Noname manuscript No.
(will be inserted by the editor)

A Fully-Autonomous Aerial Robot for Search and Rescue Applications in Indoor Environments Using Learning-Based Techniques

Carlos Sampedro · Alejandro Rodriguez-Ramos · Hriday Bavle · Adrian Carrio · Paloma de la Puente · Pascual Campoy

Received: date / Accepted: date

Abstract Search and Rescue (SAR) missions represent an important challenge in the robotics research field as they usually involve exceedingly variable-nature scenarios which require a high-level of autonomy and versatile decision-making capabilities in order to be solved. This challenge becomes even more relevant in the case of aerial robotic platforms owing to their limited payload and computational capabilities. In this paper, we present a fully-autonomous aerial robotic solution, for executing complex SAR missions in unstructured indoor environments. The proposed system is based on the combination of a complete hardware configuration and a flexible system architecture which permits the execution of high-level missions in a fully unsupervised manner. In order to obtain flexible and versatile behaviors from the proposed aerial robot, several learning-based capabilities have been integrated for target recognition and interaction. The target recognition capability includes a supervised learning classifier based on a computationally-efficient Convolutional Neural Network (CNN) model trained for target/background classification, while the target interaction capability introduces a novel Image-Based Visual Servoing (IBVS) algorithm which integrates a recent Deep Reinforcement Learning method named Deep Deterministic Policy Gradients (DDPG). In order to train the UAV for performing IBVS tasks, our own reinforcement learning framework has been developed, which integrates a deep reinforcement learning agent (e.g. DDPG) with a Gazebo-based simulator for aerial robotics. The proposed system has been validated in a wide range of simulation flights, using Gazebo and PX4 Software-In-The-Loop, and real flights in unstructured indoor environments, demonstrating the versatility of the proposed system in complex SAR missions.

Keywords Autonomous Robots · Search and Rescue · Supervised Learning · Reinforcement Learning · Deep Learning · Image-Based Visual Servoing

Carlos Sampedro
Technical University of Madrid
Tel.: +34 913 36 30 60
E-mail: carlos.sampedro@upm.com

1 Introduction

Research interests in Unmanned Aerial Vehicles (UAVs) have increased considerably in the last decade, especially regarding multirotor aerial robots, owing to their good maneuverability for solving complex missions. In this context, the research community has mostly focused on outdoor applications where the availability of Global Navigation Satellite System (GNSS) information facilitates the autonomous navigation of the UAV. Furthermore, in this kind of outdoor scenarios there are usually no disturbances in the flight-path of the UAV related to ground obstacles. However, nowadays there is an increasing demand for solving complex indoor missions, such as surveillance, search and rescue, package delivery in large indoor facilities, etc. In this kind of applications, the absence of GNSS makes necessary the usage of the exteroceptive sensors mounted onboard the UAV for navigation and decision-making purposes.

The development of fully-autonomous missions in such indoor environments requires research on Micro Aerial Vehicles (MAVs) systems, which pose a great challenge on the system design, due to their limited payload and computational capabilities. These limitations impose the necessity to develop efficient and lightweight algorithms in order to find a proper balance between mission performance and computational resources available onboard the aerial robot. This balance becomes especially critical when particularly complex tasks have to be performed by the aerial robot without human supervision (e.g. target recognition and interaction in search and rescue missions) as more complex decision-making algorithms are required. In this paper, we focus our efforts on the development of an autonomous aerial robot capable of performing such high-level missions in an unsupervised manner (i.e. without human intervention) with a special interest in the versatility and ease of adaptation of the algorithms developed for object recognition and interaction.

Regarding the object recognition problem, classic computer vision algorithms are generally very dependent on the conditions of the environment where they operate (e.g. lighting conditions, variety of backgrounds, presence of clutter, etc.) which implies the need to readjust the parameters of the algorithm for each new environment in order to obtain a precise detection of the object and remove possible false positives in the image plane. Furthermore, this adjusting procedure can become an onerous task as it is usually carried out by trial and error tests. The use of machine learning techniques, when trained on meaningful datasets, allows overcoming these limitations, providing more versatile solutions which can be executed in a wide range of environments. However, some of the recent machine learning models for object recognition [21, 29, 38] consist of heavy models with a considerable amount of parameters that have not been designed for operating onboard a robotic platform with hard computational constraints such as a UAV.

With respect to the object interaction problem, in this paper we characterize these tasks by means of IBVS methods which allow the interaction with targets in a wide variety of SAR missions. Within these missions, operation in SAR disaster scenarios has an special interest as it usually involves the interaction with the detected target by means of delivering the required items such as medicines, food, etc [13], where IBVS techniques can provide versatile and lightweight solutions. In this direction, classical IBVS methods usually require a tedious tuning stage of their parameters when changing to different operating conditions. This fact can

eventually appear in SAR scenarios where different aerial robotic platforms may be used depending on the environment. Furthermore, classic IBVS methods can suffer from convergence and stability problems if the task is performed in a zone away from the operating point [9, 11].

Inspired by the aforementioned limitations, in this paper we propose a fully-autonomous UAV featured with learning-based techniques which can provide flexible and versatile solutions to indoor SAR missions. The main contributions of the proposed system are summarized here: i) A custom UAV has been built featured with a flexible system architecture which permits the efficient coordination of planning, situation awareness, perception and execution systems for solving complex SAR missions. ii) This flexibility has allowed the integration of learning-based techniques for object recognition and object interaction. Concretely, several supervised learning classifiers, including computationally-efficient CNN models, have been trained and evaluated for target/background classification. In addition, a novel Image-Based Visual Servoing (IBVS) algorithm based on Deep Deterministic Policy Gradients (DDPG) [28] has been implemented and validated for solving IBVS tasks, comparing its performance with classic IBVS techniques. iii) An extensive evaluation of the previous learning-based components and the whole system in cluttered indoor SAR scenarios has been conducted both in simulated and real flights.

In order to obtain a reliable testbed for experimentation in SAR missions, in this paper we have adopted as the main use case the missions proposed for the 2016 International Micro Air Vehicle Competition[1] (IMAV), where several SAR problems had to be addressed. In the IMAV 2016 indoors competition, challenging high-level missions were designed, ranging from autonomous building entering and exiting, indoor exploration of unknown scenarios, object recognition, etc. In addition, UAVs were required to perform object interaction tasks, such as grasping a cylindrical item and releasing it into a cylindrical bucket, with the option of pre-loading the items previous to the takeoff maneuver. The latter scenario has been extensively studied in this work in order to provide our previous system [41] with more high-level functionalities based on learning-based techniques.

The remainder of this paper is organized as follows: Section 2 presents the related work; Section 3 describes the hardware configuration adopted in our aerial robotic platform. The system architecture is explained in Section 4. Section 5 presents the experiments performed in simulated and real scenarios, with their respective results, and finally, Section 6 concludes the paper, and points out future research directions.

2 Related Work

In the following paragraphs, some of the most relevant solutions for the autonomous operation of UAVs in SAR missions are covered in chronological order [1, 5, 13, 15, 39, 44, 46, 47]. We also refer to articles in which the delivery of specific items from UAVs in emergency situations has been studied [18, 20, 48]. Right after, we cover other relevant developments aimed towards the execution of autonomous UAV missions outside the field of SAR applications [2, 3, 22, 51]. We conclude by

[1] IMAV 2016 official website: http://www.imavs.org/2016/

refering to other works applying vision-based deep reinforcement learning to UAV navigation [37, 40] as well as other relevant uses of deep reinforcement learning in visual control tasks [27, 49, 50].

There is an increasing number of recent studies aiming at UAVs as a potentially useful complement to SAR applications. Early developments in high-level artificial intelligence applied to aerial robotics were introduced in Doherty et al. [13]. In particular, UAV autonomous missions were implemented for the SAR of injured civilians, with robots being able to scan designated areas, trying to identify injured civilians and attempting to deliver medical and other supplies to identified victims in realistic urban scenarios. The specific techniques from this work, used to detect humans at a high frame rate onboard an autonomous UAV, were described in detail in [39]. These techniques were applied in a real-world outdoor environment using visible and thermal infrared cameras. In their work, detected human positions were geolocated and a map of points of interest was built. The resulting map is proposed to plan medical supply delivery during a disaster relief effort.

In the context of SAR technology developments, the UAV ChallengeOutback Rescue has been established as an important international competition held annually in Queensland, Australia. Since 2011, participants have been required to perform UAV SAR missions, which typically have involved executing autonomous take off, navigation for aerial search, and landing maneuvers. These exercises had associated image processing and control tasks needed to identify and deliver an emergency medical package to a mannequin simulating a lost person, placed in a 4 km $\times$ 6 km area. The UAV developed by the Missouri University of Science and Technology, who finished in the second place in the 2008 edition, published a peer-reviewed article describing their work [15]. They used a standard hobby fixed-wing airframe, modified for autonomous flight. Their UAV featured GPS-based navigation, ground image acquisition, and payload delivery, all implemented in a low-cost platform.

The work by Tomic et al.[47] introduced a modular and extensible software and hardware framework designed for the autonomous execution of SAR missions using aerial robots, which was successfully tested on a quadrotor platform. However, while using multiple sensors (four cameras and a laser scanner) the proposed system did not feature any collision avoidance capabilities.

In 2015, a pilot study was conducted by Abrahamsen [1] to assess the concept and feasibility of using a remotely piloted aircraft (RPA) system to support remote sensing in simulated major incident exercises. A custom-made, remotely controlled UAV with vertical take-off and landing was equipped with visible and thermal infrared cameras, a laser beam, a mechanical gripper arm and an avalanche transceiver. Successful missions were executed for five simulated exercises: a mass casualty traffic accident, a mountain rescue, an avalanche with buried victims, a fisherman through thin ice and searching for casualties in the dark. These missions proved UAVs to be suitable for carrying small payloads as well as useful tools to support situation assessment and information exchange at major incident scenarios.

In [44], Scherer et al. tested another interesting modular architecture of a UAV system for SAR missions in an outdoor environment. The objective of the mission was to detect a ground target by means of color, text or shape, and to provide a live aerial video stream for remote monitoring. Their proposal consisted of a swarm of multicopters coordinated to operate as a communications relay using a distributed

control system. The system was implemented using the Robot Operating System (ROS) and was capable of providing a real-time video stream from a UAV to one or more base stations using a wireless communication infrastructure. The proposed system supported a heterogeneous set of UAVs and image sensors and allowed the operator to select different levels of autonomy.

In the study published by Sun et al., a camera-based target detection and positioning system was developed and integrated into a fully autonomous fixed-wing UAV [46]. The system was capable of on board and real-time target identification, post-target identification and localization, and aerial image collection for further mapping applications. Its performance was assessed using several simulated SAR missions, demonstrating its reliability and efficiency.

Deep learning was applied for supporting UAV SAR operations in [5]. In this work, a sequence of images of avalanche debris captured by a UAV was processed with a pretrained CNN model to extract discriminative features. A trained linear Support Vector Machine (SVM) was integrated at the output of the CNN to detect objects of interest. Moreover, they introduced a preprocessing method to increase the detection rate and a postprocessing method based on a Hidden Markov Model to improve the prediction performance of the classifier. Experimental results conducted on two different datasets at different levels of resolution showed that the detection performance increased when incrementing the resolution, at the cost of raising the computation time.

Developments for drone delivery of emergency items in search and rescue missions have as well been analysed in the literature. Examples include: drug shipments [20], delivery of defibrillators [18] and life rings [48].

Other recent developments focused on providing UAVs with high levels of autonomy outside the field of application of SAR missions are discussed next. In the work of Bacharach et al. [3], a quadrotor helicopter equipped with a laser rangefinder, was designed and implemented to autonomously explore and map unstructured and unknown indoor environments. The paper highlighted the difficulties of applying to UAVs algorithms that were originally developed for Unmanned Ground Robots (UGVs). Interesting solutions were described in this work, such as: a multilevel sensing and control hierarchy, a high-speed laser scan-matching algorithm, an Extended Kalman Filter (EKF) for data fusion, a high-level SLAM implementation, and an exploration planner. The manuscript showed experimental results demonstrating the helicopter's ability to navigate accurately and autonomously in unknown environments. Algorithms originally conceived for UGVs were also exploited by Grzonka et al.[22] to increase UAV autonomy. In this case, they proposed a general navigation system that enabled a small-sized quadrotor platform to autonomously operate in indoor environments. A similar work was published by Achtelik et al.[2], which presented a software architecture providing a quadrotor helicopter with the capabilities to autonomously navigate, explore and locate objects of interest in unknown, unstructured indoor environments.

Results specific to autonomous navigation in indoor corridors were presented by Zingg et al.[51]. In their approach for wall collision avoidance, a depth map based on optical flow from images captured by an onboard omnidirectional fisheye camera was used. Inertial Measurement Unit (IMU) data was also used for compensating rotational effects of the optical flow.

Several implementations of visual control for UAVs can be found in the literature, but very few presenting it as one capability among several other ones in

the context of an autonomous mission. Some of the aforementioned developments make use of computer vision geometry to determine navigation waypoints, but only [47] used visual information to provide control feedback in real time.

There are as well very few developments exploiting vision-based deep reinforcement learning for UAV navigation. In particular, Sadeghi et al. [40] introduced the CAD^2RL learning method, which allows collision-free navigation in a real indoor environment using synthetic data from 3D CAD models as the only training data. Another example is the work by Polvara et al.[37], who made use of Deep Q-Networks for the autonomous landing of a quadrotor.

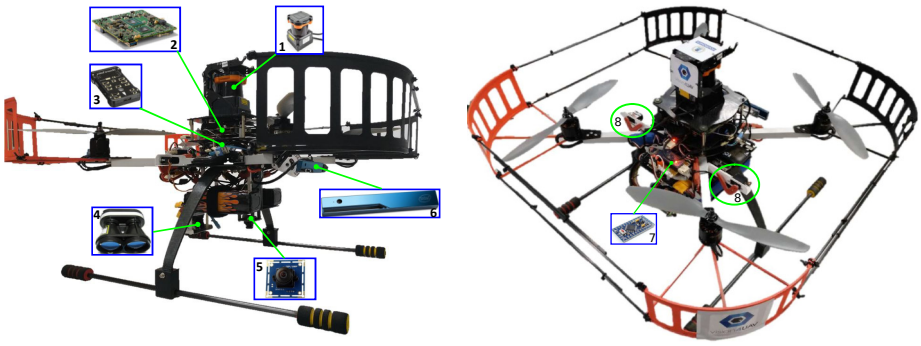
Other interesting applications of deep reinforcement learning for solving visual control tasks outside the field of aerial robotics are mentioned next. Lee et al.[27] proposed to perform a visual servoing task by extracting deep features instead of using pixels or keypoints. The best features to solve the task were then selected using a Q-iteration algorithm. Also Zhang et al.[49] proposed to use vision-based deep reinforcement learning for controlling the motion of a three-joint robot manipulator. Finally, Zhu et al.[50] presented an efficient algorithm for visual navigation in indoor scenes using deep reinforcement learning. The algorithm was trained using high-quality 3D scenarios allowing for physical interaction with the objects in the scene.

