Deep learning based semantic situation awareness system for multirotor aerial robots using LIDAR

Jose Luis Sanchez-Lopez[1] and Carlos Sampedro[2] and Dario Cazzato[1] and Holger Voos[1,3]

***Abstract*— In this work, we present a semantic situation awareness system for multirotor aerial robots, based on 2D LIDAR measurements, targeting the understanding of the environment and assuming to have a precise robot localization as an input of our algorithm. Our proposed situation awareness system calculates a semantic map of the objects of the environment as a list of circles represented by their radius, and the position and the velocity of their center in world coordinates. Our proposed algorithm includes three main parts. First, the LIDAR measurements are preprocessed and an object segmentation clusters the candidate objects present in the environment. Secondly, a Convolutional Neural Network (CNN) that has been designed and trained using an artificially generated dataset, computes the radius and the position of the center of individual circles in sensor coordinates. Finally, an indirect-EKF provides the estimate of the semantic map in world coordinates, including the velocity of the center of the circles in world coordinates.**

We have quantitative and qualitative evaluated the performance of our proposed situation awareness system by means of Software-In-The-Loop simulations using VRep with one and multiple static and moving cylindrical objects in the scene, obtaining results that support our proposed algorithm. In addition, we have demonstrated that our proposed algorithm is capable of handling real environments thanks to real laboratory experiments with non-cylindrical static (i.e. a barrel) and moving (i.e. a person) objects.

I. INTRODUCTION

The design of a robust real-time obstacle detection and avoidance system for autonomous aerial robot navigation is a complex problem, thus it is not surprising that many works in the state of the art have focused on the problem in exam [1], [2], [3]. These systems usually process data coming from sensors that are suitable to be easily put on board like monocular [4] or stereo [5] cameras, depth sensors [6], radars [7], or laser scanner [8]. Often multiple sensors are carried on board, and methods for integrating different data sources have also been proposed [9], [10].

On the other hand, in order to handle dynamic complex environments, semantic representations are preferred over grid-based ones, when applying deliberative path planning [11], [12], or reactive obstacle avoidance [13].

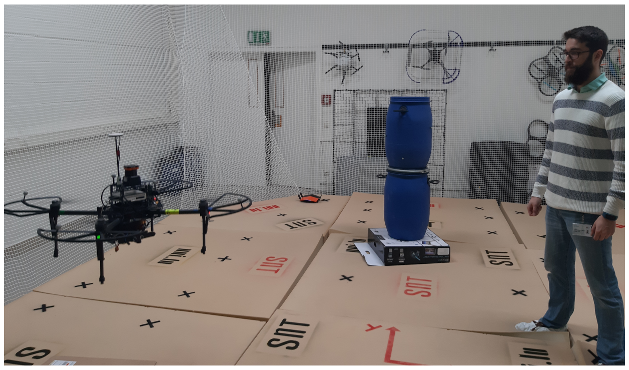


Fig. 1. Lab experiment with one quasi-cylindrical static object (i.e. blue barrels) and one non-cylindrical moving object (i.e. person).

In this paper, we focus on LIDAR based semantic situation awareness of the environment for aerial robots (see Fig. 1). We limit the semantic representation of the objects to their shape. Despite many solutions to estimate and classify a shape from different data sources have been presented, few works focus on deep neural networks to process 2D LIDAR data to detect shapes for aerial robotics, being most of the state of the art work related to 3D LIDAR range data for ground robots.

A. Problem formulation and hypothesis

In this work, we assume to have a multirotor aerial robot equipped with a 2D LIDAR rigidly attached to its body frame with a known calibration. The aerial robot includes a localization component that provides an accurate fast estimation of the pose of the robot with respect to the world reference frame (e.g. GNSS/INS based localization or a motion capture based localization).

We assume that the environment is populated only by objects with an approximately cylindrical shape, moving in the horizontal plane. The objects are assumed to have infinite height. This last hypothesis can be relaxed by constraining the movement of the aerial robot to only the horizontal plane without changing its flight altitude. Having an aerial robot moving only in the horizontal plane might be a requirement for some applications (e.g. inspection of crop trees for precision agriculture).

The goal of this work is to compute a semantic map of the objects of the environment using the measurements provided by the onboard LIDAR and the information of the aerial robot localization. The semantic map is described as a list of

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

[1] Dr. Jose Luis Sanchez-Lopez and Dr. Dario Cazzato and Prof. Dr.-Ing. Holger Voos are with Automation and Robotics Research Group (ARG), 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

[2] Carlos Sampedro is with the Computer Vision and Aerial Robotics (CVAR), Centre for Automation and Robotics (CAR), Technical University of Madrid (CSIC-UPM). Address: Calle de Jose Gutierrez Abascal, 2, 28006 Madrid (Spain)

[3] Prof. Dr.-Ing. Holger Voos is with the Faculté des Sciences, de la Technologie et de la Communication, University of Luxembourg. e-mail: holger.voos@uni.lu

circles represented by their radius, the position of their center and the velocity of their center, both in world coordinates. Every circle includes a unique identifier.

B. Contributions and outline

Our main contribution is the design and development of a semantic situation awareness system for multirotor aerial robots based on 2D LIDAR measurements. The proposed algorithm is the combination of three main modules. The first module encompasses two components that preprocess the LIDAR measurements, together with a third component that carries out an object segmentation to cluster the candidate objects present in the environment. The second one is a Convolutional Neural Network (CNN) that has been designed and trained using an artificially generated dataset to compute the radius and the position of the center of individual circles in sensor coordinates. Lastly, an indirect-EKF estimates the semantic map of the environment in world coordinates, including the velocity of the center of the circles in world coordinates.

The remainder of the paper is organized as follows: Sect. II reviews the related works. Sect. III presents the overall proposed semantic situation awareness system, getting into the details of the preprocessing and object segmentation components. In Sect. IV, the key component that detects the circles using machine learning techniques is presented. Sect. V covers the details of the semantic circle mapping. The evaluation of the performance of the proposed semantic situation awareness system is done in Sect. VI. Finally, Sect. VII concludes the paper.

II. RELATED WORK

In [14], pedestrian detection in urban areas using only LIDAR-based features is proposed. Using a set of features specifically designed for the task, a minimum-redundancy maximum-relevancy method is used to select features and their combination [15]. Five different classifiers have been tested, showing the feasibility of pure LIDAR based obstacle recognition. In the work of [16], shape is extracted from multi-layer LIDAR data. Objects are then classified and tracked by support vector machine (SVM) and a Kalman filter. In [17], obstacles in outdoor navigation are modeled as basic geometric shapes whose locations can be found in the space by processing 3D LIDAR raw point clouds. In particular, shapes are extracted from supervoxels by a RANSAC-based method and the output is optimized by means of a density loss estimation. A solution to detect circular shapes from 2D LIDAR data has been proposed in [18]: first of all, data is grouped in segments by a variation of a Point-Distance-Based Segmentation method improved by adding a copy process to detect circular shape and an angle thresholding procedure; five features are extracted from valid segments and fed to a SVM classifier to detect specific circular objects. Another example of circle detection in LIDAR data is in [19]; the work proposes a method to separate wood and leaf data; different geometric primitives (circles, arcs or lines) have been identified depending on the different tree height, thus circle from points projected in the 2D x-y plane are fitted by means of the Hough transform. Another example of object shapes detected from LIDAR data are building roofs [20] and vehicle L-shapes fitting [21]. Circle fitting for incomplete and noisy 3D LIDAR data has been proposed in [22]. The method combines Principal Component Analysis (PCA) and robust regression with an algebraic circle fitting method. A circle detector suited for LIDAR data and that utilizes M-Split estimation has recently been proposed in [23].

