2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

VO-Safe Reinforcement Learning for Drone Navigation

Feiqiang Lin[1], Changyun Wei[2], Raphael Grech[3], Ze Ji[1]

***Abstract*— This work is focused on reinforcement learning (RL)-based navigation for drones, whose localisation is based on visual odometry (VO). Such drones should avoid flying into areas with poor visual features, as this can lead to deteriorated localization or complete loss of tracking. To achieve this, we propose a hierarchical control scheme, which uses an RL-trained policy as the high-level controller to generate waypoints for the next control step and a low-level controller to guide the drone to reach subsequent waypoints. For the high-level policy training, unlike other RL-based navigation approaches, we incorporate awareness of VO performance into our policy by introducing pose estimation-related punishment. To aid robots in distinguishing between perception-friendly areas and unfavoured zones, we instead provide semantic scenes, as input for decision-making instead of raw images. This approach also helps minimise the sim-to-real application gap.**

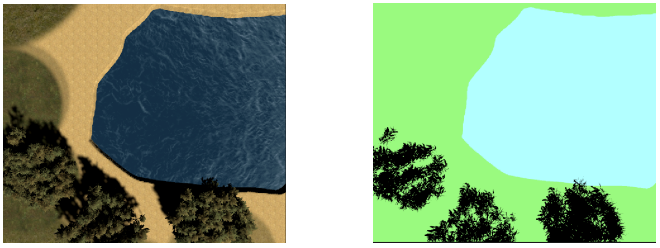
I. Introduction

Reinforcement learning (RL) has been proven effective in training robots to handle point-goal-reaching navigation tasks [1]. In point-goal-reaching tasks, robots need to navigate to designated goal positions presented in coordinates. In such tasks, robots need to calculate the relative goal position based on the current pose, for decision-making of the next action. However, most existing RL-based approaches simply assume robots' ground truth poses are available, and these methods optimise the RL policies in terms of the length of the paths, ignoring the fact that, in real-world applications, poses need to be estimated by algorithms, such as Visual Odometry (VO) or Simultaneous Localization and Mapping (SLAM), based on observations from onboard sensors. For cameras-equipped drones in GNSS-denied environments, VO is commonly used for pose estimation.

However, the performance of such approaches heavily relies on the quality of observations. Observations with ambiguous or insufficient features along planned trajectories will lead to failed localisation and feeding robots with wrong poses can result in catastrophic consequences, especially for safety-critical tasks. On the other hand, the assumption of the availability of the ground truth poses will introduce a huge gap between the simulation and real-world environments for deployment on the real robot. Fig. 1 illustrates a scenario where a drone is undertaking a navigation task with VO-based localisation. The water area should be avoided due to the poor visual features that can lead to deteriorated pose estimation. Instead, the green path, despite being longer, is more suitable for enhanced VO-based navigation.

Fig. 1: A drone travels from the start to the destination, crossing a lake. The red trajectory, although shorter, lacks reliable visual features for pose tracking. In contrast, the longer green trajectory enhances visual odometry performance by maximising feature quality over terrain, houses, etc.

Most current research in path planning focuses solely on either efficiency, aiming for shorter paths, or safety, prioritizing collision-free routes. The above issue has been seldom noticed in previous works. Also, most RL-based robots are usually trained in simulation environments, where robot poses are always accessible. As a result, agents after training with ground truth poses tend to choose shorter paths (Fig. 1), ignoring the possible failed localisation in the real world, where ground truth poses are unavailable. On the other hand, visual observations gathered in simulators present considerable differences from images collected in the real world. To mitigate such problems, we propose to utilize semantically segmented images (Fig. 2a), instead of raw images (Fig. 2b) as input, to ensure input consistency between simulation-based training and real-world deployments.



