Semantic-aware Active Perception for UAVs using Deep Reinforcement Learning

Luca Bartolomei, Lucas Teixeira and Margarita Chli
Vision For Robotics Lab, ETH Zürich, Switzerland

***Abstract*— This work presents a semantic-aware path-planning pipeline for Unmanned Aerial Vehicles (UAVs) using deep reinforcement learning for vision-based navigation in challenging environments. Driven by the maturity of works in semantic segmentation, the proposed path-planning architecture uses reinforcement learning to distinguish the parts of the scene that are perceptually more informative using semantic cues, in effect guiding more robust, repeatable, and accurate navigation of the UAV to the predefined goal destination. Assuming that the UAV performs vision-based state estimation, such as keyframe-based visual odometry, and semantic segmentation onboard, the proposed deep policy network continuously evaluates the optimal relative perceptual informativeness of each semantic class in view. A perception-aware path planner uses these informativeness values to perform trajectory optimization in order to generate the next best action with respect to the current state and the perception quality of the surroundings, essentially guiding the UAV to avoid flying over perceptually degraded regions. Thanks to the use of semantic cues, the policy can be trained in a large number of non-photorealistic randomly-generated scenes, and results to an architecture that is generalizable to environments with the same semantic classes, independently of their visual appearance. Extensive evaluations on challenging, photorealistic simulations reveal a remarkable improvement in robustness and success rate with the proposed approach over the state of the art in active perception.**

Video – `https://youtu.be/Ra03whUBVnc`

I. Introduction

The capability of a robot to achieve precise localization and perform safe navigation is a fundamental skill for genuinely autonomous systems. In the case of Unmanned Aerial Vehicles (UAVs), these abilities play an even more significant role due to the agility of these platforms. For example, in industrial inspection and package delivery, UAVs have to map the environment with sufficient accuracy to avoid collisions with obstacles and reach the goal position accurately. However, both real-time localization and mapping are still open problems since the most used sensors, such as GPS, can fail numerous situations due to bad weather, jamming, or in the presence of mountains or tall buildings. Avoiding failures in localization systems for aerial robots is fundamental because, as opposed to ground robots, they cannot instantaneously halt all motion, but instead, they either need to reach an emergency landing spot or to continue flying towards their destination. Vision-based Simultaneous Localization And Mapping (SLAM) is now well accepted for UAV navigation [1]; however, its performance (i.e. robustness, accuracy) is heavily influenced by the condition of the navigation environment. For example, vision-based SLAM's accuracy degrades significantly (and even fails) in the presence of dynamic objects (e.g. people, cars, trees moving in the wind) and areas with no texture or textures exhibiting specularities (e.g. oceans, lakes).

This work was supported by the Swiss National Science Foundation (SNSF, Agreement no. PP00P2183720), NCCR Robotics, the Amazon Research Awards and the HILTI group.

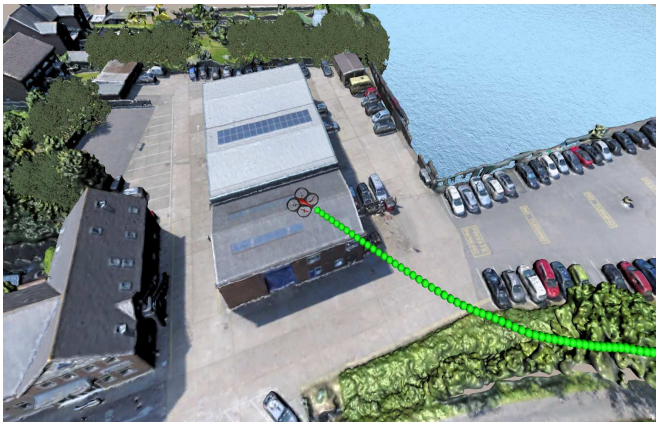


Fig. 1: 3D-view of the path flown by the UAV towards the destination in the *Baxall* experiment. The travelled path (in green) indicates that the robot is able to successfully avoid areas that are problematic for camera-based state estimators, such as water, while navigating on top of reliable textures, such as terrain and buildings.

This work proposes an active perception path-planning algorithm for vision-based UAV navigation based on Reinforcement Learning, guiding the UAV to reach a predefined goal position while avoiding texture-less and poorly textured areas, as well as regions that are less reliable for localization, such as lakes, streets with moving cars, and trees (Fig. 1). Given as input a semantic mask of the scene, the proposed framework outputs online the optimal policy that allows steering the robot away from potentially dangerous areas by assigning a perceptual informativeness score to each semantic class. We train our policy in a set of non-photorealistic randomly-generated 3D models, and we test it in a set of previously unseen environments, including photorealistic 3D models of real-life places. The algorithm is agnostic to the semantic segmentation algorithm and works with any set of semantic classes.

In brief, the contributions of this work are the following:

- the design of a reinforcement learning-based semantic-aware path-planning algorithm for vision-based aerial navigation in challenging environments,
- the design of a training framework and a policy archi-

tecture able to generalize to unseen environments, and
- an extensive quantitative evaluation of the performances of the proposed system with respect to the state-of-the-art in active perception in photorealistic simulations.

II. Related Work

The performance of Visual SLAM algorithms, providing estimates of the robot's pose and surroundings, is greatly influenced by the motion and the path followed by the robot, as sharp and highly dynamic movements can lead to wrong estimates or even failure. Active perception constitutes a fundamental step towards the creation of fully autonomous systems, able to cope with the uncertainties present in a real mission. The core concept in active perception is that sensory performance can be improved by proper selection of motion-control actions [2]. Modern works propose solutions integrating perception, path planning and control in a unified framework [3], [4]. From a theoretical perspective, the problem of active perception is formulated as Partially Observable Markov Decision Processes (POMDPs) [5], which are in general complex to solve. The drive to find efficient solutions led to the conception of belief roadmaps [6] and belief trees [7], while more recently, receding horizon approaches and perception-aware path-planning algorithms emerged [8], [9], showing how texture-less regions of the environment could be avoided [10] using the internal state of a SLAM system.

