Drones and Autonomous Vehicles

SciEPublish

Article

Tunnel Navigation with a Nano Quadrotor via Signal Temporal Logic-Guided Reinforcement Learning

Mahrokh Hosseinkhani Hezaveh [1,*], Howard Schwartz [1] and Mehdi Foroozan Koodiyani [2]

[1] Department of Systems and Computer Engineering, Carleton University, 1125 Colonel By Drive, Ottawa, ON K1S 5B6, Canada; schwartz@sce.carleton.ca (H.S.)

[2] Department of Mechanical Engineering, K. N. Toosi University of Technology, 470 Mirdamad Ave., Tehran 19697, Iran; forouzan.mehdi73@gmail.com (M.F.K.)

* Corresponding author. E-mail: mahrokhhosseinkhanih@cmail.carleton.ca (M.H.H.)

Received: 21 August 2026; Revised: 19 October 2026; Accepted: 3 November 2026; Available online: 24 November 2026

ABSTRACT: Autonomous navigation in confined environments, such as narrow tunnels, presents a notable challenge due to restricted maneuverability, constant proximity to obstacles, and the need for reliable perception and control systems. Nanoscale multirotors are particularly well suited to such tasks. This study introduces a control strategy for tunnel navigation based on reinforcement learning, employing the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm. To enable training, a custom three-dimensional tunnel environment with range sensor emulation was developed. The reward function incorporates Signal Temporal Logic (STL) robustness measures, formulated with a smooth minimum over a sliding temporal window. Three STL shaping modes are supported: safety, centering, and alignment, allowing the training objective to emphasize wall clearance, centerline tracking, or heading alignment. The mode is selected according to the behavioral requirement to be emphasized during training. A simulation comparison with TD3 without STL shaping found 100% completion for both centering and alignment shaping across the evaluated tunnels and seeds. The learned centering policy was deployed on a Crazyflie 2.1 Brushless (Bitcraze AB, Malmö, Sweden) nano quadrotor and flown through a physical tunnel course 0.6 m wide and over 17 m long at Carleton University.

Keywords: Reinforcement learning; Signal temporal logic; Confined-space inspection; Unmanned aerial vehicle

1. Introduction

Confined-space workplace incidents result in approximately 130 fatalities annually on average in the United States (88 to 166 per year between 2011 and 2018) and over 100 in Canada [1,2]. Smaller counts are reported elsewhere, with 29 confined-space fatalities reported to the UK Health and Safety Executive between 2003 and 2011 and 8 in Australia over the same period, although under-reporting is suspected [3]. Such environments, including tunnels, pipelines, and industrial conduits, present significant risks to operators due to limited accessibility, hazardous atmospheres, and complex geometries. Advances in

© 2026 The authors. This is an open access article under the Creative Commons Attribution 4.0 International License (https://creativecommons.org/licenses/by/4.0/).

unmanned aerial vehicle (UAV) technology have enabled smaller aerial vehicles, which can autonomously navigate confined environments such as tunnels. Their compact form factor, agility, and ease of deployment make them particularly suitable for tasks such as infrastructure inspection, environmental mapping, and search and rescue missions [4,5]. Although navigation in tunnel-like environments may appear straightforward in theory, practical implementation reveals numerous operational challenges. These challenges include the limitations of low-resolution onboard sensing [6], the unreliability of state estimation due to constrained localization cues, the presence of irregular and non-smooth wall geometries, and the restrictions on visual perception [7]. Such factors necessitate the development of robust control and perception strategies tailored specifically to the requirements of confined-space navigation.

The domain of confined-space inspection remains relatively underexplored, particularly in the context of autonomous aerial navigation. Recent efforts have begun to address this challenge from varying perspectives. For example, in ref. [7], the authors focus on one of the key difficulties in tunnel navigation, strong ego-induced airflow disturbances that significantly affect flight stability and perception. To mitigate this, they propose a novel system that integrates a virtual omnidirectional perception module constructed from three red-green-blue-depth (RGB-D) cameras with a motion planning framework that explicitly accounts for perception uncertainties and aerodynamic disturbances. The system leverages both optical flow analysis and computational fluid dynamics (CFD) modeling to generate control policies that are robust to airflow-induced deviations and visibility limitations in narrow environments. The work in ref. [8] introduces a real-time, event-based vision system to estimate aerodynamic disturbances in drones navigating narrow pipes. Key contributions include the first event-based smoke velocimetry method with submillisecond latency and high accuracy, a disturbance estimator that measures force and torque effects in real time, and a neural controller based on reinforcement learning that improves flight stability. Although the approach is technically strong and achieves excellent performance, it relies on specialized hardware, particularly event-based cameras [9], which are currently expensive and not widely available as mass-market components.

This work develops an STL-guided reinforcement learning framework for nano-quadrotor navigation in narrow tunnels. Its contribution lies in designing and experimentally implementing a task-specific reward-shaping framework, rather than in a new TD3 algorithm or a new definition of STL robustness. The framework combines instantaneous navigation rewards with a smooth minimum of recent predicate margins and an adaptive violation penalty. Three selectable predicates (wall clearance, centering, and heading alignment) provide a common formulation for investigating different navigation objectives. The two contributions are as follows.

- The first contribution is a windowed STL reward-shaping formulation. Unlike constrained reinforcement learning formulations that optimize performance subject to explicit cost constraints, it incorporates STL-derived robustness into a scalar reward. It therefore encourages specification satisfaction without guaranteeing it. The temporal window retains the influence of recent poor margins, while the adaptive penalty increases the cost of negative surrogate robustness during training.
- The second contribution is implementing the framework in a custom three-dimensional tunnel simulator and deploying the learned centering policy through the Crazyflie velocity-control interface. Physical flights through a tunnel approximately 0.6 m wide and over 17 m long demonstrate deployment feasibility.

2. Related Work

Autonomous aerial navigation in underground and confined structures has been studied from several directions. A review of UAV navigation for underground structure inspection (ref. [10]) describes the unavailability of satellite positioning and the scarcity of visual features in old tunnels and caves. It discusses the fusion of LiDAR and cameras for localization and learning-based obstacle avoidance, whose use is

limited by computing power and training requirements. Legged and aerial robots have also been deployed for autonomous exploration in tunnels and urban circuits of the DARPA Subterranean Challenge [11], and micro aerial vehicle navigation in unknown and dark mining tunnels has been investigated in ref. [12]. Reinforcement learning has been applied to quadrotor control in ref. [13]. Data-driven neural predictive control has been applied to trajectory tracking of autonomous vehicles [14]. In contrast, the present study uses a nano quadrotor with four low-resolution range sensors and learns the navigation policy with STL-shaped reinforcement learning.

Designing effective reward signals remains a major challenge in reinforcement learning, especially for systems where safety and timing constraints are critical. Reward-function design has been investigated using a wide range of methods [15]. The objective of inverse reinforcement learning is to recover a reward function that explains the observed expert behavior [16]. According to ref. [17], human-in-the-loop reinforcement learning highlights the importance of integrating human feedback and supervision at all stages of the learning process. These methods improve safety, interpretability, and robustness, especially in practical scenarios where fully autonomous decision making might fail.

STL enables the encoding of complex temporal and spatial requirements, allowing designers to embed domain knowledge and behavioral constraints in robotic learning and control systems [18]. Recent research has investigated incorporating STL specifications into autonomous control systems. In the path planning layer, ref. [19] introduced a strategy in which agents navigate through temporally scheduled waypoints, ensuring coordinated task execution and formal adherence to STL constraints, including collision prevention.