LIDAR-based obstacle detection has also been successfully employed in autonomous navigation scenarios [24]. At this aim, the work of [25] proposes a precise negative obstacle detection system for Unmanned Ground Vehicle (UGV) that can detect obstacles farther than 10 meters, exploiting the less restrictive payload limitations of UAVs. 2D laser range data is for rescue robots autonomous navigation in [26]. Authors propose an algorithm to detect lines and their boundaries. Lines are fitted by a Hough transform and, after removing noisy and sparse points, extracted by a recursive split algorithm. People are identified and tracked by a Convolutional Neural Network (CNN) trained with 2D line scanner data in [27].

III. SEMANTIC SITUATION AWARENESS OF THE ENVIRONMENT

In this section, we first introduce some essential concepts used in our work, then we describe the overall proposed situation awareness system, and we finally describe the preprocessing and object segmentation components.

One single measurement of a 2D LIDAR, $\boldsymbol{z}$, represents a list where each entry is the distance between the measured point of the environment and the sensor, ρ_i, for every orientation of the LIDAR, ψ_i, of its angular range, $\forall \psi_i \in [\psi_{\min},\ \psi_{\max}]$:

$$\boldsymbol{z} = \{z_i = \rho_i(\psi_i)\}, \quad \forall \psi_i \in [\psi_{\min},\ \psi_{\max}] \tag{1}$$

Therefore, a LIDAR measurement can be seen as a point cloud in $\mathbb{R}^2$ represented in polar coordinates:

$$\boldsymbol{z} = \{\boldsymbol{t}_{P_i}^{L} = [\psi_i,\ \rho_i]\}, \quad \forall i \tag{2}$$

It is worth to mention that in case of no physical point in the environment within the linear range of the LIDAR for a particular orientation ψ_i, then, the measured distance by the LIDAR is set to an out-of-range value (e.g. $z_i = \infty$, or $z_i = \text{NaN}$).

A circle in a 2D environment is represented by its radius, r, and the position of its center in world coordinates, $\boldsymbol{t} = [t_x,\ t_y]$. Its analytical equation is the following:

$$(x - t_x)^2 + (y - t_y)^2 = r^2 \tag{3}$$

where x and y are the coordinates of its circumference in the world reference frame.

As mentioned before, our proposed environment situation awareness takes in input, the measurements given by a 2D LIDAR and the information of the pose of the aerial robot, generating as output, the semantic map of the environment as a list of circles. The proposed algorithm is formed by the five components shown in Fig. 2, which are described below.

A. LIDAR RANGE LIMITATION

This preprocessing component limits the operation range of the onboard LIDAR to a user-defined 3D volume described by 2D planes in the world reference frame, filtering the objects in the environment that are outside the operational volume.

The position of any point, P_i, measured by the LIDAR in world coordinates is calculated as:

$$\boldsymbol{t}_{P_i}^{W} = \boldsymbol{p}_R^W \oplus \boldsymbol{p}_L^R \oplus \boldsymbol{t}_{P_i}^L \tag{4}$$

where $\boldsymbol{p}_R^W$ is the pose of the aerial robot in world coordinates, $\boldsymbol{p}_L^R$ is the calibrated pose of the LIDAR with respect to the robot reference frame, and $\oplus$ is the pose composition operation.

If the point, P_i, is located outside the operational volume, the i-th element of the measurement of the LIDAR is set to out-of-range (e.g. $z_i = \infty$, or $z_i = \text{NaN}$).

B. LIDAR HORIZONTAL PROJECTION

Common multirotor aerial robots are underactuated platforms that require to change its tilting (pitch and roll) when moving in the horizontal plane. This tilting distorts the measurements of the LIDAR when perceiving cylinders in the sense that the intersection between the measurement plane and the cylinder becomes an ellipse instead of a circle.

This preprocessing component corrects the LIDAR measurements by compensating the pitch and roll movements of the aerial platform.

The projection of the raw measurement of the LIDAR, $\boldsymbol{z} = \{\rho_i\}$, in the horizontal plane is calculated as follows:

$$\rho_{h_i} = \rho_i \cdot \cos\left(\theta_S\right) \cdot \cos\left(\phi_S\right), \quad \forall i \tag{5}$$

where θ_S and ϕ_S are the pitch and roll values of the LIDAR sensor in world coordinates.

C. OBJECT SEGMENTATION

In order to deal with environments with multiple objects, we use an approach based on object segmentation. With this method, we create from every coming LIDAR measurement, a set of LIDAR measurements, one per object present in the scene. This way, every created LIDAR measurement contains only the measures of one single object present in the scene.

The first step consists of a clustering algorithm that groups into the same cluster the points of the coming LIDAR measurement that belong to the same object. To do that, both the Cartesian and the Polar spaces are used to cluster the points that satisfy a user-defined distance threshold. After that, clusters formed by a less than a user-defined number of points (i.e. 0.3% of the elements of the LIDAR measurement) are filtered as they are considered to be spurious measurements. This algorithm generates a vector, $\boldsymbol{m}$, of the same dimension of the LIDAR measurement, whose values indicate the belonging of a cluster of every point P_i of the LIDAR measurement:

$$\boldsymbol{m} = \{m_i\}, \quad \forall i \tag{6}$$

being $m_i = 0$ if the point P_i is not associated to any cluster, and $m_i = k$ if the point P_i is associated to the cluster $k \in \mathbb{N}$.

Finally, one LIDAR measurement is generated per clustered object, containing only the i-th elements of the coming LIDAR measurement, that belong to the same cluster k defined in the vector $\boldsymbol{m}$.

IV. MACHINE LEARNING BASED CIRCLE DETECTION

The component presented in this section computes the parameters of one single circle using a LIDAR measurement. It is worth to highlight that by means of the components presented in Sect. III, the LIDAR measurement input to this component contains only one single circle.

This component consists of a Convolutional Neural Network (CNN) described in Sect. IV-A. We have generated a synthetic dataset (see Sect. IV-B) to train the artificial neural network using a custom loss function (see Sect. IV-C).

As described in Sect. VI-B, we use a 2D LIDAR, which provides, when taking advantage of its full angular range, a measurement with a dimension of 1081 points. Therefore, the input of the component presented in this section is a vector of dimension 1081 containing all the distances ρ_i.

As mentioned before, the output of this component are the parameters of the computed circle, i.e. the pose of its center in the sensor frame, $\boldsymbol{t}_C^S = \left[t_{x_C^S},\ t_{y_C^S}\right]^T$; and its radius, r_i. Therefore, its output is gathered as a vector of dimension 3.

A. CNN architecture

The architecture of the proposed CNN is graphically represented in Fig. 3 and detailed in Tab. I.

The proposed CNN encompasses six 1-dimensional convolutional layers and two fully connected layers. The last the fully connected layer is used as output layer for regression by means of three neurons with a linear activation function. The reader must note that the typically used pooling layers are not present in our proposed CNN, replacing them by an increase of the strides in the even convolutional layers (i.e. C2, C4, and C6).