In contrast to all the aforementioned developments, our work proposes a fully-autonomous UAV that is not only capable of autonomously navigating in indoor cluttered environments with situational awareness, but can also interact with static and moving targets, which can be automatically detected and followed to precisely deliver items. Furthermore, while much of the discussed literature focuses on accomplishing specific tasks only, and many developments are evaluated in computer simulations only, the solution proposed here focuses on complex missions involving multiple heterogeneous tasks and has been evaluated in detail in both simulations and real flights.

3 Hardware Configuration

Taking into account the IMAV 2016 requirements, where the minimum distance of the passages within the indoor environment was 1 m wide, it was necessary to build a custom UAV considerably smaller in size with the adequate capacities for carrying the sensors and actuators required for localization, navigation, and object recognition and interaction. In addition, appropriate electronic components and holding devices have been designed and integrated for object interaction tasks (see Fig. 1). For this purpose, two small curved hooks have been integrated into the UAV framework (see Fig. 1b). These hooks are controlled by a servo motor which is actuated when a signal of *target locked* is commanded in order to release the preloaded items onboard the UAV.

Based on the aforementioned constraints, a custom UAV has been designed and built with a total takeoff weight of 3.2 kg, a maximum payload capacity of 1 kg and a maximum flight time of 12 min. The onboard computer consists of an Intel NUC6i5SYK having a 2.9 GHz Intel Core i5-6260U CPU. The avionics of the UAV is managed by a Pixhawk [33] autopilot, which integrates an Inertial Measurement Unit (IMU), a barometer and a magnetometer. The exteroceptive sensors mounted onboard consist of a Hokuyo laser range finder UTM-30 LX with a horizontal field



(a) (b)

Fig. 1: Proposed aerial robot, showing 1) Hokuyo laser range finder UTM-30 LX. 2) Intel NUC6i5SYK computer. 3) Pixhawk autopilot. 4) Lightware SF10/A altimeter. 5) Fisheye bottom camera. 6) Intel realsense R200 camera. 7) Arduino pro-mini. 8) Designed hooks for holding items.

of view of 270° and an angular resolution of 0.25° with a maximum range of 30 m, an Intel Realsense R200 camera, a standard RGB 180° fisheye-lens bottom-looking camera and a Lightware altimeter SF10/A with a maximum range of 25 m (see Fig. 1a). The communication between the autopilot, proprioceptive, exteroceptive sensors and the onboard computer is performed over USB connections.

4 System Architecture

The system proposed in this paper has been built on top of our software architecture named Aerostack [43]. In this architecture, the components are organized in different layers of abstraction (see. Fig. 2) which provide high modularity and flexibility. This modularity has permitted the easy integration of the proposed learning-based components which are the main contribution of the system proposed in this work.

4.1 PLANNING SYSTEM

4.1.1 Mission Planner

The mission planner subsystem is composed of two main components, named Global Mission Planner (GMP) and Agent Mission Planner (AMP). In the following paragraphs a detailed description of both components is provided.

1. Global Mission Planner (GMP). In this component resides the higher level of intelligence in terms of mission planning. The design of the GMP allows the mission generation for a single agent or a swarm of agents. In the latter case, the GMP is able to distribute the mission between the agents in the swarm in

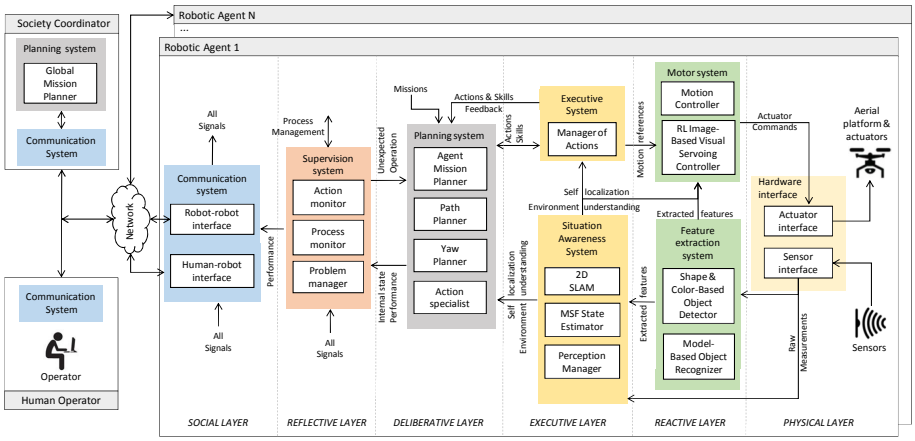


Fig. 2: System architecture. All the components developed in this work have been integrated into the different layers of the Aerostack framework.

order to optimize the accomplishment of such mission in terms of the explored area [42].

The main responsibility of the GMP is focused on the mission generation. For this purpose, its flexible design allows the specification of the mission in two different operating modes: manual and automatic. In the manual mission generation mode, the definition of the mission is performed using an XML-based language, in which the human operator can define the mission task by task. In the automatic mission generation mode, the GMP interprets a high-level mission command provided by a human operator (e.g. *find a target*, *explore*, etc.). Additional inputs in this mode comprise the dimensions of the area to be explored as well as the number of mission points. Using these inputs, the GMP is able to automatically generate mission points by applying a *K-means* clustering over points randomly distributed over the area to be explored. For a detailed explanation of this functionality, we refer the reader to [42]. Once the global mission is prepared, it is distributed through the agent or agents in the swarm and sent to the AMP.

One important functionality implemented in the GMP is its capability of concatenating several missions [41]. Based on this, the GMP has an active *list of missions* per agent. Once the last mission in the list is accomplished, the GMP is able to recover the previous one. This functionality acquires an utmost importance in SAR missions, where a new mission for object interaction has to be generated when the target is recognized, and the current exploration mission gets queued up. Once the UAV finishes the interaction with the target, the previous exploration mission is recovered in its corresponding state (i.e. current task). This behavior provides the system with the ability to perform complex high-level missions, such as the ones presented in Section 5.

2. Agent Mission Planner (AMP). This component is located at the agent's level and is responsible for scheduling task by task the received mission. For this purpose, the AMP acts as an interface between the GMP and the rest of components in the architecture.

Since the area to be explored is unknown *a priori*, the mission points generated by the GMP can fall within an obstacle. In order to address this problem, the AMP is capable of generating *safety points* when the current mission point falls within an obstacle. In order to generate a *safety point*, the AMP implements an iterative method in which random points lying on a *safety circumference* of predefined radius are generated. After several iterations of the algorithm, and if no *safety point* has been obtained in the current circumference, its radius is incremented and the iterative method continues. In this iterative procedure, the AMP communicates with the Path Planner in each iteration, until an obstacle-free point is obtained.

4.1.2 Path Planner

The path planner component relies both on the use of a precise Lidar sensor and a robust localization and mapping algorithm. The path planning algorithm utilized in the proposed architecture is based on an existent Robot Operating System (ROS) navigation package [32], which was originally designed for differential-drive and holonomic-wheels robots, and has been adapted in this work to the Aerostack architecture, enabling its operation with multirotor UAVs. Furthermore, the original 2D functionality of the mentioned ROS planner package has been extended in order to provide 3D navigation capabilities by adding the remaining altitude coordinate as a constant value (by default) to each intermediate path point. However, the AMP can dynamically modify its value within the execution of a mission in order to fulfill specific requirements relative to the current environment.

The Path Planner component requires a 2D occupancy grid map (see Section 4.2.1) as well as a mission point generated by the AMP (in world frame of reference) for its normal operation. The 2D occupancy grid map is subsequently translated into a 2D cost map in which cost values are propagated out of occupied cells based on an inflation radius parameter. A detailed explanation of the algorithm and its components can be reviewed at [32].

4.1.3 Yaw Planner

The yaw planner is in charge of associating a yaw angle to each point in a 2D path, based on the AMP directives. Taking into consideration normal mission conditions, a specific policy has been defined:

- Middle waypoint: Orientation is set to a constant value of $[0, \pi/2, \pi, 3\pi/2]$, according to the direction of navigation at each time step.
- Last waypoint: Orientation is derived from the AMP directives.

Following this policy, a UAV is considered to maximize the area covered by both the Lidar and image sensor's field of view, in order to plan throughout the optimum path and to avoid blind zones which can lead to a collision.

4.2 SITUATION AWARENESS SYSTEM

4.2.1 2D Localization and Mapping

In this work, localization and mapping capabilities have been integrated by means of a state-of-the-art 2D-SLAM algorithm [24], which has been extensively tested for ground robots in Urban Search and Rescue (USAR) missions. In this algorithm, a 2D-SLAM subsystem based on Lidar information and a 3D-Navigation subsystem based on IMU measurements are combined in order to provide a stable scan matching together with a reliable 3D pose estimation. The latter is only achieved if a source of altitude information is also provided. The mapping and scan matching stability are guaranteed by using a fast approximation of map gradients and a multi-resolution grid map representation for mitigating local minimum problems. Finally, the map representation is encoded into a 2D occupancy grid map, including *occupied*, *non-occupied* and *non-explored* cell types. The 2D-SLAM algorithm can perform properly without odometry information as well as on platforms that exhibit roll/pitch motion, but the publicly available implementation only provides a 2D estimate (including rotation around the vertical axis). An example of a 2D occupancy grid map of a real indoor scenario can be found in Fig. 16a.

4.2.2 Multi-sensor Fusion State Estimation

The objective of this component is to provide a full 6 d.o.f pose and the respective velocities of the UAV in cluttered indoor environments, enabling navigation using the Path Planner component described in Section. 4.1.2. In this direction, our proposed architecture integrates two separate state estimator components which can be combined in order to provide a higher level situation awareness functionality or be used separately depending on the requirements of the mission.

Flight Altitude State Estimator: The 2D SLAM algorithm explained in Section 4.2.1, provides a 2D map of the environment, enabling obstacle detection and avoidance in a 2D plane at the given flight altitude of the UAV. Sensors such as laser altimeters or similar range sensors can be used in order to estimate the altitude of the UAV while providing some knowledge about obstacles located bellow the flying robot. However, these sensors can cause errors in the flight altitude measurements when flying above ground obstacles, as the measurements get referred to them instead of the ground surface. In order to accurately estimate the flight altitude of the UAV in the presence of several ground obstacles, we propose an EKF-based algorithm which is able to estimate the flight altitude of the UAV as well as the elevation of the ground obstacles [4]. This is achieved by fusing the measurements coming from the IMU, the barometer, and the laser range altimeter sensor.

The proposed state estimator considers a state vector $\boldsymbol{x} \in \mathbb{R}^{10}$ based on the combination of four main components: $\boldsymbol{x_R}$, $\boldsymbol{x_G}$, $\boldsymbol{x_I}$, and $\boldsymbol{x_B}$, which represent the state of the robot, ground object, IMU sensor and barometer sensor respectively. The corresponding state of the aerial robot is defined by $\boldsymbol{x_R} = \begin{pmatrix} \boldsymbol{\Omega}_{xy}^T & \boldsymbol{\omega}_{xy}^T & t_{z_R} & v_{z_R} & a_{z_R} \end{pmatrix}$, where $\boldsymbol{\Omega}_{xy}^T = (\phi, \theta)^T$ are the roll and pitch Euler angles, $\boldsymbol{\omega}_{xy}^T = (\omega_x, \omega_y)^T$ represent the x and y angular velocities of the aerial robot in the UAV frame, and t_{z_R}, v_{z_R} and a_{z_R} are the vertical coordinates of

the position, velocity and acceleration of the aerial robot in the world frame of reference. Assuming that the robot changes its vertical acceleration and angular velocity slowly, we adopt a constant vertical acceleration and constant angular velocity as process model. The ground object state is defined by $\boldsymbol{x_G} = t_{z_G}$, where t_{z_G} is the altitude of the ground object in the world frame. Obstacles are set so that they always have positive altitude with respect to the XY (ground) plane. Finally, the IMU and the barometer sensors contribute to the state with their corresponding biases, thus $\boldsymbol{x_I} = b_{a_z}$ and $\boldsymbol{x_B} = b_{b_z}$, where b_{a_z} and b_{b_z} are the biases in the vertical acceleration and flight altitude measurements respectively.

Robot State Estimator: The robot state estimator is able to combine the measurements from the 2D SLAM (see Section. 4.2.1) with the flight altitude estimator (see Section. 4.2.2) or the IMU sensor in order to provide complete pose and velocity estimates of the UAV in the world frame of reference.

The robot state estimator [34] is a standard ROS package which implements an EKF-based estimator with state vector $\boldsymbol{x_R} \in \mathbb{R}^{12}$, being $\boldsymbol{x_R} = \begin{pmatrix}\boldsymbol{\Omega}^T & \mathbf{p}^T & \mathbf{v}^T\end{pmatrix}$, where $\boldsymbol{\Omega}^T = (\phi, \theta, \psi)^T$ are the roll, pitch and yaw Euler angles, $\mathbf{p}^T = (p_x, p_y, p_z)^T$ represents the position of the robot, and $\mathbf{v}^T = (v_x, v_y, v_z, \omega_x, \omega_y, \omega_z)$ is the vector containing the linear and angular velocities of the aerial robot. It includes a non-linear measurement model able to fuse any robot pose or velocity measurements, provided by any number of sensors. In contrast to the Flight Altitude State Estimator this EKF model does not incorporate in its state vector $\boldsymbol{x_R}$ any biases present in the sensor measurements.

Our system integrates the previously described components for multi-sensor state estimation in a versatile manner, so they can be employed in different configurations depending on the selected hardware and mission requirements at hand. Since the Flight Altitude State Estimator incorporates the height of ground obstacles and the sensors' bias into the state, it provides enhanced robustness as compared to the Robot State Estimator.