One of the most relevant rival to this work is the perception-aware planner in [8], proposing to generate motion primitives and evaluate them by considering the concentration of landmarks in each area, the probability of collision and the distance to the goal. In [11], however, it is demonstrated that landmark concentration alone is not enough to identify the best areas to fly through, and instead, a perception-aware planner employing semantics to evaluate the quality of the candidate areas for navigation is proposed. Using the semantic segmentation of the SLAM input image as an additional cue in a perception-aware planning algorithm, [11] was shown to achieve the best results in terms of accuracy and robustness of localization to date. Nonetheless, it is not able to adapt dynamically to changes in the navigation area at flight time, as it assigns fixed binary informativeness scores to every semantic class in the scene. Furthermore, this approach requires manual score definition, which can become challenging with an increasing number of classes. Addressing this, this work proposes a pipeline that tackles these limitations using reinforcement learning-based active perception, dynamically identifying the most reliable regions for localization from semantic information. Reinforcement Learning (RL) can allow autonomous systems to learn policies for complex tasks, such as control [12] and obstacle avoidance [13], reducing the engineering effort to the design of a suitable reward function [14]. In combination with deep neural networks, deep RL provides a powerful tool capable of mapping high-dimensional sensory inputs to optimal actions, showing promising results in fields, such as autonomous driving [15] and robot navigation [16], [17].

In this spirit, this work tackles the limitations of [11] using a RL-trained deep-policy-based perception-aware path-planning pipeline that is able to reduce the drift in vision-based SLAM, without the need for manual setting of informativeness scores, but with real-time and onboard auto-tuning of scores instead. In a nutshell, given an input semantic mask of the scene, the proposed planner learns to encourage navigation through regions suitable for visual localization, by adapting to the scene online and assigning different importance to the different semantic classes. Our approach exhibits great improvement in the success rate of missions when compared to purely reactive planning and to state-of-the-art perception-aware planners, such as the ones proposed in [8] and [11].

III. Method

The problem targeted in this work is for a UAV to reach a predefined goal pose, while minimizing the drift in the onboard vision-based SLAM algorithm. Our main objective is to identify and fly through suitable areas for visual localization, avoiding spatial regions that lead to high error and pose estimation failure. We formulate this as a path-planning problem, where a deep RL agent is trained to identify reliable areas from semantically labelled images. Semantics is a valuable source of information for the agent, as drift in pose estimation is generally consistent for areas belonging to the same semantic class in case of good illumination conditions. Moreover, using mid-level representations, such as semantic masks as input is proven to yield better generalization of the policy [18]. At the same time, our architecture allows to decouple the problems of semantic labeling and path planning. This decoupling is essential for deployability, as learning directly the mapping from raw camera data to perceptual informativeness for each semantic class requires an implicit semantic segmentation step, which would take extremely long training time in an RL fashion.

Given that real-time semantic segmentation has already reached a high degree of maturity [19], this work assumes a high-quality semantic segmentation image as input. Using an adaptation of the approach of [11], we generate smooth and collision-free trajectories in a receding horizon fashion, considering the limitations of keyframe-based Visual Odometry (VO) systems used for pose estimation. Semantic classes are mapped to a certain score of perceptual informativeness by a policy trained in randomly generated scenes using Unity[1] and the Flightmare frameworks [20]. These scores, recorded per class, are successively fed to the path planner, which encourages navigation in areas that cause less drift and keep the best 3D landmarks estimated by the VO pipeline in the field of view of the camera.

A. System Overview

As shown in Fig. 2, the proposed pipeline consists of three main components; a monocular camera-based pose estimation module, a deep RL agent and a path-planning

[1] https://unity.com/

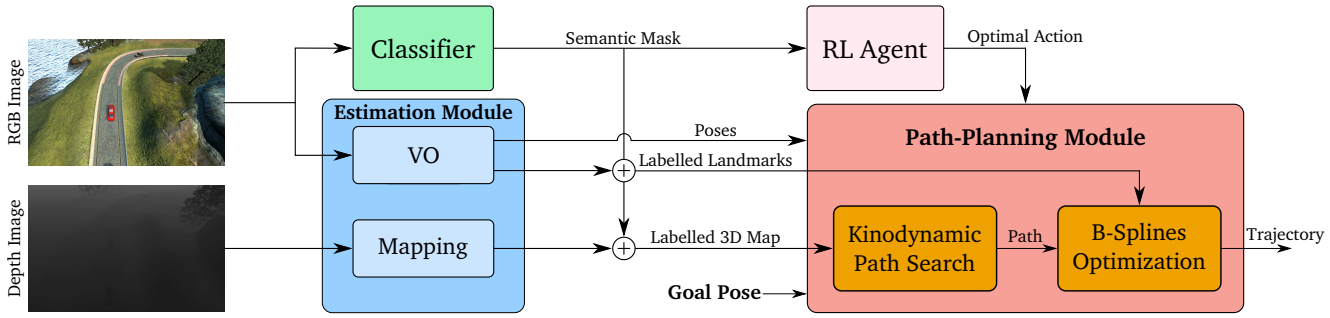


Fig. 2: Schematic overview of the pipeline. In the Estimation Module, we process the sensor inputs to estimate the pose of the UAV and the 3D landmarks. The landmarks, together with the occupancy map obtained from the depth images, go through a classification step, where each point is assigned to a semantic class. The RL agent utilizes the semantic mask to generate the optimal action, which consists of the set of weights to assign to each semantic class. The optimal choice of weights is then communicated to the perception-aware path planner from [11]. Finally, the Path-Planning Module outputs the optimal trajectory, while considering both the dynamics of the platform and perceptional quality.

module. We assume a stream of RGB and depth images is available. The depth images are used to generate 3D reconstructions of the local surroundings of the robot, while the VO algorithm utilizes the RGB images to estimate the robot's pose. The dense point cloud is stored in an occupancy map employing a 3D circular buffer [21].

We employ the keyframe-based VO system ORB-SLAM [22] without loop closures to estimate the pose of the camera, but our pipeline is agnostic to the adopted VO algorithm. Since ORB-SLAM is a monocular vision-only system, the scale cannot be retrieved. However, as our agent is trained in simulation, we have access to ground-truth information that is used to re-scale the estimated position and the 3D landmarks.