In learning-based controllers, more specifically in reinforcement learning controllers, STL provides structured guidance for policy learning [20]. STL incorporates quantitative semantics [21,22] that assign robustness values to system executions [23,24]. These values measure the degree to which a trajectory complies with or deviates from a given STL specification, providing a continuous metric for guiding learning. Refs. [22,25–27] explore using STL to formally specify desired behaviors for reinforcement learning agents. Instead of relying on hand-crafted or informal reward functions, the authors use STL to guide behavior and propose methods for locally shaping rewards at each state to help the agent satisfy these formal specifications. They validate the approach in simulation with a cart-pole balancing task [26,27] and quadrotor navigation [22]. Safe reinforcement learning methods instead handle constraints through cost-constrained or Lagrangian formulations [28,29] or through shielding [30].

Our approach draws on quantitative STL semantics and smooth robustness approximations [23,31]. These concepts provide numerical measures associated with explicitly stated requirements. In this work, one navigation predicate is selected at a time; the framework does not jointly optimize a weighted conjunction of all three specifications.

Adapting this idea to reinforcement learning, we apply a smooth minimum computation to a short history of recent physical safety, centering, or alignment margins. The term worst margin denotes the minimum robustness margin identified within this sliding window, indicating the moment the agent was nearest to violating a constraint. For example, in safety mode, this corresponds to the moment when the drone approached a tunnel wall most closely, that is, distance minus the required clearance. Even if the agent is well positioned most of the time, a single near collision lowers the worst margin and thus reveals the most critical short-term risk.

To incorporate this history of robustness margins into reinforcement learning, we replace the exact minimum with a log-sum-exp (LSE) soft minimum [31]. This smooth approximation emphasizes poor recent margins while allowing other samples within the window to contribute. Its approximation properties and limitations are described in Section 5.3.2.

Related developments in networked control include asynchronous bipartite consensus over quasi-structurally balanced signed networks [32] and consensus tracking over asynchronous cooperation-

competition networks [33]. These studies address coordination among interacting agents, whereas the present work concerns STL-guided reinforcement learning for single-vehicle tunnel navigation.

3. Preliminaries

3.1. Reinforcement Learning

Reinforcement learning (RL) is a framework for sequential decision-making, typically modeled as a Markov Decision Process (MDP). An MDP is defined by the tuple $M = (S, A, r, P, \gamma)$, where S is the state space, A is the action space, $r(s, a)$ is the reward function, $P(s' \mid s, a)$ is the state transition probability, and $\gamma \in (0,1)$ is a discount factor that weights future rewards [34]. The objective is to determine a policy π that maximizes the expected cumulative discounted reward

$$\eta(\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right].$$

The policy is represented as a mapping

$$\pi(s, a) = P(a_t = a \mid s_t = s),$$

which specifies the probability of selecting action a in state s. In practice, a common exploration strategy is the ϵ-greedy policy: with probability $1 - \epsilon$ the agent selects the greedy action $\text{argmax}_a Q(s, a)$, and with probability ϵ an action is selected uniformly at random from the action set.

Q-learning estimates the action-value function $Q^\pi(s, a)$, defined as the expected discounted return obtained by taking action a in state s and thereafter following policy π:

$$Q^\pi(s_t, a_t) = \mathbb{E}[r(s_t, a_t) + \gamma Q^\pi(s_{t+1}, a_{t+1})].$$

The optimal action-value function satisfies the Bellman optimality condition

$$Q^*(s, a) = \mathbb{E}\left[r(s, a) + \gamma \max_{a' \in A} Q^*(s', a')\right].$$

Deep Q-learning extends this framework by approximating $Q(s, a)$ with a neural network parameterized by θ, enabling the method to scale to high-dimensional and continuous observation spaces through mechanisms such as experience replay.

3.2. Twin Delayed Deep Deterministic Policy Gradient

The tunnel navigation task requires continuous lateral, vertical, and yaw commands, so value-based methods such as deep Q-learning, which select actions by maximizing over a discrete action set, are not directly applicable. We therefore train the policy with the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm [35], an off-policy actor-critic method designed for continuous control.

TD3 maintains a deterministic actor $\pi_\phi(s)$, which maps an observation to an action, and two critics $Q_{\theta_1}(s, a)$ and $Q_{\theta_2}(s, a)$, which estimate the action-value function. Target networks $\pi_{\phi'}$, $Q_{\theta_1'}$, and $Q_{\theta_2'}$ track the online networks through slow Polyak averaging. Transitions are stored in a replay buffer and sampled in minibatches, and exploration during training is induced by adding Gaussian noise to the actor output.

Three mechanisms distinguish TD3 from its deterministic policy gradient predecessors and address the value overestimation and brittleness that affect them. First, clipped double-Q learning forms the critic target from the smaller of the two target critics,

$$y = r + \gamma \min_{i=1,2} Q_{\theta_i'}(s', \pi_{\phi'}(s') + \epsilon),$$

which suppresses the systematic overestimation that arises when a single critic is maximized. Both critics are then regressed onto this shared target by minimizing the mean squared Bellman error.

Second, target policy smoothing adds clipped noise to the target action,

$$\epsilon \sim \mathrm{clip}(\mathcal{N}(0,\sigma), -c, c),$$

so the value estimate averages over a small neighborhood of the target action rather than evaluating it at a single point. This regularizes the critic against sharp peaks in the value landscape that a deterministic policy would otherwise exploit.

Third, the actor and the target networks are updated less frequently than the critics, typically once every two critic updates. Delaying the policy update allows the value estimates to stabilize before each policy improvement step, reducing the error accumulation that follows from optimizing a policy against a rapidly changing critic. The actor is updated by the deterministic policy gradient.

$$\nabla_\phi J(\phi) = \mathbb{E}\left[\nabla_a Q_{\theta_1}(s,a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)\right].$$

These properties make TD3 well suited to the present task, where the reward is dense but shaped by several competing terms, and the policy must produce smooth continuous commands over long episodes. The hyperparameters used for training are listed in Table 1.

Table 1. Hyperparameters and STL margins used in TD3 training.

Hyperparameter	Value
Policy network	MLP (2 hidden layers, 400 and 300 units)
Optimizer	Adam
Learning rate	linear decay $3 \times 10^{-3} \rightarrow 3 \times 10^{-6}$
Discount factor γ	0.99
Replay buffer size	20,000
Batch size	64
Action space	$[a_{\mathrm{lat}}, a_{\mathrm{ver}}, a_{\mathrm{yaw}}]$
Training step	0.1 s
Episode length	task-dependent
Safety margin m^*	0.15 m
Centering bounds α, β	0.25 m
Alignment bound γ	0.30 rad
STL window W	5 samples at 0.1 s intervals
Soft-min temperature τ	0.05 m (safety/centering); 0.05 rad (alignment)
Reward gain k (Equation (6))	4
Positive shaping weight w_{STL}	0.9
Comfort threshold ρ_{target} (normalized)	0.25
Training steps per run	400,000
Exploration noise	Gaussian, $\sigma = 0.1$ decaying linearly to 0
Dual-ascent rate η	10^{-3}
Penalty bound $\lambda_{\max}$	50
Initial penalty $\lambda(0)$	2

3.3. Signal Temporal Logic

STL [31] provides a formal language for specifying high-level behavioral requirements that combine logical predicates with temporal constraints. At the core of STL are predicates, which are conditions on the system state. A predicate ϕ is defined by a real-valued function $h: S \rightarrow \mathbb{R}$ over the state space S:

$$\phi(s) := \begin{cases} \text{True}, & \text{if } h(s) \geq 0, \\ \text{False}, & \text{if } h(s) < 0. \end{cases}$$

