     $world_x = \cos\theta \cdot robot_x - sen\theta \cdot robot_y$
14:     $world_y = \cos\theta \cdot robot_x + sen\theta \cdot robot_y$
15:     $world_z = robot_z$
16:     {Apply translations}
17:     $world_x = world_x + t_{rx}$
18:     $world_y = world_y + t_{ry}$
19:     $world_z = world_z$
20:     $x_0^{[m]} = (world_x, world_y, world_z)$
21:     $X_0 = X_0 + \langle x_0^{[m]}, \frac{1}{M} \rangle$
22: end for
23: return $X_0$
```

B. OBJECT DETECTION

Inside the particle filter, observations are used to correct W_i weights. At the end of this procedure, the degrees of confidence are used to determine the degree of confidence about observation α. The degree of confidence about α observation plays an important role within the particle filter as it is used during weight correction. In the stage of detecting the object of interest, the image from the Kinect is taken as input. As an initial step, the stages of image formation and acquisition together with digitization are carried out through the Robotic Operating System (ROS). Every image generated by the sensor is sent to ROS, which in turn passes it on to the system responsible for tracking the object of interest. But before any operation on the image, a smoothing operation by a median filter is performed in order to reduce the noise present in the image using a 5x5-size mask. Then, the image is converted from the RGB color for HSV in order to facilitate the passage of the object of interest's perceived color by the user to the system.

As the object of interest has a distinct color, this information is used to differentiate it from other objects present in the scene [21], [22]. For this, a threshold operation is performed, which is described by equation 6 where variable I represents the input image, variables $I(x, y)_H$ and $I(x, y)_S$ represent, respectively, the value of hue's components and saturation of the pixel coordinates (x, y) of the input image I and the variable I_{seg} represents the output image of the threshold process. This operation receives the image in the HSV color space and returns a binary image where each pixel of the input image that is within the threshold generates the assignment of the maximum value for the pixel of the output image of the same position or the assignment of the minimum value if contrary. The values used in the threshold were obtained empirically.

$$I_{seg}(x, y) = \begin{cases} 1, & \text{if } 55 \geq I(x, y)_H \geq 25 \\ & \text{and } I(x, y)_S \geq 45, \\ 0, & \text{else.} \end{cases} \tag{6}$$

As an operation of the post-processing step, the opening morphological operation is performed with the purpose of removing sets of pixels of dimensions smaller than the structuring element since it is expected that the object of interest always has a fixed minimum size. The structuring element is described as

$$B = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}. \tag{7}$$

Then, an operation is performed to find the contours within the binary image. This operation outputs the contours found in the input image separated into sets of connected pixels. The next operation consists of classifying the contours where each contour generated by the previous operation is treated as a single object. For each contour, the smallest possible circle is found that encloses all its pixels. From the circle's radius and area, together with the area bounded by the contour, a value is obtained that determines how close the contour resembles the object of interest. This value was denominated as α_1 detection confidence and is calculated by

$$\alpha_1 = \frac{A_{obs}}{A_{esp}}, \tag{8}$$

where A_{obs} represents the observed area, which consists of the area formed by the region that is encompassed by the contour, and A_{esp} represents the expected area, which is the area of the circle that encloses all points of the contour.

The value of the observed area A_{obs} is obtained by the formula of the area from Green's theorem [23], [24] and the value of the expected area A_{esp} is obtained by equation 9 where the variable r_{enc} represents the radius of the circle that encloses all points of the contour.

$$A_{esp} = \pi \cdot r_{enc}^2. \tag{9}$$

The idea for this step arises from the fact that the object of interest has attributes that can be easily seen in the image, and even after the previous operations these attributes remain despite suffering deformations. Thus, we can compare attributes extracted from the image with those of spherical objects in order to determine how close they are to each other. In the calculation described above, the area of the object was selected and how it is distributed in the image as a classification attribute.

After calculating the α_1 detection confidence of all the contours found in the image, the contours that have the α_1 detection confidence value less than 0.5 are discarded and, of the remaining contours, the contour with the highest α_1 detection confidence value is selected. as the outline of the object of interest. As a final part of the detection process, the position of the object of interest in the screen coordinate system $P_t = (x_t, y_t)$ is taken as the centroid of the smallest possible circle that encompasses all points of the selected contour.

C. OBJECT STATE ESTIMATION

The key to estimate object poses is matching feature points in the captured image with predefined ones of the 3D model of the object [25]–[27]. Here we use the perspective projection. Also called the perspective transformation, it is formalized as

$$\lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & c_x & 0 \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}, \tag{10}$$

where $(X, Y, Z)^T$ represents the position of a point in $3D$ space, (u, v) represents the projection of the point (X, Y, Z) on the plane image, f represents the focal length, (c_x, c_y) represent the coordinates of the main point of the image plane and $\lambda = Z$ represents the homogeneous scale factor. The calculations performed in the perspective projection that take points from the $3D$ space to the $2D$ space are necessary to carry out the reverse process.

In this step, we have obtained the position of the object of interest in the $2D$ space of the screen coordinate system in pixels [28]. It is necessary to convert the position of the object from $2D$ space to the position of the object in $3D$ space in the global coordinate system in meters [29]. This conversion is not performed directly, requiring a few steps to reach the desired result. The initial step is to convert the screen coordinate system to the camera coordinate system. This operation takes the position of the object from $2D$ space to $3D$ space using equations 11, 12 and 13, which were derived from

$$x_c = \frac{x_t - c_x}{f} \cdot z, \tag{11}$$

$$y_c = \frac{y_t - c_y}{f} \cdot z, \tag{12}$$

$$z_c = z, \tag{13}$$

where x_c, y_c and z_c represent the position of the object in the coordinate system of the camera in space $3D$, x_t and

y_t represent the position of the object in the screen coordinate system, z represents the distance between the object and the camera on the Z axis.

