Semantic situation awareness of ellipse shapes via deep learning for multirotor aerial robots with a 2D LIDAR

Jose Luis Sanchez-Lopez[1] and Manuel Castillo-Lopez[1] and Holger Voos[1]

***Abstract*— In this work, we present a semantic situation awareness system for multirotor aerial robots equipped with a 2D LIDAR sensor, focusing on the understanding of the environment, provided to have a drift-free precise localization of the robot (e.g. given by GNSS/INS or motion capture system). Our algorithm generates in real-time a semantic map of the objects of the environment as a list of ellipses represented by their radii, and their pose and velocity, both in world coordinates. Two different Convolutional Neural Network (CNN) architectures are proposed and trained using an artificially generated dataset and a custom loss function, to detect ellipses in a segmented (i.e. with one single object) LIDAR measurement. In cascade, a specifically designed indirect-EKF estimates the ellipses based semantic map in world coordinates, as well as their velocity. We have quantitative and qualitatively evaluated the performance of our proposed situation awareness system. Two sets of Software-In-The-Loop simulations using CoppeliaSim with one and multiple static and moving cylindrical objects are used to evaluate the accuracy and performance of our algorithm. In addition, we have demonstrated the robustness of our proposed algorithm when handling real environments thanks to real laboratory experiments with non-cylindrical static (i.e. a barrel) objects and moving persons.**

I. INTRODUCTION

Acquiring a complete situational awareness of the surroundings is an essential capability to facilitate the reasoning, needed by any mobile robotic system, concretely multirotor aerial robots equipped with a 2D LIDAR sensor (see Fig. 1). Some works, [1], [2], elude this complex task by feeding their planning and control algorithms with the raw measurements of the LIDAR, generating a reactive behavior, which is prone to fall on inefficient or blocking situations. On the other hand, deliberative approaches, [3], [4], [5], [6], [7], present an optimum behavior, with the requirement of the knowledge of the environment.

The geometric representation of the environment can be done by using point clouds, [3]; grid maps, [4], or geometric shapes, [5], [6], [7]. While the first two approaches are mainly used for static environments with a limited number of moving objects, the later is well suited to handle dynamic complex environments. Moreover, to facilitate the reasoning, the situational information of the surroundings must include, not only the geometrical features of the environment (i.e. overall location of the objects) but also other useful details such as the semantic information of every object (e.g. type or shape) or their dynamic features (e.g. velocity).

Fig. 1. Lab experiment with one static object (i.e. blue barrel) and two moving persons.

In this paper, we focus on building a semantic map of the environment with dynamic information, capable of handling dynamic complex environments, for multirotor aerial robots equipped with a 2D LIDAR sensor, as well as, a localization component providing an accurate drift-free estimation of the pose of the robot with respect to the world reference frame (e.g. GNSS/INS or motion capture system). We limit the semantic representation of the objects to their geometric shapes, concretely to elliptical shapes. In our approach, we combine classic filtering-based mapping techniques adapted to handle such semantic representation of the environment, with cutting edge deep neural network techniques to detect geometric shapes on 2D LIDAR data.

A. Problem formulation and hypothesis

In this work, we assume that the environment is composed only by objects with an approximate ellipse-base cylindrical shape, moving in the horizontal plane. The movement of the aerial robot is constrained to the horizontal plane, without any change in its flight altitude, which is a requirement in multiple applications (e.g. inspection of crop trees for precision agriculture).

The goal of this work is to compute in real-time a semantic map of the objects of the environment by using the measurements given by the LIDAR sensor, as well as, the localization of the aerial robot. The semantic map is described as a list of ellipses with a unique identifier, represented by their radii, their pose (position and orientation) and their velocity (linear and angular), both in world coordinates.

B. Contributions and outline

This paper presents our latest advances in semantic situation awareness using deep learning techniques for multirotor

This work was supported by the "Fonds National de la Recherche" (FNR), Luxembourg, under the project C15/15/10484117 (BEST-RPAS).

[1] Automation and Robotics Research Group, Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg. Address: 29, avenue J. F. Kennedy, L-1855 Luxembourg (Luxembourg). e-mail: joseluis.sanchezlopez@uni.lu

aerial robots equipped with a 2D LIDAR sensor. In our previous work, [8], we built a semantic map of the environment as a list of circles, while in this work, we use a more complex conic section, the ellipse, to represent in a more rich and accurate way, the objects existing in the environment.

The first contribution is a Convolutional Neural Network (CNN) based ellipse detector, which takes as input, a segmented (i.e. with one single object) LIDAR measurement, generating as output, the parameters that describe the detected ellipse, in sensor coordinates. Two different CNN architectures, inspired from the image recognition field, have been designed. We have trained them using an artificially generated dataset and a custom loss function.

As the second contribution, using the detected ellipses, we propose an indirect-EKF (that uses quaternions to represent the orientation) to estimate the semantic map of the environment as a list of ellipses in world coordinates, including an estimation of their velocities (both linear and angular).

The remainder of the paper is organized as follows: In Sect. II we review the related works. Sect. III introduces some essential concepts used in our work, presenting afterward, the overall proposed environment situation awareness architecture. In Sect. IV, we describe our machine learning based ellipse detector, while in Sect. V we detail our indirect-EKF based semantic ellipse mapping. A thoroughful evaluation of our proposed semantic situation awareness system is carried out in Sect. VI. Finally, Sect. VII concludes the paper.

II. RELATED WORK

A. Semantic situation awareness using LIDAR point clouds

Very little works can be found related to semantic situation awareness using LIDAR point clouds or multirotor aerial robots equipped with a 2D LIDAR, being this field monopolized by ground robots and autonomous vehicles. [9] presents a good attempt to build a 2D map that consists of three kinds of geometric primitives (i.e. lines, circles, and ellipses), using 2D LIDAR data. The measured point cloud, converted to Cartesian coordinates, is processed using classic techniques (i.e. not machine learning), to obtain and then map the geometric shapes. It has a limited performance when computing circles and ellipses, being unable to handle dynamic environments. [10] uses exclusively segments as geometric shapes computed using classic techniques, given a 2D LIDAR point cloud in Cartesian coordinates, focusing on handling moving obstacles. [11] presents a solution for simultaneous localization and mapping (SLAM) combining detection and tracking of moving objects using 2D LIDAR point clouds in Cartesian coordinates. The moving objects are simply segmented from the point cloud without providing any semantic or geometric information. People are detected and tracked with the help of a Convolutional Neural Network (CNN) trained with 2D point clouds in Cartesian coordinates in [12]. [13] uses a 3D LIDAR to carry out a SLAM, building a 3D map that consists of static geometric shapes using 2D geometric features (i.e. lines, rectangles, and circles), being unable to track moving objects. When provided a whole 3D point cloud map, [14] provides semantic labeling of these points by using a CNN. Regarding vehicle detection, [15] uses a single measurement of a 3D LIDAR as a spherical 2D plane image, to feed a CNN algorithm, whereas [16] requires the measurement to be a bids view in Cartesian coordinates. [17] provides by means of a CNN, not only the semantic information of the vehicles but also the estimate of their dynamic features by using two consecutive 3D point clouds as spherical 2D plane images.

B. Ellipse detection

The problem of ellipse detection (also known as ellipse fitting), is a deeply researched topic, that consists of computing the parameters of an ellipse, given a set of coplanar points. Ellipse fitting algorithms can be classified into three categories: clustering techniques [18], [19], least-squares algorithms, [20], [21], and machine learning based methods, [22]. While the first two classic methods heavily rely on geometric or algebraic properties of the ellipse, the later is based on learning using artificial neural networks. Machine learning techniques are more flexible than classic methods and can be adapted to other geometric shapes at the cost of having a labeled dataset, but more importantly, they are capable to implicitly (i.e. without any further design action) learn how the sensor acquires the measurements, improving, therefore, their performance. Since some work has been carried out on machine learning-based ellipse fitting for images, [22], there is a very limited work regarding 2D LIDAR measurements.