Both the 3D occupancy map and the sparse landmarks go through a classification step providing the semantic labels for all the points, which are then fed to the path-planning module. Given that the detection of moving objects (e.g. cars, people, trees) and ground classification are not the core contribution of this work, we use ground-truth semantic masks in our experiments, but there are a plethora of off-the-shelf algorithms available [23], [24]. The semantic mask also serves as input to the deep RL policy, which outputs values associated with the perceptual informativeness of each semantic class. These values are communicated to the planner, which uses them to reason about the next best action. The policy output is utilized as a set of weights in the objective function to be optimized in the path-generation step, favoring the tracking and triangulation of points belonging to parts of the scene useful for camera-based state estimation. In the next section, the deep RL agent structure and its interface with the path-planning module are explained in detail.

B. Perception-aware Path Planning

Our objective is to let the agent fly through areas well-suited for VO by learning which semantic classes are less likely to generate localization drift. The robot learns this by interacting with the environment, selecting an action, and receiving a reward value as feedback. Here, an action corresponds to a set of weights for each semantic class in the perception objective function, to be optimized in the path-planning module. The planning pipeline is an adaptation of [11], where originally semantic weights are manually assigned, and used in a kinodynamic A^* path search, followed by B-Spline trajectory optimization.

1) Kinodynamic Path Search: In the first planning step, the aim is to encourage navigation in well-textured areas. The path search is limited to the robot's position in $\mathbb{R}^3$ and, using the differential flatness of the multirotor systems, the trajectory is represented as three independent time-parametrized polynomial functions $\mathbf{p}(t)$:

$$\mathbf{p}(t) := [p_x(t), p_y(t), p_z(t)]^T,\ p_d(t) = \sum_{k=0}^{K} a_{d,k} t^k \qquad (1)$$

with $d \in \{x, y, z\}$. We assume the multirotor system to be linear and time-invariant, and we define its state as $\mathbf{s}(t) := [\mathbf{p}(t)^T, \dot{\mathbf{p}}(t)^T, \ldots, \mathbf{p}^{(n-1)}(t)^T]^T \in \chi \subset \mathbb{R}^{3n}$, with control input $\mathbf{u}(t) := \mathbf{p}^{(n)}(t) \in \mathcal{U} := [-u_{max}, u_{max}]^3 \subset \mathbb{R}^3$ and $n = 2$, corresponding to a double integrator. Given the current robot's state $\mathbf{s}(t)$, the control input $\mathbf{u}(t)$ and a labelled occupancy map $\mathcal{M}$ of the environment, we define the cost of a trajectory as

$$\mathcal{J}(T) = \int_0^T \Big(w_u \|\mathbf{u}(t)\|^2 + \sum_{j=0}^{N} w^j d_M^j(\mathbf{p}(t), \mathcal{M}) \Big) dt + w_T T, \qquad (2)$$

where $\|\mathbf{u}(t)\|^2$ is the control cost; $d_M^j(\mathbf{p}(t), \mathcal{M})$ represents a penalty for navigating far away from areas associated to the semantic class $j \in \{0, \cdots, N\}$ with N the total amount of classes; and T is the total time of the trajectory.
The terms w_u and w_T are constant weights associated with the respective costs, while the w^j is the weight associated with the semantic class j assigned by the current optimal policy and is subjected to changes as the agent gathers additional experience. The cost $d_M^j(\mathbf{p}(t), \mathcal{M})$ is defined as

$$d_M^j(\mathbf{p}(t), \mathcal{M}) := \sum_{\mathbf{v}_j \in \mathcal{M}_j \subseteq \mathcal{M}} d_v(\mathbf{p}(t), \mathbf{v}_j) = \sum_{\mathbf{v}_j \in \mathcal{M}_j \subseteq \mathcal{M}} d_{xy}(\mathbf{p}(t), \mathbf{v}_j) + d_z(\mathbf{p}(t), \mathbf{v}_j), \qquad (3)$$

where $\mathbf{v}_j = [v_x, v_y, v_z]^T$ are the voxels of the occupancy map $\mathcal{M}$ with semantic label j, indicated with $\mathcal{M}_j \subseteq \mathcal{M}$. The

cost $d_M^j(\mathbf{p}(t), \mathcal{M})$ is composed of the two potential functions that are calculated as

$$d_{xy}(\mathbf{p}(t), \mathbf{v}_j) := (p_x(t) - v_x)^2 + (p_y(t) - v_y)^2, \quad (4)$$

and, by defining $\Delta z := |p_z(t) - v_z|$,

$$d_z(\mathbf{p}(t), \mathbf{v}_j) := d^* \Delta z + \frac{1}{2}\frac{{d^*}^4}{\Delta z^2} - \frac{3}{2}{d^*}^2, \quad (5)$$

where d^* controls the minimum height of the robot with respect to the voxels in $\mathcal{M}_j$. In order to speed up the search in the A^* algorithm, we use the same heuristic as in [11], adapted to match the new cost definitions.

2) Trajectory Optimization: While the trajectory computed in the path-searching step encourages navigation towards informative areas, the trajectory optimization step leverages the additional information given by the 3D landmarks in the VO module. The trajectory $\boldsymbol{\pi}(t)$ is parametrized as a uniform B-Spline of degree K and it is defined as

$$\boldsymbol{\pi}(t) = \sum_{i=0}^{N} \mathbf{q}_i B_{i,K-1}(t), \quad (6)$$

where $\mathbf{q}_i$ are the control points at time t_i with $i \in \{0, \ldots, N\}$, and $B_{i,K-1}(t)$ are the basis functions. Each control point in $\{\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_N\}$ encodes both the position and orientation of the robot, i.e. $\mathbf{q}_i := [x_i, y_i, z_i, \theta_i]^T \in \mathbb{R}^4$ with $\theta_i \in [-\pi, \pi)$. The B-Spline is optimized in order to generate smooth, collision-free trajectories, encouraging the triangulation and tracking of high-quality landmarks. For a B-Spline of degree K defined by $N+1$ control points $\{\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_N\}$, our optimization acts on $\{\mathbf{q}_K, \mathbf{q}_{K+1}, \ldots, \mathbf{q}_{N-K}\}$ while keeping the first and last K control points fixed due to boundary constraints. The optimization problem is formulated as a minimization of the cost function