Intuitively, the function $h(s)$ measures how well the state s satisfies a constraint, with $h(s) \geq 0$ indicating satisfaction and $h(s) < 0$ indicating violation. For example, $h(s) = d(s) - m^*$ could represent

a safety requirement that the distance $d(s)$ to an obstacle must remain larger than a margin m^*. Using predicates as atomic building blocks, STL formulas φ are constructed according to the syntax

$$\varphi ::= \text{True} \mid \phi \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid F[a,b]\varphi \mid G[a,b]\varphi,$$

where $a, b \in \mathbb{R}_{\geq 0}$ with $a \leq b$, and φ_1, φ_2 are STL formulas. The logical operators $\neg$, $\wedge$, and $\vee$ denote negation, conjunction, and disjunction, respectively. The temporal operators $F[a,b]$ and $G[a,b]$ express requirements over a time interval $[a,b]$: $F[a,b]\varphi$ means that φ must hold at least once within the interval, that is, eventually within $[a,b]$, and $G[a,b]\varphi$ means that φ must hold for all times within the interval. Formally, for a signal $s: \mathbb{R}_{\geq 0} \to S$ that describes the trajectory of the system, satisfaction is defined recursively as

$$\begin{aligned}
s_t &\models \phi && \Leftrightarrow \phi(s_t) \text{ is True},\\
s_t &\models \neg\varphi && \Leftrightarrow s_t \not\models \varphi,\\
s_t &\models \varphi_1 \wedge \varphi_2 && \Leftrightarrow s_t \models \varphi_1 \text{ and } s_t \models \varphi_2,\\
s_t &\models \varphi_1 \vee \varphi_2 && \Leftrightarrow s_t \models \varphi_1 \text{ or } s_t \models \varphi_2,\\
s_t &\models F[a,b]\varphi && \Leftrightarrow \exists t' \in [t+a, t+b]: s_{t'} \models \varphi,\\
s_t &\models G[a,b]\varphi && \Leftrightarrow \forall t' \in [t+a, t+b]: s_{t'} \models \varphi.
\end{aligned}$$

Thus, $s_t \models \phi$ means that the atomic predicate is satisfied at time t, while $\neg\varphi$, $\varphi_1 \wedge \varphi_2$, and $\varphi_1 \vee \varphi_2$ behave as in classical logic. $F[a,b]\varphi$ means that φ becomes true at some point between $t+a$ and $t+b$, and $G[a,b]\varphi$ means that φ is true at every point between $t+a$ and $t+b$. This semantics allows STL to express rich temporal behaviors. For example, the formula

$$G[0,\infty)(d(s) \geq m^*)$$

requires that the system always maintains at least m^* clearance from obstacles, while

$$F[0{,}5](|e_\ell(s)| < \alpha)$$

requires that the lateral error e_ℓ falls below the threshold α at least once within the next 5 s. STL robustness provides a numerical measure of satisfaction. Standard minimum and maximum operators are not differentiable everywhere; smooth approximations can be introduced when desired. TD3 learns through its critics and does not require differentiation through the reward calculation.

$$\begin{aligned}
\rho_\phi(s_t) &= h(s_t),\\
\rho_{\neg\varphi}(s_t) &= -\rho_\varphi(s_t),\\
\rho_{\varphi_1\wedge\varphi_2}(s_t) &= \min\left(\rho_{\varphi_1}(s_t), \rho_{\varphi_2}(s_t)\right),\\
\rho_{\varphi_1\vee\varphi_2}(s_t) &= \max\left(\rho_{\varphi_1}(s_t), \rho_{\varphi_2}(s_t)\right),\\
\rho_{F[a,b]\varphi}(s_t) &= \max_{t'\in[t+a,t+b]} \rho_\varphi(s_{t'}),\\
\rho_{G[a,b]\varphi}(s_t) &= \min_{t'\in[t+a,t+b]} \rho_\varphi(s_{t'}).
\end{aligned}$$

These robustness formulas provide a continuous measure of STL satisfaction, where ρ_ϕ captures how strongly a predicate is satisfied or violated, negation inverts this measure, conjunction and disjunction reflect the weakest or strongest satisfaction among subformulas, and the temporal operators eventually (F) and always (G) take the maximum or minimum over the specified time interval. These quantitative robustness values can be used to construct rewards for reinforcement learning. Their use as rewards does not itself guarantee satisfaction of the specification. In this work, we restrict attention to globally quantified specifications.

4. Tunnel Navigation Problem

To enable reinforcement learning for autonomous tunnel navigation, a structured representation of the tunnel environment and onboard sensing modalities must first be established. This representation defines the perceptual inputs available to the UAV as well as the set of admissible control actions. A task-specific

reward function is then formulated to capture key performance objectives, including collision avoidance, centerline tracking, and proper orientation. With the environment, observation model, and reward function in place, a reinforcement learning algorithm can be trained to synthesize a control policy capable of guiding the UAV reliably through the tunnel.

4.1. Range Sensor Model

In this research, we deploy an RL agent to enable autonomous navigation of the Crazyflie nanocopter, a compact and open-source platform well suited for research purposes [36]. The Crazyflie features a range sensor deck, equipped with several sensors that detect nearby obstacles and surfaces to support safe navigation. Figure 1 illustrates the Multi-ranger deck (Bitcraze AB, Malmö, Sweden). Figure 2 provides a top view of the tunnel environment with range-sensor beams intersecting the tunnel walls. Tunnel walls are modeled with added noise to create a wavy surface, to represent irregularities in real tunnel surfaces.

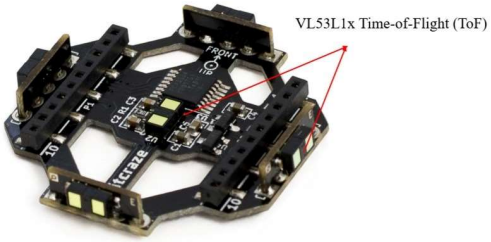


Figure 1. Multi-ranger deck of the Crazyflie quadcopter.

We modeled the range sensors in our simulated environment to replicate the real-world behavior of the Crazyflie inside a tunnel. The sensor model provides distance measurements in four directions, left, right, up, and down, allowing the agent to perceive its spatial relationship to the tunnel walls. These measurements are crucial for the agent to maintain safe distances and navigate effectively. The method computes distances from the drone to the tunnel boundaries in each direction, taking into account the drone's current position and orientation. These distances are then aggregated into the observation vector provided to the reinforcement learning agent, forming the perceptual input used for decision making during training and evaluation.