III. SEMANTIC SITUATION AWARENESS OF THE ENVIRONMENT

In this section, we first introduce some essential concepts used in our work, and then we describe the overall proposed environment situation awareness architecture.

A. LIDAR sensor measurement

One single measurement of a 2D LIDAR sensor, $\boldsymbol{z}$, is a point cloud in $\mathbb{R}^2$ represented in polar coordinates:

$$\boldsymbol{z}_L = \{[\alpha_i,\ \rho_i]\}, \quad \forall i$$

being ρ_i the distance between the measured point of the environment and the sensor, corresponding to the angular coordinate with respect to the sensor frame, α_i.

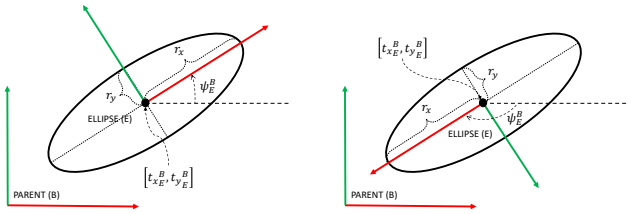
The dimension of the point cloud is characterized by the angular resolution, $\delta\alpha$, of the sensor, and its angular range, $[\alpha_{\min},\ \alpha_{\max}]$. In case of no physical point in the environment within the linear range of the sensor, $[\rho_{\min},\ \rho_{\max}]$, for a particular angular coordinate, α_i, then, the measured distance is set to an out-of-range value (i.e. $\rho_i = \text{NaN}$).

Our 2D LIDAR sensor, described in Sect. VI-B, provides, when taking advantage of its full angular range, a point cloud measurement with 1081 points.

B. Geometric shape: ellipse

Ellipses are 2D conic sections considered as the generalization of a circle. We define a reference frame rigidly attached to the ellipse, E, which origin is located in the center point of the ellipse, and its x-, and y-axis is aligned with the x-, and y-axis of the ellipse, respectively. An ellipse in a 2D environment is represented, with respect to an arbitrarily defined parent reference frame B, by the following five parameters: (1) the position of its center, $\boldsymbol{t}_E^B = \left[t_{x_E}^B,\ t_{y_E}^B\right]$; (2) the orientation of its x-axis, ψ_E^B; and (3) its two radii, $\boldsymbol{r} = [r_x,\ r_y]$.

The aforementioned definition of the ellipse is not unique, since there exist four different values of the previously defined set of parameters that represent the same ellipse shape. Enforcing r_x to be the largest axis of the ellipse, i.e. $r_x \geq r_y$, reduces to two, the number of cases, as can be visualized in Fig. 2. Unfortunately, if no additional information about the absolute orientation of the ellipse is available, we cannot uniquely determine the orientation of the ellipse. In the case that there is no additional information available about the orientation of the ellipse, we enforce its orientation to be in the range $\psi_E^B \in \left[-\frac{\pi}{2},\ \frac{\pi}{2}\right]$ (i.e. the case presented in Fig. 2a).



(a) Option 1. (b) Option 2.

Fig. 2. Ellipse representation in 2D using the five parameters: (1) the position of its center, $\boldsymbol{t}_E^B = \left[t_{x_E}^B,\ t_{y_E}^B\right]$; (2) the orientation of its x-axis, ψ_E^B; and (3) its two radii, $\boldsymbol{r} = [r_x,\ r_y]$.

C. ENVIRONMENT SITUATION AWARENESS ARCHITECTURE

Our proposed environment situation awareness is an adaptation of the one presented in our previous work [8], and it is formed by the five components shown in Fig. 3. It takes two inputs: (1) the measurements given by a 2D LIDAR sensor, and (2) the existing information of the pose of the aerial robot. It generates as output, a semantic map of the environment as a list of ellipses with their velocity estimates.

The first three components are extracted from our previous work [8]. They preprocess and segments the input raw LIDAR measurement, generating a filtered and horizontally projected set of point clouds in polar coordinates (similar to the one taken as input), with one single object per cluster. The last two components of the architecture are the two main contributions of this paper, and are presented in Sections IV and V, respectively.

IV. MACHINE LEARNING BASED ELLIPSE DETECTION

The ellipse detector presented in this section computes the parameters of one single ellipse using the previously segmented LIDAR measurement point cloud with 1081 points. Consequently, the output of this component are the parameters of the computed ellipse (see Sect. III-B), i.e. its radii, $\boldsymbol{r} = [r_x,\ r_y]$, and its pose (position and orientation) with respect to the sensor frame, $\boldsymbol{t}_E^S = \left[t_{x_E}^S,\ t_{y_E}^S\right]^T$ and ψ_E^B, respectively. These parameters are gathered as a vector of dimension 5. It is important to highlight that the radii can only take a positive real value (i.e. $\boldsymbol{r} \in \mathbb{R}^2_{\geq 0}$).

This component consists of a Convolutional Neural Network (CNN) described in Sect. IV-A. We define a custom loss function (see Sect. IV-B) that is used to train our artificial neural network (Sect. IV-D) using a synthetically generated dataset (see Sect. IV-C).

We gather in a vector of dimension 1081 all the distances, ρ_i, of the segmented point cloud measurement, and we preprocess it as follows before feeding it to our CNN: First of all, the out-of-range values (i.e. $z_i = \text{NaN}$) are replaced by a real number that is numerically treatable. We chose the number 0, since it is convolution neutral. Second, the input data is normalized in the range of $[0,\ 1]$, based on the normalization parameters calculated during the training stage.

A. CNN architectures

We propose two different CNN architectures, inspired from the image recognition field: Alex-Net [23], and VGG-Net [24]. Despite having been introduced some years ago, these two very well-known CNN architectures are still a very popular choice in multiple applications [25].

Unlike the original counterparts, our ellipse detector is used for regression, where it has to predict the values of the parameters of the ellipse. The output layer uses the linear activation function for both the position and the orientation of the ellipse, but a ReLu activation for the radii, since the latter can only take a positive real value.

1) Alex-Net: We propose the CNN architecture presented in Table I, inspired from Alex-Net [23]. Our architecture encompasses six 1-dimensional convolutional layers, three 1-dimensional max polling layers, and two fully connected layers, having a total of $1,814,105$ trainable parameters.

2) VGG-Net: Our VGG-Net [24] inspired CNN architecture is presented in Table II. It has twelve 1-dimensional convolutional layers, and three fully connected layers, having a total of $1,846,505$ trainable parameters.

B. Loss function

Comparing ellipses using well-known loss functions such as the mean squared error (MSE) or the mean absolute error (MAE), might lead to poor results because of the fact of mixing the five parameters with a very different nature that define an ellipse (i.e. position, orientation, and radii). To overcome this limitation, we have defined a custom loss function with geometric meaning, considering the four characteristic points

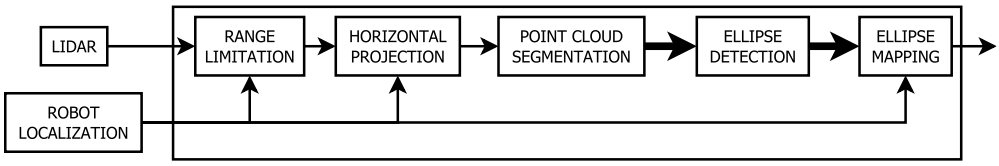


Fig. 3. Architecture of our proposed environment situation awareness. Adaptation of [8].