4.2.3 Perception Manager

The Perception Manager component is in charge of managing and centralizing the perception events that can occur during the execution of a mission (e.g. object *recognized*, *picked* and *released*).

In order to obtain a proper management of the perception events, this component integrates the information regarding the current situation of the states of the different objects which the UAV can interact with, together with its internal state (e.g. *Exploring*, *Going For Picking Item*, *Going For Releasing Item*). An example of the initial configuration of such states for a SAR mission applied to the use case of IMAV 2016 is provided in Table 1. When a perception event is detected by the Perception Manager, (e.g. target recognized) the current state of the objects as well as its internal state are evaluated and updated. Based on this evaluation, the Perception Manager can request a *mission adaptation* event to the GMP. As an example, and taking the initial configuration presented in Table 1, if a *bucket* object is recognized in the current time instant, and since both corresponding *items* are already picked up, the Perception Manager will generate a *mission adaptation* event consisting of *Going For Releasing Item*.

Table 1: Example of the initial state of the objects in a Search and Rescue Mission, where 0/1 represent a false/true statement

State / Object	Recognized	Picked	Released
Item A	0	1	0
Bucket A	0	0	0
Item B	0	1	0
Bucket B	0	0	0

Furthermore, another important functionality implemented in the Perception Manager consists in mapping the objects of interest (e.g. targets) in a SAR mission. For obtaining an adequate mapping of the objects that are being recognized during the execution of the mission, the Perception Manager provides a higher level of understanding over the recognition events coming from the Model-Based Object Recognizer component (see Section 4.3.2). For this purpose, the Perception Manager is responsible for referring the relative pose of the recognized objects in the frame of reference of the corresponding device (e.g. front camera) to the *world* frame of reference. This transformation is computed using Eq. 1. In addition, In order to provide a robust final pose of the recognized objects, the Perception Manager has an internal buffer for storing the pose of the objects being recognized over time. This buffer is utilized by a filter in which the poses away from the mean are removed in an iterative process.

$$ {}^{W}\boldsymbol{T}_{O} = {}^{W}\boldsymbol{T}_{U} \cdot {}^{U}\boldsymbol{T}_{C} \cdot {}^{C}\boldsymbol{T}_{O} \tag{1} $$

where $\boldsymbol{T}$ stands for a 4×4 homogeneous matrix transformation. ${}^{W}\boldsymbol{T}_{O}$ computes the global transformation from world to object frame of reference. ${}^{W}\boldsymbol{T}_{U}$ is the transformation from world to UAV frame of reference, which is given by the Robot State Estimator component. ${}^{U}\boldsymbol{T}_{C}$ represents a rigid transformation from UAV to front camera reference frame, and ${}^{C}\boldsymbol{T}_{O}$ provides the transformation from camera to object frame of reference, computed online using a Perspective-n-Point (PnP) algorithm (see Fig. 3).

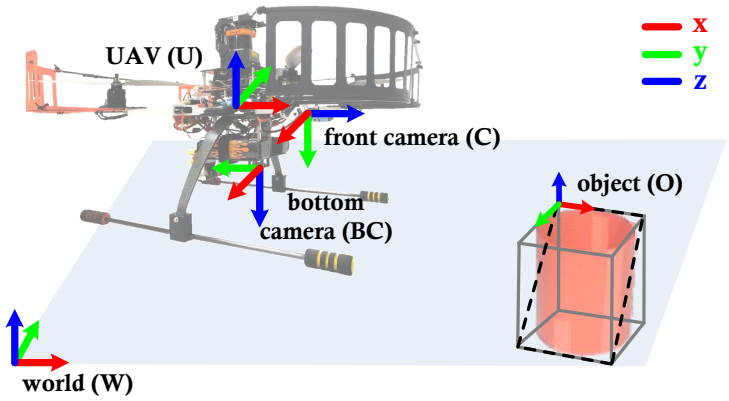


Fig. 3: Frames of reference defined in the proposed solution. the dotted line in the object depicts the virtual tilted plane used for pose estimation purposes.

4.3 FEATURE EXTRACTION SYSTEM

4.3.1 Shape and Color-Based Object Detector

This component of the system has been designed taking into account the hard computational constraints required when running real-time computer vision algorithms onboard an aerial robot. For this reason, this component is based on "low-cost" algorithms meant for increasing the effectiveness of posterior recognition stages. Its main objective is to generate candidate proposals to the Model-Based Object Recognizer component (see Section 4.3.2), and it is also utilized as the main object detection algorithm when the image processing is performed using the bottom camera of the UAV.

For addressing the aforementioned constraints, the functionalities implemented within this component are based on two main features: color and shape information. The former is based on heuristics extracted from the use case of IMAV 2016, in which the targets were defined as red and blue *buckets* or *items*. In order to segment the image based on color information, the HSV color space has been utilized, whose optimal ranges for each channel have been empirically derived. Shape information is mainly utilized for differentiating between identical objects in terms of 3D shape and color (e.g. a red *bucket* is defined as a cylinder of 0.25 m radius and 0.3 m height, while a red *item* consists of a cylinder of 0.1 m radius and 0.1 m height).

Using these functionalities, the Shape and Color-Based Object Detector acts as an effective preliminary stage in order to provide candidates to the Model-Based Object Recognizer component. Furthermore, using the combination of shape and color information, this component provides an intra-class classification of the detected object (e.g. *itemA*, *bucketA*, *itemB*, *bucketB*, where all these objects are cylinders and only differ in the color and shape as stated previously). In addition, it is important to remark the importance of this component in the posterior stage for computing the relative pose of the object being detected. As this component provides object proposals based on color information, the detected Region Of Interest (ROI) in the image fits accurately the contour of the object (see Figures 17d and 18d for an example), which will lead to a better pose estimation as compared to other object proposal algorithms based on predefined ROI sizes such as sliding window approaches, where the ROI can be slightly away from the object contour.

4.3.2 Model-Based Object Recognizer

Object detectors based on predefined knowledge of the object to be detected, such as color, shape, etc, can be very specific and prone to false positives. In order to provide a robust detection of the target, reducing its vulnerability to the environment conditions (e.g. lighting conditions) we implement this component, whose core is based on a supervised learning classifier for target/background segmentation.

The objective of this component is to recognize and locate the object of interest (target) within the image plane, by providing its corresponding ROI, and in addition it is responsible for the recognition of the target in terms of its 3D location with respect to the frame of reference of the camera. For achieving the

aforementioned capabilities, the Model-Based Object Recognizer is composed of four main blocks:

- Object proposal: this block is in charge of generating the candidates within the image to be introduced to the classifier. In this work, we have utilized the candidates generated by the Shape and Color-Based Object Detector (see Section 4.3.1), which can implement an independent object detector component itself or be a part of a higher level recognizer, which is the case when the Model-Based Object Recognizer is operating.
- Feature Extractor: This module is only used in the case of the supervised learning classifiers considered in this work which are not CNN-based models, as CNNs perform an unsupervised feature extraction in the convolution layers. In each candidate ROI generated by the Object Proposal module, Histogram of Oriented Gradients (HOG) [12] features are computed, obtaining a descriptor vector of size 1728. The configuration of the HOG feature extractor is summarized here:
  - Window Size: 56×72 pixels.
  - Cell Size: 8×8 pixels.
  - Block Size: 16×16 pixels (2×2 cells).
  - Block Stride: 8 pixels (50% of block overlapping).
  - Histogram configuration: 9 bins, 20° each (unsigned gradient).
- Classifier: this block implements a supervised learning classifier that has been trained for bucket/background classification. In this work, three supervised learning classifiers have been evaluated: L2 Regularized Logistic Regression (L2R-LR), Support Vector Machines (SVMs) with linear kernel (L-SVM), and CNNs models.
  The formulation of the L2R-LR follows the implementation in [17], whose loss function is given by Eq. 2.

$$J(\omega) = \frac{1}{2}\omega^T\omega + C\sum_{i=1}^{l} log(1 + e^{-y_i\omega^T x_i}) \tag{2}$$

  where ω are the parameters to be learned by the classifier. C is the regularization parameter, and (x_i, y_i) is the instance-label pair of the i_{th} training sample.
  The SVM classifier formulation has been defined using the implementation provided in [8] for the primal form (see Eq. 3).

$$\begin{aligned} \min_{\omega,b,\xi} \frac{1}{2}\omega^T\omega + C\sum_{i=1}^{l}\xi_i, \\ subject \quad to: y_i(\omega^T\phi(x_i) + b) \geq 1 - \xi_i, \\ \xi_i \geq 0 \end{aligned} \tag{3}$$

  where ω (weights), b (intercept term) and ξ_i (*slack variables*) are the parameters to be learned by the SVM classifier. C is the regularization parameter, (x_i, y_i) is the instance-label pair of the i_{th} training sample, and $\phi(x_i)$ is a *feature mapping* function.

In the formulation presented in Eq. 3, a kernel function can be defined as:

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j) \tag{4}$$

where (x_i, x_j) are points in the input feature space, and ϕ is a *feature mapping* function.
The kernel function can lead to different type of SVM classifiers. In this paper, we consider the SVM classifier with a linear kernel computed using Eq. 5.

$$K(x_i, x_j) = x_i^T x_j \tag{5}$$

Regarding the CNN classifier, its architecture consists of 7 layers: 2 convolutional layers, 2 max pooling layers and 3 fully-connected with one hidden layer of 256 units, using ReLU activation function [35] in each layer except the final one, in which a *softmax* activation function is utilized, being the input to the CNN model an image of 56×72 pixels. After the evaluation conducted in Section 5.2.2, the selected supervised learning classifier is based on the architecture presented in Fig. 4.

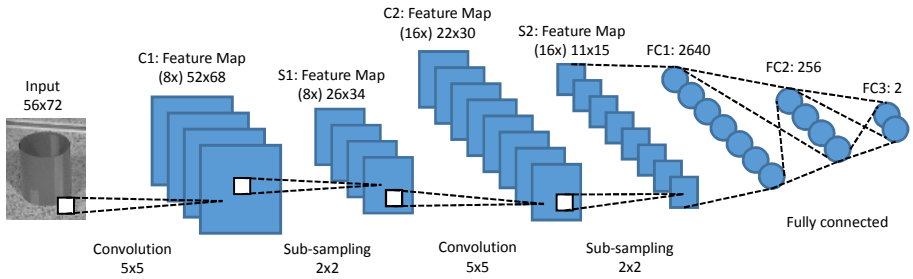


Fig. 4: CNN classifier architecture. Architecture of the CNN model (see *Conf5* in Table 2) utilized for bucket/background classification.

– Pose estimator: In order to compute the relative pose of the target with respect to the frame of reference of the camera, the Pose Estimator block uses a PnP algorithm taking as input the previous computed ROI of the detected target (camera frame of reference), a set of *object points* (object frame of reference), and the intrinsic camera parameters. The final computed pose is selected so that it minimizes the reprojection error between the detected points in the image and the projected *object points*, using Levenberg-Marquardt optimization and RANSAC for filtering the outliers. In order to compute the *object points* we assume that the UAV is always flying at a higher altitude than that of the object of interest. Based on this assumption, and taking into account the appearance of a cylindrical object in such situations, the selected *object points* are the four corners that define the tilted plane similar to the one depicted in Fig. 3. These four corners are defined in the object frame of reference given its 3D shape (height and diameter in the case of our target).

The Model-Based Object Recognizer is utilized as the main object recognition algorithm when the image processing is performed using the front camera.

The implementation details for training and evaluating the Model-Based Object Recognizer are provided in Section 5.2

4.4 MOTOR SYSTEM

4.4.1 Motion Controller

The Motion Controller utilized in our system architecture [36], is based on a cascaded control loop architecture where the external control loops act in position and the inner loops control the linear velocity of the UAV. The attitude control, which is executed by the Pixhawk autopilot, calculates the propeller rotation rates for the actuators.

Three main flight modes are available within the Motion Controller: trajectory, position or velocity, which can be switched during flight. In the experiments presented in this work, the UAV is able to automatically switch from position to velocity control mode when the UAV is commanded to interact with the target by means of a releasing maneuver of a preloaded item.

Some of the capabilities of the Motion Controller that make it suitable for our purposes include a native saturation of the linear velocity commands in all flight modes and the capability of tuning the control gains after performing system identification of a simple dynamic model. These functionalities allow a smooth navigation in cluttered environments, decreasing potential motion blur in the images captured onboard the UAV. As a result, the controller provides a high flight repeatability, even in the presence of uncertainty in the vehicle's state.

4.4.2 Image-Based Visual Servoing based on Reinforcement Learning

The Image-Based Visual Servoing based on Reinforcement Learning component (from now on RL-IBVS) is based on our own designed reinforcement learning framework which integrates an agent (i.e. an aerial robot executing a deep RL algorithm) which can interact with an environment based on RotorS Gazebo [19] simulator (see Fig. 5). The design of our reinforcement learning framework focuses on the two core levels of abstraction in reinforcement learning problems: the *agent* and the *environment*, providing a flexible and versatile interface for the development of new agents or environments. In the proposed RL framework, the interaction between the agent and the environment has been implemented following the guidelines provided by the OpenAI Gym [6] toolkit for reinforcement learning, using ROS for communication purposes.

In this work, we model the visual servoing task as a fully observable Markov Decision Process (MDP), where the final objective is to train the agent in order to find the policy π that maximizes the accumulated discounted reward $R_t = \sum_{i=t}^{T} \gamma^{i-t} r(s_i, a_i)$, given the state $\boldsymbol{s_i} \in \mathbb{R}^4$, the action $\boldsymbol{a_i} \in \mathbb{R}^2$, and a discount factor $\gamma \in [0, 1]$. The behavior of the agent is controlled by this policy which maps an observation to an action, and is usually evaluated by using the action-value function $Q^{\pi}(s_t, a_t) = \mathbb{E}_{\pi}[R_t | s_t, a_t]$.