The coordinates of the main point of the image plane (c_x, c_y) and the focal length f are the camera intrinsic parameters. The cameras attached to the robots were calibrated using the ROS monocular camera calibration procedure. After this conversion, the values are now represented in meters. The distance of the object in relation to the camera on the Z axis, which we call z, is necessary for the conversion of 2D to 3D space. After detecting the object of interest, the object's previous information and camera specifications are used to calculate the value of this variable.

The Thin Lens Model simplifies the problem by considering as if it were a simple one glass element with only the one central node point. Usually, multi-element camera lenses have two nodes for each side of lens, while camera lenses normally have several glass lens elements designed to correct optical aberrations and distortions. The Thin Lens Model does work well for practical computing purposes and it is adopted here. Thus, the focal length is measured from the sensor plane to the lens node (often inside the lens, but not always). Lens equations uses the distance d in front of the field node However, here d is the distance between the lens and the object. In addition, the standard camera magnification geometry uses the standard ratios of similar triangles, i.e., the field dimension angle in front of this lens node is the same angle (opposite angles) as the sensor dimension angle behind the lens. The ratio of distances on each side of the lens are the same as the ratio of the size dimensions on each side of the lens. These ratios are simply the trigonometry tangents of the same angle on each side of the lens.

Thus, from the optics theory we have the following relationship

$$\frac{I_{dim}}{f} = \frac{F_{dim}}{z}, \tag{14}$$

where f represents the focal distance, z represents the distance of the object in relation to the camera on the Z axis, I_{dim} is the image dimension, and F_{dim} is the field dimension.

With no loss of generality we can state here that the dimension we are calculating is the height, and that the field we are observing is the object of interes. Thus, the same equation can be written as

$$\frac{O_{H_S}}{f} = \frac{l_r}{z}, \tag{15}$$

where O_{H_S} is Object height on sensor, and l_r represents the real height of the object of interest.

In addition, through the linear magnification of a thin lens we have that

$$O_{H_S} = \frac{l_s \cdot l_o}{l_i}, \tag{16}$$

where l_s represents the size of the height of the camera sensor responsible for capturing the image, l_i represents the height of the image, and l_o represents the observed height of the object in the image.

Thus, we can obtain the distance by using equations 15 and 16 as follows

$$z = \frac{f \cdot l_r \cdot l_i}{l_o \cdot l_s}, \tag{17}$$

where the values of the variables z, f, l_r and l_s are represented in millimeters and the values of the variables l_i and l_o are represented in pixels. The value of the variables f, l_i and l_s of the Kinect version 1.0 are, respectively, 3.099 mm, 480 pixels and 2.87 mm [30]. The real radius of the object is 0.092356 meters. Using the selected contour's radius r_{enc}, the value of the variable z, in meters, is calculated as

$$z = \frac{0.003099 \cdot 0.092356 \cdot 480}{r_{enc} \cdot 0.00287} = \frac{48.6916}{r_{enc}} \tag{18}$$

After obtaining the value of the variable z, this value is used to calculate the value of the confidence variable over the distance (α_2). The α_2 is a value assigned as an exponential gain that decreases with distance. This means that the greater the object's distance from the camera, the smaller its size within the image, and, consequently, the greater the difficulty in detecting and classifying it. Thus, this confidence variable about distance α_2 is employed as

$$\alpha_2 = \frac{1}{1.5^{z_{min}-z}}, \tag{19}$$

where z represents the distance of the object in relation to the camera's Z axis and z_{min} represents the minimum distance at which the object of interest can reach from the camera in relation to the Z axis whose value used is 0.8 meters.

After obtaining the values of the confidence variables on detection α_1 and distance α_2, the value of the confidence variable on observation α_f is computed, as shown in equation 20. The confidence on observation α_f is used to assign to each detection of the object of interest a value that seeks to determine the quality of this detection.

$$\alpha_f = \alpha_1 \cdot \alpha_2. \tag{20}$$

When leaving the screen coordinate system for the camera coordinate system, it is necessary to apply a rotation to reconcile the screen coordinate system with the camera coordinate system. As shown in Fig. 4, the directions of the x and y axis are different. For this reason, a 180° clockwise rotation around the z axis is applied, as shown in equation 21, to fix this problem.

$$\begin{bmatrix} x'_c \\ y'_c \\ z'_c \\ 1 \end{bmatrix} = \begin{bmatrix} \cos 180^\circ & -\text{sen } 180^\circ & 0 & 0 \\ \text{sen } 180^\circ & \cos 180^\circ & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}. \tag{21}$$

The next step is to move the position of the object from the coordinate system of the camera on the robot to the coordinate system of the base of the robot. This operation is necessary because, when we talk about cameras fixed in robots, the positioning of the camera in relation to the robot

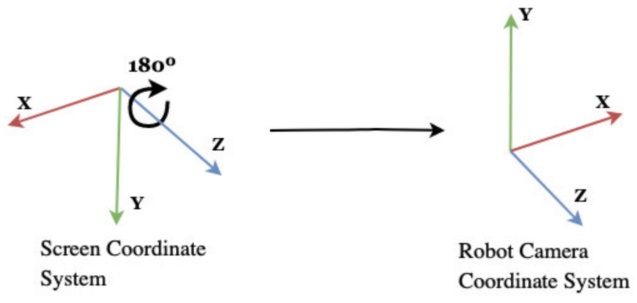


FIGURE 4. Performed rotations to transform from the screen coordinate system to the robot camera coordinate system.

can vary. An example is the Turtlebot robot (see Fig. 5) whose camera can change position according to the user's needs. The positioning of the cameras in relation to the base of the Turtlebot is identical for all robots used, therefore the same calculation for changing the coordinate system is applied for all robots.