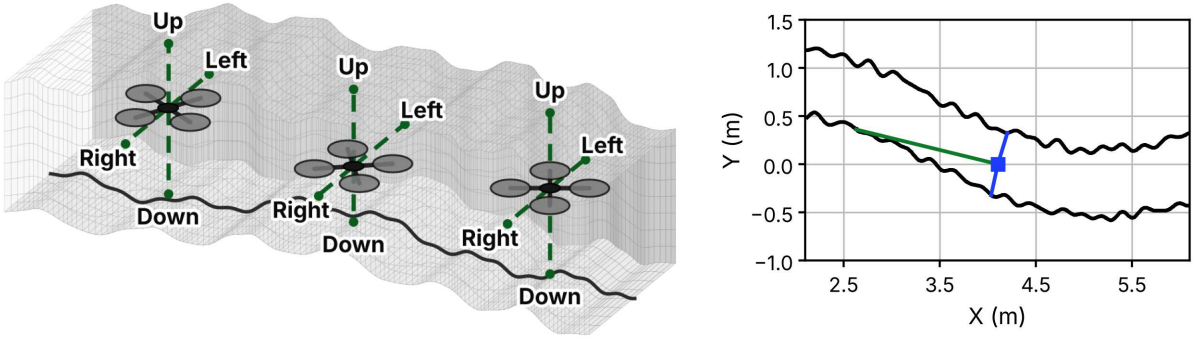


Figure 2. Three-dimensional view of the simulated tunnel with the emulated range sensor directions shown at three vehicle poses (**left**) and a cross-sectional view of the tunnel at one pose, with the measured distances to the walls (**right**).

4.2. Tunnel Simulation in the Algorithm

The tunnel environment is generated procedurally using a combination of sinusoidal functions and stochastic noise to create realistic and irregular wall boundaries. The base tunnel centerline in the lateral (y) and vertical (z) directions is defined as weighted sums of sine waves with randomized amplitudes, producing smooth oscillations along the longitudinal axis (x). To capture the natural irregularity of the real

tunnel surfaces, band-limited random noise is added independently to the right, left, upper, and lower tunnel walls. These noisy perturbations are generated as uniform samples that are resampled for continuity, resulting in small-scale waviness in the wall geometry. The tunnel cross section is defined by a nominal radius parameter R, which offsets the left and right boundaries from the centerline and determines the spacing between the upper and lower walls. Figure 3 shows a physical tunnel module. This construction produces a 3D tunnel with varying curvature, allowing the simulation of a realistic and challenging environment for UAV navigation tasks.

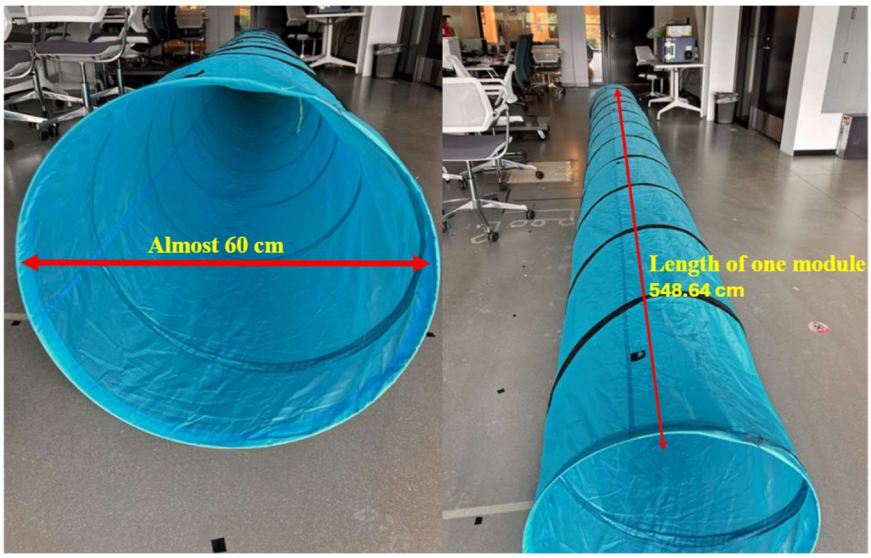


Figure 3. Tunnel emulation. This figure illustrates one of the modules in the experiment. There are three tunnel modules in total, together forming a tunnel over 17 m long with varying curves and shapes. The experiment was conducted at the Ericsson 5G Autonomous Drone Lab, located at Carleton University.

5. Learning

5.1. State, Observation, and Action Spaces

The state of the system at time t, denoted by s_t, includes the drone's position, velocity, and orientation in 3D space:

$$s_t = [x_t, y_t, z_t, \dot{x}_t, \dot{y}_t, \dot{z}_t, \theta_t, \dot{\theta}_t], \tag{1}$$

where (x_t, y_t, z_t) are the position coordinates, $(\dot{x}_t, \dot{y}_t, \dot{z}_t)$ are the corresponding velocities, θ_t is the yaw angle, and $\dot{\theta}_t$ is the yaw rate. The observation o_t available to the agent is a subset of the measurements of the state and the range sensor readings of the tunnel walls:

$$o_t = [e_{\text{left}}, e_{\text{right}}, e_{\text{up}}, e_{\text{down}}, v_{\text{lateral}}, v_{\text{vertical}}, \dot{\theta}_t], \tag{2}$$

where $e_{\text{left}}, e_{\text{right}}, e_{\text{up}}, e_{\text{down}}$ denote the distances from the drone to the tunnel boundaries measured by the range sensors, v_{lateral} and v_{vertical} are the lateral and vertical velocity components, and $\dot{\theta}_t$ is the yaw rate. The action space is continuous and three-dimensional, consisting of accelerations in the lateral, vertical, and yaw directions:

$$a_t = [a_{\text{lateral}}, a_{\text{vertical}}, a_{\text{yaw}}], \tag{3}$$

where a_{lateral} is the control input for sideways acceleration, a_{vertical} controls vertical motion, and a_{yaw} controls rotational acceleration on the yaw axis. Forward acceleration is not one of the action space components because the forward velocity is set to a constant value.

5.2. Rewards and Termination

The reward function plays a central role in shaping the agent's navigation behavior. In this work, the reward is designed to encourage safe, smooth, and efficient flight through the tunnel while discouraging risky or unstable maneuvers. At each time step t, the reward is computed as $r_t = r_t^{\text{center}} + r_t^{\text{dur}} + r_t^{\text{straight}} + r_t^{\text{effort}} + r_t^{\text{change}} + R_{\text{STL}}(t)$, where the individual terms are defined as follows.

- Centeredness (r_t^{center}): penalizes lateral and vertical deviations from the tunnel centerline, ensuring the drone remains close to the geometric middle.
- Durability (r_t^{dur}): provides a constant survival bonus to encourage forward progress and discourage premature collisions.
- Straightness (r_t^{straight}): penalizes yaw misalignment between the drone and the tunnel's local heading, promoting alignment with the tunnel direction.
- Control effort (r_t^{effort}): penalizes the magnitude of control inputs, encouraging energy-efficient and smooth actions.
- Control change (r_t^{change}): penalizes large variations in consecutive control commands, preventing abrupt maneuvers.
- STL shaping ($R_{\text{STL}}(t)$): integrates an STL robustness-based shaping term, defined in Section 5.3.3.

At each step, the drone estimates its position relative to the tunnel using range sensor readings from the left, right, up, and down walls. The differences $e_\ell = d_{\text{left}} - d_{\text{right}}$ and $e_v = d_{\text{down}} - d_{\text{up}}$ provide lateral and vertical range-difference signals, while yaw error quantifies orientation misalignment with the tunnel direction. These values directly contribute to the centeredness and straightness penalties.

5.3. STL Reward Shaping

The base reward is augmented with a term derived from the robustness of a selected STL predicate over a sliding temporal window. Whereas the base reward evaluates the current navigation state and control command, the additional term incorporates recent predicate margins, allowing a poor margin to influence the reward for several successive updates. An adaptive penalty increases the cost associated with negative surrogate robustness. These mechanisms provide reward-based guidance rather than enforcement of the specification.

5.3.1. Instantaneous Robustness Margins

At each time step t, a robustness margin is defined depending on the operating mode.