The proposed architecture of the CNN has a total of $1,826,350$ trainable parameters.

To improve its performance, the input data is preprocessed before feeding it to the CNN. First of all, it is normalized in the range of $[0,\ 1]$, based on the normalization parameters calculated during the training stage. After the normalization, the out-of-range values (e.g. $z_i = \infty$, or $z_i = \text{NaN}$) are replaced by a real number that is numerically treatable by the CNN (i.e. -1).

B. Dataset generation

We have artificially generated a synthetic dataset using MATLAB by simulating the measurements of a LIDAR as

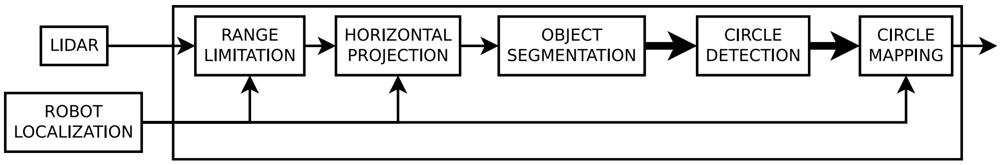


Fig. 2. Architecture of our proposed environment situation awareness.

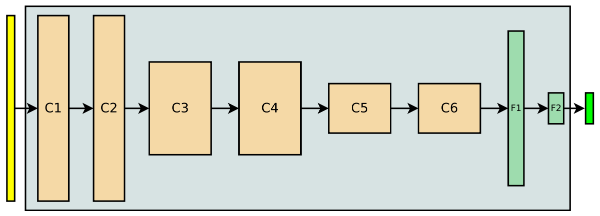


Fig. 3. Architecture of the proposed CNN for the circle detector.

TABLE I

DETAILED DESCRIPTION OF THE ARCHITECTURE OF THE PROPOSED CNN FOR THE CIRCLE DETECTOR.

Layer	Type	Description
Input	Input	Dimension=1081
C1	Conv1D	Filt=50, KerSi=7, Str=1, Act=ReLu
C2	Conv1D	Filt=50, KerSi=7, Str=2, Act=ReLu
C3	Conv1D	Filt=100, KerSi=7, Str=1, Act=ReLu
C4	Conv1D	Filt=100, KerSi=7, Str=2, Act=ReLu
C5	Conv1D	Filt=100, KerSi=7, Str=1, Act=ReLu
C6	Conv1D	Filt=100, KerSi=7, Str=2, Act=ReLu
F1	Fully Connected	Neur=125, Act=ReLu
F2	Fully Connected	Neur=3, Act=Linear
Output	Output	Dimension=3

the one used in our real experiments (described in Sect. VI-B) when perceiving circles with known radius and located in known positions.

We have simulated individually circles with a minimum radius of 0.05 m. and a maximum radius of 0.5 m., with an increasing step of 0.05 m. The center of the simulated circles are uniformly distributed all over the range of the LIDAR, with a distance of 0.05 m. in both coordinates. A Gaussian noise in the LIDAR measurement is incorporated with a mean equal to 0 and a standard deviation of 0.01. We have collected two measurements of every simulated circle, in order to facilitate the learning of noisy measurements.

Our dataset consists, therefore, of $2,011,494$ data, formed by the noisy simulated LIDAR measurements and the ground truth circle parameters (i.e. the pose of its center in the sensor frame, and its radius).

C. Training

To train the proposed CNN, we have randomly splitted the dataset as indicated in Tab. II.

The train and validation datasets are utilized to calculate the normalization parameters. The train dataset have been used to compute the parameters of the CNN, checking its performance in every epoch of the training using the validation data. The test dataset have been used to evaluate the overall performance of the CNN after the whole training.

TABLE II

DATASET RANDOM DISTRIBUTION DURING THE TRAINING PROCESS.

Usage	Percentage	# of data
Train	64%	$1,287,356$
Validation	16%	$321,839$
Test	20%	$402,299$
Dataset	100%	$2,011,494$

Since the output of the CNN is formed by two values representing the position of the center of the circle in sensor frame, and one value representing the radius of the circle, combining these parameters using widely used loss functions such as the mean squared error (MSE) or the mean absolute error (MAE), might result in a poor training due to the very different nature of these two kinds of output values. To increase the performance of the training, we have defined a custom loss function. Our loss function computes the mean of the mean squared Euclidean norm of the difference between four characteristic points, P_i, of the circles we are evaluating:

$$f = \frac{1}{N} \cdot \sum_{i=1}^{N} \left(\frac{1}{4} \sum_{j=1}^{j=4} \| \boldsymbol{t}_{P_{jb}} - \boldsymbol{t}_{P_{ja}} \|_2^2 \right) \tag{7}$$

where $\boldsymbol{t}_{P_{jk}}$ are the coordinates of the characteristic point P_{jk} in LIDAR coordinates. These characteristic points are the intersection between the principal axes of the circle and the circle surface (see Fig. 4). To overcome the symmetry of the circle, we set the principal axes of the circle to be parallel to the LIDAR frame.

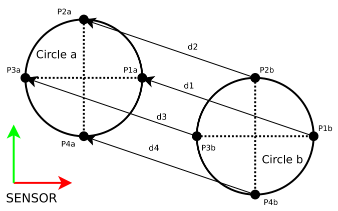


Fig. 4. Characteristic points used for the loss function used for training the proposed CNN.

We have trained the CNN using the Adam optimizer [28], during 106 epochs, with the default parameters (learning rate equals to 0.0001). Fig. 5 displays the value of our custom loss function of the train and validation data during the training of the CNN.

The performance of the whole circle detector has been evaluated using the dataset, obtaining the results shown in Tab. III.

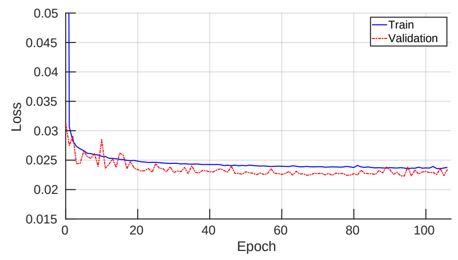


Fig. 5. Value of the loss function during the training of the CNN.

TABLE III
PERFORMANCE OF THE CIRCLE DETECTOR ON THE DATASET.

Data	Loss	MSE	MAE
Train + Validation	0.023907	0.007969	0.014166
Test	0.025035	0.008345	0.014296

V. SEMANTIC CIRCLE MAPPING

This component creates a semantic map of the environment as a list of circles with a unique identifier represented by their radius, the position of their center and its velocity, 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 detected circles by the circle detection presented in Section IV.

The algorithm is based on an error-state (or indirect) extended Kalman filter (EKF), [29], with mapping capabilities. We use the indirect formulation (vs. direct) to improve its performance when working with angular variables (i.e. orientations). Its definitions and models are presented in Section V-A, and its different stages are reviewed in Section V-B. All the noises are assumed to follow a Gaussian distribution.

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 quaternions $\bar{\boldsymbol{q}} = [q_w, \ q_z]^T$ to compactly represent orientations in the SO(2) space. Additionally, we 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 error-state formulation.

In the remainder of the section we use the following nomenclature with α is a true state variable, $\hat{\alpha}$ is an estimated variable, $\check{\alpha}$ is an input variable, $\tilde{\alpha}$ is a measurement variable.