(a) Simulated lake (b) Semantic Mask

Fig. 2: Simulated environments and semantic output

For the control system, we propose a hierarchical framework rather than an end-to-end approach. This involves a high-level controller generating short-term waypoints, cou-

The authors thank the China Scholarship Council (CSC) for financially supporting Feiqiang Lin in his PhD programme (201906020170). Corresponding author: Ze Ji

[1]School of Engineering, Cardiff University, Cardiff, UK {linf6,jiz1}@cardiff.ac.uk
[2]College of Mechanical and Electrical Engineering, Hohai University, China weichangyun@hotmail.com
[3]Spirent Communications, Paignton, UK raphael.grech@spirent.com

pled with a traditional low-level controller guiding the drone to these waypoints. This hybrid approach harnesses neural networks' capabilities while ensuring safety through the reliability of conventional controllers. Although end-to-end learning has been proven effective for tasks of drone flights, it poses a limitation in the context of VO-based pose tracking, particularly with agile motions where significant differences in observations between consecutive frames occur due to rapid and erratic movements, which can result in challenges in VO feature matching. Also, end-to-end policies are trained for specific robot configurations, hence cannot be deployed to other robots. The high-level policy of a hierarchical framework only decides the next waypoint, agnostic from robot configurations, and the low-level controller, tailored for individual robots, controls the robot to the next waypoint. This excludes the need for retraining the RL-based policies.

In this work, we address the above challenges by introducing a new RL-based navigation framework, named VO-Safe RL. The contributions of our work are summarised below:

- We propose an RL-based navigation framework to prevent odometry failures during flight.
- A novel reward space is introduced to encourage VO-safe behaviours.
- Semantic images are used to bridge the sim-to-real gap.
- Our approach combines learning-based and conventional control in a hierarchical scheme.

II. Related Work

Deep RL (DRL) incorporates deep learning with RL, allowing agents to make decisions from unstructured high-dimensional input data without manually engineering the state space. As such, DRL has been widely applied for solving complex navigation tasks with high-dimensional state input.

Tai et al. introduce two prominent works in [1] and [2], where robots are trained with RL to achieve point goal-reaching tasks without environment maps based on the DDPG algorithm with a simple reward space to penalise collisions and reward actions reducing the relative goal-robot distance.

Many variants of the above works have been introduced for further improvement from different perspectives. Various RL approaches have been employed for specific navigation tasks, such as duelling double DQN [3] that helps stabilise training, DDPG [1] with which action space can be continuous, CPO [4] that considers constraints during optimisation. Regarding algorithm implementations, efforts have been paid to improve data efficiency as RL usually requires huge amounts of training data [4], [5], [6]. Also, training neural networks with high-dimensional image inputs could be difficult. For efficient state representations (e.g. encoded images), training with auxiliary tasks is introduced in [7], [8], [9], where, in addition to direct task rewards, agents can collect bonus rewards by predicting task-related parameters, such as estimating scene depths from images.

From the control system's perspective, drone-specific RL methods can be broadly categorised into two groups, namely low-level and high-level control [10]. For low-level control, RL policies directly output individual rotor thrust commands based on observations [11], [12], [13]. This end-to-end training can empower drones' manoeuvrability. However, the policy is specific to certain drones used for training as the dynamics vary with different drones. Also, it will be difficult to deploy such policies on real-world drones due to the considerable differences between simulated aerodynamics and the real world. For high-level control, trained policies can produce the collective thrust and bodyrate commands [14], [15], [16] or linear velocity (x-y-z 3-dimension velocities and yaw rotation speed) commands [17], [18], [19]. Such high-level control commands are usually executed by a low-level controller, which is typically non-learning-based. This hierarchical control scheme has the advantage of isolating specific individual drone dynamics from high-level control. Its platform-agnostic nature allows for easy transfer to real-world deployment of agents trained in simulation environments.

In addition, drones performing complex aerobatic motions usually require external optical motion tracking systems for obtaining states, rather than estimating states using onboard sensors as discussed earlier. Excessive camera motion can lead to VO-based tracking failures, as VO relies on overlapping visual features between consecutive frames. We use a hierarchical scheme that decomposes a long-distance path into short-term waypoints that can avoid violent behaviours.

Works introduced above always assume the availability of robot poses throughout the navigation tasks, leaving the issues introduced by localisation algorithms unsolved, as illustrated in Section I. Preliminary studies have been performed by the authors [20][21] in different scenarios of 2D navigation that is based on a 2D planar Lidar for localisation and a TurtleBot. Another related work that takes VO performance into consideration is [22], which scores environment objects based on VO performance.

