Indoor Multi-Floor UAV Delivery: Energy-Aware Navigation through A* S-RRT* and Reinforcement Learning

Huynh Nhut Huy, Nguyen Ly Minh Ky, Luong Danh Doanh, Nguyen Huy Hoang
FPT University Ho Chi Minh City

November 3, 2025

Abstract

Indoor delivery using unmanned aerial vehicles (UAVs) requires robust localisation, safe path planning and energy-efficient control, especially when navigating multi-storey buildings without GPS. This report presents an updated and significantly expanded account of our work on energy-aware indoor UAV navigation. Building upon our previous two reports, we now consider a five-floor environment and provide a thorough literature review and detailed descriptions of the algorithms employed. Our system integrates visual–inertial simultaneous localisation and mapping (VI-SLAM) for centimetre-scale pose estimation, graph-based global planning via the A* algorithm, safety-oriented local replanning using a variant of Rapidly-exploring Random Tree (S-RRT*), and an energy-aware control policy learned with Proximal Policy Optimisation (PPO). The navigation task is formulated as a Markov decision process (MDP) with continuous states and actions; the reward function penalises time, thrust, jerk and collisions while rewarding goal progress. Preliminary results on a smaller environment showed approximately 25 % energy savings compared with following the A* path directly while maintaining similar collision rates. This report details the expanded methodology, training procedures, and prepares the ground for full experiments on the five-floor map.

Contents

1 Introduction

The deployment of autonomous unmanned aerial vehicles (UAVs) for last-mile delivery promises rapid transportation of goods without dependence on ground traffic. While outdoor drone delivery has been demonstrated, indoor delivery remains a major challenge. Indoor environments impose **(i) localisation constraints**: GPS signals are unavailable, necessitating visual–inertial SLAM; **(ii) path planning complexity**: buildings consist of multiple floors with narrow corridors, stairways and dynamic obstacles; **(iii) strict energy budgets**: battery capacity limits flight time; and **(iv) safety requirements**: drones must avoid collisions with people and infrastructure. Efficient indoor delivery therefore demands an integrated approach combining accurate localisation, global and local planning and energy-aware control.

Previous work on indoor UAV delivery has typically focused on either localisation or high-level planning. In our earlier reports, we presented a system that couples ORB-SLAM3 with Proximal Policy Optimisation (PPO) to learn energy-efficient trajectories on a three-floor map, achieving a success rate of 96 % and roughly 25 % lower energy consumption relative to A* planning alone [7]. This report extends those results in two directions. First, we upgrade the environment to a five-floor building with more complex layouts. Second, we expand the methodological exposition to include a comprehensive literature review and detailed algorithmic descriptions, ensuring that all steps from map generation through to reinforcement learning are reproducible. Due to the ongoing nature of training on the new environment, numerical results are currently limited to those previously reported, but the expanded methodology will enable future updates.

2 Literature Review

This section surveys key research topics relevant to indoor UAV delivery: localisation via visual–inertial SLAM, graph-based and sampling-based planning algorithms, safe variants of RRT*, and reinforcement learning for UAV navigation.

2.1 Visual–Inertial SLAM

In GPS-denied environments, simultaneous localisation and mapping (SLAM) must be performed using onboard sensors. Visual–inertial SLAM combines camera imagery with inertial measurements to achieve scale-aware pose estimation. ORB-SLAM3 is a state-of-the-art system supporting monocular, stereo and RGB-D cameras; it tightly integrates inertial measurements, builds multiple maps and performs loop closure and map merging [1]. Experiments on public datasets show that stereo-inertial ORB-SLAM3 attains centimetre-level accuracy (3.6 cm on EuRoC [1]). Its ability to restart new maps after tracking loss and subsequently merge them makes it suitable for complex indoor scenes with multiple floors. However, SLAM alone does not provide path planning or control.

2.2 Graph-Based Path Planning: A* Algorithm

Classical planning for discrete environments often employs graph search. The A* algorithm extends Dijkstra's algorithm by incorporating a heuristic estimate of the cost to the goal. For each node n it computes $f(n) = g(n)+h(n)$, where $g(n)$ is the cost of reaching n from the start and $h(n)$ is a heuristic that does not overestimate the true remaining cost; A* selects the node with minimal f for expansion [2]. With an admissible heuristic, A* is complete and returns an optimal path [2]. In three-dimensional grids, A* can be extended by considering the 26 neighbouring voxels; vertical transitions represent moving between floors. A drawback of A* is that the cost function typically measures distance, not control energy. Moreover, its performance degrades with large search spaces, although occupancy grids of moderate size (tens of thousands of cells) remain tractable.