In the proposed RL-IBVS architecture, the agent implements the DDPG [28] algorithm, which is a model-free, off-policy algorithm based on an actor-critic architecture. In this architecture, the *critic* neural network acts as a nonlinear

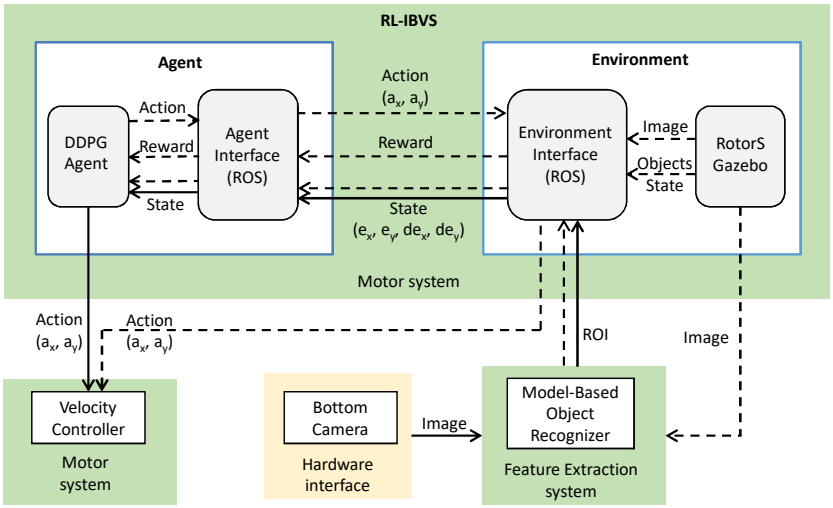


Fig. 5: Image-Based Visual Servoing based on Reinforcement Learning (RL-IBVS) component. The dotted lines represent interactions between the components in training mode, while the continuous lines depict the interactions in test mode (e.g. real flights).

function approximator for estimating the action-value function, while the *actor* neural network is in charge of learning the policy that maps a state into a continuous action. In the specific case of the RL-IBVS, at each time step t the *agent* executes a continuous action $\boldsymbol{a_t} = (a_x, a_y)$ and receives an *observation* (next state) and a *reward* from the environment (see Fig. 5). Where a_x, a_y represent the linear velocities in the x and y direction commanded to the Motion Controller in global coordinates, generated by the deterministic policy $\pi(s_t)$ computed by the *actor* network.

At each time step, the *environment* receives the action generated by the *agent* and computes the next state $\boldsymbol{s_t} = (e_x, e_y, de_x, de_y)$, and a scalar *reward* r_t (see Eq. 7). Where e_x, e_y represent the normalized error in position, measured in pixels, of the center of the detected target ROI with respect to the defined reference in the image (see Figure 9), and de_x, de_y are the normalized derivatives of the error filtered using a low-pass filter. The normalization factor utilized includes the width and the height of the image for the x and y coordinates respectively. The reference in the image is computed by projecting into the image plane the four corners of the square circumscribed to the circumference that forms the cylinder cover (in the object frame of reference) at the desired position with respect to the UAV. Once the projected rectangle is obtained using the intrinsic camera parameters, its center is calculated, which will define the reference in the image. Since the hooks responsible for releasing the objects are not aligned with the bottom camera axis, an offset is added to this reference center in the case of real flights to ensure the items will fall inside the bucket. It is worth remarking here that, although the 3D points of the object are projected considering the desired UAV altitude,

our algorithm is independent of this altitude as only the center of the projected rectangle is utilized for computing the state.

The reward at each time step (see Eq. 7) is computed by considering the evolution of a *shaping* function [14] in two consecutive time steps (see Eq. 6). This procedure permits to alleviate the temporal credit assignment problem while accelerating the learning process of the agent as a localized advice is provided to the agent in each transition via the *shaping* function [16, 26]. The latter has been carefully designed in this work for achieving a smooth desirable behavior of the UAV while approaching the target. For this purpose, as can be seen in Eq. 6, the behavior of the agent is penalized when the errors in the image, their velocities, or the commanded actions are high.

$$shaping_t = -\alpha\sqrt{e_x^2 + e_y^2} - \beta\sqrt{de_x^2 + de_y^2} - \epsilon\sqrt{a_x^2 + a_y^2} \tag{6}$$

$$r_t = shaping_t - shaping_{t-1} \tag{7}$$

where r_t is the reward computed at time step t, and α, β, ϵ are the gains that control the penalization in the error in position, velocity and action terms respectively. These gains have been empirically obtained, being $\alpha = 100$, $\beta = 10$ and $\epsilon = 1$.

In this work, the trained networks (*actor* and *critic*) that compose the DDPG model, consist of feed-forward neural networks with two hidden layers of 300 and 200 units. At test time, the only network utilized is the *actor* network, whose input consists of a 4-dimensional vector representing the state of the detected target in the image, with the output layer being composed of two units that provide the 2-dimensional action to be commanded to the Motion Controller of the UAV, in velocity control mode. In order to remove large displacements in the image of the detected ROI due to high roll and pitch commands of the UAV, an image stabilization technique is utilized taking into account the roll and pitch information provided by the IMU. In this direction, the points of the actual ROI are transformed into a stabilized image frame by using Eq. 8. This procedure has proven to be crucial in order to obtain a proper behavior of the RL-IBVS.

$$\boldsymbol{p'} = \boldsymbol{K}\boldsymbol{R}\boldsymbol{K}^{-1}\boldsymbol{p} \tag{8}$$

where $\boldsymbol{p}$ represents a point in the actual image plane, $\boldsymbol{p'}$ is the point in the stabilized image plane, $\boldsymbol{K}$ is the projection matrix and $\boldsymbol{R}$ is the rotation matrix which encapsulates the roll and pitch information in the aerial robot frame of reference expressed in the camera frame.

Additionally, in order to provide invariance to the UAV's yaw angle during the mission, a 2D rotation based on the actual yaw angle is applied to the state before being introduced to the actor network (see Eq. 9).

$$\boldsymbol{s'} = \begin{bmatrix} \cos(\psi) & \sin(\psi) \\ -\sin(\psi) & \cos(\psi) \end{bmatrix} \boldsymbol{s} \tag{9}$$

where $\boldsymbol{s'}$ is the transformed state introduced to the *actor* network, and ψ represents the UAV's yaw angle.

The implementation details for training and evaluating the RL-IBVS are provided in Section 5.3, where in all the experiments the frequency of the RL-IBVS has been set to 20 Hz.

5 Experiments and Results

The aim of this section is to present the experiments that have been conducted in order to evaluate the different components that integrate the proposed system and to analyze the obtained results. For this purpose, six main experiments have been designed. Two of these experiments are conducted in order to train the proposed learning-based algorithms for performing object recognition and IBVS tasks. Another two experiments are focused on the evaluation of the RL-IBVS algorithm both in simulated and real flight scenarios, considering the particular case of interacting with a moving target. The remaining experiments have been proposed for evaluating the whole system in SAR missions, both in simulated and real flight scenarios, with an emphasis in the object recognition and object interaction tasks. A video demonstration of the reported experiments and results is provided with this manuscript in: `https://vimeo.com/235929544`.

5.1 Experimental Setup

All the developed algorithms have been integrated into the Aerostack architecture [43] and implemented in C++ and Python, using ROS as the communication middleware. Deep learning models for object classification have been trained using Keras[2] library on a 2.6 GHz CPU Intel Core i7-6700HQ, whereas models utilized for reinforcement learning purposes have been trained using TensorFlow[3] on a GPU Nvidia GeForce GTX 970. Regarding the simulated flight experiments, the proposed setup uses RotorS Gazebo for evaluating the IBVS algorithms, and on the other hand, the PX4 Software-In-The-Loop is integrated with Gazebo in order to provide a realistic evaluation of the whole system for SAR missions. Regarding the real flight experiments, two indoor scenarios have been designed. The first scenario consists of a 3 m $\times$ 4 m area conceived for evaluating the RL-IBVS system, in which OptiTrack motion capture system has been utilized for recording the ground truth data relative to the UAV and the moving target. The second scenario consists of a 11 m $\times$ 7 m area used for evaluating the whole system in a SAR mission. In all the presented experiments, no tethers or external power supplies were utilized. In addition, in all real flight experiments, the UAV was carrying preloaded items of 100 g each, which substantially increased the complexity of the missions.

5.2 Training and Evaluation of the Supervised Learning Classifiers for Object Recognition

5.2.1 Dataset

In the SAR missions proposed for evaluating our system, the targets consist of cylindrical buckets of different colors positioned at random locations in the environment. Currently, there are no publicly available datasets containing images of cylindrical buckets that could be used for training a deep learning classifier. For

[2] https://faroit.github.io/keras-docs/1.2.2/
[3] https://www.tensorflow.org/

(a)

(b)

Fig. 6: Examples of images used for training the supervised learning classifiers for *bucket*/*background* classification. (a) Bucket training examples. (b) Background (non bucket) training examples. Images of office furniture and different types of floors have been used as examples of the background class.

this purpose, a custom dataset has been created containing images from *background* (non bucket) class and *bucket* class. After a large process of data acquisition, a total of 875 images for the *bucket* class and 1750 images for the *background* class were collected.

From this original dataset, several data augmentation techniques have been applied in order to increase the number of images used for training and evaluating the classifier and to prevent overfitting problems. The data augmentation process consisted in the application of three main techniques: *random cropping*, *horizontal flipping*, and *noise addition*. The random cropping strategy was similar to the one presented in [25], obtaining four cropped images per original image, by selecting a random offset starting from each of the four corners in the original image. The *horizontal flipping* strategy consisted in mirroring the original image from left to right direction, which allowed doubling the number of images. Finally, the last data augmentation technique was based on adding a Gaussian noise to the original image with zero mean and a standard deviation of 10 pixels. From the 875 images of *bucket* class, we performed data augmentation over 375 of these images obtaining a total of 4500 images. The remaining 500 images were added to the augmented dataset, providing a total amount of 5000 images for *bucket* class. From the 1750 images belonging to *background* class, data augmentation was conducted over 550 of these images, obtaining a total of 3300 images. The remaining 1200 images were added to the augmented set, providing a total amount of 5000 images for *background* class.

Subsequently, in order to train and evaluate the supervised learning classifiers considered in this work, the final dataset consisting of 10000 images (see Fig. 6 for an example), was divided into *train* (70 %), *validation* (15 %) and *test* (15 %) sets.

5.2.2 Training and Evaluation Methodology

In order to conduct a reliable comparison between the different supervised learning classifiers considered in this work, a similar methodology as the one presented in [7] has been applied: First, a total of 6 evaluation tests have been defined, where the *train*, *validation* and *test* sets were randomly picked for each evaluation test. In each of the 6 evaluation tests, a *5-fold* cross-validation procedure is performed in the case of the L2R-LR and L-SVM classifiers in order to select the optimal regularization parameters of the corresponding classifier (see Eqs. 2 and 3 in Section 4.3.2). The range of values considered for the regularization parameter cover a wide range starting form 10^{-4} to 10^{2} with a step of 2×10^{p}, where p ranges from -4 to 1 with a step of 1. In the case of the CNN classifier, a cross-validation procedure over the parameters of this model can be intractable. Thus, in this case we have considered 8 CNN configurations (see Table 2) varying the number of *feature maps* in each convolutional layer, the size of the convolution filters, and the number of channels of the images introduced to the network (i.e. grayscale or RGB images).

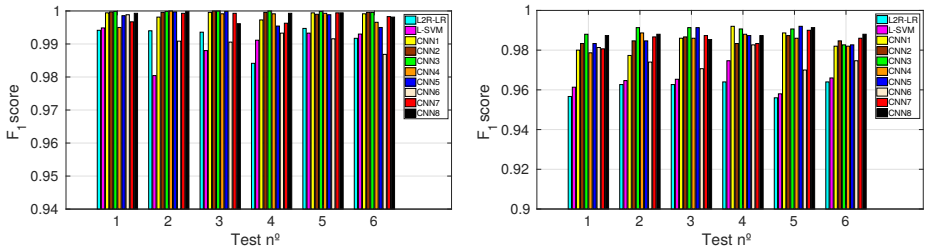
All the CNN configurations have been trained using Keras library with TensorFlow as backend. As explained in Section 4.3.2, the CNN models consisted of 7 layers: 2 convolutional layers, 2 max pooling layers and 3 fully-connected with one hidden layer of 256 units, using ReLU activation function [35] and *dropout* regularization [45] in each layer except the final one, in which a *softmax* activation function was used. The selected loss function was based on the categorical cross-entropy loss, using Adam optimizer [23] for its minimization during a training process of 60 epochs as maximum, and a mini-batch size of 128 images. In addition, the *early stopping* technique was applied for regularization purposes, taking into account the validation loss with a patience of 5 *epochs*.

Table 2: CNNs configurations evaluated.

CNN conf.	**Input dim.**	**#Feature Maps**		**Filter size**		**#param.**
		Conv1	Conv2	Conv1	Conv2	
CNN1	$56 \times 72 \times 1$	32	64	3×3	3×3	3,165,314
CNN2	$56 \times 72 \times 1$	32	64	5×5	3×3	3,165,826
CNN3	$56 \times 72 \times 1$	32	64	5×5	5×5	2,756,226
CNN4	$56 \times 72 \times 1$	16	32	5×5	5×5	1,365,698
CNN5	$\mathbf{56 \times 72 \times 1}$	**8**	**16**	$\mathbf{5 \times 5}$	$\mathbf{5 \times 5}$	**680,034**
CNN6	$56 \times 72 \times 1$	4	8	5×5	5×5	339,602
CNN7	$56 \times 72 \times 3$	32	64	$3 \times 3 \times 3$	3×3	3,165,890
CNN8	$56 \times 72 \times 3$	32	64	$5 \times 5 \times 3$	5×5	2,757,826

The results obtained after conducting the 6 evaluation tests on each classifier are shown in Figures 7a and 7b and summarized in Tables 3 and 4, where C1 and C2 stand for Class 1 (*bucket* class) and Class 2 (*background* class) respectively. In these tables, the average values over the 6 tests are presented. In Table 3, the high scores obtained in most of the CNN configurations for the training set can be noticed. More relevant are the results presented in Table 4 for the test sets, where all the CNN configurations obtained a significantly higher F_1 score than the one obtained by the L2R-LR and L-SVM classifiers. This fact can be

confirmed in Fig. 7b, where in all the evaluation tests the CNN-based classifiers obtained a higher performance than the L2R-LR and L-SVM. As shown in Table 4, the number of feature maps was confirmed as being a crucial parameter for the CNN-based classifiers, where CNN6 provided a significantly lower performance as compared to the rest of CNN configurations. Another important result extracted from Fig. 7b and Table 4 is that CNN configurations with a convolution filter of size 5×5 provided higher performance than the analogous configurations with a filter size of 3×3 (i.e. CNN3 vs CNN1, and CNN8 vs CNN7). However, these CNN configurations (CNN3 and CNN8) require the highest processing times per image and are quite demanding in terms of the number of parameters, which will also lead to a higher memory consumption. All the results presented in Tables 3 and 4 have been obtained with a CPU Intel Core i7-6700HQ except the test time computed in Table 4, which has been measured with the UAV's onboard computer, and refers to the processing time per image averaged over a mini-batch (i.e. 128 images), taking into account the feature extraction plus the inference time of the classifier. Thus, in the case of the L2R-LR and the L-SVM, this time encompasses the HOG feature extraction process plus the inference time of the classifier.