- Safety: $\rho_{\text{safety}}(t) = \min\{d_L(t), d_R(t), d_U(t), d_D(t)\} - m^*$, where d_L, d_R, d_U, d_D denote distances to tunnel walls and m^* is the safety clearance.
- Centering: $\rho_{\text{center}}(t) = \min\{\alpha - |e_\ell(t)|, \beta - |e_v(t)|\}$, where e_ℓ, e_v are the lateral and vertical range differences, and α, β the centering limits.
- Alignment: $\rho_{\text{align}}(t) = \gamma - |\delta\psi(t)|$, where $\delta\psi$ is the yaw error and γ the alignment tolerance.

Positive values denote satisfaction, while negative values indicate constraint violation.

5.3.2. Windowed Soft Minimum and Aggregated Robustness

The instantaneous margin for the selected mode is defined in Section 5.3.1 as the clearance, centering, or alignment slack relative to its specified threshold. Positive instantaneous margins indicate satisfaction, while negative instantaneous margins indicate violation.

We aggregate the most recent W margins using a log-sum-exp soft minimum. Equation (4) gives a numerically stabilized form of the unnormalized log-sum-exp expression:

$$\rho(t) = m_{\min}(t) - \tau \log\left(\sum_{i=t-W+1}^{t} \exp\left(-\frac{m_i - m_{\min}(t)}{\tau}\right)\right), \tag{4}$$

Here, $m_{\min}(t)$ is the exact minimum margin in the current window and $\tau > 0$ is the temperature, expressed in the same units as the selected predicate margin. The approximation satisfies $m_{\min}(t) - \tau \ln(W) \leq \rho(t) \leq m_{\min}(t)$. Thus, it conservatively underestimates the exact minimum, with a maximum error of $\tau \ln(W)$, and approaches that minimum as τ approaches zero. Smaller temperatures approximate the worst margin more closely; larger temperatures distribute sensitivity across more samples. The soft minimum serves as a smooth temporal aggregation over a finite window; it is not needed for gradient computation, since TD3 does not differentiate through the reward. The aggregated robustness is then normalized with respect to the relevant threshold:

$$\hat{\rho}(t) = \text{clip}\left(\frac{\rho(t)}{\text{norm}}, -1, 1\right), \tag{5}$$

The normalization threshold is m^* for safety, $\min(\alpha, \beta)$ for centering, and γ for alignment. Positive surrogate robustness implies positive margins throughout the sampled window. Negative surrogate robustness can occur even when all instantaneous margins are positive, because the approximation underestimates their minimum; it is therefore not definitive evidence of an actual predicate violation. Exact minimum robustness is the appropriate quantity for determining sampled predicate satisfaction.

The window contains five samples at a sampling interval of 0.1 s; the oldest and newest samples are separated by 0.4 s. A longer window retains poor margins for longer, including after recovery, while a shorter window provides less temporal memory. At fixed temperature, increasing the window length also increases the approximation-error bound. The selected parameters are implementation choices rather than theoretically optimal values. Their influence on policy convergence has not been isolated in the reported evaluation, which characterizes the selected configuration rather than establishing parameter-independent convergence.

5.3.3. Reward Contribution

From the normalized robustness $\hat{\rho}(t)$, three shaping terms are derived:

$$r_{\text{pos}} = \tanh\left(k\hat{\rho}(t)\right), \tag{6}$$

$$r_{\text{violate}} = \left(\max\{0, -\hat{\rho}(t)\}\right)^2, \tag{7}$$

$$r_{\text{margin}} = \max\{0, \hat{\rho}(t) - \rho_{\text{target}}\}. \tag{8}$$

Their interpretations are as follows. The term r_{pos} is a bounded shaping term that grows with robustness, meaning that larger safety margins or better alignment yield higher values, while tanh prevents unbounded growth. The term r_{violate} is a quadratic penalty that activates only when $\hat{\rho}(t) < 0$, so violations are discouraged, with severe ones penalized disproportionately more than mild ones. The term r_{margin} is an additional bonus for exceeding a comfort threshold ρ_{target}, encouraging policies that go beyond minimal satisfaction. The overall STL shaping reward is

$$R_{\text{STL}}(t) = w_{\text{STL}} r_{\text{pos}} - \lambda(t) r_{\text{violate}} + r_{\text{margin}}, \tag{9}$$

where w_{STL} is a weight for positive shaping and $\lambda(t)$ is an adaptive penalty factor updated online via dual ascent. This formulation balances three aspects: encouraging robustness, discouraging violations, and rewarding comfort margins, thereby producing learning signals that are both informative and safety-aware.

5.3.4. Adaptive Penalty via Dual Ascent

The reward includes a penalty that becomes active when the normalized surrogate robustness is negative. Its adaptive weight determines the cost of such negative values during training. This is a soft reward penalty and does not enforce a hard constraint. At each time step, we compute the surrogate violation magnitude.

$$v(t) = \max\{0, -\hat{\rho}(t)\}, \tag{10}$$

The surrogate violation magnitude is zero when normalized surrogate robustness is nonnegative and positive otherwise. The target level is set to zero in the reported experiments. The penalty weight is updated using the projected rule below:

$$\lambda(t+1) = \Pi_{[0,\lambda_{\max}]}\big(\lambda(t) + \eta\big[v(t) - v_{\text{target}}\big]\big), \tag{11}$$

Here, $\eta > 0$ is the update rate, and the projection restricts the weight to $[0, \lambda_{\max}]$. In the reported experiments, $\eta = 10^{-3}$, $\lambda_{\max} = 50$, and $\lambda(0) = 2$; λ is updated at every time step and is not reset between episodes. With a zero target level, the weight increases when surrogate robustness is negative, until it reaches the upper bound, and remains unchanged when surrogate robustness is nonnegative. The penalty therefore acts as a monotonically increasing weight, and any benefit of this adaptivity over a fixed weight was not evaluated. The update is inspired by projected multiplier methods. It does not provide a convergence or feasibility guarantee for the resulting reinforcement learning policy.

5.3.5. Examples: STL Reward Computation in Centering Mode

We illustrate how the STL shaping terms behave under both satisfaction and violation of the centering constraint. Throughout these examples, the centering bounds are $\alpha = \beta = 0.25$ m, and the STL window length is $W = 5$.

Satisfaction case. Consider the five instantaneous margins [0.15, 0.12, 0.18, 0.1, 0.14] m. Using Equation (4) with τ = 0.05 m gives $\rho(t) \approx 0.05053$ m, and normalization by 0.25 m gives approximately 0.20213. With $k = 4$ and a normalized comfort threshold of 0.25, the shaping terms are $r_{\text{pos}} \approx 0.66878$, $r_{\text{violate}} \approx 0.00000$, and $r_{\text{margin}} \approx 0.00000$. With $w_{\text{STL}} = 0.9$, the STL contribution is approximately 0.60190; the penalty weight does not affect this example because $r_{\text{violate}} = 0$.

Violation case. Consider the five instantaneous margins [0.15, 0.12, −0.05, 0.1, 0.14] m. Using Equation (4) with τ = 0.05 m gives $\rho(t) \approx -0.05584$ m, and normalization by 0.25 m gives approximately −0.22335. With $k = 4$ and a normalized comfort threshold of 0.25, the shaping terms are $r_{\text{pos}} \approx -0.71307$, $r_{\text{violate}} \approx 0.04989$, and $r_{\text{margin}} \approx 0.00000$. With $w_{\text{STL}} = 0.9$ and $\lambda(t) = 3$, the STL contribution is approximately −0.79142. The negative instantaneous margin indicates an actual violation of the centering predicate. The hyperparameters and STL margins used in training are listed in Table 1.