2.3 Sampling-Based Planning: RRT and RRT*

Rapidly-exploring Random Trees (RRTs) are sampling-based planners for continuous spaces. An RRT incrementally builds a tree rooted at the start by sampling random points in free space and connecting each sample to the nearest existing node within a maximum distance. RRT algorithms are probabilistically complete but do not guarantee path optimality. To address this, Karaman and Frazzoli proposed RRT*, which adds a rewiring step: after connecting a sample, nearby nodes are considered for rewiring if doing so reduces the path cost [3]. They prove that the expected cost of the RRT* solution converges to the optimal cost as the number of samples grows large [3]. Numerous variants improve performance, such as Informed RRT*, which restricts sampling to an ellipsoidal subset once an initial solution is found [4].

2.4 Safe Variants of RRT*

While RRT* yields asymptotically optimal paths, the sampling does not explicitly enforce safety distances or curvature constraints. In dynamic environments with moving obstacles, additional considerations are needed. Safe Interval RRT* (SI-RRT*) divides the configuration space into safe intervals in space–time and performs a low-level RRT* search within these intervals while a high-level conflict resolution algorithm (e.g., conflict-based search) resolves multi-agent interactions [5]. Cost functions can also incorporate clearance and curvature terms, penalising proximity to obstacles and excessive turning, thereby producing smoother and safer paths. These enhancements motivate our adoption of an S-RRT* planner for local replanning around dynamic obstacles.

2.5 Reinforcement Learning for UAV Navigation

Deep reinforcement learning (RL) has been applied to UAV navigation tasks with continuous actions. An RL agent perceives its state and chooses actions to maximise expected cumulative rewards. Proximal Policy Optimisation (PPO) is a widely used on-policy actor–critic method that updates the policy by maximising a surrogate objective subject to a clipping constraint to prevent large updates [6]. The objective is

$$L(\theta) = \mathbb{E}_t\big[\min(r_t(\theta)\hat{A}_t, \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\big], \tag{1}$$

where $r_t(\theta)$ is the likelihood ratio between new and old policies and $\hat{A}_t$ is the advantage estimate. This constraint stabilises training and yields good sample efficiency [6]. RL has been used for energy-aware UAV routing, yielding energy savings but often neglecting multi-floor topologies or dynamic human obstacles. Our approach integrates RL with classical planning and SLAM to address these shortcomings.

3 Problem Formulation

We cast the indoor delivery task as a Markov decision process (MDP) $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$. At each timestep t the drone observes a state $s_t \in \mathcal{S}$ and selects a control command $a_t \in \mathcal{A}$ according to a policy $\pi(a_t \mid s_t)$. The system then transitions to a new state s_{t+1} according to the environment dynamics $P(s_{t+1} \mid s_t, a_t)$ and receives a scalar reward $R(s_t, a_t)$. The goal is to maximise the expected discounted return $\mathbb{E}\big[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\big]$ with discount factor $\gamma \in (0, 1)$. Episodes terminate when the UAV reaches the goal, collides with an obstacle, or a time limit of 300 s is reached; thus the task is episodic even though the theoretical objective is defined over an infinite horizon. In this section we describe the design of the observation and action spaces and the reward function.

3.1 Observation and Action Space Design

The observation vector contains measurements that capture the UAV's pose, environment and task progress. It is continuous and 35-dimensional in total. Table 1 summarises the

components and their dimensionalities. These features are normalised using running statistics during training. The action space is continuous: the agent outputs body-frame velocity commands $a_t = [v_x, v_y, v_z, \omega]$ where v_x, v_y, v_z are translational velocities in metres per second and ω is the yaw rate in radians per second. These commands are fed through a low-level PID controller that translates them into motor thrusts. Continuous actions allow smooth acceleration and deceleration; discretising the action space would restrict manoeuvrability and increase jerk.

Table 1: Observation space design. Each episode starts with the goal vector and battery level reset; the histogram and error estimates are updated at 20 Hz.

Feature	**Dim.**	**Description**
Pose	7	3D position and quaternion orientation from VI-SLAM.
Velocity	4	Body-frame linear velocities (v_x, v_y, v_z) and yaw rate ω.
Goal vector	3	Difference between current position and goal in 3D coordinates.
Battery level	1	Remaining battery fraction in $[0, 1]$.
Occupancy histogram	24	Distances to nearest obstacles in 24 angular sectors around the UAV, capturing local free space.
Localisation error	1	Estimate of absolute trajectory error (ATE) from SLAM.