TABLE I

ARCHITECTURE OF OUR ALEX-NET CNN FOR THE ELLIPSE DETECTOR.

Type	Description		
Input	Dim=1081		
Conv1D	Filt=50, KerSi=7, Str=1, Act=ReLu		
Conv1D	Filt=50, KerSi=7, Str=1, Act=ReLu		
MaxPool1D	Pool=2		
Conv1D	Filt=100, KerSi=7, Str=1, Act=ReLu		
Conv1D	Filt=100, KerSi=7, Str=1, Act=ReLu		
MaxPool1D	Pool=2		
Conv1D	Filt=100, KerSi=7, Str=1, Act=ReLu		
Conv1D	Filt=100, KerSi=7, Str=1, Act=ReLu		
MaxPool1D	Pool=2		
FC	Neur=125, Act=ReLu		
FC	Neur=2, Act=Lin	Neur=1, Act=Lin	Neur=2, Act=ReLu
Output	$\boldsymbol{t}_E^S$, Dim=2	ψ_E^S, Dim=1	$\boldsymbol{r}$, Dim=2

TABLE II

ARCHITECTURE OF OUR VGG-NET CNN FOR THE ELLIPSE DETECTOR.

Type	Description		
Input	Dim=1081		
Conv1D	Filt=25, KerSi=3, Str=1, Act=ReLu		
Conv1D	Filt=25, KerSi=3, Str=2, Act=ReLu		
Conv1D	Filt=50, KerSi=3, Str=1, Act=ReLu		
Conv1D	Filt=50, KerSi=3, Str=2, Act=ReLu		
Conv1D	Filt=100, KerSi=3, Str=1, Act=ReLu		
Conv1D	Filt=100, KerSi=3, Str=2, Act=ReLu		
Conv1D	Filt=150, KerSi=3, Str=1, Act=ReLu		
Conv1D	Filt=150, KerSi=3, Str=2, Act=ReLu		
Conv1D	Filt=200, KerSi=3, Str=1, Act=ReLu		
Conv1D	Filt=200, KerSi=3, Str=2, Act=ReLu		
Conv1D	Filt=250, KerSi=3, Str=1, Act=ReLu		
Conv1D	Filt=250, KerSi=3, Str=2, Act=ReLu		
FC	Neur=250, Act=ReLu		
FC	Neur=250, Act=ReLu		
FC	Neur=2, Act=Lin	Neur=1, Act=Lin	Neur=2, Act=ReLu
Output	$\boldsymbol{t}_E^S$, Dim=2	ψ_E^S, Dim=1	$\boldsymbol{r}$, Dim=2

depicted in Fig. 4, as the intersection between the principal axes of the ellipse and the ellipse surface. Our loss function computes, the mean of the mean Euclidean norm of the difference between these four characteristic points, P_j, of the ellipses we are evaluating:

$$f = \frac{1}{N} \cdot \sum_{i=1}^{N} \left(\frac{1}{4} \sum_{j=1}^{j=4} \left\| \boldsymbol{t}_{P_{i,j,b}} - \boldsymbol{t}_{P_{i,j,a}} \right\|_2 \right) \tag{1}$$

where $\boldsymbol{t}_{P_{j,k}}$ are the coordinates of the characteristic point $P_{j,k}$ in coordinates of a parent frame.

C. Dataset generation

Using MATLAB, we have generated a synthetic dataset consisting of $4,000,000$ data, formed by the noisy simulated LIDAR sensor measurements and the ground truth ellipse parameters.

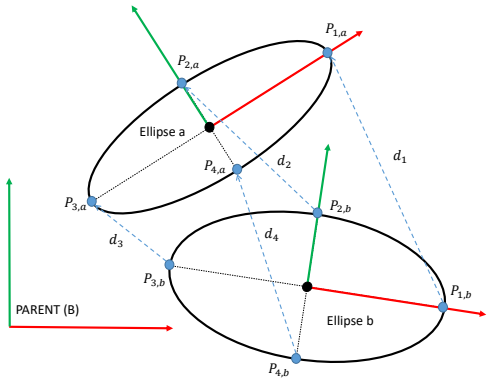


Fig. 4. Characteristic points used on our custom ellipse loss function.

We have simulated individual ellipses with their center randomly placed all over the range of our LIDAR sensor, and their orientation to be within the range $\left[-\frac{\pi}{2},\ \frac{\pi}{2}\right]$ radians. Their large radius (i.e. r_x) has been randomly selected from the interval $[0.05,\ 0.5]$ meters, while their small radius (i.e. r_y) has been randomly drawn from the interval $[0.05,\ r_x]$ meters. We have used continuous uniform distribution functions for the random selection of the aforementioned parameters of the ellipses. Our simulator reproduces our LIDAR sensor by generating the point cloud measurements associated with these ellipses and adds to the distance value of the point cloud, a Gaussian noise with a mean equal to 0 and a standard deviation of 0.01. Measurements partially laying outside the range of our sensor, or point cloud measurements with less than 5 points have been discarded.

D. Training

To train the proposed CNN, we have randomly split the dataset presented in the previous section as indicated in Table III.

TABLE III

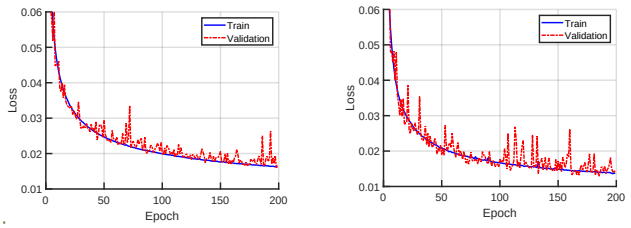
DATASET RANDOM DISTRIBUTION.

Usage	Percentage	# of data
Training	70%	$2,800,000$
Validation	15%	$600,000$
Test	15%	$600,000$
Dataset	100%	$4,000,000$

The training dataset has been used to first calculate the normalization parameters, and then to compute the parameters of the proposed CNN (i.e. train the CNN), checking its performance in every epoch of the training using the valida-

tion data. The test dataset has been only used to evaluate the overall performance of the CNN after the training.

We have trained both of the proposed CNNs during 200 epochs using the Adam optimizer [26], with the default parameters (learning rate equals to 0.0001). Fig. 5 plots the value of our custom loss function of the training and validation data during the training of the proposed CNN architectures.



(a) Alex-Net. (b) VGG-Net.

Fig. 5. Loss function value during the training of the proposed CNNs.

The performance of the two proposed CNN architectures of the ellipse detector has been evaluated using the dataset, obtaining the results presented in Table IV.

TABLE IV

PERFORMANCE OF THE CNN ARCHITECTURES ON THE DATASET.

CNN	Alex-Net			VGG-Net		
Data	**Loss**	**MSE**	**MAE**	**Loss**	**MSE**	**MAE**
Training	0.01563	0.00714	0.01396	0.01285	0.00273	0.01082
Validation	0.01679	0.00849	0.01520	0.01407	0.00451	0.01213
Test	0.01695	0.00874	0.01528	0.01408	0.00445	0.01214

E. Discussion

Analyzing Table IV, we can see that in the case of the VGG-Net, the loss function value is close to 1.4 cm., while in the case of the Alex-Net is slightly lower than 1.7 cm. Therefore, we can extract that, as expected, that the performance of the VGG-Net inspired CNN architecture is slightly better (around 14%) than the Alex-Net counterpart. It is worth to remember that our LIDAR measurements have been generated, including a Gaussian noise with a standard deviation equals to 1 cm., therefore, we can conclude that both proposed CNNs are able to filter the noise, generating an accurate estimate of the ellipse parameters.

Despite having a similar number of parameters (around 1.8 million), the VGG-Net is deeper than the Alex-Net (15 layers vs. 11 layers), which is translated into a larger computational cost. Therefore, if our computational resources are really limited, we propose to use our Alex-Net architecture at the cost of lower precision, and the VGG-Net architecture in another case.