$$\mathcal{F}_{TOT} = \lambda_s \mathcal{F}_s + \lambda_f \mathcal{F}_f + \lambda_c \mathcal{F}_c + \lambda_l \mathcal{F}_l + \lambda_v \mathcal{F}_v, \quad (7)$$

where $\mathcal{F}_s$ is the smoothness cost; $\mathcal{F}_c$ is the collision cost; $\mathcal{F}_f$ is a soft limit on the derivatives (velocity and acceleration) over the trajectory; $\mathcal{F}_l$ is the penalty associated with losing track of the high-quality landmarks currently in the field of view; and $\mathcal{F}_v$ is a soft constraint on the co-visibility between control points of the spline. The coefficients $\lambda_s, \lambda_c, \lambda_f, \lambda_l$ and λ_v are the fixed weights associated to each cost. While we maintain the original cost formulations, similarly to Eq. 2, we propose a novel perception cost that accommodates multiple semantic classes:

$$\mathcal{F}_l = -\sum_{j=0}^{N} \sum_{\mathbf{l}_C \in \mathcal{L}_j^C} w^j \prod_{k=0}^{5} o_k(\mathbf{q}, \mathbf{l}_C), \quad (8)$$

where $\mathcal{L}_j^C$ is the set of 3D landmarks associated to class j expressed in camera frame C, and o_k a smooth indicator function determining the visibility of landmark $\mathbf{l}_C$ from the control pose $\mathbf{q}$ [11]. The optimal set of weights for each semantic class is computed in real-time by a policy modeled as a neural network, trained in an episode-based deep RL-fashion. In the next section, we introduce our policy architecture and explain the training process in detail.

C. Deep RL Policy

The RL policy maps from semantic masks to actions, employing an Actor-Critic model with the architecture shown in Fig. 3. Here the action consists of the set of weights $w^j \in [0, 1]$ with $j \in \{0, \cdots, N\}$ used by the planner in Eq. 2 and Eq. 8, where N is the number of semantic classes in the scene fixed by the user. The Actor and the Critic networks share the first part, composed of a 3-layer Convolutional Neural Network (CNN), followed by a Long-Short Term Memory (LSTM) module. The LSTM is responsible for the memory of the policy and captures spatial dependencies that would otherwise be hard to identify, as some semantic classes can be linked together (e.g. cars and roads). The final part of the Critic consists of two Fully Connected (FC) layers composed of 64 units, while the optimal action is output by the Actor from three FC layers with 128 units each. Policy optimization is then performed at fixed-step intervals employing the on-policy algorithm PPO [25]. Moreover, in order to reduce the hyperspace dimension, we convert the color mask into grayscale and downsample the resulting image. We utilize the semantic image as input because this mid-level visual representation is more generic than the raw color image, and it is demonstrated to allow faster training and improved policy performances [18]. Using the raw image from the sensor directly makes the training cumbersome and generalization of the policy harder, as the same semantic class can have different textures and visual appearances (e.g. in the cases of terrain, vegetation, and cars).

The training of the policy is then performed based on the data and the rewards collected in each episode. Since we aim to reduce the localization error and increase the chances of getting to the destination, the reward function received by the RL agent at step t is defined as

$$R_t(\mathbf{p}(t), e(t)) := R_S + w_E R_E(e(t)) + w_G R_G(\mathbf{p}(t)), \quad (9)$$

where R_S is the survival reward, R_E is associated to the localization error $e(t)$ and R_G to the progress towards the goal position. The survival reward is assigned at every step, until tracking is lost:

$$R_S := \begin{cases} 0 & if\ lost\ track \\ 1 & otherwise \end{cases}. \quad (10)$$

Note that we do not penalize explicitly with a negative final reward when the VO system loses track, in order not to penalize promising actions that lead to high errors due to faulty initialization of the visual tracks at the beginning of the episode.

The reward associated to the localization error is also assigned in every step, and encourages to take actions that reduce the drift in the VO system:

$$R_E(e(t)) := \begin{cases} R_E^{max} & if\ e(t) \leq e_{min} \\ 0 & if\ e(t) \geq e_{max} \\ R_E^{max} exp(-(e(t) - e_{min})) & otherwise \end{cases}, \quad (11)$$

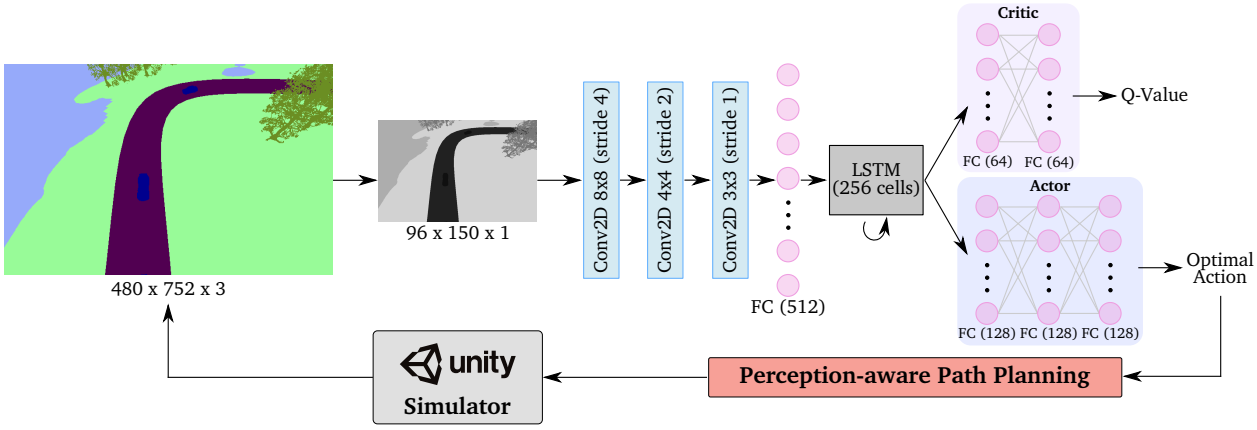
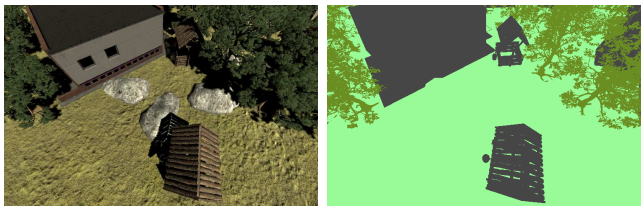