3.2 Reward Function Design

The reward function encourages goal progress, penalises time and control effort, and imposes large penalties for collisions. Following our previous work, we adopt a structured reward of the form

$$R(s_t, a_t) = 500\,\mathbf{1}_{\{\text{goal}\}} - 5\,d_t - 0.1\,\Delta t - 0.01\sum_{i=1}^{4} u_i^2 - 10\,j_t - 1000\,c_t, \tag{2}$$

where $\mathbf{1}_{\{\text{goal}\}}$ is an indicator that becomes 1 upon reaching the goal, d_t is the current Euclidean distance to the goal, Δt is the duration of the control step (0.05 s in our simulation), u_i are the motor thrusts corresponding to each rotor, j_t is a jerk penalty computed as the squared time derivative of the commanded velocities, and c_t is a binary collision indicator. This formulation matches the reward used in our earlier report [7]. The first term provides a substantial bonus upon reaching the target; the next two terms penalise remaining distance and elapsed time, promoting efficient navigation; the fourth and fifth terms minimise energy consumption and encourage smooth changes in velocity; and the final term imposes a severe cost for collisions. These numerical weights can be tuned via ablation studies to trade off success rate and energy.

Note that the reward depends implicitly on the continuous action: motor thrusts and jerk are functions of the velocity commands through the UAV dynamics. Even though the reward function is written in terms of thrust and jerk rather than the raw action vector, the action choice influences these quantities and hence the reward. Accordingly, there is no contradiction with the definition of the MDP in which R depends on (s_t, a_t).

4 Algorithmic Framework

Our system comprises four modules that operate synergistically: visual–inertial SLAM for localisation, A* for global planning, safe RRT* for local replanning, and PPO for energy-aware control. Figure 1 summarises the pipeline.

4.1 Localisation: ORB-SLAM3

We employ ORB-SLAM3 for localisation. A forward-facing stereo camera provides images at 30 Hz, while an inertial measurement unit (IMU) supplies accelerometer and gyroscope readings at 200 Hz. ORB-SLAM3 extracts ORB features, matches them across frames, integrates inertial data into a tightly coupled visual–inertial bundle adjustment and maintains multiple local maps. When tracking loss occurs (e.g., due to rapid rotation or occlusion), ORB-SLAM3 switches to a new map and later merges it when revisiting known areas. The system outputs 6-DoF poses relative to an arbitrary origin, as well as an estimate of the current localisation covariance. We use this pose as input to our state vector and the covariance to estimate localisation error.

4.2 Global Planning: A* on a Five-Floor Grid

We discretise the environment into a 3D occupancy grid. Each floor is divided into cells of size 0.5 m; vertical edges connect cells at stairs and lifts. A cell is free if no obstacle is present; otherwise, it is blocked. Given the start and goal coordinates, A* performs search on this graph. The cost $g(n)$ accumulates the Euclidean distance along the path. The heuristic $h(n)$ is the straight-line distance to the goal. To discourage unnecessary floor transitions, we add a constant vertical penalty ϕ_{floor} each time the path changes floors. A* terminates when the goal cell is popped from the open list, at which point the path is extracted. Because the grid is moderately sized (on the order of $20 \times 40 \times 5 = 4000$ cells), A* returns a path in milliseconds on a standard CPU.

4.3 Local Replanning: S-RRT*

Dynamic obstacles such as moving people or temporary barriers may invalidate the global path. Our S-RRT* module triggers when sensors detect a moving obstacle within a safety radius. The local planner constructs a space–time search region around the UAV's current position and the next waypoint in the global path. Samples are drawn from this region and

connected to the existing tree if the motion is collision-free. The cost of an edge is

$$C = \ell + \lambda_c \frac{1}{d_{\min}} + \lambda_\kappa \kappa^2, \tag{3}$$

where ℓ is the Euclidean length, $d_{\min}$ is the minimum clearance to obstacles along the edge, and κ is the curvature. The parameters λ_c and λ_κ weight the contributions of clearance and curvature, respectively. Rewiring is performed to seek lower-cost connections. If a new path to the goal (or the next safe waypoint) is found, it replaces the segment of the global path ahead. To handle moving obstacles, the planner accounts for predicted obstacle motion over a short horizon and ensures that proposed trajectories maintain a minimum separation in both space and time. This variant draws inspiration from Safe Interval RRT* [5] but operates on a single UAV by reasoning about dynamic human obstacles.