V. SEMANTIC ELLIPSE MAPPING

This component creates a semantic map of the environment as a list of ellipses with a unique identifier, containing, their radii, together with their pose (position and orientation) and velocity (linear and angular), both in world coordinates. It uses the information of the pose of the robot in world coordinates, provided by the localization component, and the list of previously detected ellipses.

The algorithm is based on an indirect (also called error-state) extended Kalman filter (EKF), [27], with mapping capabilities. We use the indirect formulation (vs. direct) to improve its performance when working with angular variables (i.e. orientations). All the noises are assumed to follow a Gaussian distribution.

Similarly than in our previous work [8], this component is implemented to follow an asynchronous operation, where the state of the map elements is updated asynchronously when a measurement is received, and the state of the map elements is published on-demand (e.g. when requested by an eventual controller or planner, or synchronously at a certain rate).

We use simplified unit quaternions $\bar{\boldsymbol{q}} = [q_w,\ q_z]^T = \left[\cos\frac{\psi}{2},\ \sin\frac{\psi}{2}\right]^T$ to compactly represent orientations in the SO(2) space. All the angles are therefore transformed into quaternions (including the measurements given by the ellipse detector presented in Sect. IV).

We use a local perturbation model to decompose the true value, ν_t, of the magnitude, ν, into its nominal value, $\breve{\nu}$, and its incremental value, $\delta\nu$, as $\nu_t = \breve{\nu} \oplus \delta\nu$. For all the magnitudes involved in our problem, except for the orientations, the dimension of the nominal value and the incremental value is always the same, and the operation $\oplus$ is simply a vectorial addition, resulting $\nu_t = \breve{\nu} + \delta\nu$. For the special case of orientations, we need to use the concept of error quaternions, $\delta\bar{\boldsymbol{q}} \approx \left[1,\ \frac{1}{2}\delta\theta\right]^T$, to reduce the dimensionality of the problem in the incremental formulation as follows:

$$\tilde{\boldsymbol{q}}_t = \breve{\bar{\boldsymbol{q}}} \oplus \delta\theta = \breve{\bar{\boldsymbol{q}}} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\delta\theta \end{bmatrix} \tag{2}$$

being $\otimes$ the quaternion product operation.

The definitions and models of this component are presented in sections V-A and V-B, respectively. For brevity, only the true-value definitions and models are specified.

The reader is referred to our previous work [8] to have the details regarding the different stages and general equations of the implemented indirect EKF. Some particularities that differ from our previous work, are presented in Sect. V-C.

In the remainder of the section, we use the following nomenclature, being α a true value variable, $\hat{\alpha}$ an estimated variable, $\breve{\alpha}$ an input variable, and $\tilde{\alpha}$ a measurement variable.

A. Definitions

1) State definition: The full state is formed by the combination of the state of the robot, $\boldsymbol{x}_R$, and the state of all the map elements, $\boldsymbol{x}_{M_i}$.

The state of the robot is given by:

$$\boldsymbol{x}_R = \left[\boldsymbol{t}_R^W,\ \bar{\boldsymbol{q}}_R^W\right]^T \tag{3}$$

where $\boldsymbol{t}_R^W$ and $\bar{\boldsymbol{q}}_R^W$ are, respectively, the position and the orientation of the robot in the world frame.

The map element, M_i, represents an ellipse, whose state is given by:

$$\boldsymbol{x}_{M_i} = \left[\boldsymbol{t}_{M_i}^W,\ \boldsymbol{v}_{M_i}^W,\ \bar{\boldsymbol{q}}_{M_i}^W,\ w_{z\,M_i}^W,\ \boldsymbol{r}_i\right]^T \tag{4}$$

where $\boldsymbol{t}_{M_i}^W$ and $\bar{\boldsymbol{q}}_{M_i}^W$ is the pose (position and orientation) of the ellipse in the world frame; $\boldsymbol{v}_{M_i}^W$ and $w_{z\,M_i}^W$ is the velocity (linear and angular) of the ellipse in the world frame; and $\boldsymbol{r}_i$ are the radii of the ellipse.

2) Inputs definition: We consider as inputs, $\check{\boldsymbol{u}}$, the pose of the robot in world coordinates given by the localization component, as:

$$\check{\boldsymbol{u}}_R = \left[\check{\boldsymbol{t}}_R^W,\ \check{\bar{\boldsymbol{q}}}_R^W\right]^T \tag{5}$$

where $\check{\boldsymbol{t}}_R^W$ and $\check{\bar{\boldsymbol{q}}}_R^W$ are the position and the orientation of the robot in the world frame given by the localization component.

The reader must note that we use the information given by the localization component as inputs instead of as measurements, as our goal is the estimation of the map elements of the environment and not the localization of the robot.

3) Measurements definition: We consider as measurements, $\tilde{\boldsymbol{z}}$, the list of detected ellipses provided by the ellipse detection component, as:

$$\tilde{\boldsymbol{z}}_{M_i} = \left[\tilde{\boldsymbol{t}}_{M_i}^S,\ \tilde{\bar{\boldsymbol{q}}}_{M_i}^S,\ \tilde{\boldsymbol{r}}_i\right]^T \tag{6}$$

where $\tilde{\boldsymbol{t}}_{M_i}^S$ and $\tilde{\bar{\boldsymbol{q}}}_{M_i}^S$ is the measured pose (position and orientation) of the ellipse in the sensor frame, and $\tilde{\boldsymbol{r}}_i$ is the measured radii of the ellipse.

B. Models

1) Process model: The true-value process model, $\boldsymbol{x}(k) = f(\boldsymbol{x}(k-1), \check{\boldsymbol{u}}(k-1))$, is decoupled into the robot model and every individual map element model.

The robot process model is given by:

$$\boldsymbol{x}_R(k) = \check{\boldsymbol{u}}_R(k-1) \tag{7}$$

The map element process model is given by:

$$\boldsymbol{t}_{M_i}^W(k) = \boldsymbol{t}_{M_i}^W(k-1) + \Delta t \cdot \boldsymbol{v}_{M_i}^W(k-1) \tag{8}$$
$$\boldsymbol{v}_{M_i}^W(k) = \boldsymbol{v}_{M_i}^W(k-1) + \boldsymbol{n}_{f_v} \tag{9}$$
$$\bar{\boldsymbol{q}}_{M_i}^W(k) = \frac{1}{2} \cdot \delta\bar{\boldsymbol{q}}_w \otimes \bar{\boldsymbol{q}}_{M_i}^W(k-1) \tag{10}$$
$$w_{z\,M_i}^W(k) = w_{z\,M_i}^W(k-1) + n_{f_w} \tag{11}$$
$$\boldsymbol{r}_i(k) = \boldsymbol{r}_i(k-1) \tag{12}$$

where Δt is the time difference between two prediction steps; $\delta\bar{\boldsymbol{q}}_w = \left[\cos\left(\frac{1}{2}\cdot\delta\psi_w\right),\ \sin\left(\frac{1}{2}\cdot\delta\psi_w\right)\right]^T$, being $\delta\psi_w = \Delta t \cdot w_{z\,M_i}^W(k-1)$; and $\boldsymbol{n}_{f_v}$ and n_{f_w} are the noises of the robot process model associated with the velocity both linear and angular, respectively. The noises $\boldsymbol{n}_{f_v}$ and n_{f_w}, are added to take into account the fact that the map elements are not always moving at a constant velocity or in a constant direction.