(a) (b)

Fig. 7: Results obtained during the comparison of the supervised learning classifiers on 6 different evaluation tests (best see in color). a) F_1 score obtained in the training sets. b) F_1 score obtained in the test sets.

Table 3: Average training results obtained for the 6 evaluation tests of Fig. 7a.

Classifier	**Precision**		**Recall**		**F_1score**		**Train time (s/epoch)**
	C1	C2	C1	C2	C1	C2	
CNN1	0.998	1.00	1.00	0.998	0.999	0.999	30
CNN2	0.999	1.00	1.00	0.999	1.00	1.00	31
CNN3	1.00	1.00	1.00	1.00	1.00	1.00	44
CNN4	0.997	1.00	1.00	0.997	0.998	0.998	20
CNN5	**0.996**	**1.00**	**1.00**	**0.996**	**0.998**	**0.998**	**10**
CNN6	0.987	0.997	0.997	0.987	0.992	0.992	7
CNN7	0.996	1.00	1.00	0.996	0.998	0.998	33
CNN8	0.998	1.00	1.00	0.998	0.999	0.999	46
L2R-LR	0.992	0.992	0.992	0.992	0.992	0.992	-
L-SVM	0.989	0.991	0.991	0.989	0.990	0.990	-

Table 4: Average test results obtained for the 6 evaluation tests of Fig. 7b.

Classifier	**Precision**		**Recall**		**F_1score**		**Test time (ms/image)**
	C1	C2	C1	C2	C1	C2	
CNN1	0.973	0.996	0.996	0.973	0.985	0.984	3.830
CNN2	0.978	0.993	0.993	0.977	0.985	0.985	3.956
CNN3	0.985	0.993	0.993	0.985	0.989	0.989	5.136
CNN4	0.973	0.998	0.998	0.972	0.985	0.985	3.027
CNN5	**0.978**	**0.996**	**0.996**	**0.978**	**0.987**	**0.987**	**2.075**
CNN6	0.963	0.989	0.989	0.962	0.976	0.975	1.921
CNN7	0.974	0.998	0.998	0.973	0.986	0.985	4.123
CNN8	0.980	0.996	0.996	0.980	0.988	0.988	5.830
L2R-LR	0.958	0.965	0.965	0.957	0.961	0.961	0.984
L-SVM	0.966	0.964	0.964	0.966	0.965	0.965	4.187

The execution of SAR missions using a fully-autonomous aerial robot implies running simultaneously multiple processes that implement the functionalities of each component of the system. Traditionally, UAVs are equipped with small and light computers with limited computational capabilities. In order to deal with these computational constraints, it is required to select the appropriate model in order to find a proper trade-off between performance and computational cost. For this purpose, in this work, we consider the selection of the most appropriate classifier based on two main variables: the performance of the classifier and the processing time. Thus, we take into consideration the average F_1 score and the test time presented in Table 4, which are plotted in Fig. 8 for a better visualization. Taking into account the results presented in Fig. 8 and considering that the best possible classifier is the one lying on point $[0, 1]$ (i.e. maximizing the F_1 score while minimizing the processing time), we have selected CNN5 as the most appropriate classifier for our purposes. This configuration provides the lower Euclidean distance (in normalized coordinates) to the desired point.

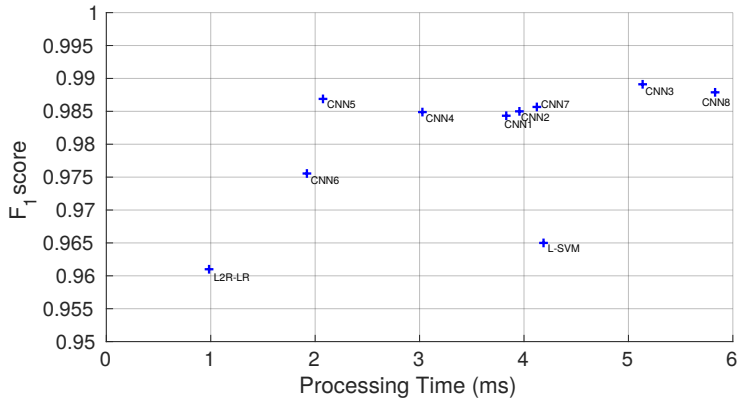


Fig. 8: Average F_1 score and processing time of the different supervised learning classifiers from the results presented in Table 4. The processing time has been measured on the Intel NUC6i5SYK onboard computer.

5.3 Training and Evaluation of the Reinforcement Learning model for Image-Based Visual Servoing

For training the agent in order to perform IBVS tasks, we use the RL-IBVS component (see Section 4.4.2) in training mode.

In the RL-IBVS training mode, the *environment* is designed in an episodic RL setting, where the agent's experience is divided into a series of episodes, each of one composed of several train steps. In each training step, the agent takes an action with added noise according to an Ornstein-Uhlenbeck distribution, and receives an *observation* and a *reward* from the environment (see Fig. 5). The current 4-dimensional observation is fed into the *actor* network, which generates a continuous 2-dimensional action in the range $[-0.5, 0.5]$ m/s.

For stabilizing and accelerating the training process, the e_x, e_y variables from the state are measured taking into account the ROI obtained by the projection into the image plane of the known 3D points of the target (see the cyan rectangle in Fig. 9). During experimentation, we have found that this procedure is critical for allowing the convergence of the training process, as it removes large displacements of the detection of the target in the image plane due to high roll and pitch commands of the UAV.

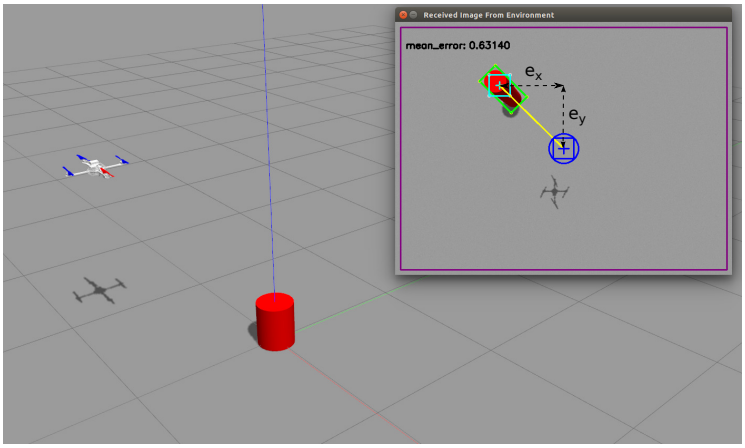
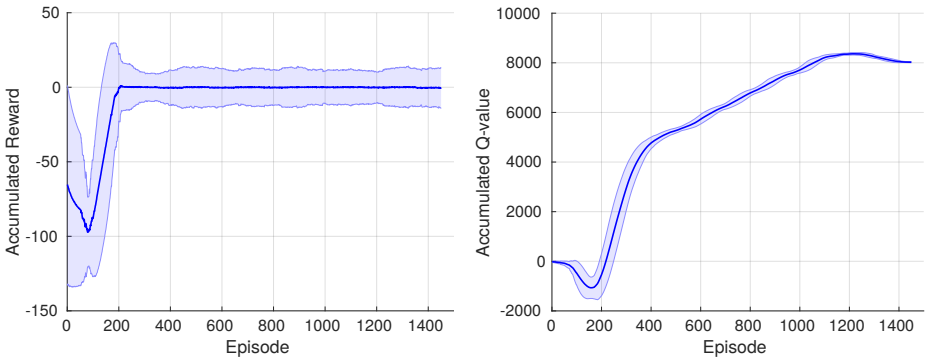


Fig. 9: RotorS Gazebo simulation environment and image captured from the bottom camera of the UAV (best see in color). The desired reference is depicted in blue color. In green color the object detection result can be seen. The cyan rectangle represents the ground truth points (in the object frame of reference) defined by the four corners of a square circumscribed to the circumference that forms the cylinder cover, projected into the image plane.

At the beginning of each episode, the UAV is placed at a random position of a 2 m $\times$ 2 m area of the environment at a constant altitude of 1.2 m. The episode is considered as finished when the maximum number of training steps per episode is reached or when the agent reaches a terminal state. In the specific case of the RL-IBVS presented in this work, the maximum number of steps per episode has

been empirically set to 400 steps. A terminal state is reached when the target is out of the boundaries defined in the image captured by the bottom camera of the UAV (see the magenta rectangle in Fig. 9). When this situation occurs, the agent is penalized with a negative reward of -100 and a new episode is started. In the rest of situations, the reward is computed using Eq. 7.

The training process of the *actor* and *critic* neural networks took approximately 7.5 hours, in which Adam optimizer [23] was utilized with a base learning rate of 10^{-4} for the *actor* and 10^{-3} for the *critic*. As explained in Section 4.4.2, the architecture of the neural networks has been empirically obtained, using two hidden layers of 300 and 200 units. The output layer of the *actor* is a *tanh* function for providing a continuous linear velocity command, bounded to $[-0.5, 0.5]$. The rest of the hyperparameters are the same as the ones presented in [28]. The results obtained during the training process are illustrated in Fig. 10. In this figure, it is important to notice the different time instants in which the reward and the action-value function stabilize. The former reaches a stable value around zero from episode 200, while the action-value function continues increasing until episode 1250. Based on these results, the learned weights adopted for the neural networks of the DDPG (*actor* and *critic*) are the ones obtained at episode 1400, whose results have been additionally confirmed in a visual manner, and can be reviewed in the video demonstration.



(a) (b)

Fig. 10: Training curves obtained during the training process of the RL-IBVS agent. a) Each point depicts the accumulated reward at the specific episode. b) Each point in the curve represents the accumulated action-value Q predicted by the *critic* network at the specific episode. Each curve is plotted taking into account a moving average of 100 episodes.

It should be noted that although this training procedure is performed in simulation using the ground truth points of the object projected into the image plane, the realistic dynamics provided by the RotorS Gazebo simulator together with the appropriate policy learned by the actor network allow an almost direct transition to previously unseen simulated and real scenarios where the location of the target is unknown. In these scenarios, where no ground truth data relative to the position of the target is available, the stabilized detection of the target in the image plane is utilized for computing the state.

5.4 Simulated flight experiments

5.4.1 Image-Based Visual Servoing

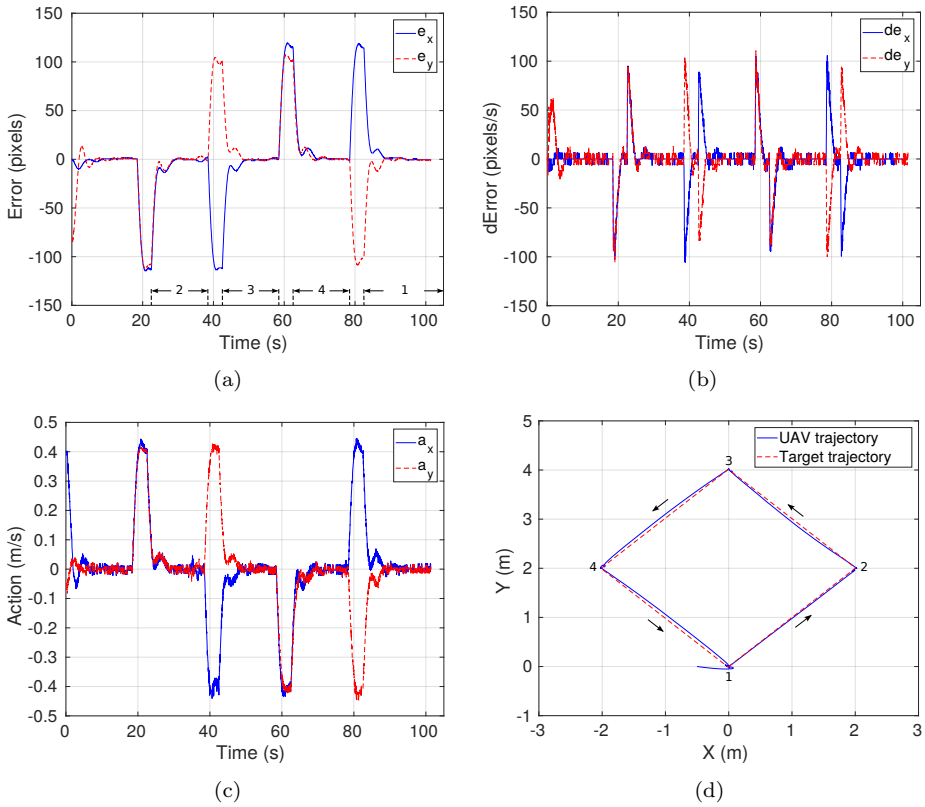


Fig. 11: Simulation results obtained for the classic IBVS approach while following a moving target executing a rhomboidal trajectory at a maximum speed of 0.5 m/s. The target stops for a few time steps on each corner of the rhomboid (1 to 4). a) Error in the image with respect to the reference. b) Derivatives of the error in the image with respect to the reference. c) Actions generated by the UAV referred in the world frame of reference. d) Trajectories followed by the UAV and the target.