III. Method

As described, the overall target of the problem is to guide a drone to the goal within a given time, while also avoiding VO-unfavoured regions. The problem can be formulated as

$$\min_{\pi(x)} \mathcal{J}_o = \int_0^T \|p(t) - Goal\|^2 \mathrm{d}t, \tag{1a}$$

$$s.t. \quad x_{t+1} = f(x_t, \pi(x_t)), \tag{1b}$$

$$x_0 \in \mathcal{X}, \tag{1c}$$

$$p(t) \in \mathcal{P}, \ \forall t \in [0, T], \tag{1d}$$

where x represents the drone state and p specifies the drone position. Eq. 1b represents the drone's dynamic model and $\mathcal{X}$ is the possible initial state set. $\mathcal{P}$ denotes areas with high-quality visual features suitable for VO-based pose tracking. $\pi(x_t)$ is the policy to be optimised subject to dynamics (Eq. 1b) and VO constraints $\mathcal{P}$. We consider collision-free environments in this work and focus on VO performance. This assumption is reasonable as collision detection is usually not required for open-space navigation.

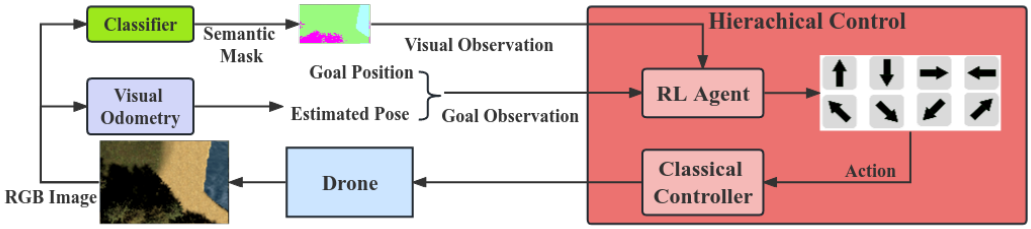


Fig. 3: Overall System Framework

A. Overall System

An RL-based hierarchical control scheme is proposed in this work. The proposed framework and the key modules are shown in Fig. 3. A drone equipped with an RGB camera is used for VO-based navigation. Firstly, raw images are processed by the VO module to perform pose estimation, calculating the relative goal positions. This is part of the RL agent observations. The raw images are also fed into a classifier to produce semantic images, which form the other part of the observations. Based on the observation, the RL agent will make a decision to select one action among eight possible moving directions. At the low level control, a classical controller then calculates individual rotor thrusts in order to guide the drone along the selected directions for a designed fixed distance. The procedure will repeat until the goal position is reached successfully.

B. Reinforcement Learning based High-Level Controller

The high-level controller is based on the Proximal Policy Optimisation (PPO) algorithm. This subsection briefly introduces the basic concepts of RL and the PPO algorithm, followed by the implementation details of our specific method.

Basic Concepts. RL algorithms solve problems under the Markov Decision Process (MDP) framework. In MDP, a physical process is described by a state transition model $p(s_{t+1}|s_t, a_t)$ where states s fall in a state space S and actions are in an action space A. A reward function $R_t(s_t, a_t)$: $S \times A \rightarrow R$ is needed to justify the actions a_t taken at state s_t according to specific tasks. Strategies for solving the tasks can be obtained by maximising the overall discounted rewards, formulated as $R(\tau) = \sum_{k=0}^{\infty} \gamma^k R_t$ received during the whole process. State-action value functions $Q^{\pi(s_t,a_t)} = E_{\tau\sim\pi}[R(\tau)|s_t, a_t]$ and/or state value function $V^{\pi}(s_t) = E\tau \sim \pi[R(\tau)|s_t]$ are usually defined to reshape the objective function, where π: $a \sim \pi(\cdot|s, \theta)$ established by parameters θ represents the policy to be learned. Also, advantage function $A^{\pi}(s, a) = Q^{\pi}(s, a) - V^{\pi}(s)$ is used to describe how much better on average an action is than others.

PPO. We use the PPO algorithm considering its reliable performance, ease of implementation, and flexibility to handle both discrete and continuous action spaces. The PPO is a policy gradient algorithm, which trains the policy π_θ directly. The reliability is ensured by limiting updates between consecutive network training steps to prevent large deviations that can cause training issues. This constraint can be applied through methods like penalising significant differences using KL-divergence (PPO-Penalty) or by setting a maximum update range (PPO-Clip) [23].