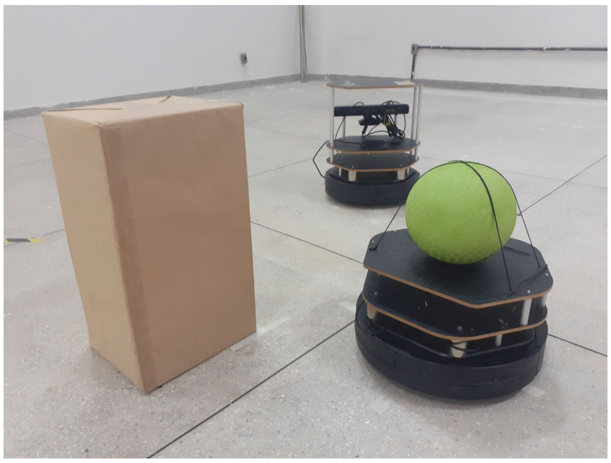


FIGURE 5. The object of interest fixed on top of a Turtlebot.

Fig. 6 shows the directions of each axis of the camera and robot coordinate system. The first rotation is a 90° clockwise rotation around the y-axis and the second rotation is a 90° clockwise rotation around the x-axis. This operation is described in the equation below.

$$\begin{bmatrix} x_r \\ y_r \\ z_r \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos 90^\circ & -\text{sen } 90^\circ & 0 \\ 0 & \text{sen } 90^\circ & \cos 90^\circ & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos 90^\circ & 0 & \text{sen } 90^\circ & 0 \\ 0 & 1 & 0 & 0 \\ -\text{sen } 90^\circ & 0 & \cos 90^\circ & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x'_c \\ y'_c \\ z'_c \\ 1 \end{bmatrix}, \quad (22)$$

As the origin of the coordinate systems of the camera and the robot is different, after the rotations, a translation is applied to take from one origin to the other, the translation is described by equation 23. The values of the

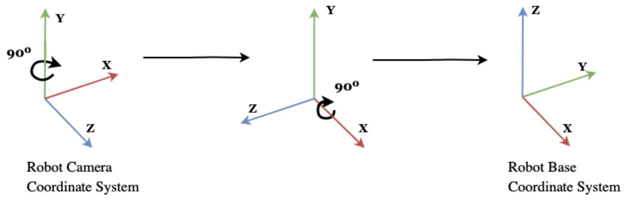


FIGURE 6. Performed rotations to transform from the robot camera coordinate system to the robot base coordinate system.

variables t_{cx}, t_{cy} and t_{cz} were obtained manually and are, respectively, -0.125 meters, 0.0 meters and 0.25 meters.

$$\begin{bmatrix} x'_r \\ y'_r \\ z'_r \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_{cx} \\ 0 & 1 & 0 & t_{cy} \\ 0 & 0 & 1 & t_{cz} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ z_r \\ 1 \end{bmatrix}. \quad (23)$$

Finally, the position of the object in the local coordinate system of the robot is converted to the global coordinate system. This operation aims to ensure that all robots involved in tracking the object of interest share the same coordinate system. For this purpose, the robot pose is used. This operation is described by

$$\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\text{sen }\theta & 0 & t_{rx} \\ \text{sen }\theta & \cos\theta & 0 & t_{ry} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x'_r \\ y'_r \\ z'_r \\ 1 \end{bmatrix}, \quad (24)$$

where t_{rx} and t_{ry} represent, respectively, the robot's x and y position in the global coordinate system and θ represents the robot's orientation angle.

The information collected after the end of the estimating stage, the object of interest's position in $3D$ space is passed on as input to a particle filter. This process aims to improve the estimation of the object's position through the use of a probabilistic model.

As a next step, weights are corrected through observations and it is at this stage that the non-cooperative and cooperative approaches are distinguished from each other. The crucial difference is the set of observations that are used to correct the weights. While in the non-cooperative approach only the observation generated by the robot itself is used to perform the correction, in the cooperative approach the observations of all members of a team of robots are used by merging the data obtained by the observations.

In the cooperative approach, for the data to merge, it is necessary that there is a communication mechanism that allows the exchange of information between members of the tracking team. This communication was implemented using the synchronous communication mechanism known as ROS services. After the end of the detection and estimation of the position of the object, the robot makes a request containing the x_w, y_w and z_w position information of the object of interest, the confidence on observation α and the identifier k. The code snippet responsible for executing the service stores the

Algorithm 2 Weight Adjustment Algorithm

```
1: Input: $(\alpha_k, \beta_k, X_t, M, N, k)$
2: {Service request - Exclusive step of the cooperative approach}
3: Send the $i^{th}$ robot's $\alpha_k$, $\beta_k$ and $k$
4: {Service response - Exclusive step of the cooperative approach}
5: Receives $\alpha$, $\beta$ from the robots within the team
6: {Trust normalization step}
7: for $r = 1$ to $N$ do
8:     $\alpha_r = \frac{\alpha_r}{\sum_{n=1}^{N} \alpha_n}$
9: end for
10: {Correction of particle weights from observations}
11: for $m = 1$ to $M$ do
12:     for $r = 1$ to $N$ do
13:         Obtain the $p_r$ probability of $x_t^{[m]}$ from the PDF $M_r$ generated by the observation $\beta_r$
14:     end for
15:     $w_t^{[m]} = w_t^{[m]} \cdot \sum_{n=1}^{N} p_n \alpha_n$
16: end for
17: {Normalization of weights associated with particles}
18: for $m = 1$ to $M$ do
19:     $w_t^{[m]} = \frac{w_t^{[m]}}{\sum_{n=1}^{M} w_t^{[n]}}$
20: end for
21: return $X_t$
```

data generated by the robot and returns the same information as that of the other robots involved in tracking as a response.