6. Simulation and Experimental Results

The evaluation investigates a specification-guided approach to tunnel navigation, connecting explicit clearance, centering, or alignment requirements to the learning objective through windowed robustness shaping. The simulations describe the three shaping configurations within a common TD3 framework, and the physical experiments demonstrate deployment on a nano quadrotor. A comparative benchmark against other learning algorithms and conventional controllers is outside the reported evaluation; relative performance advantages over such methods and statistical significance are therefore not established. In particular, a well-tuned classical feedback controller, such as a PD controller acting on the range differences, is a natural baseline for this task, and we do not claim that the learned policy outperforms it in the tunnel geometries considered here. Section 6.4 adds a limited simulation-only ablation against a TD3 agent trained without STL shaping. Sections 6.1 and 6.4 address the windowed STL reward shaping, Section 6.2 shows the simulated behavior of the learned centering policy, and Section 6.3 reports the physical flights on the Crazyflie.

6.1. Training Reward Progression

Figure 4 shows training returns for the safety, centering, and alignment shaping modes. All runs use the same total number of environment steps, but the number of episodes differs because episode lengths depend on survival time. The centering run contains fewer, longer episodes, whereas the safety run contains more, shorter episodes. Since reward definitions differ and the horizontal axis counts episodes rather than environment steps, these curves do not establish a common ranking of learning speed or statistically significant differences in convergence. The simulation time step was 0.1 s, matching the update interval of the multi-ranger deck sensors.

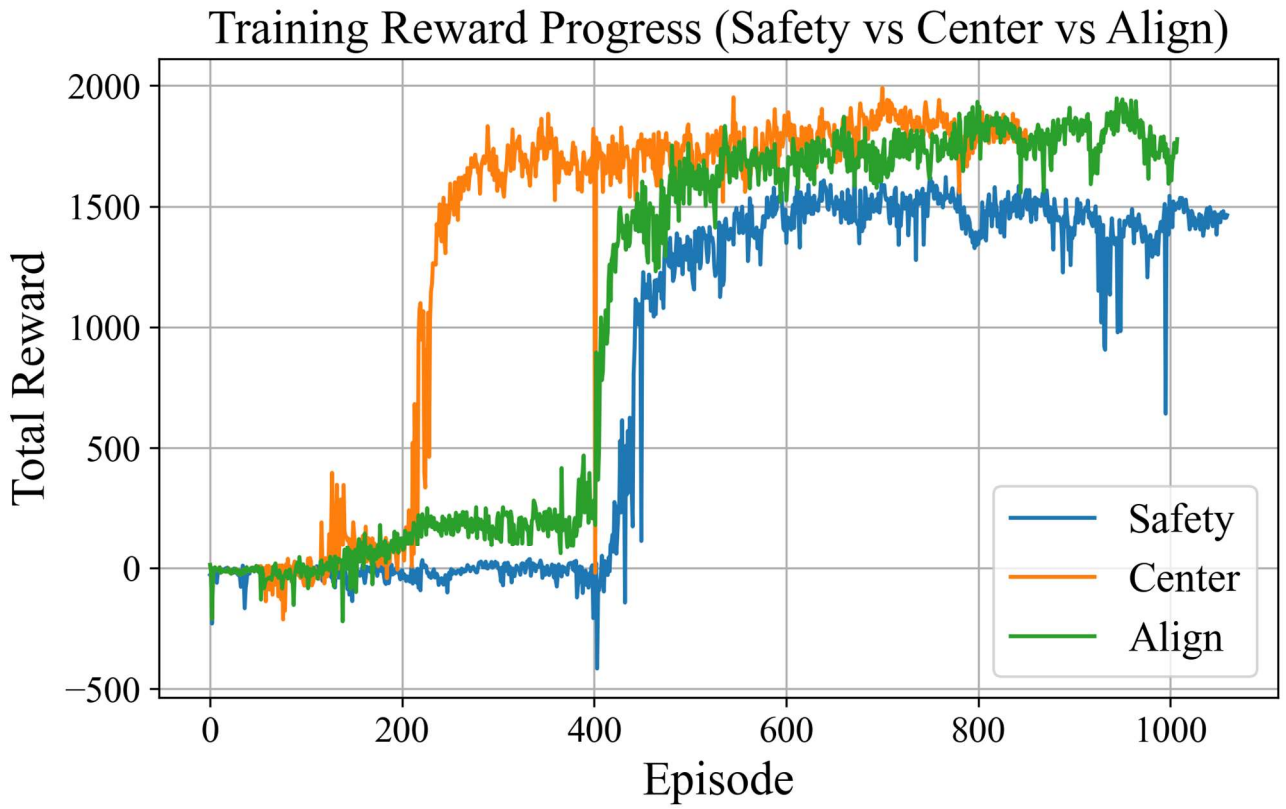


Figure 4. Training returns for the three STL shaping modes. The total environment-step budget is matched, while episode counts differ because episode lengths vary. Returns are specific to each mode and should not be interpreted as a common performance measure.

Figure 5 shows the distributions of surrogate robustness values during training. The safety-mode distribution has a wider negative tail, while the centering and alignment distributions are concentrated at positive values. Each distribution characterizes its selected predicate under the corresponding normalization threshold. Negative surrogate values are interpreted as explained in Section 5.3.2. These mode-specific distributions describe the shaping signals; they do not provide common collision, tracking-error, or control-smoothness metrics for a quantitative ablation. A limited ablation with common metrics is given in Section 6.4.

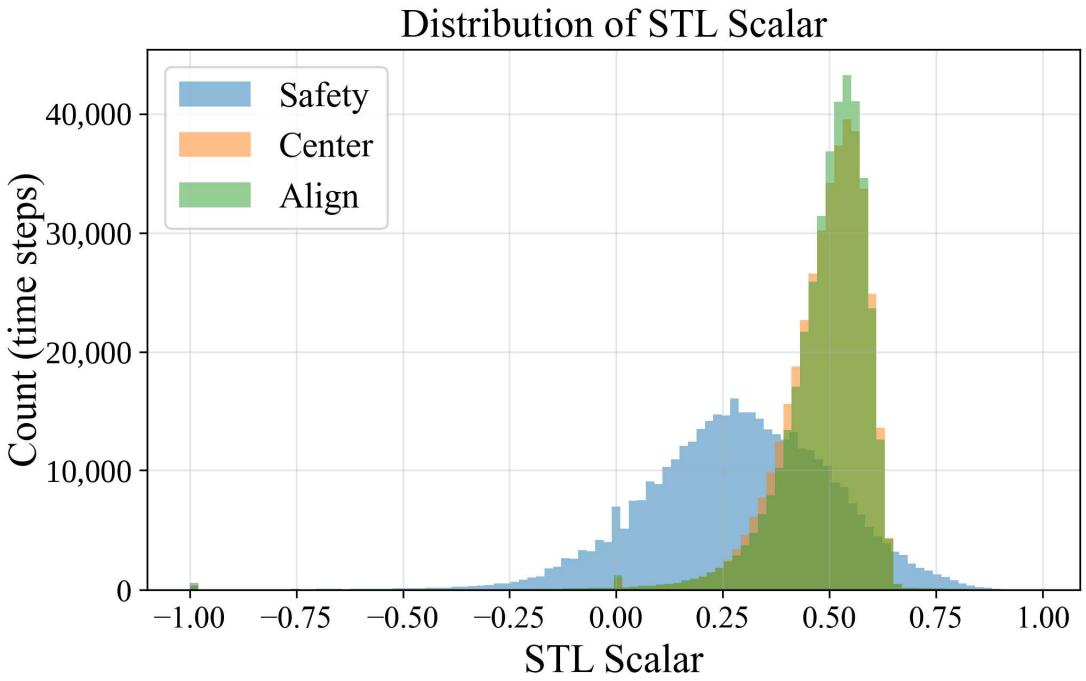


Figure 5. Distributions of mode-specific surrogate STL robustness values across training time steps. The modes use different predicates and normalization thresholds.