We adopt PPO-clip in the work. The updating process of PPO-clip is as follows:

$$\theta_{k+1} = \arg\max_{\theta} \mathop{\mathrm{E}}_{s,a\sim\pi_{\theta_k}} [L(s, a, \theta_k, \theta)] \tag{2}$$

where L is given by

$$L(s, a, \theta_k, \theta) = \min \left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s, a), g(\epsilon, A^{\pi_{\theta_k}}(s, a)) \right) \tag{3}$$

in which ϵ is a small hyperparameter and $g(\epsilon, A) = \begin{cases} (1+\epsilon)A & A \geq 0 \\ (1-\epsilon)A & A < 0. \end{cases}$

The intuition behind this is to increase the possibility of actions ($\pi_\theta(a|s)$) that are better than others, evaluated by the advantage $A^{\pi}(s, a)$ and lower the probabilities of worse actions. The clipping constrains how far the update can be made between the new and old policies to avoid divergence.

VO-safe RL Implementations. The following describes the details of our VO-safe RL. The state s is constructed by the estimate relative goal position p_g and the observed images O_{img}: $s = [p_g, O_{img}]$. For the action space A, we propose to use 8 discrete actions, which represent moving along 8 possible directions for 1 meter. The discretised action space will help accelerate the training process.

To design the reward function, as the drone relies on VO-based odometry for pose estimation, the agent should not only optimise the path length, but also needs to avoid actions leading to VO failures. Thus, unlike most previous research works, we introduce an extra punishment based on the VO status. The reward space is given as follows:

$$R_t = R_s(p(t)) + R_G(p(t)) + R_{VO} \tag{4}$$

$R_s(p(t))$ is the progress reward designed to encourage the agent to reduce the distance from the robot to the goal:

$$R_s(p(t)) = \alpha(d_{t-1} - d_t) \tag{5}$$

where d_t is the distance from the robot to the goal at time step t and α is the distance weight. The goal reaching reward $R_G(p(t))$ is:

$$R_G(p(t)) = \begin{cases} 10 & d_t \leq d_g \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

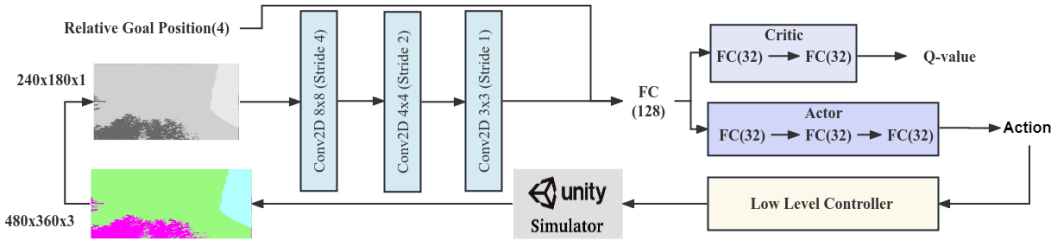


Fig. 4: High-level policy network

A positive reward is provided when the distance to the goal is within a threshold d_g. The last term R_{VO} is the VO-related reward for penalising actions leading to the deterioration of VO-based tracking. Tracking is considered lost or failed when consecutive frames are not consistent with each other. R_{VO} is defined as:

$$R_{VO} = \begin{cases} -10 & \text{if visual odometry failed} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

C. Low-Level Controller

The low-level controller uses a classic PID controller to compute control signals (u) for motor controllers. This enables the drone to move toward the next waypoint determined by the high-level policy. The low-level controller follows a cascaded control structure, comprising a position controller followed by an attitude controller, as depicted in Fig. 5.

The position PID controller receives position commands $p_d = [p_x, p_y, p_z]_d$, which outputs the desired attitude $\mathbf{\Theta}_d$ for the attitude controller and also the collective thrust:

$$\begin{aligned} \mathbf{\Theta}_{hd} &= -g^{-1} A_\psi^{-1}(-K_{hD}(\dot{p_h}) - K_{hP}(p_h - p_{hd})), \\ f &= m(g + K_{zD}(\dot{p_z} - \dot{p_{zd}}) + K_{zP}(p_z - p_{zd})), \\ \mathbf{\Theta}_d &= [\mathbf{\Theta}_{hd}, \psi_d], \end{aligned} \tag{8}$$

where the control of horizontal position $p_h = [p_x, p_y]$ and altitude p_z are decoupled. The attitude control $\mathbf{\Theta} = [\psi, \theta, \phi]$ is also separated into two parts: $\mathbf{\Theta}_h = [\theta, \phi]$ and yaw ψ. The desired yaw ψ_d is provided by the task and $A_\psi = \begin{bmatrix} \sin\psi & \cos\psi \\ -\cos\psi & \sin\psi \end{bmatrix}$. m and g are the drone mass and gravity respectively. K_{hD}, K_{hP}, K_{zD} and K_{zP} are the corresponding derivative and proportional weights.