A. Definitions and models

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, $\boldsymbol{x}_{M_i}$ as $\boldsymbol{x} = \left[\boldsymbol{x}_R, \ \boldsymbol{x}_{M_1}, \ .., \ \boldsymbol{x}_{M_i}, \ .., \ \boldsymbol{x}_{M_{nm}}\right]^T$.

The state of the robot is given by:

$$\boldsymbol{x}_R = \begin{bmatrix} \boldsymbol{t}_R^W \\ \hdashline \bar{\boldsymbol{q}}_R^W \end{bmatrix} \tag{8}$$

where $\boldsymbol{t}_R^W = \left[t_{x}{}_R^W, \ t_{y}{}_R^W\right]^T$ and $\bar{\boldsymbol{q}}_R^W = \left[q_{w}{}_R^W, \ q_{z}{}_R^W\right]^T$ are, respectively, the position and the orientation (as a simplified quaternion) of the robot in world frame in a SE(2) space.

The map element, M_i, represents a circle, whose state is given by:

$$\boldsymbol{x}_{M_i} = \begin{bmatrix} \boldsymbol{t}_{M_i}^W \\ \hdashline \boldsymbol{v}_{M_i}^W \\ \hdashline r_i \end{bmatrix} \tag{9}$$

where $\boldsymbol{t}_{M_i}^W = \left[t_{x}{}_{M_i}^W, \ t_{y}{}_{M_i}^W\right]^T$ and $\boldsymbol{v}_{M_i}^W = \left[v_{x}{}_{M_i}^W, \ v_{y}{}_{M_i}^W\right]^T$ are the position and velocity of the center of the circle in world frame, and r_i is the radius of the circle.

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

$$\check{\boldsymbol{u}}_R = \begin{bmatrix} \check{\boldsymbol{t}}_R^W \\ \hdashline \check{\bar{\boldsymbol{q}}}_R^W \end{bmatrix} \tag{10}$$

where $\check{\boldsymbol{t}}_R^W = \left[\check{t}_{x}{}_R^W, \ \check{t}_{y}{}_R^W\right]^T$ and $\check{\bar{\boldsymbol{q}}}_R^W = \left[\check{q}_{w}{}_R^W, \ \check{q}_{z}{}_R^W\right]^T$ are the position and the orientation (as a simplified quaternion) of the robot in world frame given by the localization component in a SE(2) space.

The reader must note that we use the information given by 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 circles provided by the circle detection component, as:

$$\tilde{\boldsymbol{z}}_{M_i} = \begin{bmatrix} \tilde{\boldsymbol{t}}_{M_i}^S \\ \hdashline \tilde{r}_i \end{bmatrix} \tag{11}$$

where $\tilde{\boldsymbol{t}}_{M_i}^S = \left[\tilde{t}_{x}{}_{M_i}^S, \ \tilde{t}_{y}{}_{M_i}^S\right]^T$ is the measured position of the center of the circle in the sensor frame, and $\tilde{r}_i$ is the measured radius of the circle.

4) Process model: The 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:

$$\boldsymbol{x}_R(k) = \check{\boldsymbol{u}}_R(k-1) \tag{12}$$

The map element process model is given by:

$$\boldsymbol{t}_{M_i}^W(k) = \boldsymbol{t}_{M_i}^W(k-1) + \Delta t \cdot \boldsymbol{v}_{M_i}^W(k-1) \tag{13}$$

$$\boldsymbol{v}_{M_i}^W(k) = \boldsymbol{v}_{M_i}^W(k-1) + \boldsymbol{n}_{f_v} \tag{14}$$

$$r_i(k) = r_i(k-1) \tag{15}$$

where Δt is the time difference between two prediction steps, and $\boldsymbol{n}_{f_v}$ is the noise of the robot process model associated with the velocity. The noise $\boldsymbol{n}_{f_v}$, is added to take into account the fact that the map elements are not always moving at a constant velocity.

5) 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:

$$\tilde{\boldsymbol{t}}^{S}_{M_i} = \left(\boldsymbol{R}^{R}_{S}\right)^{T} \cdot \left(\boldsymbol{R}^{W}_{R}\right)^{T} \cdot \left(\boldsymbol{t}^{W}_{M_i} - \boldsymbol{R}^{W}_{R} \cdot \boldsymbol{t}^{R}_{S} - \boldsymbol{t}^{W}_{R}\right) + \boldsymbol{n}_{h_t} \tag{16}$$

$$\tilde{r}_i = r_i + n_{h_r} \tag{17}$$

where $\boldsymbol{t}^{R}_{S}$ and $\boldsymbol{R}^{R}_{S}$ represents the calibrated pose (translation and rotation matrix) of the sensor in the robot frame; $\boldsymbol{R}^{W}_{R}$ is the rotation matrix of the attitude of the robot in world frame, calculated using the simplified quaternion $\bar{\boldsymbol{q}}^{W}_{R}$; and $\boldsymbol{n}_{h_t}$ and n_{h_r} are the measurement noises.

6) 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:

$$\boldsymbol{t}^{W}_{M_i} = \boldsymbol{R}^{W}_{R} \cdot \boldsymbol{R}^{R}_{S} \cdot \tilde{\boldsymbol{t}}^{S}_{M_i} + \boldsymbol{R}^{W}_{R} \cdot \boldsymbol{t}^{R}_{S} + \boldsymbol{t}^{W}_{R} \tag{18}$$

$$\boldsymbol{v}^{W}_{M_i} = \boldsymbol{0}_{2\times 1} + \boldsymbol{n}_{g_v} \tag{19}$$

$$r_i = \tilde{r}_i \tag{20}$$

where $\boldsymbol{n}_{g_v}$ is the mapping noise associated with the estimation of the velocity of the center of the circle in world frame, as it cannot be directly observable from the measurement.

B. Stages

1) Prediction stage: During the prediction stage, the state of the robot and the previously mapped circles are estimated. The nominal-state is estimated following the process model presented above, as:

$$\hat{\boldsymbol{x}}(k) = f(\hat{\boldsymbol{x}}(k-1), \check{\boldsymbol{u}}(k-1)) \tag{21}$$

The covariance of the error-state is estimated following:

$$\begin{aligned}\boldsymbol{P}_{\delta \boldsymbol{x}}(k|k-1) = {} & \boldsymbol{F}_{\delta \boldsymbol{x}} \cdot \boldsymbol{P}_{\delta \boldsymbol{x}}(k-1|k-1) \cdot \left(\boldsymbol{F}_{\delta \boldsymbol{x}}\right)^{T} \\ & + \boldsymbol{F}_{\delta \boldsymbol{u}} \cdot \boldsymbol{Q}_{\delta \boldsymbol{u}} \cdot \left(\boldsymbol{F}_{\delta \boldsymbol{u}}\right)^{T} \\ & + \boldsymbol{F}_{\delta \boldsymbol{n}} \cdot \boldsymbol{Q}_{\delta \boldsymbol{n}} \cdot \left(\boldsymbol{F}_{\delta \boldsymbol{n}}\right)^{T}\end{aligned} \tag{22}$$

where $\boldsymbol{F}_{\delta \boldsymbol{x}}$, $\boldsymbol{F}_{\delta \boldsymbol{u}}$, and $\boldsymbol{F}_{\delta \boldsymbol{n}}$ are the Jacobian matrices of the error-state process model, which are omitted for brevity; $\boldsymbol{Q}_{\delta \boldsymbol{u}}$ is the covariance matrix associated with the inputs, and $\boldsymbol{Q}_{\delta \boldsymbol{n}}$ is the covariance matrix associated with the process model.