In this section, the experiments conducted in order to evaluate the proposed RL-IBVS algorithm using the trained agent are described in detail. For the sake of comparison, and in order to perform a thorough evaluation of the RL-IBVS system, the IBVS controller proposed in [10, 30] (referred to as *classic* IBVS) has been implemented using the Visual Servoing Platform (ViSP) library [31], and is utilized in this work as the baseline for evaluating our RL-IBVS. The *classic* IBVS is based on the well-known visual servo control law which establishes that

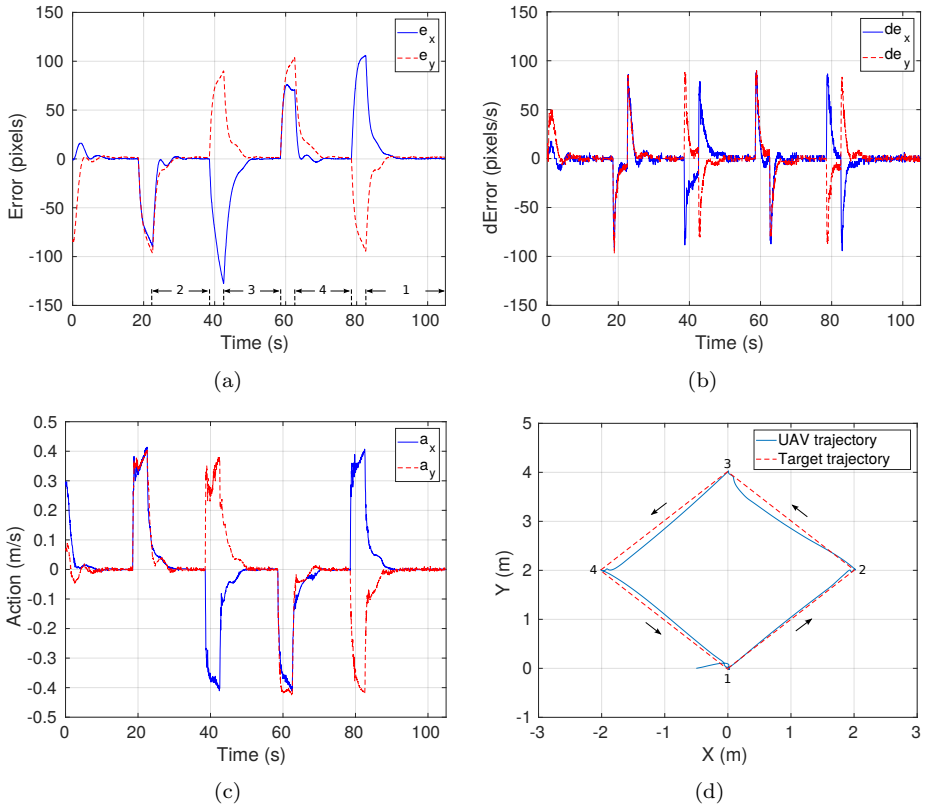


Fig. 12: Simulation results obtained for the proposed RL-IBVS approach while following a moving target executing a rhomboidal trajectory at a maximum speed of 0.5 m/s. The target stops for a few time steps on each corner of the rhomboid (1 to 4). a) Error in the image with respect to the reference. b) Derivatives of the error in the image with respect to the reference. c) Actions generated by the UAV referred in the world frame of reference. d) Trajectories followed by the UAV and the target.

$\mathbf{v} = -\lambda \widehat{\mathbf{L}_\mathbf{s}^+} \mathbf{e}$, where $\mathbf{v}$ represents the spatial velocity of the camera, $\mathbf{L_s}$ is the *interaction matrix*, and $\mathbf{e} = \mathbf{s} - \mathbf{s}^*$ is the error computed as the difference between the actual and desired features in the image plane. The estimation of the interaction matrix is obtained as proposed in [30], where $\widehat{\mathbf{L}_\mathbf{s}^+} = \frac{1}{2}(\mathbf{L_s} + \mathbf{L_{s^*}})^+$, being $\mathbf{L_{s^*}}$ the interaction matrix at the desired feature location, and $(.)^+$ represents the pseudoinverse operation as defined in [10].

In order to obtain a complete evaluation of the *classic* IBVS and the RL-IBVS, a simulated experiment using RotorS Gazebo has been designed with the aim of following a moving target. In this experiment, the target has to perform a rhomboidal trajectory (see Figures 11d and 12d), moving at a constant speed of 0.5 m/s along the edges of the rhomboid and stopping during several simulation steps in its corners. Using this setup, a complete validation of the IBVS approaches

can be conducted as it involves the interaction with a static and a moving target in the same experiment.

The results obtained after the execution of the proposed experiment are shown in Figures 11 and 12 for the *classic* IBVS and the RL-IBVS respectively. As can be seen in Figures 11a and 12a, both controllers produced a smooth response in the error signal, with the RL-IBVS producing a little less oscillatory response. A similar result can be observed in Figures 11c and 12c for the commanded actions. The trajectories followed by the UAV are depicted in Figures 11d and 12d, where the classic IBVS produced a trajectory with a very small deviation with respect to the trajectory of the moving target. Finally, Table 5 summarizes the results obtained for both IBVS approaches in the case of the error signal. In Table 5 it can be noticed the appropriate behavior of both controllers, highlighting the very high precision of the RL-IBVS controller with an average error of less than one pixel in both x and y directions. Furthermore, the standard deviation of the error signal in the case of the RL-IBVS is about 7 pixels less than the *classic* IBVS, revealing the faster response of the former while following the target.

It should be remarked that the results obtained for the *classic* IBVS approach (see Fig. 11) are the result of a long tuning process in which the corresponding gains of this controller were adjusted by a trial an error process. Conversely, the results presented in Fig. 12 for the RL-IBVS have been obtained by directly using the trained *actor* network without any further tuning.

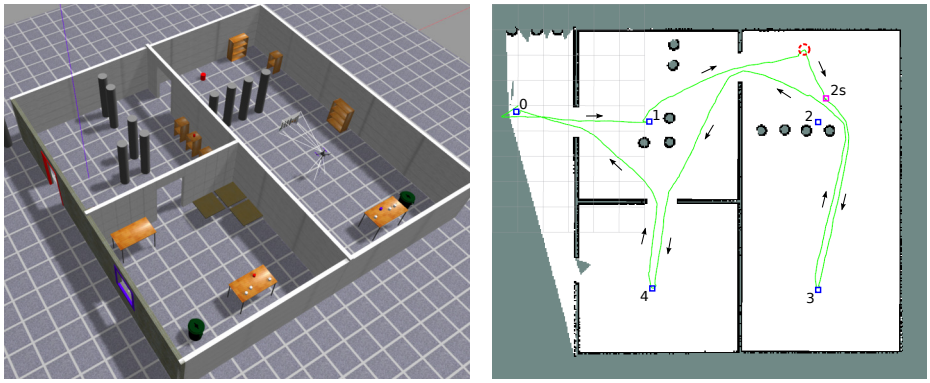
Table 5: Mean and standard deviation errors in pixels obtained for the simulation flight experiment of Figures 11 and 12.

IBVS approach	e_x	e_y
Classic	0.5 ± 42.5	-1.3 ± 40.0
RL	-0.4 ± 34.6	0.5 ± 34.2

5.4.2 Search and Rescue

This section presents the simulation experiments that have been designed in order to evaluate the coordination between the components of the whole aerial robotic system and its capabilities to successfully accomplish SAR missions. For the evaluation of the whole system, several simulated environments using Gazebo simulator (see Fig. 13a) have been designed based on the scenario proposed for the IMAV 2016 competition. Inspired by the use case of IMAV 2016, different layouts of indoors rooms have been created. In this experiment, the objective of the UAV is to explore the *a priori* unknown indoor scenario searching for a predefined target (red bucket). Once the target has been recognized, the UAV has to interact with the former to release an item inside of it. Once the interaction is finished, the UAV has to continue exploring the scenario in order to locate more possible targets. As shown in Fig. 13b, the UAV is able to explore the entire indoor scenario, readapting the exploration mission when the target is recognized in order to perform the release maneuver. Another remarkable behavior that can be noticed in Fig. 13b is the safety point (point 2s in Fig. 13b) generated by the AMP in order to avoid

the mission point number 2 which is within the boundaries of an obstacle, taking into account the inflation radius of the obstacles.



(a) (b)

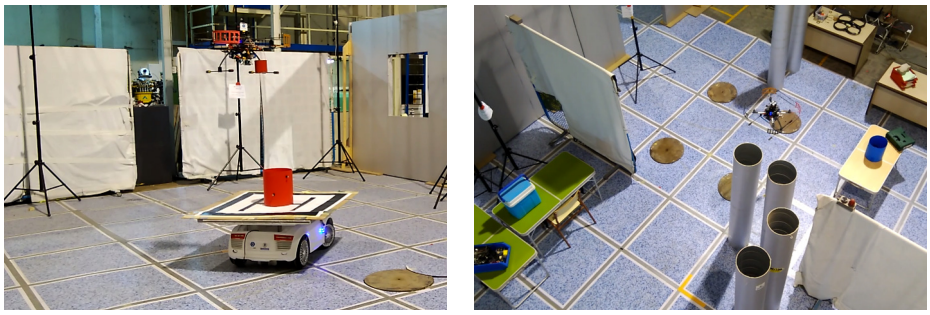
Fig. 13: Fully-autonomous simulation flight experiment for a Search and Rescue mission in a cluttered indoor environment (best see in color). The target object is represented by a red bucket. a) Gazebo simulation environment. b) 2D occupancy grid map and trajectory followed by the UAV (green color). Blue squares depict the mission points (0 to 4) automatically generated by the GMP. The magenta square (2s) represents the safety mission point generated in substitution of mission point 2.

5.5 Real flight experiments

This section presents the experiments conducted and the results obtained during the execution of real flight experiments. Two main experiments have been designed (see Fig. 14), one for evaluating the RL-IBVS algorithm, and the second experiment which aims to evaluate the whole proposed system in a SAR mission.

5.5.1 Image-Based Visual Servoing

This experiment has been designed for evaluating the proposed RL-IBVS algorithm in real flight conditions. In order to obtain a detailed evaluation of the system, the experiment integrates a moving target which describes a random trajectory commanded by a human operator. The goal of the UAV is to autonomously track the moving target until a secure position is reached for performing the release maneuver of a preloaded item. The results obtained during the execution of the experiment are presented in Fig. 15, which shows how the error in the image and the control actions tend to zero at the end of the path of the moving target, where it starts decreasing the speed until it gets stopped. In this instant (see instant 41.6 s in Fig. 15a), the error in pixels of the detected object in the image decreases below a predefined threshold and the *target locked* action for releasing the item



(a) (b)

Fig. 14: Indoor scenarios designed for real flight experiments. a) Real flight experiment conducted for evaluating the RL-IBVS. b) Real flight experiment conducted for evaluating the complete proposed system in a SAR mission.

is commanded. In Fig. 15c, the decremental tendency of the control actions can be clearly seen as soon as the UAV is approaching the target. Around second 41.6, the release operation is commanded. The release of a 100 g item, causes a perturbation on the dynamics of the UAV, successfully handled by our system. It is also very important to notice that, despite the turbulences generated while flying very close to the target (empty bucket), the UAV is able to maintain a stable flight, being the control actions under 0.04 m/s when the UAV is exactly above the target. Finally, the 3D trajectories followed by the UAV and the moving target are depicted in Fig. 15d, which shows the stable tracking trajectory followed by the UAV on top of the moving target. This trajectory is only perturbed in height when the release operation is performed. This fact can be clearly seen around point $x : -0.82, y : -0.84$ where the UAV moves slightly up.

5.5.2 Search and Rescue

In this section, a real flight SAR experiment for evaluating the whole system proposed in this work is presented. In the designed experiment the UAV is required to explore an indoor environment with two possible entries: a 1.2-m-wide door or a 1-m-wide window. The indoor scenario is composed of several obstacles, mainly tables, chairs and columns, whose location is completely unknown (see Fig. 14b). Based on the use case of the IMAV 2016, the targets are represented by two colored buckets. For increasing the difficulty of the experiment, one of these objects is located on top of a table (blue bucket) while the other (red bucket) is positioned on the floor (see Fig. 16b). The objective of the UAV is to autonomously explore the indoor environment, localize the targets, and interact with them by means of performing a release maneuver of the corresponding item. The items, both red and blue, are preloaded before takeoff and held in the hooks of the UAV. Once the interaction with the targets is finished, the UAV has to return to the initial takeoff point. Fig. 16 shows the 2D map generated and the 3D trajectory followed by the UAV when the SAR mission is completed. In addition, the estimated positions of the targets computed by the Model-Based Object Recognizer component are provided (see dotted circles in Fig. 16a). The ground truth positions of both targets