The attitude PID controller calculates the desired moment τ_d such that the drone can fly in desired attitudes:

$$\begin{aligned} \omega_d &= -K_{\mathbf{\Theta}}(\mathbf{\Theta} - \mathbf{\Theta}_d), \\ e_\omega &= \omega - \omega_d, \\ \tau_d &= -K_{\omega P} e_\omega - K_{\omega I} \int e_\omega - K_{\omega D} \dot{e_\omega}, \end{aligned} \tag{9}$$

$K_{\mathbf{\Theta}}$ is designed to control the drone attitude to converge to desired attitude following an expected trajectory, and $K_{\omega P}$, $K_{\omega I}$, $K_{\omega D}$ denote the coefficients for the proportional, integral, and derivative terms respectively. $u = [\tau_d, f]$ can then be used to calculate individual motor control signals $[f_1, f_2, f_3, f_4]$.

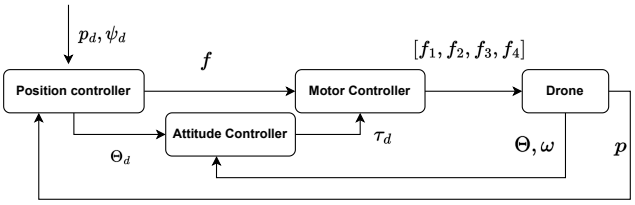


Fig. 5: Low level controller

IV. Evaluation

A. Experiments Setup

We test our algorithm by deploying a simulated drone equipped with a monocular camera in the Flightmare [24] simulator. The VO used is ORB-SLAM2 [25] and the policy network is shown in Fig. 4.

Hierarchical RL Agent (Ours). To train the PPO agent, in each episode, the robot is spawned at a random position at an altitude of $15m$. The goal position is also randomly assigned. At each time step, the drone flies along the RL-decided direction for 1 meter. Each episode is terminated and restarted when any of the following cases occurs: 1) the drone reaches the goal; 2) VO reports failures; or 3) maximum time steps are reached.

End-to-end State-based RL Agent. We also train an end-to-end state-based agent from previous research [10], as the baseline for comparison. Ground truth state information $s = [p_g, \Theta, \omega]_{gt}$ is provided instead of images and VO inputs. In this case, the agent decides individual motor control signals $[f_1, f_2, f_3, f_4]$ directly.

B. PPO Training Results

Fig. 6 shows the average rewards during the training of the policy. The state-based agent has access to ground-truth state information from the simulator. For convenience, we name it GT-state-based here. As expected, this agent obtains the highest average reward after training. This can be viewed as a benchmarking reference to evaluate the performance of our vision-based agent. We can see that the performance of our visual agent is comparable to the GT-State-based agent in terms of the total rewards received.

The test success rates can be seen in Table I. During the test, our algorithm achieved a success rate of 0.78. For the GT-state-based agent fed with ground truth poses for training, it can always reach the goal as the ground truth poses are

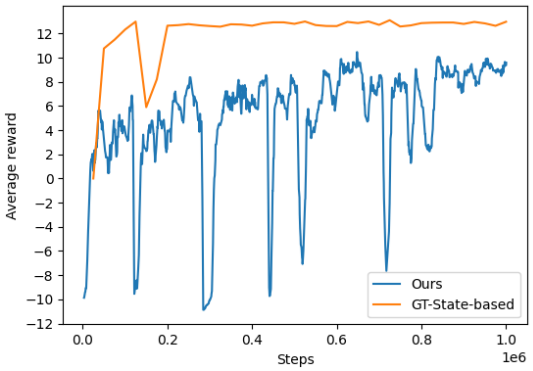


Fig. 6: Average rewards

provided during testing too. However, if we feed the state-based agent with VO-estimated poses (named VO-State-based), the success rate drops significantly to 0.56. The main failure reasons include the drone flying over VO-unfavoured regions, such as water, and occurrences of excessive flying velocities $\dot{p}$ or attitudes Θ.