4.4 Energy-Aware Control: PPO Policy

The high-level planners output a sequence of waypoints in Cartesian coordinates. To follow this sequence while minimising energy use, we train a PPO controller. The actor network maps the state vector described in Section **??** to a mean action; the critic estimates the value function. Gaussian exploration noise with learnable standard deviation is applied during training; at deployment time, we set the variance to zero. The PPO update uses mini-batches of 2048 transitions and 10 epochs per update, with a clipping parameter $\epsilon = 0.2$ and Adam optimiser. The reward weights in Eq. (**??**) are tuned via grid search on the two-floor environment; we found values $w_g = 500$, $w_t = 0.1$, $w_u = 0.01$, $w_j = 10$, $w_c = 1000$ and $w_s = 1000$ to yield good performance [7]. These weights prioritise goal reaching and collision avoidance while penalising energy consumption and jerks. Training proceeds for several million steps until the policy converges.

5 Implementation Details

This section presents the environment construction, training procedures and baselines used to evaluate our system. We emphasise reproducibility by describing the simulation setup and hyperparameters.

5.1 Environment Construction

The five-floor building used for evaluation is based on a realistic layout comprising corridors, rooms and staircases. Each floor measures approximately 20 m by 40 m with a ceiling height of 3 m. A vertical shaft at one corner houses a lift and staircase connecting the floors. Walls and obstacles are extracted from architectural blueprints and encoded into the occupancy grid. Dynamic obstacles are simulated as human agents walking along corridors at speeds up to 1.5 m / s. The UAV model matches a small quadrotor: mass 1.5 kg, maximum thrust per motor 15 N and maximum translational speed 5 m / s. We implement the simulation using the AirSim platform, which provides realistic physics and sensor models. An altitude hold controller stabilises the UAV's vertical position; horizontal velocities are controlled by our PPO policy.

5.2 Data Collection and Training

Episodes are generated by sampling start and goal positions uniformly across all floors. Each episode ends when the UAV reaches the goal within a 0.5 m tolerance or when a maximum time of 300 s elapses. Training proceeds in two phases. First, we employ curriculum learning: the policy is initially trained on a single floor with static obstacles, then two floors with dynamic obstacles, and finally all five floors. Curriculum learning facilitates stable training and faster convergence. Second, we fine-tune the policy on the full environment with reward weights adjusted for smoother control. During training, the global and local planners run synchronously: the global path is recomputed whenever start or goal changes; local replanning triggers when dynamic obstacles appear. Training uses a batch size of 2048 steps and a total of 5 million steps. Each update uses Generalised Advantage Estimation with $\lambda = 0.95$ and the discount factor $\gamma = 0.99$. We record energy consumption by integrating the motor thrust squared over time.

5.3 Baselines and Evaluation Metrics

To contextualise our results, we compare against three baselines:

- **A* Only**: The UAV follows the global A* path using a PID controller; there is no learning. This baseline minimises path length but does not optimise energy. Collisions are handled by stopping when an obstacle is detected.
- **RRT* + PID**: We replace A* with classical RRT* (without safety terms); the controller uses a PID to follow the resulting path. This baseline tests sampling-based planning without RL.
- **Random Exploration**: The UAV moves randomly until it reaches the goal or times out, providing a lower bound on performance.

Evaluation metrics include success rate (fraction of episodes reaching the goal), mean energy consumption per successful episode, mean flight time, collision rate (fraction of episodes with any collision), and average trajectory error (ATE) measured as the root-mean-square deviation between the VI-SLAM estimated trajectory and the ground truth in simulation [7]. We emphasise energy efficiency because battery life is the primary constraint for indoor delivery.

5.4 Training Hyperparameters

Table 2 lists the key training hyperparameters for the PPO agent. These settings mirror those used in our earlier experiments and provide a baseline for future tuning. The rollout length n_{steps} determines the number of environment steps per policy update; the batch size controls the number of samples used per gradient step; the clip range is the PPO trust region parameter; and the entropy coefficient encourages exploration.

Table 2: PPO training hyperparameters. These values follow the configuration reported in our prior work [7].

Parameter	**Value**	**Description**
Learning rate	3×10^{-4}	Adam optimiser step size.
Rollout length n_{steps}	2048	Environment steps per update.
Batch size	64	Size of mini-batches for gradient updates.
Epochs per update	10	Number of passes over each batch.
Clip range	0.2	PPO clipping parameter.
Discount factor γ	0.99	Weighting of future rewards.
GAE parameter λ	0.95	Generalised advantage estimation decay.
Entropy coefficient	0.01	Encourages exploration.
Hidden layer sizes	(256, 128, 64)	Sizes of the three hidden layers in actor and critic networks.
Activation function	`tanh`	Non-linearity after each hidden layer.