Even when the robot is unable to obtain information about the object of interest from the data collected by its sensors, the robot still performs the request but passes the confidence value on the observation α equal to zero, making sure that its data does not affect estimating the position of the object of interest during data fusion. it performs this action in order to find out if any other member of the team has any information regarding the object at that time.

One of the advantages of using ROS services is the simplicity of use and the speed at which information is exchanged. In contrast, the information that is recorded within the service needs to be updated frequently so that it is as close to the current state of the environment. After initialization, the weight of each particle is corrected based on the observations of each of the robots. Unlike the original particle filter design, more than one observation is used during the weight correction process where each observation is responsible for a part of this correction. This part is determined by the degree of confidence in the observation.

As a first step in correcting the weights, the degree of confidence in the α observation is normalized so that each one of them becomes the part of the correction that the β observation is responsible for. The correction algorithm is detailed in Algorithm 2. For each β_k observation a probability density function M_k is generated with normal distribution of average μ equal to the position estimated by observation β_k and covariance of distribution σ as described in equation 25. The correction of the weight of a particle $x_t^{[m]}$ according to the observations and obtained by the sum of the corrections generated by each observation β where the correction generated by the observation β_k is calculated as the probability of $x_t^{[m]}$ obtained by the multiplied probability density function M_k by the degree of confidence in the α_k observation.

$$\sigma = \begin{bmatrix} 1e^{-3} & 0 & 0 \\ 0 & 1e^{-3} & 0 \\ 0 & 0 & 1e^{-3} \end{bmatrix}. \tag{25}$$

The weight correction is applied to the prediction by multiplying the probability obtained through the observations by the weight of each particle. Finally, weights are normalized. After normalization, low variation re-sampling, or low variance re-sampling, is performed [7]. Right after re-sampling, the most likely position of the object of interest is estimated by the weighted average of the particles by the weights as shown in equation 26.

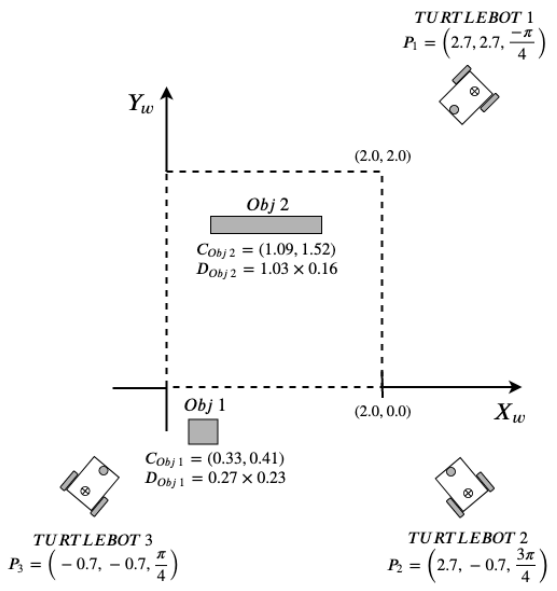
$$\mu_t = \frac{1}{N} \sum_{i=1}^{N} x_t^{[i]} w_t^{[i]}. \tag{26}$$

As a final step in the particle filter, a prediction is made about the position of the object of interest in the next instant of time. In this work, no motion model was implemented that could be used to estimate the next position of the object of interest through the instantiations represented by the particles. Within the prediction stage, we seek to predict the next position of the object by randomly moving the current position stored in each particle. For each coordinate, its value is changed at random by adding a value obtained from a sample of a uniform distribution with a half-open interval of $[-0.5, 0.5)$ meters. All previous steps, with the exception of initializing the weights, are repeated until the visual tracking task is finished.

IV. EXPERIMENTS AND RESULTS

To test the two visual tracking approaches, an experiment was carried out in which an environment was created for the execution of a task of tracking a spherical object by a group of robots. The setup of the environment can be seen in Fig. 7 and Fig. 8 where the dashed line represents the path that was planned to be made by the spherical object. The position of the objects present in the environment and the poses of the robots involved in the visual tracking were determined based on the following criteria:

- The objects were positioned within the environment in order to generate partial and total occlusion on the observations of the robots at certain points in the trajectory of the spherical object since the data fusion seeks to bypass this problem.
- The relationship between the positions of the objects and the poses of the robots was idealized so that none of the robots could obtain the complete trajectory of



$TURTLEBOT\ 1$
$P_1 = \left(2.7, 2.7, \frac{-\pi}{4}\right)$
Y_w
(2.0, 2.0)
$Obj\ 2$
$C_{Obj\,2} = (1.09, 1.52)$
$D_{Obj\,2} = 1.03 \times 0.16$
$Obj\ 1$
(2.0, 0.0)
X_w
$C_{Obj\,1} = (0.33, 0.41)$
$D_{Obj\,1} = 0.27 \times 0.23$
$TURTLEBOT\ 3$
$P_3 = \left(-0.7, -0.7, \frac{\pi}{4}\right)$
$TURTLEBOT\ 2$
$P_2 = \left(2.7, -0.7, \frac{3\pi}{4}\right)$

FIGURE 7. Setup of the environment designed to carry out the experiment. The dashed line represents the trajectory that was planned to be made by the object of interest. The variable $C_{Obj\,i}$ represents the position of the centroid of object *i* in the global coordinate system in meters, the variable $D_{Obj\,i}$ represents both the width and length of object i in meters, and $P_i = (x_i, y_i, \theta_i)$ represents the position x_i and y_i in the global coordinate system in meters and the orientation angle θi in radians of robot *i*.