Fig. 3: Schematic representation of the policy architecture and its interaction with the path planner. The semantic images are rendered in the Unity engine and are first converted into grayscale images and downsampled. The obtained images go through the Actor-Critic network, composed of three CNN layers, a linear FC layer and a LSTM module. The Critic then outputs the Q-Value from two FC layers, while the Actor feeds the optimal action to the planner, consisting of a set of weights associated to each semantic class. The policy optimization is performed by means of the PPO algorithm [25].



(a) Visual Image. (b) Semantic Mask.

Fig. 4: Example of visual image and its corresponding semantic mask rendered by the Unity engine at training time. The training scenes are generated by the game engine, placing objects (e.g. trees, rocks, cars, houses) randomly in the space.

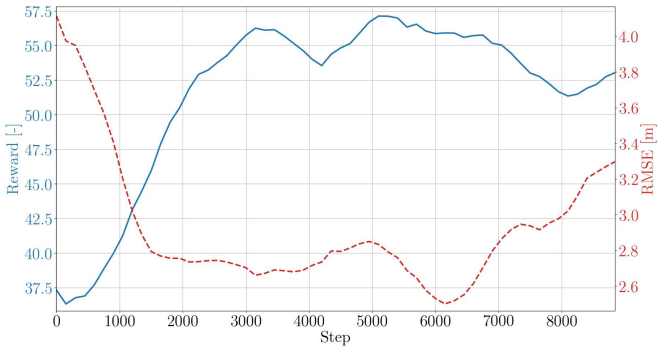


Fig. 5: Reward and RMSE trends over the training steps, shown as blue solid and red dashed curves, respectively. In the initial phase of training, the sharp increase in the reward demonstrates that the RL agent learns quickly the most reliable semantic classes for localization, followed by a plateau, where the optimal policy is reached.

where e_{min} and e_{max} are the minimum and the maximum acceptable errors, respectively, and R_E^{max} is the maximum reward value. Finally, the last component of the reward function favours the progress towards the goal position $\mathbf{p}_G(t)$ and is inversely proportional to the distance between the current robot position and the destination:

$$R_G(\mathbf{p}(t)) := R_G^{max} \frac{1}{\|\mathbf{p}(t) - \mathbf{p}_G(t)\|}, \tag{12}$$

where R_G^{max} is the maximum achievable reward value. When the agent reaches the goal, it receives a final reward equal to R_G^{max}.

At the beginning of each episode, we place the robot at a given starting position, initialize the VO tracking system, and set an end target position. The agent navigates towards the goal generating trajectories by optimizing the cost functions defined in Eq. 2 and Eq. 7, given the set of weights output by the current policy. During the flight, the agent monitors the localization error and the episode ends when either the goal is reached or the VO system loses track.

IV. Experiments

A. Policy Training

In order to maximize the generalization of the policy and to avoid overfitting to a specific scene, we train the agent in a set of randomly generated environments using the game engine Unity and the Flightmare framework [20]. The simulated UAV is equipped with a front-looking camera mounted with a pitch of 60°. The Unity engine provides the images required by the VO systems, the semantic masks, as well as the depth images to perform obstacle avoidance. Based on [26], the depth images are corrupted by zero-mean Gaussian noise in order to mimic the noise in real sensors, such as stereo or depth cameras.

At the beginning of each episode, a new scene is generated, and a goal destination is placed randomly in the scene (Fig. 4). The UAV starts navigating towards the goal position, and the episode ends when either the goal is reached or the VO system loses track. The agent outputs actions at fixed time intervals (or *steps*), communicates them to the perception-aware planner, and collects the reward as feedback. In the first episode of the training process, the policy is initialized randomly. The training continues until the maximum number of steps across all episodes is reached. Here, we set this maximum to 9000 steps.

All scenes are characterized by the same semantic classes (pavement/road, terrain, water, trees, buildings and cars). We use common classes used by several state-of-the-art semantic-segmentation algorithms, but our system can handle any set of classes. The reward parameters are set to $R_E^{max} = 5$ and $R_G^{max} = 50$, with minimum and maximum localization