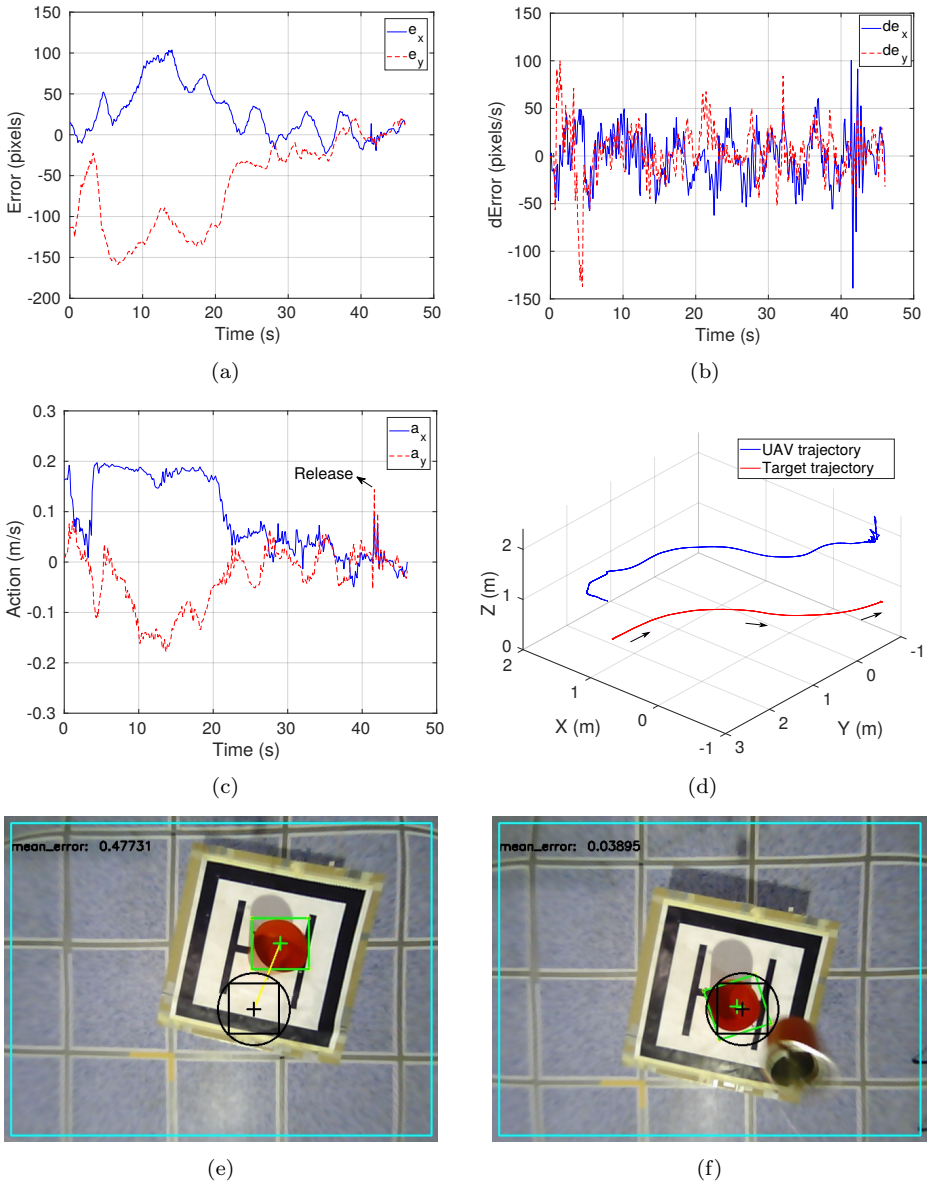
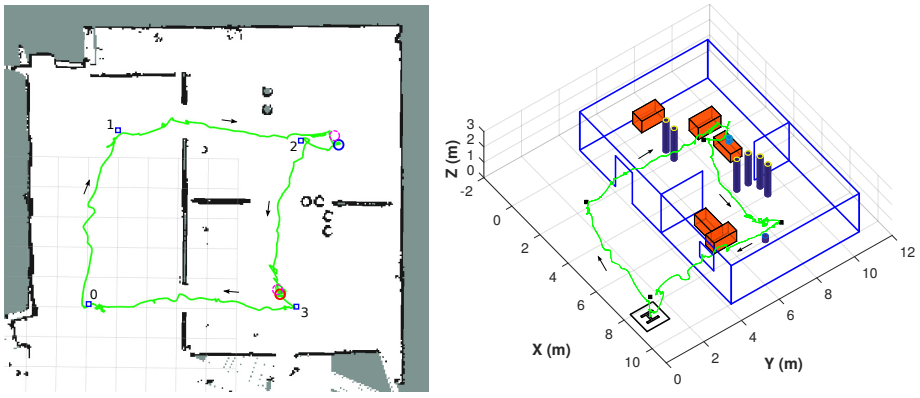


Fig. 15: Real flight experiment for evaluating the RL-IBVS algorithm while following a moving target. a) Error in the image w.r.t the reference. b) Derivatives of the error in the image w.r.t the reference. c) Action commands generated by the UAV. d) 3D paths followed by the UAV and the moving target. e), f) Errors computed by the RL-IBVS component in two different instants of the mission. f) When the target is locked, a release operation is commanded. OptiTrack system is used for recording the ground truth positions of the UAV and the moving target.

were measured by a human operator, being [x : 7.84 m, y : 7.5 m, z : 0.0 m] for *bucket A* (red bucket) and [x : 2.8 m, y : 9.2 m, z : 0.54 m] for *bucket B* (blue bucket). The estimated positions were [x : 7.71 m, y : 7.4 m, z : 0.068 m] for *bucket A* and [x : 2.54 m, y : 9.14 m, z : 0.596 m] for *bucket B*. Thus, the average error of each component for both targets is [x : 0.19 m, y : 0.08 m, z : 0.062 m] what reveals the acceptable position estimation, taking into account the accumulated errors that might occur while measuring the ground truth positions, the ones introduced by the estimation of the UAV position, and the errors produced while computing the detection of the object in the image plane.

In Figures 17 and 18 the recognition of the targets in different time instants during the execution of the real flight SAR mission is depicted. It is important to remark the capability of our system to automatically switch between different recognition modes. Based on this, once the object is considered as recognized using the front camera image, the system stops processing the images coming from that device and starts processing the images coming from the bottom camera. Subsequently, when the release maneuver has finished, the recognition mode using the front camera is enabled again, stopping the processing with the bottom device.

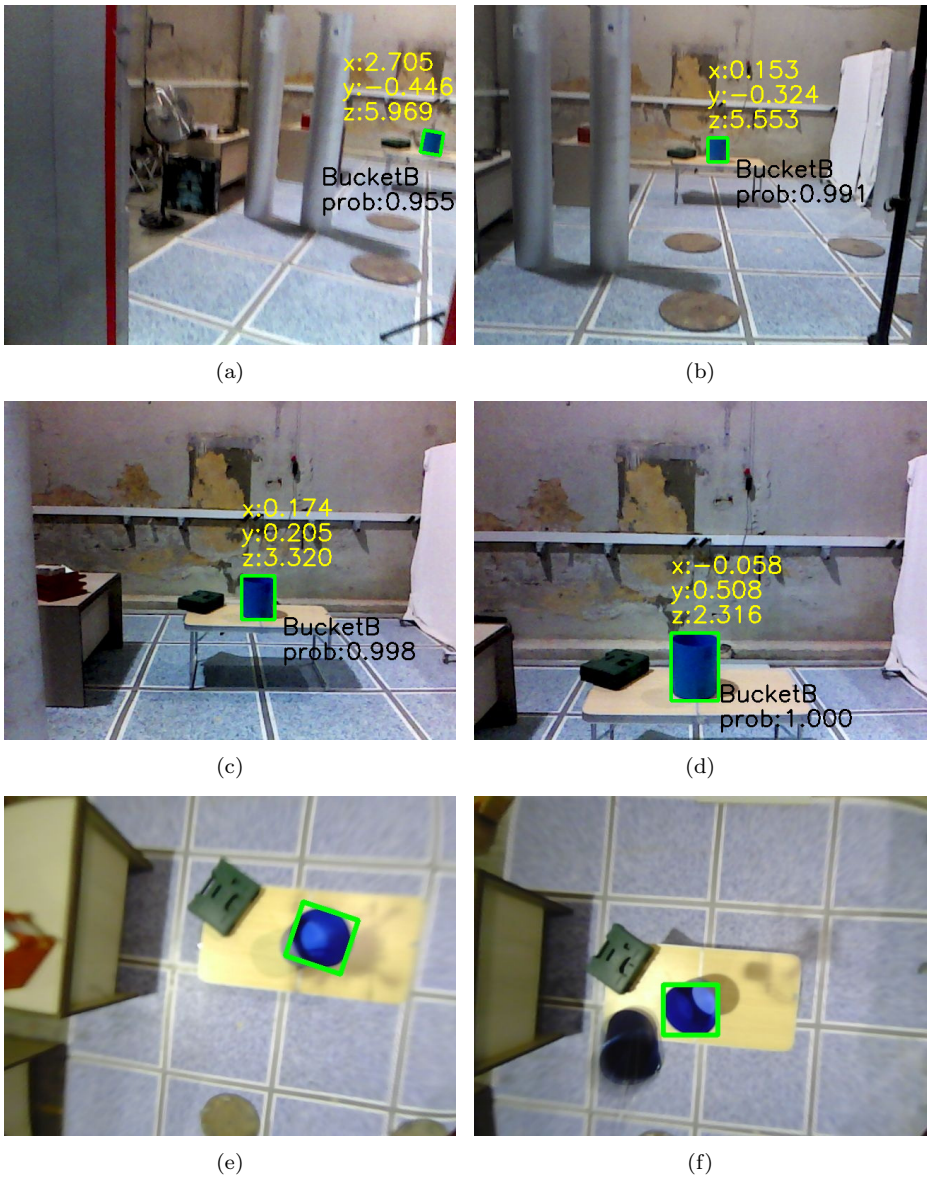


(a) (b)

Fig. 16: Fully-autonomous real flight experiment for a SAR mission in a cluttered indoor environment (best see in color). The targets are represented by two colored buckets. The trajectory followed by the UAV during the mission is depicted in green color. a) 2D occupancy grid map and trajectory followed by the UAV. Dotted pink circles represent the estimated location provided by the Model-Based Object Recognizer. b) 3D outline of the indoor environment. Obstacles are represented by cuboids and columns. Squares depict the commanded mission points.

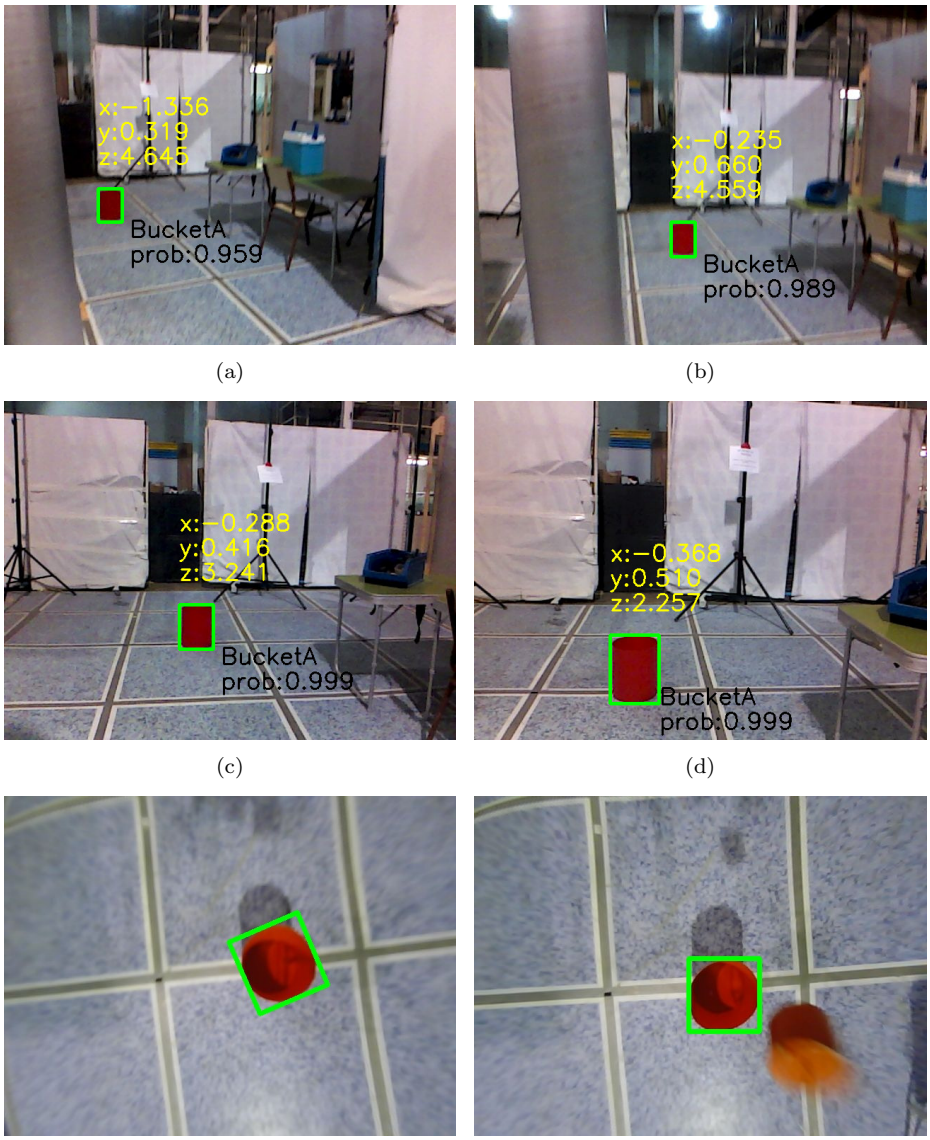
5.6 Discussion

The development of a fully-autonomous aerial robot requires the implementation of robust and efficient algorithms in a wide range of components ranging from



(a) (b) (c) (d) (e) (f)

Fig. 17: Object recognition results for target 1 (best see in color). a), b), c), d) Recognition of the blue bucket (bucket B) from the front camera. e), f) Detection of the bucket from the bottom camera. f) Detection of the bucket in the moment when the release command has just been triggered. The positions indicated in the figure refer to the computed relative pose of the target with respect to the front camera using a Perspective-n-Point (PnP) algorithm. The output probabilities from the CNN classifier are indicated together with the object category.



(a) (b) (c) (d) (e) (f)

Fig. 18: Object recognition results for target 2 (best see in color). a), b), c), d) Recognition of the red bucket (bucket A) from the front camera. e), f) Detection of the bucket from the bottom camera. f) Detection of the bucket in the moment when the release command has just been triggered. The positions indicated in the figure refer to the computed relative pose of the target with respect to the front camera using a Perspective-n-Point (PnP) algorithm. The output probabilities from the CNN classifier are indicated together with the object category.

control, mission planning, path planning, mapping, to object recognition, object following, etc. In order to successfully accomplish these missions, a precise coordination between the different components is required. In the specific case of SAR missions, and other similar ones such as package delivery, this coordination becomes more critical when it involves the interaction with the detected target, and even more in the presence of dynamic targets.

In this paper, we have focused our attention on this type of missions that have not been extensively addressed in the literature. In order to develop more robust and generic algorithms, here we extend our previous works [41, 42] by integrating learning-based algorithms for object recognition and object following. The experiments and results presented in this section, demonstrate the appropriate capabilities of our proposed system for the accomplishment of such high-level missions in a fully unsupervised manner (i.e. without human intervention).

Regarding the object recognition task, several supervised learning classifiers have been evaluated in order to address the hard computational constraints imposed in the real-time operation of an aerial robotic platform. After a thorough evaluation, a compact CNN model has been selected showing very high performance for target/background classification in cluttered environments. The trained CNN model has fewer parameters as compared to standard architectures, reducing the computational cost while maintaining a very high accuracy. This accuracy reflects the accurate detection of the object in the image plane (see Figures 17 and 18), even in frames with a considerable amount of blur as shown in Figures 17a, 17b, 18a and 18b. The accurate detection of the object in the image plane is a critical step for pose estimation purposes. The latter problem has been addressed in this paper by means of a Perspective-n-Point algorithm, which provides an acceptable pose estimation using monocular information. The errors in the position estimation of the targets shown in Section 5.5.2 are adequate for SAR missions and had low effect on our system as a fisheye-lens bottom-looking camera with a considerable field of view is utilized. Owing to this field of view, and based on the proposed RL-IBVS, our system can handle quite big errors in the position estimation of the bucket where the release operation has to be performed. It is worth highlighting the low errors obtained in the estimation of the y and z coordinates, both under 8 cm. The error in the x coordinate is slightly higher, which can be caused by sudden roll commands of the UAV while detecting the target.