TABLE I: Success rates and VO performance

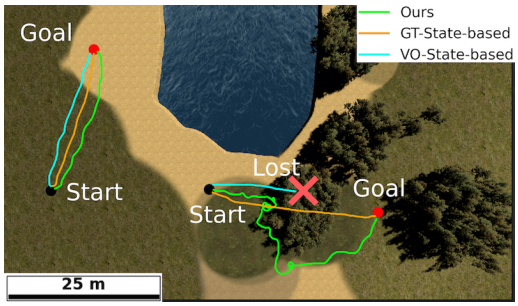
Metrics	Ours	GT-State-based	VO-State-based
Success Rate	0.78	1.0	0.56
RMSE (m)	0.38 ± 0.31	N/A	1.10 ± 0.5

Fig. 7a shows trajectories of different agents, where the red circles represent the goals. It is clear that the state-based agent chooses to fly directly towards the goal as it does not consider visual information for localization, while the agent trained with our algorithm chose the path avoiding dangerous regions such as water or trees which present ambiguous and less distinct features. Fig. 7b shows the trajectories in more photo-realistic environments, which have not been seen during training. Our agent still can accomplish tasks successfully, despite the new environments. Retraining is not required, as the agent uses semantic images as the input.

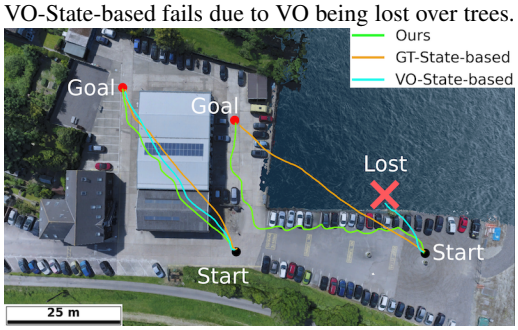
C. Hierarchical Architecture

In this section, we justify the use of hierarchical architecture by analysing the impact of drone dynamics on VO performance. As shown in Table I, poses estimated by VO are considerably more accurate, with the trajectories generated by our hierarchical method, compared to the VO-State-based approach. This is mainly attributed to the short-term goals generated by the high-level policy, preventing the drone from acting violently. In comparison, the end-to-end VO-State-based approach tends to produce excessive motions to achieve high speed, causing large discrepancies between consecutive images, hence inaccurate localisation. This is further depicted visually in Fig. 8.

To verify the impact of excessive motions on VO performance, we further show two examples. The drone is tasked to reach four goals located at the corners of a square in a simulation environment (Fig. 9a and Fig.9b). Our method



(a) Our agents can navigate successfully while the VO-State-based fails due to VO being lost over trees.



(b) In a photo-realistic environment. VO-State-based fails above the water region, while ours succeeds without retraining as with GT-State-based.

Fig. 7: Trajectories in different environments with three agents. (Red circles: the goals).

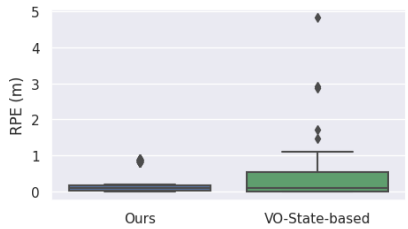


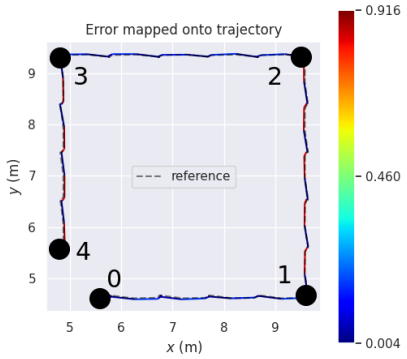
Fig. 8: Ours has a smaller relative position error (rpe).

decomposes each task into a list of short segments, with each 1m long. The VO estimation errors are projected on the drone's trajectory. However, for the end-to-end agent (VO-state-based), the drone directly moves to the next target, with each waypoint 5m away. The drone acts rapidly and violently to reach every next target. The accuracy of localisation with VO drops, as shown in Fig. 9b. VO even lost its track during the last segment and was unable to navigate to target 4.