6 Preliminary Results

Due to ongoing training on the five-floor map, quantitative results are currently available only for the two-floor environment from our prior work. Table 3 summarises the performance of our PPO-based controller relative to baselines. The RL policy achieves a success rate of 96 % and significantly reduces energy consumption compared with the A* only baseline (610 J vs 820 J), while maintaining a low collision rate. It also performs slightly better than RRT*+PID. We anticipate similar or improved performance on the five-floor map, though additional training is required to handle the greater complexity.

Table 3: Performance on the two-floor environment (mean ± standard deviation). The RL controller reduces energy use while maintaining high success rates.

Method	**Success Rate** (%)	**Energy** (J)	**Time** (s)	**Collisions** (%)	**ATE** (m)
A* Only	92	820 ± 40	32 ± 8	1.2	0.11 ± 0.02
RRT* + PID	94	720 ± 60	35 ± 10	2.0	0.12 ± 0.03
PPO (ours)	**96**	610 ± 30	31 ± 7	**0.7**	0.08 ± 0.02
Random	15	> 1000	> 300	5.0	0.30 ± 0.10

These results demonstrate that combining classical planning with reinforcement learning yields substantial energy savings while preserving safety and accuracy. The RL controller learns to exploit smooth trajectories and maintain moderate velocities, thereby reducing thrust squared costs and jerk penalties. Even though the action space lacks explicit energy variables, the reward design couples control effort to energy via thrust squared and jerk terms [7].

7 Discussion

Our system integrates multiple components to address the challenges of indoor delivery. The global A* planner efficiently computes a near-optimal path through a discretised representa-

tion of the building, while the S-RRT* local planner ensures safety in the presence of dynamic obstacles by enforcing clearance and curvature constraints. The RL controller bridges the gap between discrete planning and continuous control, optimising energy consumption without sacrificing task completion. The use of visual–inertial SLAM provides accurate localisation, enabling the MDP to be Markovian. Compared with purely model-based approaches, RL can account for aerodynamic effects and unmodelled disturbances. Nevertheless, training remains computationally intensive, particularly in complex environments with multiple floors. Curriculum learning partially alleviates this but does not obviate the need for large amounts of simulation data.

Our approach builds on existing literature in several ways. First, we demonstrate that energy-aware reinforcement learning can be effectively integrated with A* planning, producing trajectories that are both safe and energy-efficient. Second, we adapt S-RRT* to handle human obstacles in indoor spaces, drawing inspiration from safe interval planning and curvature penalties [5]. Third, we provide a detailed description of the state representation, reward function and network architecture, which should facilitate replication by other researchers. A limitation of our current work is the reliance on simulation; transferring policies to real UAVs requires domain randomisation or additional robustness training. Furthermore, the reward weights in Eq. (**??**) must be tuned empirically and may not generalise to all buildings.

8 Conclusion and Future Work

We have presented an expanded report on our indoor UAV delivery project, detailing the algorithms and system architecture for navigating a five-floor building with limited energy. The combination of VI-SLAM, A* global planning, S-RRT* local replanning and PPO-based energy-aware control offers a promising solution for safe and efficient indoor delivery. Preliminary results on a smaller environment show substantial energy savings and high success rates compared with baselines. Ongoing work focuses on completing training on the five-floor environment, conducting ablation studies on the influence of clearance and curvature weights in S-RRT*, and integrating multi-robot safe interval planners for cooperative delivery. Future experiments will evaluate the transfer of the learned policy to a physical UAV and investigate learning shared representations across multiple buildings.

References

[1] C. Campos *et al.*, "ORB–SLAM3: An accurate open-source library for visual, visual–inertial and multi-map SLAM," *arXiv*, 2021.

[2] "A* search algorithm," Wikipedia, accessed Oct. 2025.

[3] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[4] J. Gammell, S. Scherer and S. Singh, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2014.

[5] J. Sim *et al.*, "Safe interval RRT* for scalable multi-robot path planning in continuous space," *arXiv*, 2025.

[6] J. Schulman *et al.*, "Proximal policy optimisation algorithms," *arXiv*, 2017.