2) Measurement model: The measurement model, $\tilde{\boldsymbol{z}}(k) = h(\boldsymbol{x}(k))$, is decoupled for every individual map element, depending only on the robot state, and it is given by:

$$\tilde{\boldsymbol{t}}_{M_i}^S = \left(\boldsymbol{R}_S^R\right)^T \cdot \left(\left(\boldsymbol{R}_R^W\right)^T \cdot \left(\boldsymbol{t}_{M_i}^W - \boldsymbol{t}_R^W\right) - \boldsymbol{t}_S^R\right) + \boldsymbol{n}_{h_t} \tag{13}$$
$$\tilde{\bar{\boldsymbol{q}}}_{M_i}^S = \left(\bar{\boldsymbol{q}}_S^R\right)^* \otimes \left(\bar{\boldsymbol{q}}_R^W\right)^* \otimes \bar{\boldsymbol{q}}_{M_i}^W \otimes \bar{\boldsymbol{q}}_{n_{h_\psi}} \tag{14}$$
$$\tilde{\boldsymbol{r}}_i = \boldsymbol{r}_i + \boldsymbol{n}_{h_r} \tag{15}$$

where $\boldsymbol{R}_R^W$ is the rotation matrix of the attitude of the robot in the world frame, calculated using the simplified quaternion $\bar{\boldsymbol{q}}_R^W$; $\boldsymbol{t}_S^R$ and $\bar{\boldsymbol{q}}_S^R$ represents the calibrated pose (translation and orientation) of the sensor in the robot frame ($\boldsymbol{R}_S^R$ is the associated rotation matrix); $\bar{\boldsymbol{q}}_{n_{h_\psi}} = \left[\cos\left(\frac{1}{2}\cdot n_{h_\psi}\right),\ \sin\left(\frac{1}{2}\cdot n_{h_\psi}\right)\right]^T$; and $\boldsymbol{n}_{h_t}$, n_{h_ψ}, and n_{h_r} are the measurement noises.

3) Mapping model: The mapping model, $\boldsymbol{x}'(k) = g\left(\boldsymbol{x}(k), \tilde{\boldsymbol{z}}(k)\right)$, depends only on the robot state and every individual unassociated measurements, generating a new map element, following the equations:

$$\boldsymbol{t}_{M_i}^W = \boldsymbol{R}_R^W \cdot \left(\boldsymbol{R}_S^R \cdot \tilde{\boldsymbol{t}}_{M_i}^S + \boldsymbol{t}_S^R\right) + \boldsymbol{t}_R^W \tag{16}$$
$$\boldsymbol{v}_{M_i}^W = \boldsymbol{0}_{2\times 1} + \boldsymbol{n}_{g_v} \tag{17}$$
$$\bar{\boldsymbol{q}}_{M_i}^W = \bar{\boldsymbol{q}}_R^W \otimes \bar{\boldsymbol{q}}_S^R \otimes \tilde{\bar{\boldsymbol{q}}}_{M_i}^S \tag{18}$$
$$w_{z\,M_i}^W = 0 + n_{g_w} \tag{19}$$
$$\boldsymbol{r}_i = \tilde{\boldsymbol{r}}_i \tag{20}$$

where $\boldsymbol{n}_{g_v}$ and n_{g_w} are the mapping noise associated with the estimation of the velocity of the ellipse in the world frame, as it cannot be directly observed from the measurement.

C. Stages: Considerations

As presented in Sect. III-B, the same elliptical shape can represent two different ellipses with different orientations (see Fig. 2). Our ellipse detector is therefore unable to determine the measured absolute orientation of the ellipse, which can only be recovered by taking into consideration our available environment map. This fact needs to be implemented in our algorithm, concretely when carrying out the data association between the predicted measurements and the actual measurements. The actual measurement candidate may need to be rotated π radians according to the predicted measurement candidate in a way that the orientation difference is lower than π radians. In the case of an association is found, the rotated measurement is used in the update stage. In the case that the measurement has not been successfully associated with any map element, the measurement with an orientation in the range $\psi_{E_j}^S \in \left[-\frac{\pi}{2},\ \frac{\pi}{2}\right]$ is mapped.

VI. EVALUATION AND RESULTS

A. Evaluation methodology

The validation of the proposed environment situation awareness is done considering three different aspects: accuracy, performance, and robustness.

In Sect. VI-C we use our synthetic dataset presented in Sect. IV-C to compare the accuracy of our ellipse detector with other ellipse detection algorithms. In Sect. VI-D, we deeply evaluate thanks to a simulator, the accuracy and performance of the proposed system when it is facing an environment with ellipsoidal cylinders. Finally, in Sect. VI-E we evaluate the robustness of the proposed system in a real environment that is populated with objects without a perfect elliptical shape.

B. Experimental setup

All the components presented on this work have been implemented as ROS nodes [28] programmed in C++, except the ellipse detector that has been implemented with Python. They are all executed on a consumer Laptop running Ubuntu 18.04. We have implemented our CNN using Keras [29] running on top of TensorFlow [30]. We select our proposed VGG-Net architecture for the ellipse detector.

Our aerial platform is a DJI Matrice 100[2] quadrotor, equipped with a Hokuyo UTM 30LX[3] LIDAR mounted onboard the aerial platform in a calibrated location. This LIDAR has an angular range of $270°$ with an angular resolution of $0.25°$, i.e. 1081 points. Its linear range goes from 0.1 m. to 30 m., although we have limited it to 10 m and its measurement rate is set to 20 Hz.

We have simulated our aerial platform and LIDAR sensor in an environment populated with several static and moving ellipsoidal cylinders using the CoppeliaSim simulator (see Fig. 6). The simulator is integrated with the proposed components with a Software-In-The-Loop (SITL) methodology to carry out the simulated evaluation presented in Sect. VI-D.

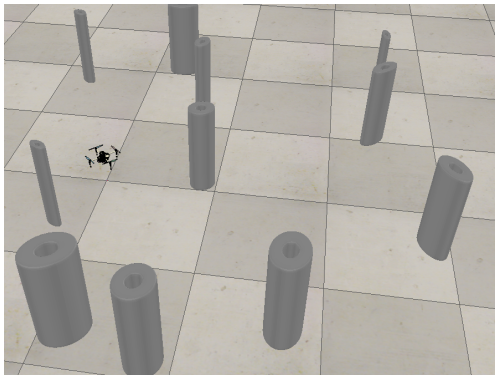


Fig. 6. Scene used in our Coppeliasim simulation with both static and moving elliptical cylinders.

For the experimental results presented in Section VI-E, we command our real aerial robot to hover inside our flying arena (see Fig. 1). Our range limiter restricts the operation volume of our flight arena to a cube of x: -2.5 m. to 2.5 m., y: -3 m. to 3 m., and z: 0.1 m. to 5 m. We use as the robot localization component the multi-sensor fusion state estimator presented in [31], feeding it with the pose measurements provided by an Optitrack motion capture system. Two moving persons and a quasi-cylindrical (i.e. barrel) static object are used as the existing objects of the environment.

[2]https://www.dji.com/matrice100

[3]https://www.hokuyo-aut.jp/search/single.php?serial=169

C. Dataset results

We compare the accuracy of our ellipse detector with two other ellipse detection algorithms using our synthetic dataset (see Sect. IV-C). The results are presented in Table V. Some examples are displayed in Fig. 7.

The first baseline detector is a Least Squares (LS) algorithm based on the Dogleg method, which uses all the points of the point cloud measurement, transformed to Cartesian coordinates, to compute the parameters of the ellipse, which have been bounded according to the parameters of our synthetic dataset. The residuals have been computed using the analytical equation of the ellipse in Cartesian coordinates.

The second baseline detector is a clustering Random sample consensus (RANSAC) algorithm that uses three points of the point cloud measurement to generate a candidate using the aforementioned LS. Then, it determines if the candidate is a good fit or a new one needs to be generated by using the distance between the candidate ellipse and the rest of the points of the measurement. Finally, a refined ellipse is computed with the inlier points.