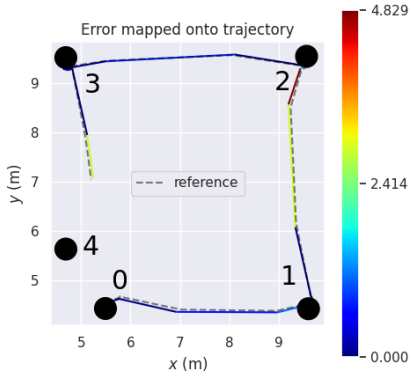
Excessive attitude change and high velocity negatively impact VO performance and even result in VO failures. Fig. 10 shows the attitude response to different distance commands. Thus, we deploy the hierarchical architecture for enhanced VO performance and reduced VO failure rates, through decomposing long-distance paths into short-range goals ($1m$ in this work).

D. Real-World Experiments

We also carried out real-world experiments to verify the proposed method. The drone configuration is shown in Fig. 11a. A down-looking camera RealSense D435 is used



(a) localisation error (Ours)



(b) localisation error (VO-State-based)

Fig. 9: VO Performances (Ours vs VO-State-based)

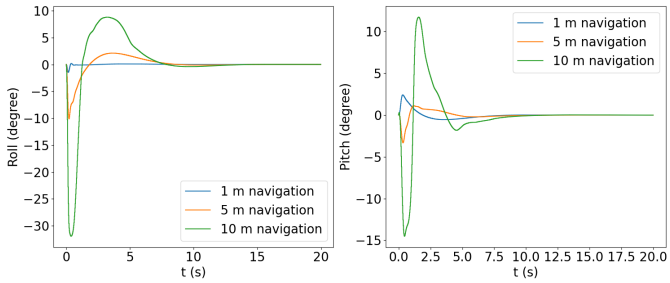


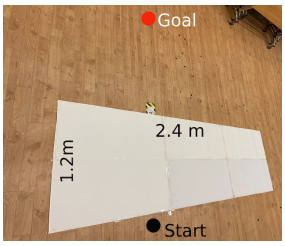
Fig. 10: Drone attitude responses to navigation with different goal distances.

for drone perception and VO. An onboard computer (Intel NUC 11 pro) is used for image segmentation, high-level policy calculation and VO computation. No ground station computers were required.

The test scenario is shown in Fig. 11b. A region of the floor covered by white paper is used as the featureless region. The starting point (black) and the goal (red) are on two sides of the white paper region. The floor has rich visual textures suitable for VO tracking, while the white paper region lacks textures. The drone should avoid entering the featureless area to prevent VO failures. The policy is not retrained to adapt to real-world experiments because the hierarchical scheme and semantic inputs are used.

As shown in Fig. 12a, the drone does not follow a straight line towards the goal, and instead it follows a longer

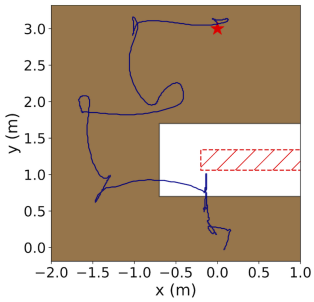
(a) Drone configuration



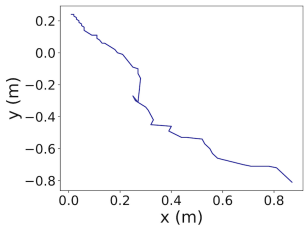
(b) Test scenario

Fig. 11: Real world implementation

trajectory bypassing the paper. The plotted trajectory is the VO-estimated result that is continuous without failures. Even though the trajectory initially crossed the white region's edge, the drone can still capture the textured floor within its field of view at an altitude of 0.7m.



(a) Real-world experiment result.



(b) Odometry drifted while the drone is stationary with poor visual features.

Fig. 12: Real-world implementation results

Last, we demonstrate the failed VO behaviour when entering entirely the white paper region (see the area with red diagonal lines in Fig. 12a, where no floor textures can be observed). The drone was holding still initially. However, due to the lost track of VO, the drone started to rely on the IMU integration for pose estimation, which caused drifted pose estimation. This explains the drifting motion along the diagonal line.