2) Update stage: During the update stage, the state of the existing map elements is corrected using the measurements, and the new map elements are added to the estimated map. It requires several steps:

First of all, the values of the measurements are predicted using the measurement model and the las predicted state as:

$$\hat{\tilde{\boldsymbol{z}}}(k) = h\left(\hat{\boldsymbol{x}}(k)\right) \tag{23}$$

The predicted measurements need to be matched with the actual ones following a process called data association. The first step of this process consists of calculating the innovation of all the predicted measurements, j, with the actual measurements, i, as:

$$\boldsymbol{\nu}_{ij}(k) = \tilde{\boldsymbol{z}}_i(k) - \hat{\tilde{\boldsymbol{z}}}_j(k) \tag{24}$$

$$\begin{aligned}\boldsymbol{S}_{ij}(k) = {} & \boldsymbol{H}_{\delta \boldsymbol{x}} \cdot \boldsymbol{P}_{\delta \boldsymbol{x}}(k|k-1) \cdot \left(\boldsymbol{H}_{\delta \boldsymbol{x}}\right)^{T} \\ & + \boldsymbol{H}_{\delta \boldsymbol{n}} \cdot \boldsymbol{R}_{\delta \boldsymbol{n}} \cdot \left(\boldsymbol{H}_{\delta \boldsymbol{n}}\right)^{T}\end{aligned} \tag{25}$$

where $\boldsymbol{H}_{\delta \boldsymbol{x}}$ and $\boldsymbol{H}_{\delta \boldsymbol{n}}$ are the Jacobian matrices of the error-state measurement model, which are omitted for brevity; and $\boldsymbol{R}_{\delta \boldsymbol{n}}$ is the covariance matrix associated with the measurement noise. After that, the Mahalanobis distance is computed following:

$$d^2_{ij} = \left(\boldsymbol{\nu}_{ij}\right)^{T} \cdot \left(\boldsymbol{S}_{ij}\right)^{-1} \cdot \boldsymbol{\nu}_{ij} \tag{26}$$

We select as candidates to be associated together, the i and j elements with the smallest Mahalanobis distance for every actual measurement. The candidates are associated if their Mahalanobis distance is lower than a user-defined threshold.

Once the data association process has finished, we correct the state of the map elements that have been associated with an actual measurement. We calculate the Kalman gain as:

$$\boldsymbol{K} = \boldsymbol{P}_{\delta \boldsymbol{x}}(k|k-1) \cdot \left(\boldsymbol{H}_{\delta \boldsymbol{x}}\right)^{T} \cdot \left(\boldsymbol{S}_{ij}\right)^{-1} \tag{27}$$

And we update the nominal-state and the covariance of the error-state, following:

$$\delta \boldsymbol{x}(k|k) = \boldsymbol{K} \cdot \boldsymbol{\nu}_{ij}(k) \tag{28}$$

$$\boldsymbol{x}(k|k) = \boldsymbol{x}(k|k-1) \oplus \delta \boldsymbol{x}(k|k) \tag{29}$$

$$\boldsymbol{P}_{\delta \boldsymbol{x}}(k|k) = \left(\boldsymbol{I} - \boldsymbol{K} \cdot \boldsymbol{H}_{\delta \boldsymbol{x}}\right) \cdot \boldsymbol{P}_{\delta \boldsymbol{x}}(k|k-1) \tag{30}$$

where $\oplus$ is the operation that updates the nominal-state with the error-state, which in our case is simply a standard addition $+$, as we are not updating the robot state, and the map elements state does not include orientations.

Finally, the unassociated measurements are mapped, adding to the state new map elements as follows:

$$\hat{\boldsymbol{x}}'(k) = g\left(\hat{\boldsymbol{x}}(k), \tilde{\boldsymbol{z}}(k)\right) \tag{31}$$

$$\begin{aligned}\boldsymbol{P}'_{\delta \boldsymbol{x}}(k|k) = {} & \boldsymbol{G}_{\delta \boldsymbol{x}} \cdot \boldsymbol{P}_{\delta \boldsymbol{x}}(k|k) \cdot \left(\boldsymbol{G}_{\delta \boldsymbol{x}}\right)^{T} \\ & + \boldsymbol{G}_{\delta \boldsymbol{z}} \cdot \boldsymbol{R}_{\delta \boldsymbol{u}} \cdot \left(\boldsymbol{G}_{\delta \boldsymbol{z}}\right)^{T} \\ & + \boldsymbol{G}_{\delta \boldsymbol{n}} \cdot \boldsymbol{T}_{\delta \boldsymbol{n}} \cdot \left(\boldsymbol{G}_{\delta \boldsymbol{n}}\right)^{T}\end{aligned} \tag{32}$$

where $\boldsymbol{G}_{\delta \boldsymbol{x}}$, $\boldsymbol{G}_{\delta \boldsymbol{z}}$, and $\boldsymbol{G}_{\delta \boldsymbol{n}}$ are the Jacobian matrices of the error-state mapping model, which are omitted for brevity; $\boldsymbol{R}_{\delta \boldsymbol{u}}$ is the covariance matrix associated with the measurement; and $\boldsymbol{T}_{\delta \boldsymbol{n}}$ is the covariance matrix associated with the mapping model.

3) Map management stage: During the map management stage, the list of map elements is updated, deleting those elements with a high covariance, as their state is assumed to be outdated and therefore not useful anymore.

We use the eigendecomposition of the covariance matrix of the estimated state of every map element, $\boldsymbol{P}_{\delta \boldsymbol{x}_{M_i}}$, to calculate its principal values as:

$$\boldsymbol{P}_{\delta \boldsymbol{x}_{M_i}} = \boldsymbol{Q}_e \cdot \boldsymbol{\Sigma}_{\delta \boldsymbol{x}_{M_i}} \cdot \boldsymbol{Q}_e^{-1} \tag{33}$$

where $\boldsymbol{Q}_e$ is the matrix of the eigenvectors; and $\boldsymbol{\Sigma}_{\delta \boldsymbol{x}_{M_i}} = diag\left(\lambda_1, .., \lambda_5\right)$, is the diagonal matrix of the eigenvalues, λ_j, of the matrix $\boldsymbol{P}_{\delta \boldsymbol{x}_{M_i}}$.

We set a threshold for the values of every eigenvalue, and we delete the associated map element when any of its value is over this threshold.

VI. EVALUATION AND RESULTS

In this section, we evaluate the performance of the proposed semantic situation awareness of the environment.

A. *Evaluation methodology*

The validation of the proposed semantic situation awareness of the environment is done considering two different aspects.

On the one hand, in Sect. VI-C, we deeply evaluate the performance of the proposed system when it is facing an environment with perfect simulated cylinders.

On the other hand, in Sect. VI-D we evaluate the robustness of the proposed system in a real environment that is populated by objects without a perfect cylindrical shape.

B. *Experimental setup*

All the presented components of this work have been implemented in C++, except the machine learning based circle detector that has been implemented in Python. They all are executed on a Laptop DELL Latitude 5470 equipped with an Intel Core i7-6820HQ CPU, 8 GB of RAM, and no dedicated GPU, running Ubuntu 16.04. We use ROS [30] as middleware between all the components. We have implemented our CNN using Keras [31] running on top of TensorFlow [32].