TABLE V
LOSS FUNCTION VALUE ON THE DATASET.

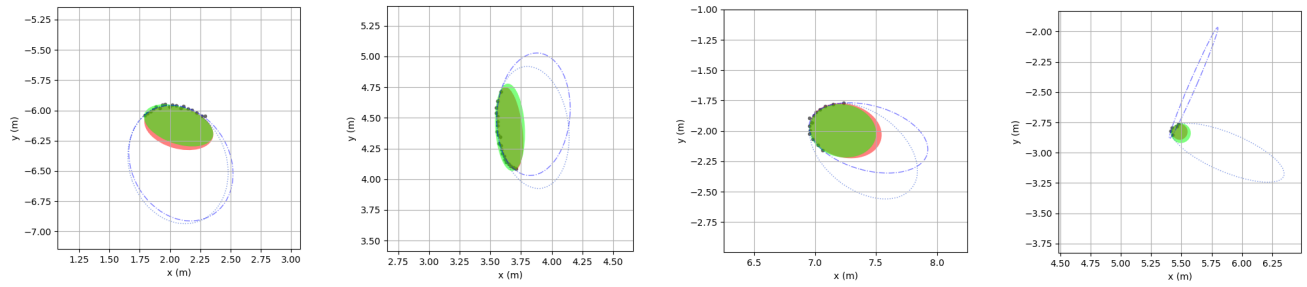
Data	Least Squares	RANSAC	Ours VGG-Net
Training	0.34359	0.37042	0.01285
Validation	0.34375	0.37009	0.01407
Test	0.34349	0.37044	0.01408

We first can extract from Table V that the LS detector has better accuracy than the RANSAC one. This is due to the fact that our dataset includes noisy measurements, but not outliers, where RANSAC outperforms LS.

Secondly, our proposed ellipse detector has an accuracy of around 25 times better than the best of the baseline detectors (i.e. LS). We believe that this is due to the fact that the LS (and RANSAC) is using the point cloud to compute the parameters of an ellipse, without adding any extra information about the sensor into the algorithm, while our detector also is implicitly learning how the LIDAR sensor acquires the measurements, using this knowledge to compute a more accurate ellipse. Looking at Fig. 7, it can be agreed that all the ellipses generated are properly fitting the given point cloud, but only the ellipses generated by our proposed detector are compatible too with the sensor acquisition (i.e. there would be more points in the measurement if the ellipses proposed by the LS and RANSAC would be the actual one).

D. Simulation results

We use the aforementioned CoppeliaSim simulator to carry out two different sets of experiments to quantitatively and qualitatively evaluate the accuracy and performance of the proposed situation awareness system.



(a) Example 1. (b) Example 2. (c) Example 3. (d) Example 4.

Fig. 7. Comparison of ellipse detection between our proposed CNN architecture (green fill), ground truth (red fill), LS (blue dash-dotted edge), and RANSAC (blue dotted edge). The gray dots represent the point cloud measurement. Best viewed in color.

The first set of experiments consists of placing one single elliptical cylindrical moving object inside the range of the LIDAR. The values of the pose and radii of the ellipse provided by the simulator are used as the ground truth. We use the following baseline situation: an elliptical cylinder with $r_x = 0.35$ m., $r_x/r_y = 2$, $\psi = 45°$, following a circular trajectory in front of the LIDAR (trajectory 1). We modify one by one, the parameters that define this baseline situation to deeply analyze the accuracy and performance of the proposed system. The quantitative evaluation is carried out by using the following two indicators: mean squared error (MSE), in m^2 and in deg^2 (for ψ); and the maximum absolute error (MaAE), in m and in deg (for ψ).

Table VI presents the results of the simulation when changing the angle ψ.

TABLE VI

SET OF EXPERIMENTS WHERE THE ANGLE IS CHANGED.

	r_x		r_y		t_x		t_y		ψ	
Angle	MSE	MaAE	MSE	MaAE	MSE	MaAE	MSE	MaAE	MSE	MaAE
0°	0.0013	0.1281	1.4403e-04	0.0404	0.0014	0.1372	3.4209e-04	0.0549	11.3785	19.0814
30°	6.8667e-04	0.1124	1.8435e-04	0.0371	6.7569e-04	0.0782	5.3434e-04	0.0639	12.6173	8.6418
−30°	6.4019e-04	0.0914	1.4418e-04	0.0388	6.3202e-04	0.0776	4.9114e-04	0.0484	11.0509	9.7839
45°	2.7868e-04	0.0488	1.8034e-04	0.0343	3.3412e-04	0.0647	4.7924e-04	0.0443	8.0020	9.0519
−45°	3.1640e-04	0.0910	2.1245e-04	0.0365	3.9307e-04	0.0808	5.3060e-04	0.0482	10.1390	10.3137
60°	1.3598e-04	0.0552	2.0517e-04	0.0399	2.2522e-04	0.0552	4.9468e-04	0.0408	8.4616	10.4655
−60°	1.3829e-04	0.0517	1.8364e-04	0.0380	2.8324e-04	0.0549	4.6909e-04	0.0444	8.5898	9.3156
±90°	1.0550e-04	0.0350	2.1527e-04	0.0486	1.7801e-04	0.0385	5.3101e-04	0.0458	987.2600	89.8826

Table VII presents the results of the simulation when changing the major radius r_x.

TABLE VII

SET OF EXPERIMENTS WHERE THE MAJOR RADIUS IS CHANGED.

	r_x		r_y		t_x		t_y		ψ	
r_x	MSE	MaAE	MSE	MaAE	MSE	MaAE	MSE	MaAE	MSE	MaAE
0.1 m	2.7538e-04	0.0718	2.7598e-04	0.0430	5.4090e-04	0.0691	3.2995e-04	0.0425	271.9011	50.5280
0.2 m	2.4007e-04	0.0795	1.4158e-04	0.0388	3.7681e-04	0.0689	4.0013e-04	0.0387	25.1857	19.4990
0.3 m	2.0730e-04	0.0446	1.9387e-04	0.0481	3.2370e-04	0.0666	4.8198e-04	0.0428	10.7164	11.9074
0.35 m	2.7868e-04	0.0488	1.8034e-04	0.0343	3.3412e-04	0.0647	4.7924e-04	0.0443	8.0020	9.0519
0.4 m	3.6936e-04	0.0715	2.1963e-04	0.0520	4.1325e-04	0.0700	5.3854e-04	0.0479	7.1554	8.2359
0.5 m	6.3557e-04	0.0749	3.9815e-04	0.0598	7.1862e-04	0.0770	6.5887e-04	0.0587	6.7921	8.2385

Table VIII presents the results of the simulation when changing the radii relationship r_x/r_y.

TABLE VIII

SET OF EXPERIMENTS WHERE THE RADII RELATIONSHIP IS CHANGED.

	r_x		r_y		t_x		t_y		ψ	
r_x/r_y	MSE	MaAE	MSE	MaAE	MSE	MaAE	MSE	MaAE	MSE	MaAE
1	4.1598e-04	0.0612	2.5299e-04	0.0515	3.7889e-04	0.0491	4.1892e-04	0.0500	4.0128e+03	134.2301
2	2.7868e-04	0.0488	1.8034e-04	0.0343	3.3412e-04	0.0647	4.7924e-04	0.0443	8.0020	9.0519
3	3.3721e-04	0.0534	1.8274e-04	0.0463	3.4662e-04	0.0608	5.3771e-04	0.0452	5.7994	8.0498
4	3.1170e-04	0.0666	1.7837e-04	0.0484	3.8516e-04	0.0780	5.2775e-04	0.0471	4.7916	7.0188
5	2.7925e-04	0.0635	1.8321e-04	0.0338	3.3470e-04	0.0511	5.2124e-04	0.0491	4.2823	5.6857

Table IX presents the results of the simulation when changing the trajectory of the ellipse to the right-side of the robot (trajectory 2) or the left-side (trajectory 3).