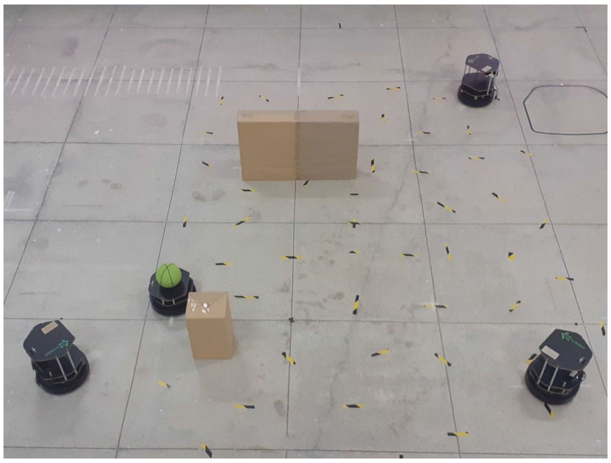


FIGURE 8. Experiment setup.

the spherical object, but the fusion of the data would generate this trajectory.

- The minimum-maximum distance, which can be inferred by visual tracking during the trajectory of the spherical object, between the position of the object and any of the robots is, respectively, 1.0 and 4.0 meters.

In order to avoid errors arising from the estimation of the robot pose, considering that the robot pose is used to estimate the position of the object in 3D space, each robot was positioned in the environment manually and its pose was passed to the tracking algorithm directly, without using an inference algorithm. None of the tracking robots move during the experiment. The spherical object was fixed on top of a robot that did not belong to the tracking team so that it was possible to carry out the previously planned trajectory. The robot was given the command to perform the trajectory that would form a square with 2 meters on its side from the origin of the global coordinate system directing towards the X+ axis rotating counterclockwise.

The odometry data of the robot that was responsible for displacing the object were obtained to compare the trajectory of the spherical object inferred by the visual tracking with the trajectory actually performed by the object. The trajectory obtained through odometry was considered as the ground truth of the experiment and it is used to validate the trajectories inferred by each robot.

To generate the results, every 30 milliseconds, counting from the beginning of the experiment, the positioning estimation error of the spherical object in the space of the world was calculated as the Euclidean distance, in meters, between the position of the spherical object inferred by tracking and the position obtained by odometry. The error of estimating the position in the world space of each robot during the experiment is shown in Fig. 9. When a robot cannot infer the position of the object, the last position of the object of interest is given as undefined, the error value is not computed or displayed in the chart.

The results of the non-cooperative approach shown in Fig. 9 have peaks of error that arise due to partial occlusion. The partial occlusion affects the information of the spherical object that is obtained through the detection steps that ends up influencing the calculation of the distance of the object in relation to the camera around the Z-axis, which ultimately influences the estimation of the position of the object in 3D space in the global coordinates.

These peaks stop happening in the cooperative approach. This is due to the fact that when an observation suffers from partial occlusion, the value of the confidence variable on the observation α decreases, making the influence of that observation on the estimation of the position of the object of interest during the data fusion be less influential than the influence of the other robots with a better observation of the object. This, in turn, leads to the attenuation of the error.