Experiment	Ours	[11]	[8]	Reactive
Racetrack				
Success Rate [%]	**100**	80	35	40
RMSE [m]	**2.4 ± 0.2**	2.5 ± 0.2	8.4 ± 4.0	6.4 ± 0.9
Missed Distance [m]	**4.5 ± 0.7**	4.6 ± 0.4	17.3 ± 11.8	6.5 ± 1.3
Path Length [m]	458.4 ± 11.1	464.1 ± 7.4	476.4 ± 56.0	**402.4 ± 1.3**
Villages				
Success Rate [%]	**90**	85	10	75
RMSE [m]	**1.2 ± 0.1**	1.3 ± 0.4	2.8 ± 1.0	5.8 ± 1.3
Missed Distance [m]	**3.8 ± 0.9**	4.5 ± 1.2	9.8 ± 1.5	10.8 ± 2.5
Path Length [m]	378.2 ± 12.8	377.5 ± 10.5	343.7 ± 3.5	**338.4 ± 6.5**
Highway				
Success Rate [%]	**95**	90	35	80
RMSE [m]	**2.1 ± 0.4**	6.1 ± 1.4	3.1 ± 2.1	2.7 ± 0.5
Missed Distance [m]	**6.4 ± 1.2**	8.6 ± 1.1	8.6 ± 3.9	**6.4 ± 0.6**
Path Length [m]	**583.7 ± 33.3**	638.0 ± 26.8	673.2 ± 11.3	660.0 ± 88.1
Baxall				
Success Rate [%]	**100**	**100**	45	70
RMSE [m]	**0.9 ± 0.2**	2.2 ± 0.2	3.7 ± 2.3	1.8 ± 0.2
Missed Distance [m]	**1.8 ± 0.3**	2.8 ± 0.7	3.6 ± 0.8	2.5 ± 1.2
Path Length [m]	439.0 ± 10.7	424.3 ± 12.1	738.1 ± 10.9	**397.8 ± 8.3**
Fraser				
Success Rate [%]	**100**	90	10	20
RMSE [m]	**1.3 ± 0.1**	2.3 ± 0.3	2.0 ± 0.1	2.3 ± 0.3
Missed Distance [m]	**5.0 ± 1.4**	5.7 ± 1.4	5.4 ± 0.1	5.2 ± 1.6
Path Length [m]	**644.4 ± 9.6**	652.7 ± 9.8	747.5 ± 10.0	678.9 ± 40.3
House Garden				
Success Rate [%]	**100**	70	15	20
RMSE [m]	**0.8 ± 0.1**	2.6 ± 0.7	1.6 ± 0.5	2.1 ± 0.1
Missed Distance [m]	**2.6 ± 0.4**	5.5 ± 0.9	3.8 ± 0.5	7.1 ± 0.3
Path Length [m]	435.1 ± 17.8	500.0 ± 81.9	**420.3 ± 6.8**	503.1 ± 79.0

TABLE I: Results of the experiments in the six simulated scenes. We report the success rate of the goal-reaching task for the different planners, as well as the average localization RMSE, the missed distance from the goal and path length, all averaged over 20 runs. The averages are computed only for the experiments where the goal is reached. The first three scenes (*Racetrack*, *Villages* and *Highway*) are built using the game engine Unity, while the others (*Baxall*, *Fraser* and *House Garden*) are 3D photorealistic models of real-life places obtained from photogrammetry. The best performance in each scene is shown in bold.

errors $e_{min} = 0.5\,\mathrm{m}$ and $e_{max} = 2.5\,\mathrm{m}$. To compute the total reward as in Eq. 9, the weights for components associated to the localization error and to the goal-reaching task are set to $w_E = 3$ and $w_G = 0.1$, respectively.

The training performance of the agent is shown in Fig. 5, where the reward and the Root Mean Square Error (RMSE) of the VO system with respect to ground-truth position are reported. As depicted by the initial sharp increase in the reward curve, the agent learns quickly to identify the semantic classes that allow robust localization, resulting in a decrease in the pose estimation error. The training performance successively decreases, as visible from the plateau in the reward curve and the small increase in the translational error. Despite the decrease due to slightly higher RMSE, the reward does not drop, as the agent is able to reach the target destination more frequently. This indicates that the optimal behavior is reached and that the oscillations in performance are linked more to the randomness of the scene and consequently, of the VO algorithm's performance rather than to wrong action selection.

B. Tests in Previously Unseen Environments

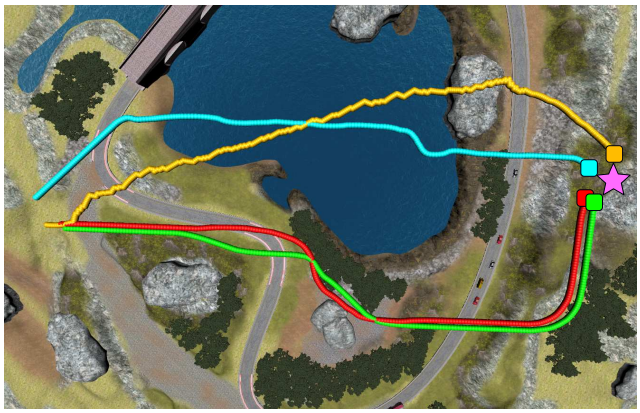
A series of experiments in challenging scenes are conducted to show the effectiveness of the proposed method. The test environments are not experienced by the policy at training time, and, similarly to [11], they are specifically built to create problematic situations for camera-based odometry algorithms, such as water with moving waves, cars, and trees moving in the wind. The scenes are either built from scratch using the Unity game engine or obtained from 3D models of real-life places from photogrammetry[2]. In total, we test six different scenes as shown in Fig. 6.

1) Test Set-Up: In each scenario, we command the UAV to fly to a set of 20 goal positions placed randomly within a target area. To evaluate the approach, we use the state estimation error as a metric since we do not encounter crashes against obstacles, despite high localization errors, thanks to the usage of local occupancy maps for obstacle avoidance. We compare the performance of our learning-based method with a purely reactive planning strategy, the semantic-based path planner from [11], and the perception-aware method proposed in [8]. The reactive planner is based on our planner without considering perception, i.e. ignoring the action from the RL agent and setting the weight w^j in Eq. 2 and λ_l and λ_v in Eq. 7 to zero.
The semantic-based planner of [11] uses its original implementation, but we modify it to accommodate an unlimited number of semantic classes, since the original version considers only two (i.e. terrain and water). We tune it once for all scenes to trust only reliable classes for vision-based localization, such as terrain, buildings and roads. As this planner assigns binary weights to each class, it is conceptually equivalent to a supervised learning classifier, trained to discriminate only between reliable and unreliable classes, but unable to adapt dynamically to the environment.
In all the scenes, the goal positions are the same for all the planners for fairness of comparisons, while the initial positions may differ by a couple of meters due to the movements necessary to initialize ORB-SLAM.