TABLE IX

SET OF EXPERIMENTS WHERE THE TRAJECTORY IS CHANGED.

	r_x		r_y		t_x		t_y		ψ	
Trajectory	MSE	MaAE	MSE	MaAE	MSE	MaAE	MSE	MaAE	MSE	MaAE
1	2.7868e-04	0.0488	1.8034e-04	0.0343	3.3412e-04	0.0647	4.7924e-04	0.0443	8.0020	9.0519
2	3.7685e-04	0.0657	1.9319e-04	0.0418	4.5475e-04	0.0485	5.2950e-04	0.0648	12.1150	9.4462
3	2.6057e-04	0.0665	1.9313e-04	0.0386	2.0410e-04	0.0372	4.3469e-04	0.0573	9.3525	7.9145

We can conclude after analyzing this first set of experiments that, in general, our proposed system is able to estimate both radius with an MSE lower than $7.0 \cdot 10^{-4}$ m^2 and a MaAE lower than 0.09 m. Similarly, the position is estimated with an MSE lower than $7.0 \cdot 10^{-4}$ m^2 and a MaAE lower than 0.08 m. The angle, ψ, is computed with an MSE lower than 12 deg^2 and a MaAE lower than 11 deg.

As expected (see Table VI), when the ellipses have a relative angle (i.e. with respect to the sensor) of 0°, our solution has difficulties to estimate the r_x and t_x, since these parameters become difficult to observe, as well as computing the angle. When the relative angle is ±90°, our detector has a large error in the estimation of the angle, due to the singularity on the angle representation. Improving this remains as future work.

From Table VII, we can conclude that the smaller the ellipse is, the lower the accuracy on the angle estimation is. This expected result is due to the increasing number of points on the measurement (i.e. used information) when the ellipse is larger.

It is also straightforward to conclude from Table VIII, that the larger the radii relationship is, the more accurate is the ellipse angle estimation. This estimation degenerates in the case of a circle (i.e. relationship equals 1), being unable to observe it.

In the second set of experiments, we randomly place multiple static and dynamic elliptical cylindrical objects in the simulated environment presented in Fig. 6. We use this simulation to qualitatively evaluate the performance of our proposed situation awareness system.

The whole experiment can be visualized in `https://youtu.be/Xy1HbmlMJ04`. Fig. 8 displays some screenshots of the experiment. The static objects are represented with red-colored ellipses and the moving objects with blue

ones. The estimated ellipses calculated by our proposed situation awareness system are displayed in green. As can be seen, the estimated ellipses are practically overlapped with the ground truth objects. The proposed situation awareness system is capable of effectively build and maintain a semantic map, adding and removing objects when occlusions occur.

E. Experimental results

We qualitatively evaluate the robustness of our proposed situation awareness system when facing a real environment where the objects do not have an exact elliptical shape, as described in Sect. VI-B. The whole experiment can be visualized in `https://youtu.be/EpM7VclnmIk`. The reader should put attention, as detailed in Fig. 9, to the measurement of the LIDAR (gray dots), being able to easily perceive the non-elliptical shape of the persons containing the two arms and the body.

As can be concluded from the experiment, the proposed situation awareness system is capable of handling a real environment with dynamic non-elliptical objects, as well as the previously mentioned occlusions. The reader must note that in some time instants, the persons are represented with two or three ellipses instead of with only one, due to the fact that the arms and the body are calculated as independent shapes. Nevertheless, the proposed situation awareness system handles properly this change of representation by adding and deleting elements to the semantic map. The whole system was running in real-time.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented a semantic situation awareness system for multirotor aerial robots, based on 2D LIDAR measurements, as the continuation of our previous work [8]. Assuming to have a precise robot localization, we generate in real-time a semantic map of the environment that consists of a list of ellipses, represented by their radii, their pose, and their velocity, both in world coordinates. Two different CNN architectures have been proposed and trained using an artificially generated dataset and a custom loss function, to detect ellipses in a segmented 2D LIDAR measurement. In cascade, an indirect-EKF estimates the ellipses based semantic map in world coordinates, including an estimate of their velocities.

We have carried out a thoroughful evaluation of our proposed semantic situation awareness system considering three different aspects: accuracy, performance, and robustness. First, we compared our detector with other ellipse fitting algorithms. Second, two sets of SITL simulations using CoppeliaSim produced both quantitative and qualitative evaluations that support our proposed algorithm. Lastly, real laboratory experiments with real non-cylindrical static objects (i.e. a barrel) and two moving persons, demonstrated the robustness of our proposed algorithm handling such real situations.

Our roadmap plan includes the addition of other geometric shapes more than ellipses (e.g. rectangles and lines) to enrich the representation of the environment. We also need to increase the performance of our algorithm handling occlusions. Finally, our long-term plans include the use of neural network techniques for one step object detectors (instead of running the object detector once per segmented object).

REFERENCES

[1] C. Sampedro, H. Bavle, A. Rodriguez-Ramos, P. de la Puente, and P. Campoy, "Laser-based reactive navigation for multirotor aerial robots using deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2018, pp. 1024–1031.

[2] S. Ladha, D. K. Kumar, P. Bhalla, A. Jain, and R. Mittal, "Use of lidar for obstacle avoidance by an autonomous aerial vehicle," *Paper presentation at IARC-2012*, 2012.

[3] F. Gao and S. Shen, "Online quadrotor trajectory generation and autonomous navigation on point clouds," in *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, 2016, pp. 139–146.

[4] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Robotics Research*. Springer, 2016, pp. 649–666.

[5] J. L. Sanchez-Lopez, J. Pestana, and P. Campoy, "A robust real-time path planner for the collision-free navigation of multirotor aerial robots in dynamic environments," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, June 2017, pp. 316–325.

[6] J. L. Sanchez-Lopez, M. Wang, M. A. Olivares-Mendez, M. Molina, and H. Voos, "A real-time 3d path planning solution for collision-free navigation of multirotor aerial robots in dynamic environments," *Journal of Intelligent & Robotic Systems*, vol. 93, no. 1, pp. 33–53, Feb 2019. [Online]. Available: https://doi.org/10.1007/s10846-018-0809-5

[7] M. Castillo-Lopez, P. Ludivig, S. A. Sajadi-Alamdari, J. L. Sanchez-Lopez, M. A. Olivares-Mendez, and H. Voos, "A real-time approach for chance-constrained motion planning with dynamic obstacles," *arXiv preprint arXiv:2001.08012*, 2020.

[8] J. L. Sanchez-Lopez, C. Sampedro, D. Cazzato, and H. Voos, "Deep learning based semantic situation awareness system for multirotor aerial robots using lidar," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, June 2019, pp. 899–908.

[9] C. Premebida and U. Nunes, "Segmentation and geometric primitives extraction from 2d laser range data for mobile robot applications," *Robotica*, vol. 2005, pp. 17–25, 2005.

[10] M. Dekan, D. František, B. Andrej, R. Jozef, R. Dávid, and M. Josip, "Moving obstacles detection based on laser range finder measurements," *International Journal of Advanced Robotic Systems*, vol. 15, no. 1, p. 1729881417748132, 2018.

[11] C.-C. Wang, C. Thorpe, and S. Thrun, "Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas," in *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, vol. 1. IEEE, 2003, pp. 842–849.