The main difference between the non-cooperative and the cooperative approach is the fact that, despite the partial and total occlusions that each robot suffers, in the cooperative approach all robots are able to keep track of the spherical object during the whole experiment. In contrast, in the non-cooperative approach, there are time intervals in which one or more robots are not able to estimate the spherical object's position.

One of the negative points of the cooperative approach was the increase in the computational cost for performing the tracking. Considering an execution as the set of operations that are performed on the inputs to infer the position of the object of interest, the algorithm responsible for the non-cooperative approach had, on average, 24.07 runs per second. In contrast, the algorithm responsible for the

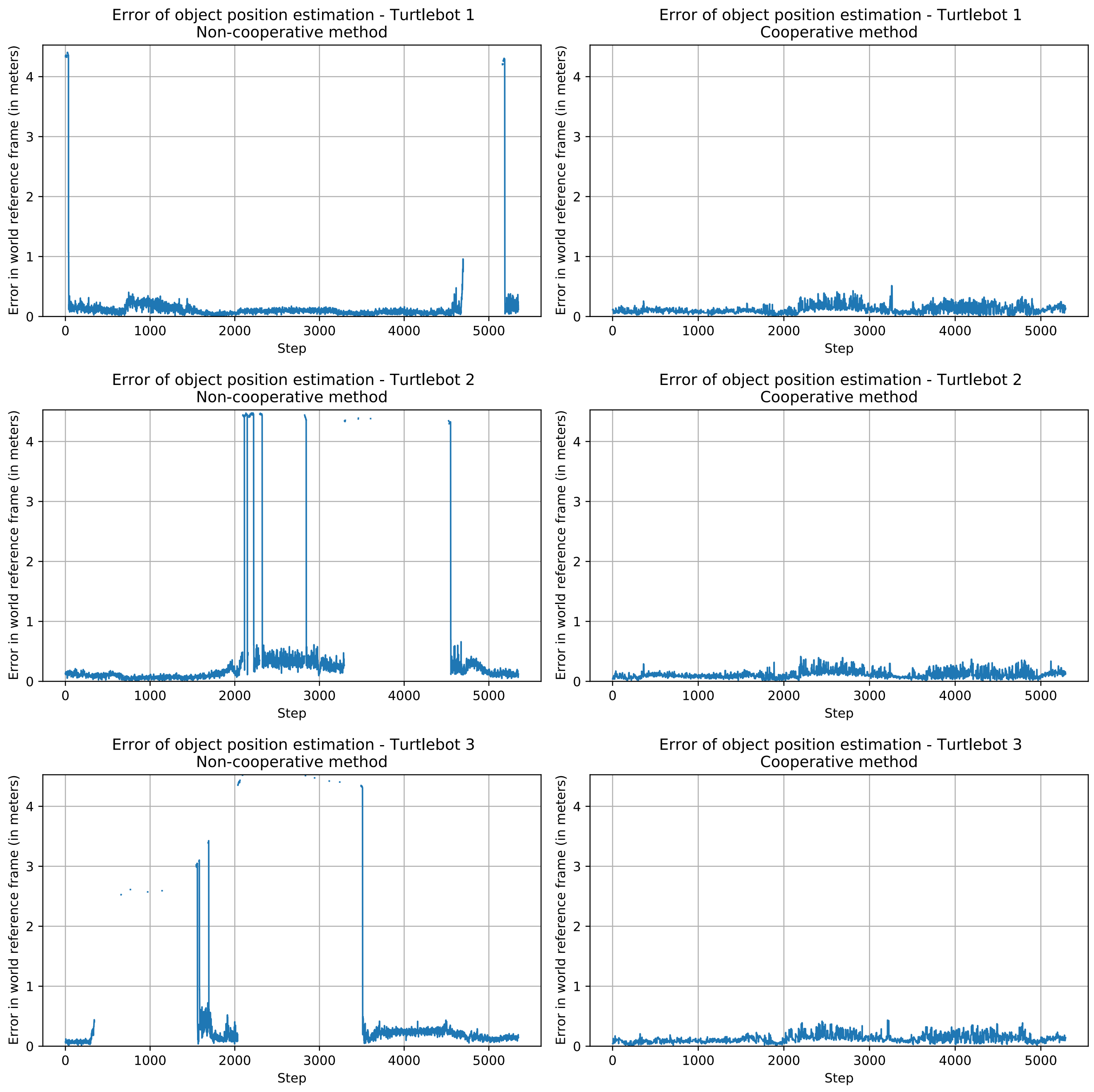


FIGURE 9. **Every 30 milliseconds, counting from the beginning of the experiment, the error of estimating the position of the spherical object in world frame was calculated as the Euclidean distance, in meters, between the position of the spherical object inferred by tracking and the position of the spherical object obtained by odometry. The graphics in the left column represent the error in the space of the tracking world of the spherical object in the non-cooperative approach, while the graphics in the right column represent the error in tracking the spherical object in the cooperative approach in the world space. When a robot is unable to infer the position of the object, the last position of the object of interest is given as undefined, the error value is neither computed nor displayed on the graph.**

cooperative approach had, on average, 8.67 runs per second, as shown in Table 1. This decrease is justified by the increase in computational cost caused by operations related to information exchange between members of the tracking team.

Despite having an average value of executions per second greater than that of the cooperative approach, the non-cooperative approach obtained a considerable percentage of executions in which the position of the spherical object cannot be inferred, as shown in Table 2.

In the cooperative approach, the position of the object was not lost because during the whole experiment, at least one of the robots had an observation of the spherical object without any type of occlusion and this information was passed on to the others. Another positive aspect of the cooperative

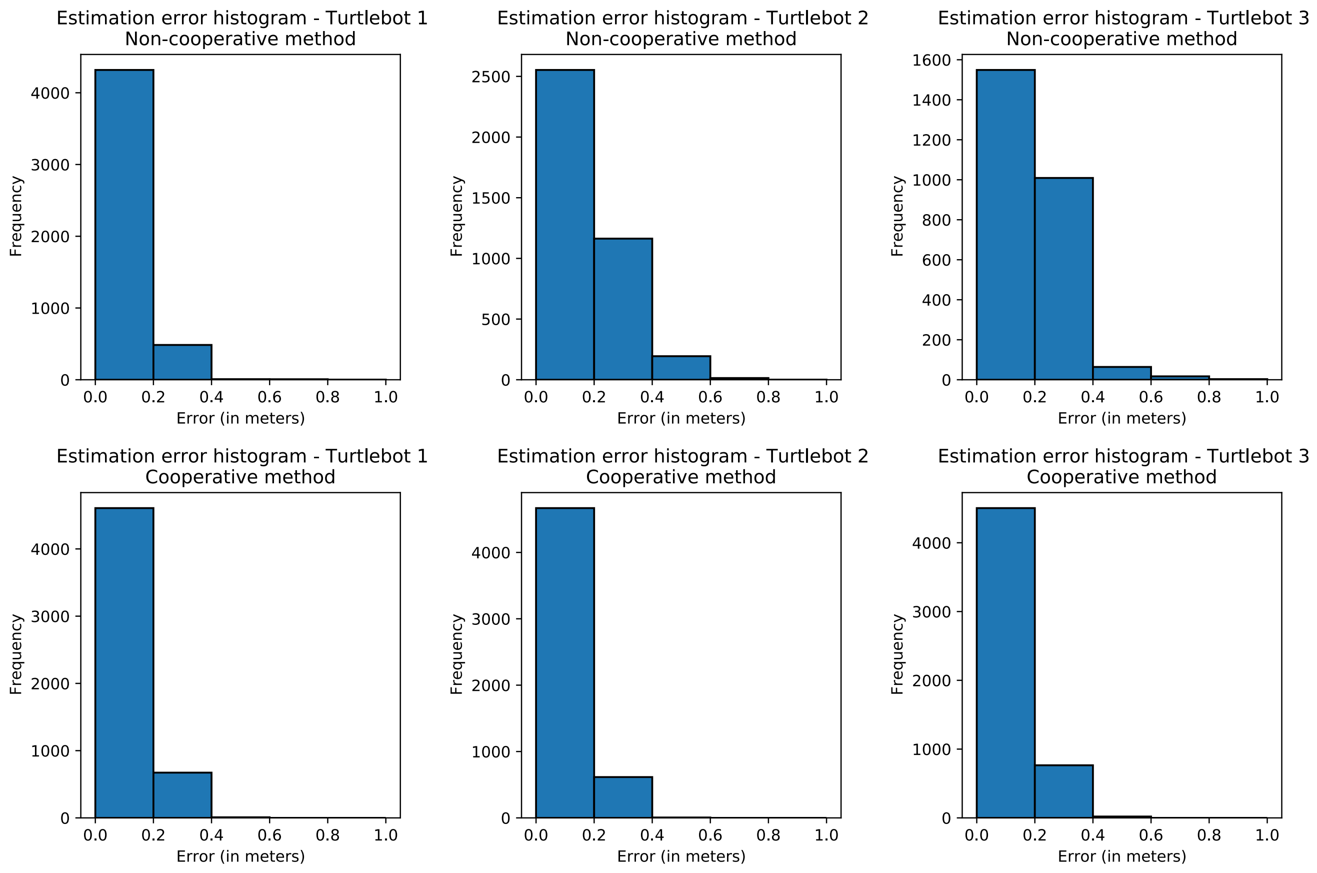


FIGURE 10. The graphs at the top represent the histogram of the estimation error of the position of the spherical object in the space of the robot world during the non-cooperative approach. The graphs at the bottom represent the histogram of the estimation error of the position of the spherical object in the space of the robot world during the cooperative approach.