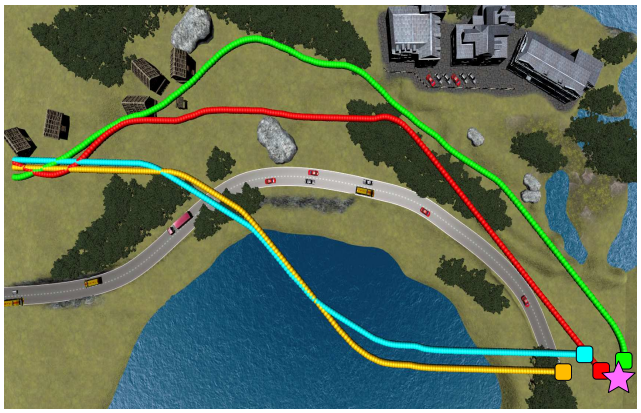
2) Result Metrics: For each experiment, we report the success rate, the translational RMSE of the VO system, the traveled path length, and the missed distance, i.e. the distance between the end position of the robot and the real position of the goal. We define a run to be successful if the goal is reached and the VO system does not lose track. Notice that we do not explicitly include the missed distance in the reward function at training time, as it is included implicitly in the minimization of the estimation error. The results are summarized in Table I, where the numerical averages are computed only for the successful runs. Fig. 6 shows the most representative trajectories for the different planners in the test scenes.

3) Result Discussion: Our deep RL-based planner reaches the highest success rate across all the experiments, showing an increase in robustness compared to the other planning strategies. More importantly, we demonstrate that our policy network can safely transfer from the randomly generated training scenes to the test environments without the need of additional fine-tuning. We also experience a general decrease

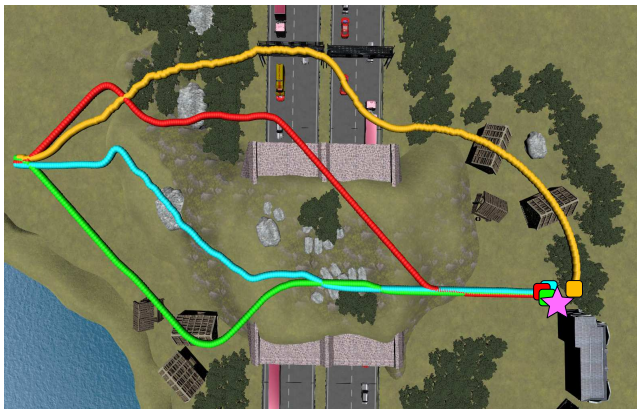
[2] `https://sketchfab.com`



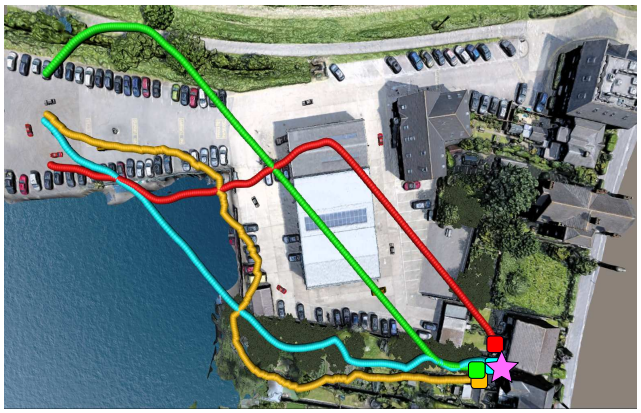
(a) Top-view of the scene *Racetrack*.



(b) Top-view of the scene *Villages*.

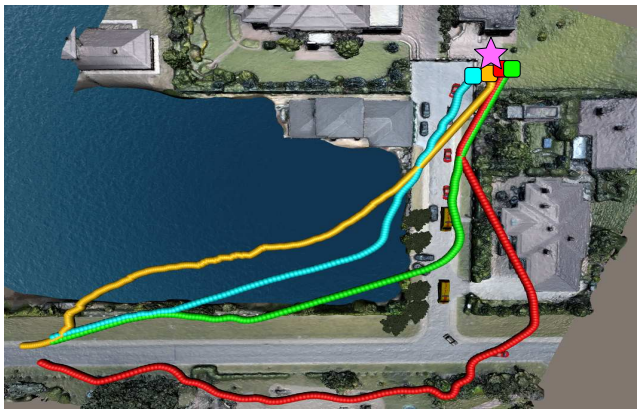


(c) Top-view of the scene *Highway*.



(d) Top-view of the scene *Baxall*.

(e) Top-view of the scene *Fraser*.



(f) Top-view of the scene *House Garden*.

Fig. 6: Top-views of the test scenes with example trajectories for the different planners, with the goal position represented with a magenta star and final positions as colored squares. As an illustration, we select one good representative successful trajectory for each planner in every scene. While our planner (green) is consistently able to avoid dangerous areas by adapting dynamically to the environment, the semantic-aware planner [11] (red) adopts a binary weighting scheme for the different semantic classes, leading to worse success rates. The reactive planner (cyan) instead finds the shortest path to the goal passing through dangerous areas (e.g. lakes, woods), while the perception-aware planner of [8] (orange) follows trails of unsuitable landmarks for localization, causing large detours and failures in the camera-based state estimation system.

in the estimation errors, especially in the experiments for the *Villages*, *Highway*, *Baxall* and *Fraser* scenes. The trajectories generated by our deep RL-based planner favor flights through the most reliable regions of the environments, such as terrain and roads, avoiding moving water and trees. However, as our approach and [11] have the same path-generation back-end, the localization errors of these pipelines are similar, since the performance of vision-based state estimators is directly affected by the path followed by the camera. This behavior is expected, since a qualitative comparison of the trajectories generated by the two planners shows their similarity, especially in *Racetrack* (Fig. 6a) and *Villages* (Fig. 6b). Moreover, [11] is tuned manually to assign a higher weight on the most reliable semantic classes in the scene, and, consequently, we expect that the behavior of their planner is close to optimal. Nonetheless, our RL-based approach reaches higher robustness, as the possibility to change the weighting scheme of the semantic classes online allows to better adapt to

changes in the scene and to avoid areas with unreliable texture for localization. This indicates that, while semantics are a powerful source of information, a binary approach, as in the case of supervised learning-based planners, is not capable of handling more complex scenes and safety requirements necessary in real-world applications.

Instead, the reactive planning strategy takes the shortest path to the goal positions, forcing the robot to fly through areas, such as water and woods that decrease the success rate and the localization accuracy. This effect is exacerbated in the experiments *Villages* and *House Garden*.