With respect to the target interaction and following task, it is very important to remark that the proposed RL-IBVS has been trained only in simulation and with the dynamics of the UAV available in the RotorS Gazebo simulator (i.e. AscTec Hummingbird quadrotor). For the transition to real flights using our custom aerial quadrotor, no modifications have been made to the trained model. This fact is even more relevant in the field of aerial robotics, where directly training a reinforcement learning algorithm in a real testbed scenario is very complicated and to the authors' knowledge has not been yet addressed in the literature. Another important feature of the proposed RL-IBVS algorithm is its simplicity and versatility as the functionality of the RL-IBVS is independent of the object to be recognized, being the only required input the center of the ROI corresponding to the detected object in the image. The comparison results of the proposed RL-IBVS with respect to a classic IBVS approach shown in Figures 11 and 12 and Table 5 demonstrate that the RL-IBVS can be utilized as an alternative to state-of-the-art IBVS approaches, which usually require a long tuning process of their parameters.

Moreover, as shown in Fig. 10, the convergence in the training process of the agent is achieved relatively fast due to the training strategy implemented in this work. This strategy explained in Section 4.4.2, consists in using as the detected ROI the 3D points of the target projected into the image plane, which provides a more stable state of the target. The main limitation of the proposed RL-based algorithm is its dependency on the stability of the detected object in the image, which will be further explored in future works.

6 Conclusions and Future Work

In this paper, a fully-autonomous aerial robotic system for executing Search and Rescue (SAR) missions in cluttered indoor environments has been presented. The aerial robotic system developed in this work is based on the combination of a complete hardware configuration and a flexible system architecture which provide the appropriate capabilities for performing very high-level missions in a fully unsupervised manner. These capabilities include a dynamic mission planning system that allows mission re-planning in real-time, a reactive collision avoidance navigation system based on laser information, a complete multi-sensor fusion system for accurate pose estimation and altitude filtering, an accurate object recognizer component based on a Convolutional Neural Network (CNN) model, and a novel Image-Based Visual Servoing (IBVS) algorithm using deep Reinforcement Learning (RL) for object interaction and following, among others.

Regarding the SAR paradigm, in this paper we have focused on object recognition and object interaction tasks, which we consider still as an open field of research in SAR missions and have been addressed in this work by means of learning-based techniques. For object recognition purposes, several supervised learning classifiers have been extensively evaluated for target/background segmentation. The final selected model consists of a 7-layered CNN which exhibits a good compromise between accuracy and computational cost, tackling the hard computational constraints of a real-time aerial robot. With respect to object interaction and following, a recent deep reinforcement learning algorithm, named Deep Deterministic Policy Gradients (DDPG), has been adapted in order to perform IBVS tasks. For training and evaluating the RL-IBVS algorithm, our own reinforcement learning framework has been developed. The proposed RL framework integrates a deep RL agent (e.g. DDPG) with a simulation environment for aerial robotic platforms (e.g. RotorS Gazebo simulator). After a thorough evaluation of the proposed RL-IBVS taking as baseline a classic IBVS controller, it has been demonstrated that reinforcement learning techniques can be efficiently trained and used for solving several tasks in SAR scenarios such as the object delivering maneuvers studied in this work.

The proposed system has been thoroughly evaluated by means of several simulated and real flight experiments. The RL-IBVS algorithm has been validated both on simulated and real scenarios with static and dynamic targets. Additionally, another set of experiments have been designed for validating the whole system on a complete SAR mission conducted in a cluttered indoor scenario, revealing its appropriate capabilities for the accomplishment of such high-level missions. One of the key capabilities of our system for the accomplishment of such missions is the versatile coordination between the mission planner and the perception com-

ponents, providing high-level decision-making capabilities to the system, which is able to efficiently respond to different events that can occur during SAR missions (e.g. target recognized).

The development of the proposed system has motivated several future research directions that are summarized here. One of these directions is aimed towards the accomplishment of SAR missions using a swarm of aerial robots. We believe that an appropriate coordination of a team of UAVs can lead towards the execution of SAR missions in a more optimized manner. As it has been stated throughout this document, object interaction is one of our major concerns. For this reason, another future line of research will be focused on the exploration of additional deep reinforcement learning techniques applied to object interaction tasks such as object release, object grasping, etc, which will increase the functionalities of the proposed RL framework. According to the outstanding results obtained in this document by using learning-based techniques, the extension of the capabilities of other components in the architecture (e.g. reactive navigation) using learning-based approaches will be considered.

Acknowledgements This work was supported by the Spanish Ministry of Science (Project DPI2014-60139-R). The LAL UPM and the MONCLOA Campus of International Excellence are also acknowledged for funding the predoctoral contract of the corresponding author.

References

1. Abrahamsen, H.B.: A remotely piloted aircraft system in major incident management: concept and pilot, feasibility study. BMC emergency medicine **15**(1), 12 (2015)
2. Achtelik, M., Bachrach, A., He, R., Prentice, S., Roy, N.: Autonomous navigation and exploration of a quadrotor helicopter in gps-denied indoor environments. In: First Symposium on Indoor Flight, 2009. Citeseer (2009)
3. Bachrach, A., He, R., Roy, N.: Autonomous flight in unknown indoor environments. International Journal of Micro Air Vehicles **1**(4), 217–228 (2009)
4. Bavle, H., Sanchez-Lopez, J.L., Rodriguez-Ramos, A., Sampedro, C., Campoy, P.: A flight altitude estimator for multirotor uavs in dynamic and unstructured indoor environments. In: 2017 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 1044–1051 (2017). DOI 10.1109/ICUAS.2017.7991467
5. Bejiga, M.B., Zeggada, A., Nouffidj, A., Melgani, F.: A convolutional neural network approach for assisting avalanche search and rescue operations with uav imagery. Remote Sensing **9**(2), 100 (2017)
6. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: Openai gym. arXiv preprint arXiv:1606.01540 (2016)
7. Caruana, R., Niculescu-Mizil, A.: An empirical comparison of supervised learning algorithms. In: Proceedings of the 23rd international conference on Machine learning, pp. 161–168. ACM (2006)
8. Chang, C.C., Lin, C.J.: LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology **2**, 27:1–27:27 (2011). Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
9. Chaumette, F.: Potential problems of stability and convergence in image-based and position-based visual servoing. In: The confluence of vision and control, pp. 66–78. Springer (1998)
10. Chaumette, F., Hutchinson, S.: Visual servo control. i. basic approaches. IEEE Robotics Automation Magazine **13**(4), 82–90 (2006). DOI 10.1109/MRA.2006.250573
11. Chaumette, F., Malis, E.: 2 1/2 d visual servoing: a possible solution to improve image-based and position-based visual servoings. In: Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on, vol. 1, pp. 630–635. IEEE (2000)

12. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, vol. 1, pp. 886–893. IEEE (2005)
13. Doherty, P., Rudol, P.: A uav search and rescue scenario with human body detection and geolocalization. In: Australian Conference on Artificial Intelligence, vol. 4830, pp. 1–13. Springer (2007)
14. Dorigo, M., Colombetti, M.: Robot shaping: an experiment in behavior engineering. MIT press (1998)
15. Erdos, D., Erdos, A., Watkins, S.E.: An experimental uav system for search and rescue challenge. IEEE Aerospace and Electronic Systems Magazine **28**(5), 32–37 (2013)
16. Erez, T., Smart, W.D.: What does shaping mean for computational reinforcement learning? In: Development and Learning, 2008. ICDL 2008. 7th IEEE International Conference on, pp. 215–219. IEEE (2008)
17. Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J.: LIBLINEAR: A library for large linear classification. Journal of Machine Learning Research **9**, 1871–1874 (2008)
18. Fleck, M.: Usability of lightweight defibrillators for uav delivery. In: Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems, pp. 3056–3061. ACM (2016)
19. Furrer, F., Burri, M., Achtelik, M., Siegwart, R.: Rotors-a modular gazebo mav simulator framework. In: Robot Operating System (ROS), pp. 595–625. Springer (2016)
20. Gatteschi, V., Lamberti, F., Paravati, G., Sanna, A., Demartini, C., Lisanti, A., Venezia, G.: New frontiers of delivery services using drones: A prototype system exploiting a quadcopter for autonomous drug shipments. In: Computer Software and Applications Conference (COMPSAC), 2015 IEEE 39th Annual, vol. 2, pp. 920–927. IEEE (2015)
21. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 580–587 (2014)
22. Grzonka, S., Grisetti, G., BUrgard, W.: A fully autonomous indoor quadrotor. IEEE Transactions on Robotics **28**(1), 90–100 (2012)
23. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
24. Kohlbrecher, S., Meyer, J., von Stryk, O., Klingauf, U.: A flexible and scalable slam system with full 3d motion estimation. In: Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR). IEEE (2011)
25. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp. 1097–1105 (2012)
26. Laud, A.D.: Theory and application of reward shaping in reinforcement learning. Tech. rep. (2004)
27. Lee, A.X., Levine, S., Abbeel, P.: Learning visual servoing with deep features and fitted q-iteration. arXiv preprint arXiv:1703.11000 (2017)
28. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015)
29. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: European conference on computer vision, pp. 21–37. Springer (2016)
30. Malis, E.: Improving vision-based control using efficient second-order minimization techniques. In: Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on, vol. 2, pp. 1843–1848. IEEE (2004)
31. Marchand, É., Spindler, F., Chaumette, F.: Visp for visual servoing: a generic software platform with a wide class of robot control skills. IEEE Robotics & Automation Magazine **12**(4), 40–52 (2005)
32. Marder-Eppstein, E., Berger, E., Foote, T., Gerkey, B., Konolige, K.: The office marathon: Robust navigation in an indoor office environment. In: International Conference on Robotics and Automation (2010)
33. Meier, L., Tanskanen, P., Fraundorfer, F., Pollefeys, M.: Pixhawk: A system for autonomous flight using onboard computer vision. In: Robotics and automation (ICRA), 2011 IEEE international conference on, pp. 2992–2997. IEEE (2011)
34. Moore, T., Stouch, D.: A generalized extended kalman filter implementation for the robot operating system. In: Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13). Springer (2014)

35. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th international conference on machine learning (ICML-10), pp. 807–814 (2010)
36. Pestana, J., Mellado-Bataller, I., Sanchez-Lopez, J.L., Fu, C., Mondragón, I.F., Campoy, P.: A general purpose configurable controller for indoors and outdoors gps-denied navigation for multirotor unmanned aerial vehicles. Journal of Intelligent & Robotic Systems **73**(1-4), 387–400 (2014)
37. Polvara, R., Patacchiola, M., Sharma, S., Wan, J., Manning, A., Sutton, R., Cangelosi, A.: Autonomous quadrotor landing using deep reinforcement learning. arXiv preprint arXiv:1709.03339 (2017)
38. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems, pp. 91–99 (2015)
39. Rudol, P., Doherty, P.: Human body detection and geolocalization for uav search and rescue missions using color and thermal imagery. In: Aerospace Conference, 2008 IEEE, pp. 1–8. IEEE (2008)
40. Sadeghi, F., Levine, S.: (cad)^{2}rl: Real single-image flight without a single real image. arXiv preprint arXiv:1611.04201 (2016)
41. Sampedro, C., Bavle, H., Rodríguez-Ramos, A., Carrio, A., Fernández, R.A.S., Sanchez-Lopez, J.L., Campoy, P.: A fully-autonomous aerial robotic solution for the 2016 international micro air vehicle competition. In: Unmanned Aircraft Systems (ICUAS), 2017 International Conference on, pp. 989–998. IEEE (2017)
42. Sampedro, C., Bavle, H., Sanchez-Lopez, J.L., Fernandez, R.A.S., Rodriguez-Ramos, A., Molina, M., Campoy, P.: A flexible and dynamic mission planning architecture for uav swarm coordination. In: Unmanned Aircraft Systems (ICUAS), 2016 International Conference on, pp. 355–363. IEEE (2016)
43. Sanchez-Lopez, J.L., Molina, M., Bavle, H., Sampedro, C., Fernández, R.A.S., Campoy, P.: A multi-layered component-based approach for the development of aerial robotic systems: The aerostack framework. Journal of Intelligent & Robotic Systems pp. 1–27
44. Scherer, J., Yahyanejad, S., Hayat, S., Yanmaz, E., Andre, T., Khan, A., Vukadinovic, V., Bettstetter, C., Hellwagner, H., Rinner, B.: An autonomous multi-uav system for search and rescue. In: Proceedings of the First Workshop on Micro Aerial Vehicle Networks, Systems, and Applications for Civilian Use, pp. 33–38. ACM (2015)
45. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research **15**(1), 1929–1958 (2014)
46. Sun, J., Li, B., Jiang, Y., Wen, C.y.: A camera-based target detection and positioning uav system for search and rescue (sar) purposes. Sensors **16**(11), 1778 (2016)
47. Tomic, T., Schmid, K., Lutz, P., Domel, A., Kassecker, M., Mair, E., Grixa, I.L., Ruess, F., Suppa, M., Burschka, D.: Toward a fully autonomous uav: Research platform for indoor and outdoor urban search and rescue. IEEE robotics & automation magazine **19**(3), 46–56 (2012)
48. Xiang, G., Hardy, A., Rajeh, M., Venuthurupalli, L.: Design of the life-ring drone delivery system for rip current rescue. In: Systems and Information Engineering Design Symposium (SIEDS), 2016 IEEE, pp. 181–186. IEEE (2016)
49. Zhang, F., Leitner, J., Milford, M., Upcroft, B., Corke, P.: Towards vision-based deep reinforcement learning for robotic motion control. arXiv preprint arXiv:1511.03791 (2015)
50. Zhu, Y., Mottaghi, R., Kolve, E., Lim, J.J., Gupta, A., Fei-Fei, L., Farhadi, A.: Target-driven visual navigation in indoor scenes using deep reinforcement learning. In: Robotics and Automation (ICRA), 2017 IEEE International Conference on, pp. 3357–3364. IEEE (2017)
51. Zingg, S., Scaramuzza, D., Weiss, S., Siegwart, R.: Mav navigation through indoor corridors using optical flow. In: Robotics and Automation (ICRA), 2010 IEEE International Conference on, pp. 3361–3368. IEEE (2010)