TABLE 1. The average number of executions per second of the tracking algorithm for each approach by each robot, considering an execution as the set of operations that are performed on the inputs to infer the position of the object of interest.

	Non-cooperative	Cooperative
Robot 1	23.86	9.41
Robot 2	23.72	9.36
Robot 3	24.62	7.23
Average	24.07	8.67

TABLE 2. Percentage of runs in which the position of the spherical object cannot be inferred.

	Non-cooperative	Cooperative
Robot 1	11.02%	0.0%
Robot 2	27.37%	0.0%
Robot 3	54.74%	0.0%
Average	31.04%	0.0%

approach was the decrease in the frequency of errors greater than 0.4 meters in comparison with the non-cooperative approach, showing that the fusion of the data generates a correction in the estimation of the object of interest's position. This can best be seen in Fig. 10 where the graphs at the top represent the histogram of the estimation error in the robot world space during the non-cooperative approach and the graphs at the bottom represent the histogram of the estimation error in the robot world space during the cooperative approach.

V. CONCLUSION

In this manuscript, we presented a modified particle filter applied to a constrained target tracking multi-robot system. The robots were not able to move and the camera used had a limited field of view. Nevertheless, we proposed an approach that was able to overcome occlusion, regardless of the existence of a map of the environment. The particle filter was able to track a spherical object in 3D space, minimizing the error of observation through the cooperative version of the modified particle filter. As future works, we aim to expand the form of the detected object to a generic contour. Another future work is to use our approach in multi-rotor unmanned aerial vehicles (multi-rotor UAVs) for human detection or multiple target detection, and thus consider input noise such as camera vibration and illumination variation.

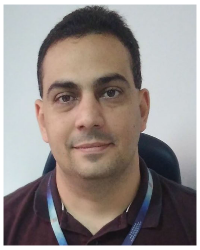
REFERENCES

[1] Y. Wu, J. Lim, and M. H. Yang, "Object tracking benchmark," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1834–1848, Sep. 2015.

[2] J. Ghommam, M. S. Mahmoud, and M. Saad, "Robust cooperative control for a group of mobile robots with quantized information exchange," *J. Franklin Inst.*, vol. 350, no. 8, pp. 2291–2321, Oct. 2013.
[3] M. M. Alyannezhadi, A. A. Pouyan, and V. Abolghasemi, "An efficient algorithm for multisensory data fusion under uncertainty condition," *J. Electr. Syst. Inf. Technol.*, vol. 4, no. 1, pp. 269–278, May 2017.
[4] B. Khaleghi, A. Khamis, F. O. Karray, and S. N. Razavi, "Multisensor data fusion: A review of the state-of-the-art," *Inf. Fusion*, vol. 14, no. 1, pp. 28–44, Jan. 2013.
[5] M. Hu, Z. Liu, J. Zhang, and G. Zhang, "Robust object tracking via multi-cue fusion," *Signal Process.*, vol. 139, pp. 86–95, Oct. 2017.
[6] A. Ahmad and P. Lima, "Multi-robot cooperative spherical-object tracking in 3D space based on particle filters," *Robot. Auto. Syst.*, vol. 61, no. 10, pp. 1084–1093, Oct. 2013.
[7] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA, USA: MIT Press, 2005.
[8] K. Zhou and S. I. Roumeliotis, "Multirobot active target tracking with combinations of relative observations," *IEEE Trans. Robot.*, vol. 27, no. 4, pp. 678–695, Aug. 2011.
[9] J. Munjani and M. Joshi, "A non-conventional lightweight auto regressive neural network for accurate and energy efficient target tracking in wireless sensor network," *ISA Trans.*, pp. 1–20, Jan. 2021.
[10] J. Dou, Q. Qin, and Z. Tu, "Robust visual tracking based on generative and discriminative model collaboration," *Multimedia Tools Appl.*, vol. 76, no. 14, pp. 15839–15866, Jul. 2017.
[11] M. Wan, G. Gu, W. Qian, K. Ren, X. Maldague, and Q. Chen, "Unmanned aerial vehicle video-based target tracking algorithm using sparse representation," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 9689–9706, Dec. 2019.
[12] A. Elnakeeb and U. Mitra, "Line constrained estimation with applications to target tracking: Exploiting sparsity and low-rank," *IEEE Trans. Signal Process.*, vol. 66, no. 24, pp. 6488–6502, Dec. 2018.
[13] J. Hu and X. Fan, "Robust visual tracking via a collaborative model based on locality-constrained sparse coding," *IEEE Access*, vol. 8, pp. 76737–76751, 2020.
[14] I. Leang, S. Herbin, B. Girard, and J. Droulez, "On-line fusion of trackers for single-object tracking," *Pattern Recognit.*, vol. 74, pp. 459–473, Feb. 2018.
[15] F. Li and S. Liu, "Object tracking via a cooperative appearance model," *Knowl.-Based Syst.*, vol. 129, pp. 61–78, Aug. 2017.
[16] H. Zhang, L. Li, and W. Xie, "Constrained multiple model particle filtering for bearings-only maneuvering target tracking," *IEEE Access*, vol. 6, pp. 51721–51734, 2018.