[12] Á. M. Guerrero-Higueras, C. Álvarez-Aparicio, M. C. C. Olivera, F. J. Rodríguez-Lera, C. Fernández-Llamas, F. M. Rico, and V. Matellán, "Tracking people in a mobile robot from 2d lidar scans using full convolutional neural networks for security in cluttered environments," *Frontiers in neurorobotics*, vol. 12, 2018.

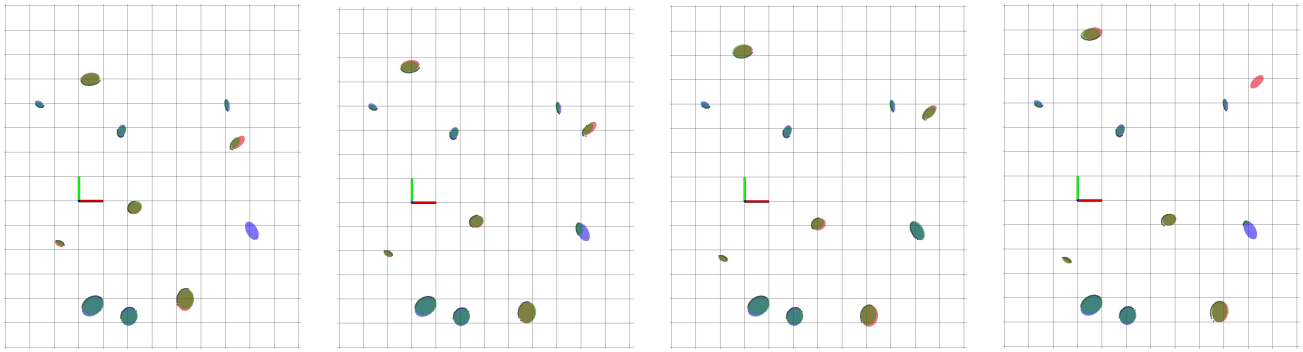
[13] J. Han, J. Kim, and D. H. Shim, "Precise localization and mapping in indoor parking structures via parameterized slam," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 12, pp. 4415–4426, 2018.

[14] J. Huang and S. You, "Point cloud labeling using 3d convolutional neural network," in *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 2016, pp. 2670–2675.

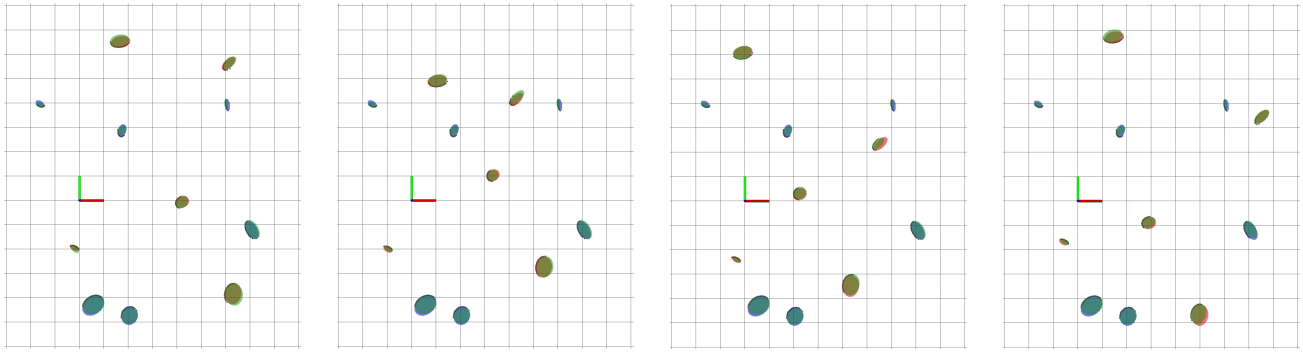
[15] B. Li, T. Zhang, and T. Xia, "Vehicle detection from 3d lidar using fully convolutional network," *arXiv preprint arXiv:1608.07916*, 2016.

[16] J. Beltran, C. Guindel, F. M. Moreno, D. Cruzado, F. Garcia, and A. De La Escalera, "Birdnet: a 3d object detection framework from lidar information," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 3517–3523.

[17] V. Vaquero, A. Sanfeliu, and F. Moreno-Noguer, "Deep lidar cnn to understand the dynamics of moving vehicles," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–6.



(a) Time 2.0 s. (b) Time 6.5 s. (c) Time 9.0 s. (d) Time 14.0 s.



(e) Time 20.0 s. (f) Time 30.5 s. (g) Time 40.0 s. (h) Time 50.0 s.

Fig. 8. Simulation experiment using the CoppeliaSim simulator with multiple static (blue) and moving (red) objects. The estimated ellipses are displayed in green. The gray dots represent the point cloud measurement. Best viewed in color. The whole experiment can be visualized in `https://youtu.be/Xy1HbmlMJ04`

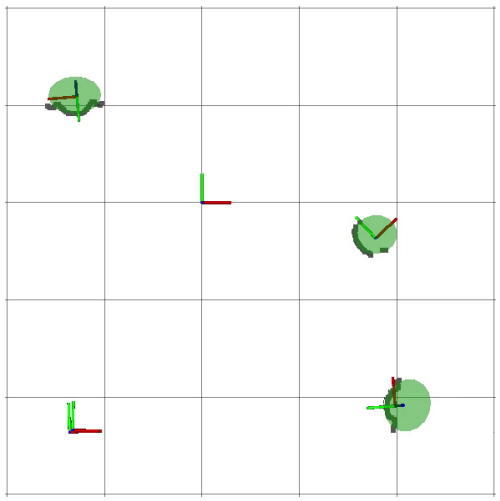


Fig. 9. Detail of the LIDAR measurement (gray dots) and ellipses estimation (green) on the Laboratory experiment with a barrel (center right) and two persons (top-left and bottom-right). The reference frames represent the robot (bottom-left) and the world (center), as well as the ground truth pose of the center of the objects. Best viewed in color. The whole experiment can be visualized in `https://youtu.be/EpM7VclnmIk`

[18] P. L. Rosin, "Further five-point fit ellipse fitting," *Graphical Models and Image Processing*, vol. 61, no. 5, pp. 245–259, 1999.

[19] Y. Xie and Q. Ji, "A new efficient ellipse detection method," in *Object recognition supported by user interaction for service robots*, vol. 2. IEEE, 2002, pp. 957–960.

[20] D. K. Prasad, M. K. Leung, and C. Quek, "Ellifit: An unconstrained, non-iterative, least squares based geometric ellipse fitting method," *Pattern Recognition*, vol. 46, no. 5, pp. 1449–1465, 2013.

[21] J. Liang, P. Li, D. Zhou, H. So, D. Liu, C.-S. Leung, and L. Sui, "Robust ellipse fitting via alternating direction method of multipliers," *Signal Processing*, vol. 164, pp. 30–40, 2019.

[22] W. Dong, P. Roy, C. Peng, and V. Isler, "Ellipse r-cnn: Learning to infer elliptical object from clustering and occlusion," *arXiv preprint arXiv:2001.11584*, 2020.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[24] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[25] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, M. Hasan, B. C. Van Essen, A. A. Awwal, and V. K. Asari, "A state-of-the-art survey on deep learning theory and architectures," *Electronics*, vol. 8, no. 3, p. 292, 2019.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.

[27] R. W. Beard and T. W. McLain, *Small unmanned aircraft: Theory and practice*. Princeton university press, 2012.

[28] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009, p. 5.

[29] "Keras web," https://keras.io/.

[30] "Tensorflow web," https://www.tensorflow.org/.

[31] J. L. Sanchez-Lopez, V. Arellano-Quintana, M. Tognon, P. Campoy, and A. Franchi, "Visual marker based multi-sensor fusion state estimation," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 16 003 – 16 008, 2017, 20th IFAC World Congress.