Our aerial platform is a DJI Matrice 100[2] quadrotor, equipped with a DJI Manifold companion onboard computer.

We use 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 values. Its linear range goes from 0.1 m. to 30 m., although we have limited it to 10 m. We have set its measurement rate to 20 Hz.

For the simulation results presented in Sect. VI-C, we use the VRep simulator, using all the presented components with a Software-In-The-Loop (SITL) methodology. We have developed a VRep scene with our simulated aerial platform and LIDAR, and populated with several static as well as moving cylinders (see Fig. 6).

For the experimental results presented in Section VI-D, we use the real aerial robot flying inside our Laboratory (see Fig. 1). We limit 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. thanks to the range limiter presented in Sect. III-A. We use as the robot localization component the multi-sensor fusion state estimator presented in [33], feeding it with the pose measurements provided by an Optitrack motion capture system. A moving person and a quasi-cylindrical static object are used as the two existing objects of the environment.

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

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

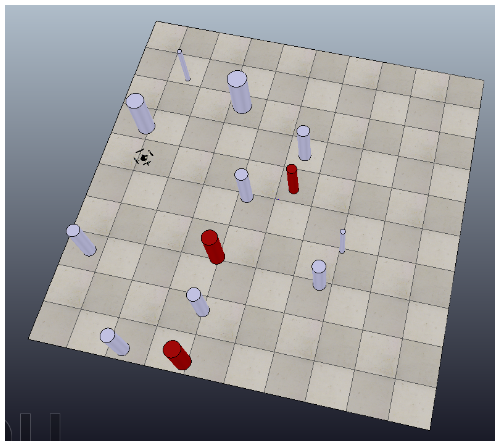


Fig. 6. Scene used in our VRep simulation. The gray cylinders are static objects, whereas the red ones are moving objects.

C. *Simulation results*

We have carried out an intensive quantitative and qualitative evaluation of the proposed situation awareness system using the simulator.

Our first set of experiments consisted of placing one single cylindrical moving object inside the range of the LIDAR. We have modified both the radius of the object and the trajectory it is following. We have compared the ground truth values provided by the simulator with the estimated values calculated by our proposed situation awareness system. We have computed the mean squared error (MSE), in m^2; and the maximum absolute error (MaAE), in m.

Fig. 7 shows the values of the position of the center in world coordinates and the radius of the single object of $r = 0.3$ m. following the second trajectory.

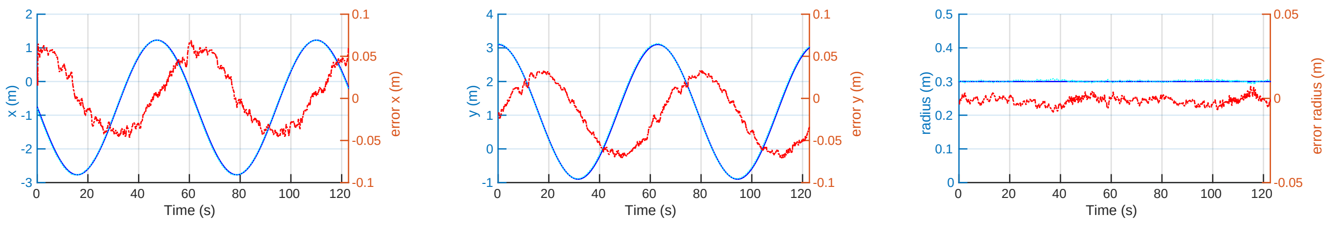
Tab. IV gathers the results of the simulations of one single object following the second trajectory for different values of its radius.

TABLE IV
SET OF EXPERIMENTS WITH A SINGLE OBJECT WITH DIFFERENT RADIUS FOLLOWING TRAJECTORY 2. MEAN SQUARED ERROR (MSE) IN M^2 AND MAXIMUM ABSOLUTE ERROR (MAAE) IN M.

	r		t_x		t_y	
Radius	MSE	MaAE	MSE	MaAE	MSE	MaAE
0.1 m.	8.7482e-06	0.0073	0.0014	0.0610	0.0011	0.0680
0.15 m	1.1542e-05	0.0097	0.0014	0.0638	0.0012	0.0715
0.3 m.	7.5769e-06	0.0081	0.0013	0.0691	0.0015	0.0711
0.45 m.	7.7660e-06	0.0075	0.0013	0.0735	0.0014	0.0714
0.6 m.	0.0019	0.0751	0.0054	0.1552	0.0012	0.0724

Tab. V shows the results of the simulations of a single object of $r = 0.3$ m, following different trajectories.

Analyzing this first set of experiments, we see that the MSE of the estimation of the radius of the object is always lower than $3.0 \cdot 10^{-5}$ m^2 and its MaAE is lower than 0.015 m. The MSE of the estimation of the position of the center



(a) t_x. (b) t_y. (c) r.

Fig. 7. Position of the center in world coordinates, (a) and (b), and radius, (c) of the single object of $r = 0.3$ m. of one of the simulations. This test corresponds to the second trajectory. Ground truth is plotted in solid blue, estimated in dotted cyan, and error in dashed red (scale on the right axis).

TABLE V
SET OF EXPERIMENTS WITH A SINGLE OBJECT OF $r = 0.3$ M. FOLLOWING DIFFERENT TRAJECTORIES. MEAN SQUARED ERROR (MSE) IN M^2 AND MAXIMUM ABSOLUTE ERROR (MAAE) IN M.

	r		t_x		t_y	
Trajectory	MSE	MaAE	MSE	MaAE	MSE	MaAE
1	1.3187e-05	0.0088	0.0012	0.0628	0.0016	0.0766
2	7.5769e-06	0.0081	0.0013	0.0691	0.0015	0.0711
3	9.2896e-06	0.0141	0.0015	0.0725	0.0013	0.0632
4	2.8435e-05	0.0099	0.0018	0.0703	0.0010	0.0588
5	1.8651e-05	0.0111	0.0014	0.0795	0.0016	0.0792

of the object in world coordinates is always lower than 0.002 m^2 and its MaAE is lower than 0.08 m.

The reader must note (last row of Tab. IV) that when we set the radius of the object to 0.6 m., the errors (both MSE and MaAE) grew significantly. This is due to the fact that we have trained our CNN with a dataset containing objects with a maximum radius of 0.5 m. Despite the expected performance drop, the system can still provide significant answers even when it is used beyond its specifications.

In the second set of experiments, we have placed multiple static and dynamic cylindrical objects in the simulated environment (see 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/cd0LXYoBOkw`. Fig. 8 displays some screenshots of the experiment. The static objects are represented with golden colored circles and the moving objects with blue ones. The estimated circles calculated by our proposed situation awareness system are displayed in red. As can be seen, the estimated circles are practically overlapped with the ground truth objects. The proposed situation awareness system is capable of effectively handling occlusions, deleting from the semantic map, the objects whose covariance has grown over the user-defined threshold since they have not been observed for a certain time. It is also capable to add to the semantic map the objects that are newly observed. The reader must note that the estimated position and radius of the occluded objects fluctuates due to the errors in the detection caused by the partial observation of the objects. Improving this behavior remains future work.