[17] J. Lee, R. Sandhu, and A. Tannenbaum, "Particle filters and occlusion handling for rigid 2D–3D pose tracking," *Comput. Vis. Image Understand.*, vol. 117, no. 8, pp. 922–933, Aug. 2013.
[18] S. Yi, Z. He, X. You, and Y.-M. Cheung, "Single object tracking via robust combination of particle filter and sparse representation," *Signal Process.*, vol. 110, pp. 178–187, May 2015.
[19] M. T. N. Truong, M. Pak, and S. Kim, "Single object tracking using particle filter framework and saliency-based weighted color histogram," *Multimedia Tools Appl.*, vol. 77, no. 22, pp. 30067–30088, Nov. 2018.
[20] C. Xu, X. Wang, S. Duan, and J. Wan, "Spatial-temporal constrained particle filter for cooperative target tracking," *J. Netw. Comput. Appl.*, vol. 176, Feb. 2021, Art. no. 102913.
[21] L. Zhou and P. Tokekar, "Active target tracking with self-triggered communications in multi-robot teams," *IEEE Trans. Autom. Sci. Eng.*, vol. 16, no. 3, pp. 1085–1096, Jul. 2019.
[22] L. Zhou, V. Tzoumas, G. J. Pappas, and P. Tokekar, "Resilient active target tracking with multiple robots," *IEEE Robot. Autom. Lett.*, vol. 4, no. 1, pp. 129–136, Jan. 2019.
[23] L. Yang and F. Albregtsen, "Fast and exact computation of Cartesian geometric moments using discrete Green's theorem," *Pattern Recognit.*, vol. 29, no. 7, pp. 1061–1073, Jul. 1996.
[24] M. Jacob, T. Blu, and M. Unser, "An exact method for computing the area moments of wavelet and spline curves," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 6, pp. 633–642, Jun. 2001.
[25] X. Jiang, Z. Fang, N. N. Xiong, Y. Gao, B. Huang, J. Zhang, L. Yu, and P. Harrington, "Data fusion-based multi-object tracking for unconstrained visual sensor networks," *IEEE Access*, vol. 6, pp. 13716–13728, 2018.
[26] Y. Liu, J. Cheng, H. Zhang, H. Zou, and N. Xiong, "Long short-term memory networks based on particle filter for object tracking," *IEEE Access*, vol. 8, pp. 216245–216258, 2020.
[27] J.-K. You, C.-C.-J. Hsu, W.-Y. Wang, and S.-K. Huang, "Object pose estimation incorporating projection loss and discriminative refinement," *IEEE Access*, vol. 9, pp. 18597–18606, 2021.
[28] S. Chae, J. H. Hong, H. Choi, and I.-J. Kim, "POP: A generic framework for real-time pose estimation of planar objects," *IEEE Access*, vol. 8, pp. 164065–164076, 2020.
[29] C. Papaioannidis, V. Mygdalis, and I. Pitas, "Domain-translated 3D object pose estimation," *IEEE Trans. Image Process.*, vol. 29, pp. 9279–9291, 2020.
[30] D. Pagliari and L. Pinto, "Calibration of Kinect for xbox one and comparison between the two generations of microsoft sensors," *Sensors*, vol. 15, no. 11, pp. 27569–27589, Oct. 2015. [Online]. Available: http://www.mdpi.com/1424-8220/15/11/27569

THULIO G. S. AMORIM received the B.Sc. degree in computer science from the Universidade Federal da Paraiba, Brazil, in 2019, where he is currently pursuing the M.Sc. degree, under the supervision of Prof. Tiago Nascimento. He is also working as a Research Associate with the Laboratory of Systems Engineering and Robotics (LASER), Department of Computer Systems, Universidade Federal da Paraiba.

LEONARDO A. SOUTO received the degree in computer science from the University Center of João Pessoa, in 2011, the master's degree in computer science from the Graduate Program in Informatics, Federal University of Paraíba (UFPB), in 2016, and the Ph.D. degree from the Graduate Program in Electrical Engineering and Computing (PPGEEC), UFRN. He is currently a member of the Embedded Systems and Robotics Laboratory (LASER), UFPB, and the NatalNet Laboratory, UFRN. His research interests include mobile robotics, control systems, artificial intelligence and intelligent control, computer vision, and computer networks.



TIAGO P. DO NASCIMENTO (Member, IEEE) received the B.S. degree in mechatronics engineering from the College of Technology and Science - FTC, Brazil, in 2007, the M.S. degree in electrical engineering from the Federal University of Bahia, Brazil, in 2009, and the Ph.D. degree in electrical and computer engineering from Oporto University, Portugal, in 2012. He joined the Department of Computer Systems, Universidade Federal da Paraiba, João Pessoa, Brazil, as an Assistant Professor, in 2013. Since 2019, he has been a Senior Researcher with Czech Technical University in Prague. He is currently an Assistant Professor with the Universidade Federal da Paraiba, João Pessoa, Brazil. His main research interests include mobile robotics, multi-robot systems, process control, vehicle dynamics, and intelligent control. He has been a member of the IEEE RAS, since 2004. He is also the IEEE Bahia Section RAS/CSS Chair.

MARTIN SASKA (Member, IEEE) received the M.Sc. degree from Czech Technical University in Prague, in 2005, and the Ph.D. degree from the University of Wuerzburg, Germany. He was a Visiting Scholar with the University of Illinois at UrbanaChampaign, Champaign, IL, USA, in 2008, and the University of Pennsylvania, Philadelphia, PA, USA, in 2012, 2014, and 2016. Since 2009, he has been a Research Fellow with Czech Technical University in Prague, where he has founded and since leads the Multi-Robot Systems Laboratory and co-founded the Center for Robotics and Autonomous Systems. He is the author or a coauthor of more than 70 publications in peer-reviewed conferences and more than 20 publications in impacted journals.