6.2. Trajectory Analysis in Simulation

The agent's behavior was further evaluated using trajectory and state data collected during test episodes in centering mode. A three-dimensional view of navigation is shown in Figure 6. The top two graphs compare the drone's positions y and z with the tunnel boundaries, indicating that the agent maintains safe distances from the walls while following the tunnel curvature. The yaw plot demonstrates close alignment with the tunnel heading, and the velocity plots show that both lateral and vertical speeds remain low and stable, which helps maintain smooth and controlled navigation.

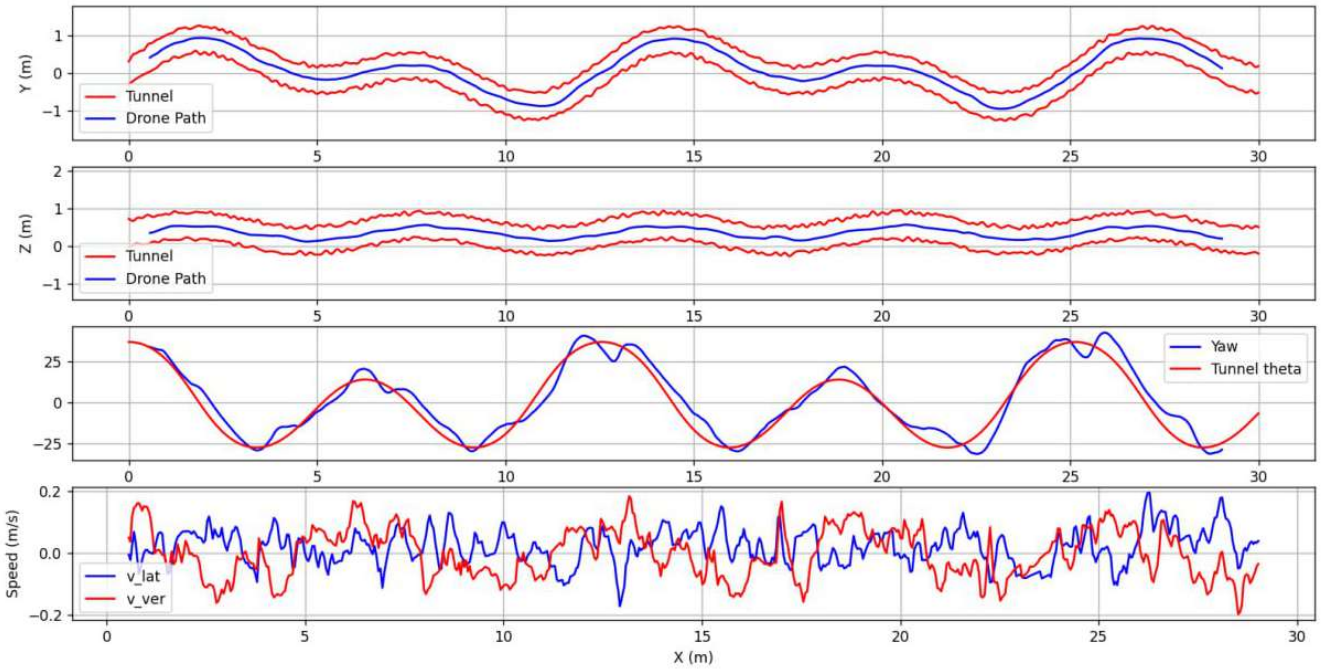


Figure 6. Trajectories flown inside the tunnel, centering mode.

6.3. Trajectory Analysis with Real Crazyflie Flights

The learned policy was flown through several arrangements of the tunnel modules, giving different curvature profiles along the course, and the vehicle completed the full course from entrance to exit in every case. The flight analyzed in this section is representative.

We disabled all integral terms in the proportional-integral-derivative (PID) loop to avoid slow error accumulation effects. Furthermore, the Crazyflie firmware was modified by removing the built-in low-pass filter on the gyroscope and increasing the bandwidth of its hardware low-pass filter, which substantially reduced control delay when executing the RL policy. In addition, the velocity filters in the software low-pass filtering of the XY PID loop, normally configured with a cutoff at 20 Hz, were disabled. These modifications reduced latency and improved responsiveness, enabling the controller to react more quickly to state deviations, albeit at the cost of slightly noisier velocity measurements. Although the RL policy outputs desired accelerations, these are integrated over short time steps to generate velocity and yaw-rate setpoints. The control signal flow is

$$\pi_{\mathrm{RL}}(o_t) \rightarrow [a_\ell, a_v, a_\psi] \rightarrow [v_\ell^{\mathrm{des}}, v_v^{\mathrm{des}}, \dot{\psi}^{\mathrm{des}}] \rightarrow \text{firmware PID.} \tag{12}$$

Thus, the RL policy produces desired accelerations, while the Crazyflie low-level controller receives desired velocities and yaw rate as its inputs.

The range difference signals in Figure 7 indicate that the quadrotor remained approximately centered throughout the tunnel, with the lateral range difference (left minus right) typically within ±0.15 m and the vertical range difference (up minus down) within ±0.1 m. The lateral range difference briefly reached approximately +0.21 m. These range differences are sensor-based indicators of centering, rather than direct measurements of displacement from the geometric centerline. Figure 7 plots the vertical difference as up minus down, the opposite sign of e_v in Section 5.2; only the magnitude $|e_v|$ enters the reward. Overall, the controller maintained stable flight with only minor deviations from the tunnel centerline. The lateral (v_y) and vertical (v_z) velocity traces in Figure 8 show that both components remain within the limits.