Finally, the planner by Zhang and Scaramuzza [8] has the lowest success rate in almost all the scenes. As this planner uses the landmark concentration to find the best route, it follows trails of landmarks without differentiating them on their appearance. Thus, it flies the robot to unnecessary detours trusting unreliable textures for localization (*Baxall*, *Fraser*), causing larger drift and failures in the state estimation algorithm.

Overall, while the different planning strategies give comparable performance of the vision-based state estimation (in case of a successful run), our method exhibits a remarkable and consistent increase in robustness and success rate for goal-reaching tasks in challenging scenarios. Thanks to its capacity of adapting dynamically to the scene, our deep RL-based planning pipeline can fly the UAV towards the destination safely and accurately, demonstrating a noticeable boost in performance compared to the state of the art.

V. Conclusion

In this paper, we propose a deep RL-based perception-aware path-planning architecture for goal-reaching tasks. Our approach pushes the limits of the state of the art by incorporating reinforcement learning into semantic-based active perception, allowing the robot to learn how to adapt dynamically to changes in the navigation area. Our RL agent can map from semantic to perceptual informativeness by assigning importance weights to each semantic class in the scene. We demonstrate that our policy architecture can be easily generalizable without fine-tuning to a set of testing environments not experienced at training time. Our design allows us to navigate through areas with more challenging perceptual conditions, showing an increase in success rate and robustness with respect to purely reactive planning and the latest available solutions in state-of-the-art of active perception.

Future directions include investigating the incorporation of semantic segmentation networks in the pipeline in replacement of high-quality masks, and the adaptation of the policy network to handle the noise and uncertainty of labels in semantic segmentation.

References

[1] L. Teixeira, I. Alzugaray, and M. Chli, "Autonomous Aerial Inspection using Visual-Inertial Robust Localization and Mapping," in *Field and Service Robotics*. Springer, 2018.

[2] R. Bajcsy, Y. Aloimonos, and J. K. Tsotsos, "Revisiting Active Perception," in *Autonomous Robots*. Springer, 2018.

[3] N. Atanasov, "Active Information Acquisition With Mobile Robots," Ph.D. dissertation, University of Pennsylvania, 2015.

[4] A.-m. Ali-Akbar, C. Suman, and M. A. Nancy, "FIRM: Sampling-based Feedback Motion-planning under Motion Uncertainty and Imperfect Measurements," *The International Journal of Robotics Research (IJRR)*, 2014.

[5] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and Acting in Partially Observable Stochastic Domains," *Journal of Artificial Intelligence (JAI)*, 1998.

[6] S. Prentice and N. Roy, "The Belief Roadmap: Efficient Planning in Belief Space by Factoring the Covariance," *The International Journal of Robotics Research (IJRR)*, 2009.

[7] A. Bry and N. Roy, "Rapidly-exploring Random Belief Trees for Motion Planning under Uncertainty," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[8] Z. Zhang and D. Scaramuzza, "Perception-aware Receding Horizon Navigation for MAVs," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[9] C. Papachristos, S. Khattak, and K. Alexis, "Uncertainty-aware receding horizon exploration and mapping using aerial robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[10] G. Costante, J. Delmerico, M. Werlberger, P. Valigi, and D. Scaramuzza, "Exploiting Photometric Information for Planning Under Uncertainty," in *Robotics Research: Volume 1*. Springer, 2018.

[11] L. Bartolomei, L. Teixeira, and M. Chli, "Perception-aware Path Planning for UAVs using Semantic Segmentation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[12] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a Quadrotor With Reinforcement Learning," *IEEE Robotics and Automation Letters*, 2017.

[13] G. Kahn, A. Villaflor, V. Pong, P. Abbeel, and S. Levine, "Uncertainty-Aware Reinforcement Learning for Collision Avoidance," *CoRR*, 2017.

[14] J. Peters and S. Schaal, "Reinforcement Learning by Reward-Weighted Regression for Operational Space Control," in *Proceedings of the 24th International Conference on Machine Learning*, 2007.

[15] T. Wang, V. Dhiman, and N. Atanasov, "Learning Navigation Costs from Demonstrations with Semantic Observations," in *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, 2020.

[16] T. Zhang, G. Kahn, S. Levine, and P. Abbeel, "Learning Deep Control Policies for Autonomous Aerial Vehicles with MPC-guided Policy Search," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[17] G. Kahn, A. Villaflor, B. Ding, P. Abbeel, and S. Levine, "Self-Supervised Deep Reinforcement Learning with Generalized Computation Graphs for Robot Navigation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[18] B. Chen, A. Sax, G. Lewis, I. Armeni, S. Savarese, A. Zamir, J. Malik, and L. Pinto, "Robust Policies via Mid-Level Visual Representations: An Experimental Study in Manipulation and Navigation," *Conference on Robot Learning (CoRL)*, 2020.

[19] M. Siam, M. Gamal, M. Abdel-Razek, S. Yogamani, M. Jagersand, and H. Zhang, "A Comparative Study of real-time Semantic Segmentation for Autonomous Driving," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition workshops*, 2018.

[20] Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza, "Flightmare: A Flexible Quadrotor Simulator," *Conference on Robot Learning (CoRL)*, 2020.

[21] V. Usenko, L. von Stumberg, A. Pangercic, and D. Cremers, "Real-time trajectory replanning for MAVs using uniform B-splines and a 3D circular buffer," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[22] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: a Versatile and Accurate Monocular SLAM System," *IEEE Transactions on Robotics*, 2015.

[23] L. Steccanella, D. D. Bloisi, A. Castellini, and A. Farinelli, "Waterline and obstacle detection in images from low-cost autonomous boats for environmental monitoring," *Robotics and Autonomous Systems (RAS)*, 2020.

[24] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *CoRR*, 2018.

[25] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *CoRR*, 2017.

[26] S. Gu, S. Guo, W. Zuo, Y. Chen, R. Timofte, L. Van Gool, and L. Zhang, "Learned dynamic guidance for depth image reconstruction," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2019.