D. *Experimental results*

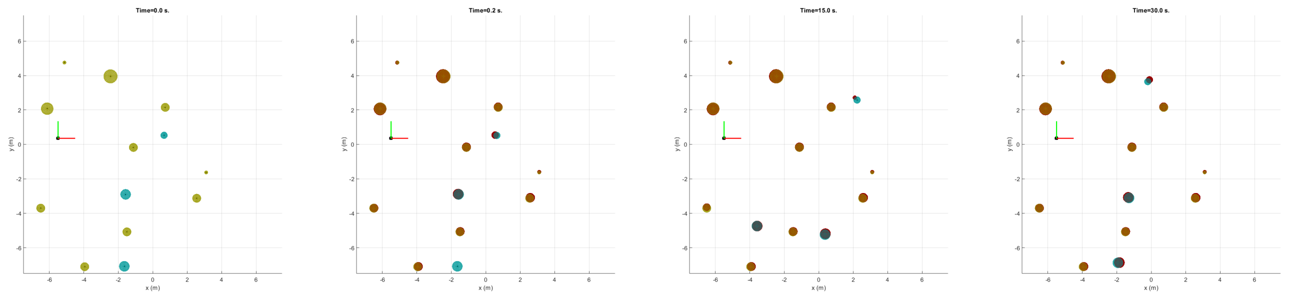
We qualitatively evaluate the performance of our proposed state estimator when facing a real environment where the objects do not have an exact cylindrical shape. A moving person and a quasi-cylindrical (i.e. a barrel) static object are used as the two existing objects of the environment. As it can be seen in Fig. 9, looking at the measurement of the LIDAR (white dots), can be easily perceived the shape of the person (right object) containing the two arms and the body, and therefore not having a circular shape. Despite this, the circle detector is able to estimate a circle (green circle) containing the person.

The whole experiment can be visualized in `https://youtu.be/7LFGTOVcCO4`. Fig. 10 displays some screenshots of the experiment. As can be concluded from the experiment, the proposed situation awareness system is capable of handling a real environment with non-cylindrical objects, as well as the previously mentioned occlusions. The reader must note that in some time instants, the person is represented with two or three circles instead of with only one, due to the fact that the arms and the body might be 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 worked in real-time executed in the aforementioned computer.

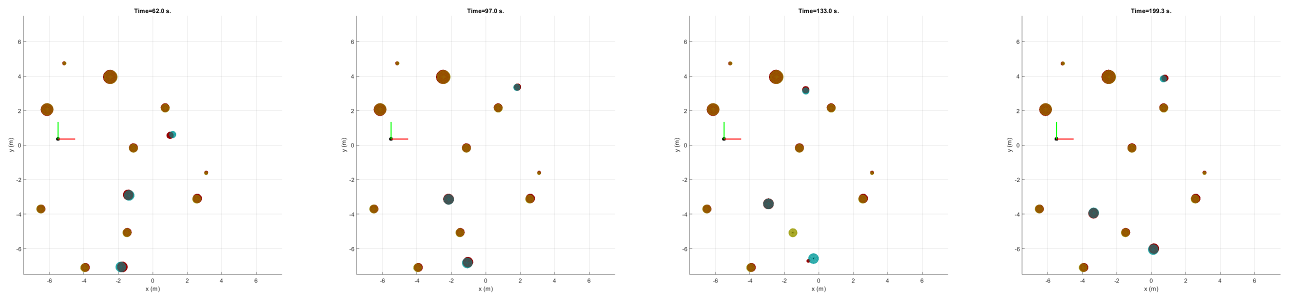
It is worth to highlight that ground truth information is not available in this experiment, due to the fact that a person has a deformable non-cylindrical and symmetrical shape, and therefore we have intentionally omitted it.

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. We have focused on the generation of a semantic map of the objects of the environment, assuming to have a precise robot localization as an input of our algorithm. The proposed semantic map consists of a list of circles represented by their radius, and the position and the velocity of their center in world coordinates. The proposed algorithm includes two components to preprocess the LIDAR measurements, followed by an object segmentation component to cluster the candidate objects present in the environment. A CNN has been designed and trained using an artificially generated dataset to compute the radius and



(a) Time 0 s. (b) Time 0.2 s. (c) Time 15 s. (d) Time 30 s.



(e) Time 62 s. (f) Time 97 s. (g) Time 133 s. (h) Time 199.3 s.

Fig. 8. Simulation experiment in VRep with multiple static (golden colored) and moving (blue colored) objects. The estimated circles are displayed in red. The whole experiment can be visualized in `https://youtu.be/cd0LXYoBOkw`

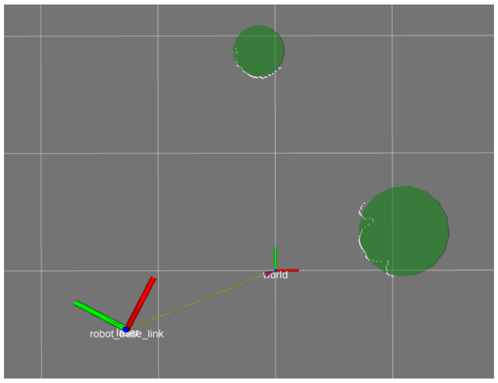


Fig. 9. Detail of the LIDAR measurement on a Laboratory experiment when perceiving a quasi-cylindrical object (top) and a person (right).

the position of the center of individual circles in sensor coordinates. Finally, an indirect-EKF is used to estimate the semantic map in world coordinates, including the velocity of the center of the circles in world coordinates.

We have evaluated the performance of our proposed situation awareness system by means of SITL simulations using VRep with one and multiple static and moving cylindrical objects in the scene, obtaining both quantitative and qualitative evaluations that support our proposed algorithm. In addition, we have carried out real laboratory experiments with real non-cylindrical static (i.e. a barrel) and moving (i.e. a person) objects, demonstrating that our proposed algorithm is capable to handle such real situations.

Our most immediate future work will consist of adding other geometric shapes more than circles (e.g. ellipses, rectangles, and lines) to improve the representation of the environment. We also plan to increase the performance of our algorithm under occlusions.Our long-term plans include the use of neural network techniques for object segmentation, together with one step object detectors (instead of running the object detector once per segmented object).

References

[1] M. Beul, N. Krombach, Y. Zhong, D. Droeschel, M. Nieuwenhuisen, and S. Behnke, “A high-performance mav for autonomous navigation in complex 3d environments,” in *2015 international conference on unmanned aircraft systems (ICUAS)*. IEEE, 2015, pp. 1241–1250.

[2] F. Azevedo, A. Oliveira, A. Dias, J. Almeida, M. Moreira, T. Santos, A. Ferreira, A. Martins, and E. Silva, “Collision avoidance for safe structure inspection with multirotor uav,” in *2017 European Conference on Mobile Robots (ECMR)*. IEEE, 2017, pp. 1–7.

[3] S. Ramasamy, R. Sabatini, A. Gardi, and J. Liu, “Lidar obstacle warning and avoidance system for unmanned aerial vehicle sense-and-avoid,” *Aerospace Science and Technology*, vol. 55, pp. 344–358, 2016.

[4] A. Al-Kaff, F. García, D. Martín, A. De La Escalera, and J. Armingol, “Obstacle detection and avoidance system based on monocular camera and size expansion algorithm for uavs,” *Sensors*, vol. 17, no. 5, p. 1061, 2017.