V. Conclusion

In this work, we proposed a RL-based method for drone navigation with visual odometry deployed for localisation, instead of being provided with the ground truth poses, as with most RL-based methods. The desired behaviour of VO-enabled drones should consider the quality of the visual features, i.e., feature-poor regions should be avoided. As such, we introduced a novel reward space to handle such problems. Furthermore, to alleviate problems for sim-to-real transfer, semantic images, instead of raw RGB images, are proposed as inputs. For control, we propose to use a hierarchical control framework. The high-level policy is trained based on the PPO algorithm, while the low-level controller is a classical method, with guaranteed safety in theory for reliable motion control. Evaluation results in both simulation and real-world environments demonstrate that our method outperforms baselines as hypothesised.

REFERENCES

[1] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 31–36.

[2] L. Tai and M. Liu, "A robot exploration strategy based on q-learning network," in *2016 ieee international conference on real-time computing and robotics (rcar)*. IEEE, 2016, pp. 57–62.

[3] X. Ruan, D. Ren, X. Zhu, and J. Huang, "Mobile robot navigation based on deep reinforcement learning," in *2019 Chinese control and decision conference (CCDC)*. IEEE, 2019, pp. 6174–6178.

[4] M. Pfeiffer, S. Shukla, M. Turchetta, C. Cadena, A. Krause, R. Siegwart, and J. Nieto, "Reinforced imitation: Sample efficient deep reinforcement learning for mapless navigation by leveraging prior demonstrations," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4423–4430, 2018.

[5] B. Moridian, B. R. Page, and N. Mahmoudian, "Sample efficient reinforcement learning for navigation in complex environments," in *2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, 2019, pp. 15–21.

[6] H.-T. L. Chiang, A. Faust, M. Fiser, and A. Francis, "Learning navigation behaviors end-to-end with autorl," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2007–2014, 2019.

[7] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, *et al.*, "Learning to navigate in complex environments," *arXiv preprint arXiv:1611.03673*, 2016.

[8] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu, "Reinforcement learning with unsupervised auxiliary tasks," *arXiv preprint arXiv:1611.05397*, 2016.

[9] J. Ye, D. Batra, A. Das, and E. Wijmans, "Auxiliary tasks and exploration enable objectgoal navigation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 16 117–16 126.

[10] E. Kaufmann, L. Bauersfeld, and D. Scaramuzza, "A benchmark comparison of learned control policies for agile quadrotor flight," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 10 504–10 510.

[11] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a quadrotor with reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2096–2103, 2017.

[12] A. Molchanov, T. Chen, W. Hönig, J. A. Preiss, N. Ayanian, and G. S. Sukhatme, "Sim-to-(multi)-real: Transfer of low-level robust control policies to multiple quadrotors," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 59–66.

[13] Y. Song, M. Steinweg, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing with deep reinforcement learning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1205–1212.

[14] G. Shi, X. Shi, M. O'Connell, R. Yu, K. Azizzadenesheli, A. Anandkumar, Y. Yue, and S.-J. Chung, "Neural lander: Stable drone landing control using learned dynamics," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 9784–9790.

[15] E. Kaufmann, A. Loquercio, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Deep drone acrobatics," *arXiv preprint arXiv:2006.05768*, 2020.

[16] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, pp. 982–987, 2023.

[17] A. Loquercio, A. I. Maqueda, C. R. Del-Blanco, and D. Scaramuzza, "Dronet: Learning to fly by driving," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1088–1095, 2018.

[18] E. Kaufmann, A. Loquercio, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Deep drone racing: Learning agile flight in dynamic environments," in *Conference on Robot Learning*. PMLR, 2018, pp. 133–145.

[19] S. Belkhale, R. Li, G. Kahn, R. McAllister, R. Calandra, and S. Levine, "Model-based meta-reinforcement learning for flight with suspended payloads," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1471–1478, 2021.

[20] F. Lin, Z. Ji, C. Wei, and H. Niu, "Reinforcement learning-based mapless navigation with fail-safe localisation," in *Annual Conference Towards Autonomous Robotic Systems*. Springer, 2021, pp. 100–111.

[21] F. Lin, Z. Ji, C. Wei, and R. Grech, "Localisation-safe reinforcement learning for mapless navigation," in *2022 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2022, pp. 1327–1334.

[22] L. Bartolomei, L. Teixeira, and M. Chli, "Semantic-aware active perception for uavs using deep reinforcement learning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 3101–3108.

[23] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[24] Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza, "Flightmare: A flexible quadrotor simulator," in *Conference on Robot Learning*, 2020.

[25] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.