[7] Huynh Nhut Huy *et al.*, "Resource-optimised indoor drone delivery via visual–inertial SLAM and reinforcement learning," 2024.

[8] X. Shu *et al.*, "Energy-saving multi-agent deep reinforcement learning algorithm for drone routing problem," *Sensors*, vol. 24, no. 20, 2024.

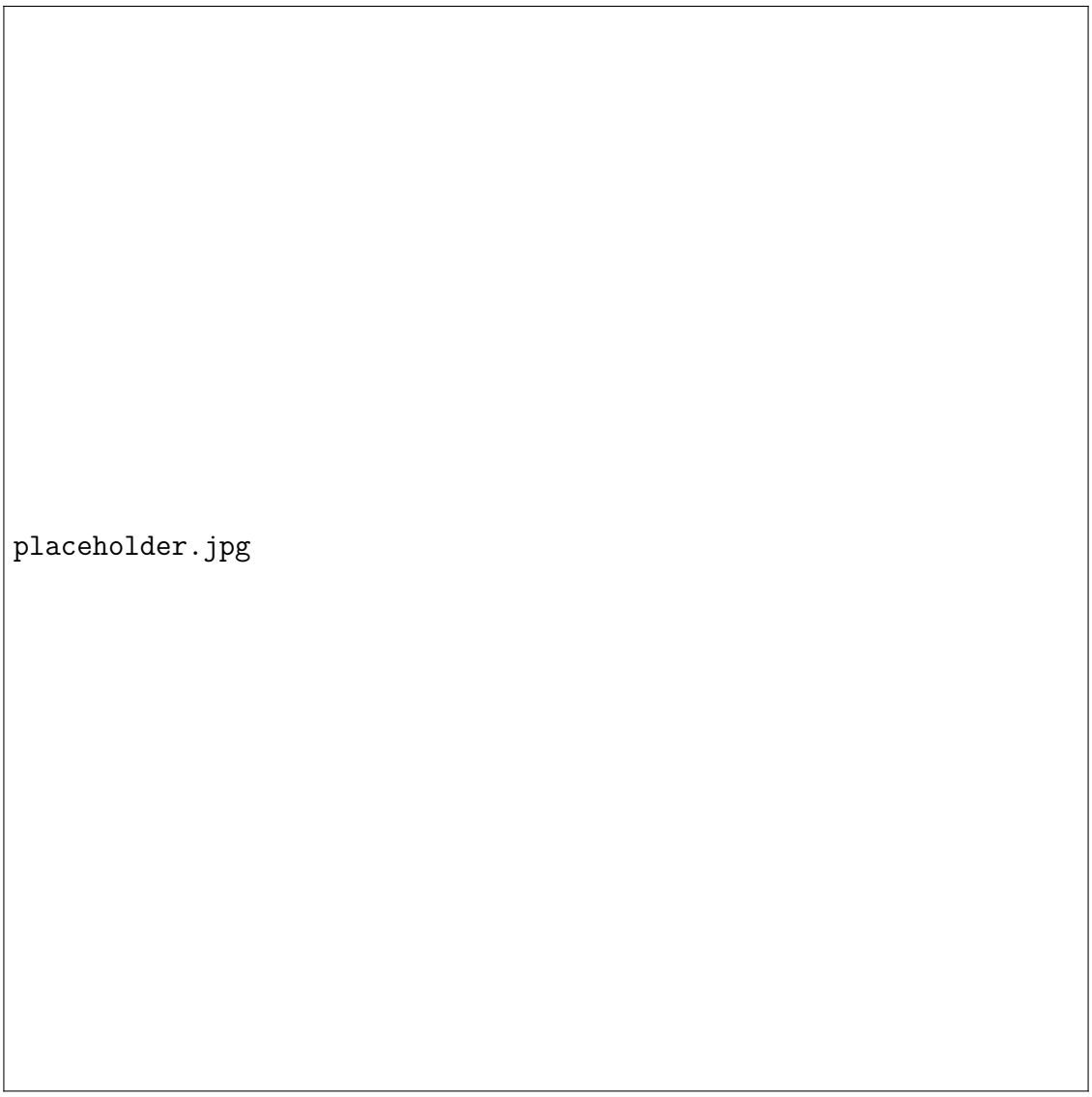

Figure 1: Overall system architecture: the VI-SLAM module estimates pose and builds a map; A* generates a global path on the five-floor occupancy grid; S-RRT* performs local replanning when dynamic obstacles are detected; PPO outputs velocity commands that track the current path while minimising energy consumption via the reward (**??**).