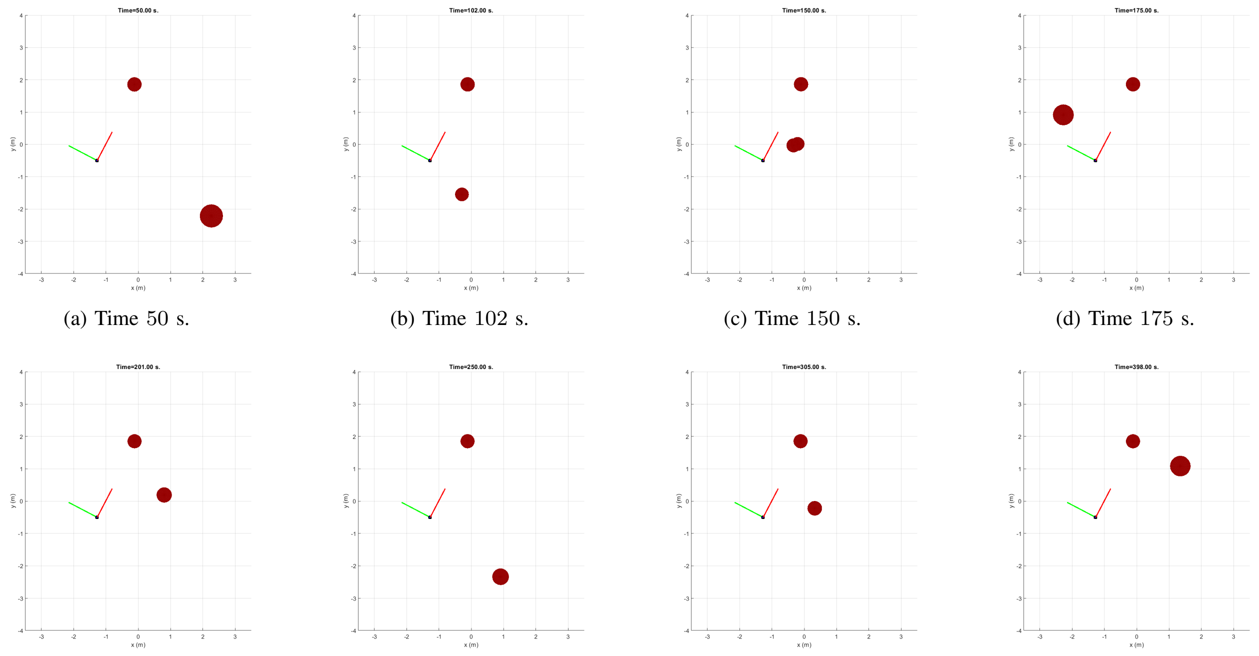
[5] S. Hrabar, “3d path planning and stereo-based obstacle avoidance for rotorcraft uavs,” in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 807–814.

[6] M. C. Santos, L. V. Santana, A. S. Brandao, and M. Sarcinelli-Filho, “Uav obstacle avoidance using rgb-d system,” in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2015, pp. 312–319.

[7] K. B. Ariyur, P. Lommel, and D. F. Enns, “Reactive inflight obstacle avoidance via radar feedback,” in *Proceedings of the 2005, American Control Conference, 2005.* IEEE, 2005, pp. 2978–2982.

[8] S. Scherer, S. Singh, L. Chamberlain, and S. Saripalli, “Flying fast and low among obstacles,” in *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, 2007, pp. 2023–2029.

[9] W. Fei, C. Jin-Qiang, C. Ben-Mei, and H. L. Tong, “A comprehensive uav indoor navigation system based on vision optical flow and laser fastslam,” *Acta Automatica Sinica*, vol. 39, no. 11, pp. 1889–1899, 2013.



(a) Time 50 s. (b) Time 102 s. (c) Time 150 s. (d) Time 175 s.

(e) Time 201 s. (f) Time 250 s. (g) Time 305 s. (h) Time 398 s.

Fig. 10. Laboratory experiment with a static quasi-cylindrical object and a moving person. The estimated circles are displayed in red. The whole experiment can be visualized in `https://youtu.be/7LFGTOVcCO4`

[10] N. Gageik, P. Benz, and S. Montenegro, “Obstacle detection and collision avoidance for a uav with complementary low-cost sensors,” *IEEE Access*, vol. 3, pp. 599–609, 2015.

[11] 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.

[12] 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

[13] M. Castillo-Lopez, S. A. Sajadi-Alamdari, J. L. Sanchez-Lopez, M. A. Olivares-Mendez, and H. Voos, “Model predictive control for aerial collision avoidance in dynamic environments,” in *2018 26th Mediterranean Conference on Control and Automation (MED)*, June 2018, pp. 1–6.

[14] C. Premebida, O. Ludwig, and U. Nunes, “Exploiting lidar-based features on pedestrian detection in urban scenarios,” in *2009 12th International IEEE Conference on Intelligent Transportation Systems*. IEEE, 2009, pp. 1–6.

[15] H. Peng, F. Long, and C. Ding, “Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy,” *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 8, pp. 1226–1238, 2005.

[16] S. Wender, M. Schoenherr, N. Kaempchen, and K. Dietmayer, “Classification of laserscanner measurements at intersection scenarios with automatic parameter optimization,” in *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.* IEEE, 2005, pp. 94–99.

[17] R. Zhao, M. Pang, and Y. Zhang, “Robust shape extraction for automatically segmenting raw lidar data of outdoor scenes,” *International Journal of Remote Sensing*, pp. 1–25, 2018.

[18] X. Zhou, Y. Wang, Q. Zhu, and Z. Miao, “Circular object detection in polar coordinates for 2d lidar data,” in *Chinese Conference on Pattern Recognition*. Springer, 2016, pp. 65–78.

[19] S. Tao, Q. Guo, S. Xu, Y. Su, Y. Li, and F. Wu, “A geometric method for wood-leaf separation using terrestrial and simulated lidar data,” *Photogrammetric Engineering & Remote Sensing*, vol. 81, no. 10, pp. 767–776, 2015.

[20] F. Tarsha-Kurdi, T. Landes, and P. Grussenmeyer, “Extended ransac algorithm for automatic detection of building roof planes from lidar data,” *The photogrammetric journal of Finland*, vol. 21, no. 1, pp. 97–109, 2007.

[21] X. Zhang, W. Xu, C. Dong, and J. M. Dolan, “Efficient l-shape fitting for vehicle detection using laser scanners,” in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 54–59.

[22] A. Nurunnabi, Y. Sadahiro, and D. F. Laefer, “Robust statistical approaches for circle fitting in laser scanning three-dimensional point cloud data,” *Pattern Recognition*, vol. 81, pp. 417–431, 2018.

[23] A. Janowski, “The circle object detection with the use of msplit estimation,” in *E3S Web of Conferences*, vol. 26. EDP Sciences, 2018, p. 00014.

[24] 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.

[25] E. Shang, X. An, J. Li, and H. He, “A novel setup method of 3d lidar for negative obstacle detection in field environment,” in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2014, pp. 1436–1441.

[26] S. M. Mavaei and H. R. Imanzadeh, “Line segmentation and slam for rescue robots in unknown environments 1,” 2012.

[27] 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.

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

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

[30] 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.

[31] “Keras web,” https://keras.io/.

[32] “Tensorflow web,” https://www.tensorflow.org/.

[33] 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.