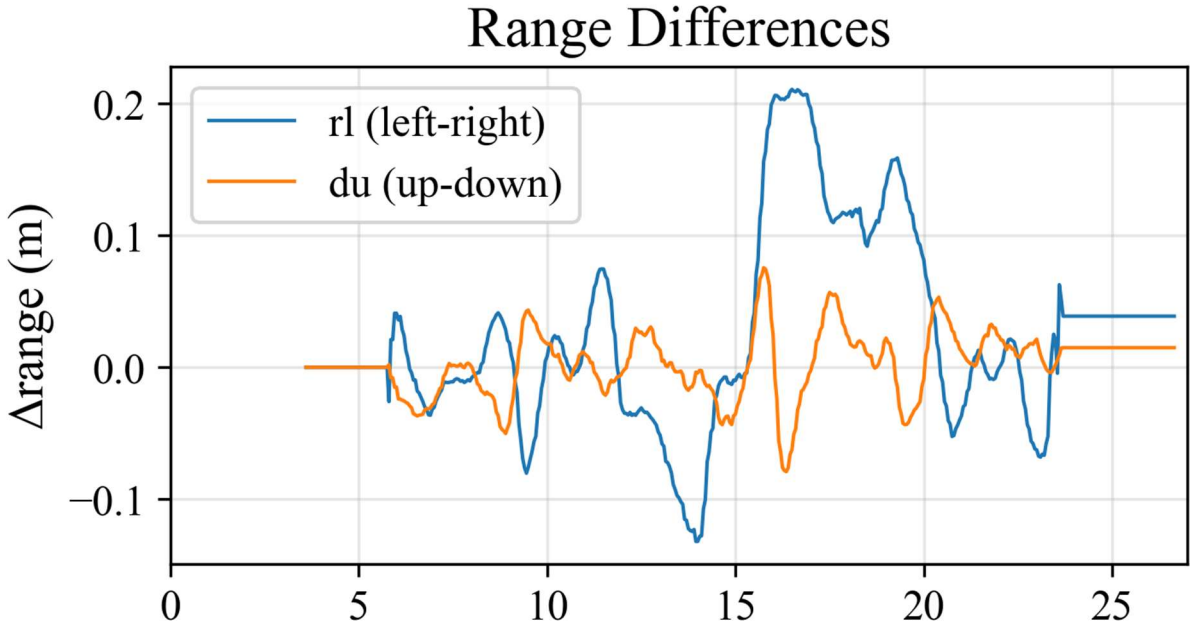


Figure 7. The range difference signals (left minus right and up minus down).

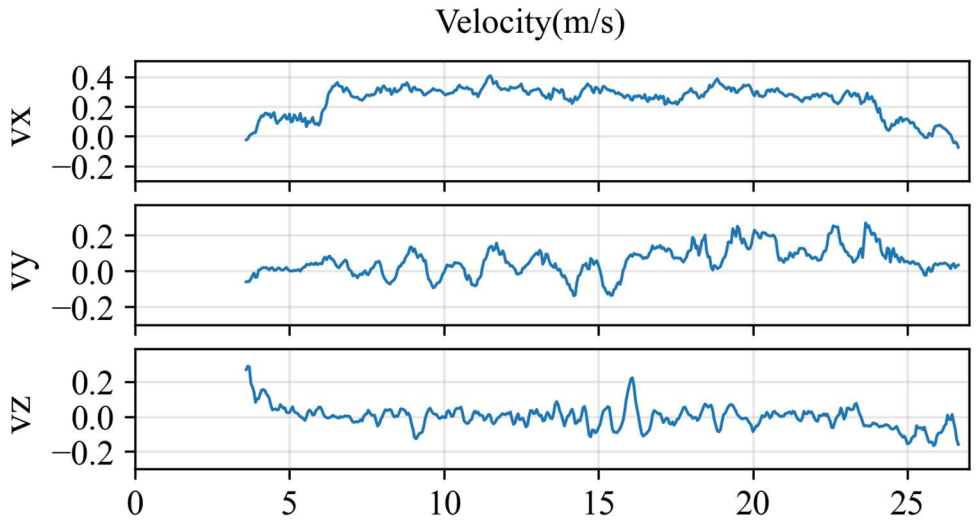


Figure 8. Lateral (v_y), longitudinal (v_x), and vertical (v_z) velocities during tunnel flight.

Figure 9 illustrates the experimental setup and tunnel mapping. The online mapping produced during flight closely matches the geometry of the physical tunnel. The middle plot shows the logged XY trajectory, confirming that the quadrotor successfully entered, followed, and exited the tunnel along the desired path. A video demonstration of the implementation is available at https://youtube.com/shorts/BNJXRSkYNYM?si=HZkXLY3GDjq6v-cB (accessed on 2 November 2026).

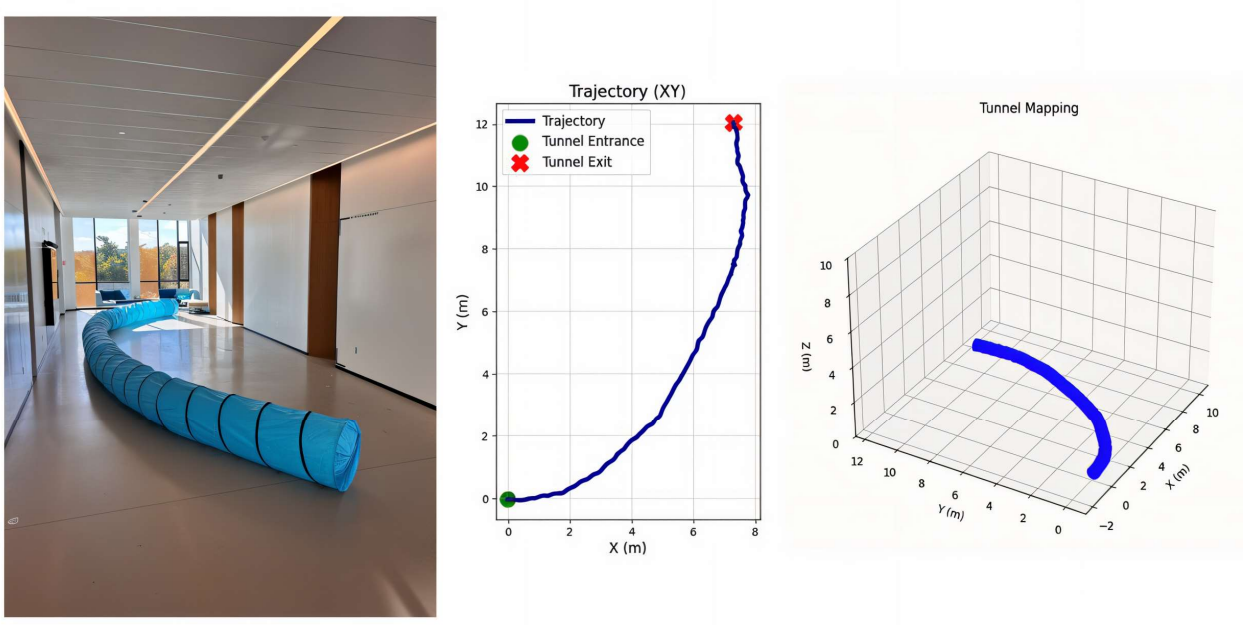


Figure 9. Experimental setup and results at Carleton University. (**Left)**: the physical tunnel environment used for flight tests. (**Middle**): logged XY trajectory showing the tunnel entrance and exit points. (**Right**): 3D tunnel mapping reconstructed from onboard range measurements.

The sim-to-real implementation used a simulation sampling interval matched to the range-sensor update interval, integration of policy acceleration outputs into velocity and yaw-rate setpoints, and the low-level controller and filtering modifications described above. The successful flights demonstrate deployment feasibility under the tested tunnel conditions. However, sensor noise, aerodynamic disturbances, model mismatch, and latency were not independently varied or quantified. Although different tunnel-module

arrangements were tested, these trials do not establish generalization to arbitrary unseen geometries. Systematic characterization of these effects remains a limitation of the present evaluation.

6.4. Ablation of STL Shaping in Simulation

Table 2 compares the three STL shaping modes with TD3 without STL shaping in simulation only. Each variant used three training seeds and a budget comparable to the main runs. Policies were evaluated without exploration noise on the same 20 unseen tunnels. Completion means reaching $x = 29$ m of a 30 m tunnel. Episodes ended upon leaving the vertical tunnel bounds, losing both lateral walls, or reaching 700 steps. Clearance violations were averaged per episode, including failed episodes up to termination, and then across tunnels; the table reports mean ± standard deviation across training seeds.

Centering and alignment both achieved 100% completion, while alignment had the lowest mean clearance-violation fraction (0.1%). These simulation results do not imply continuous clearance satisfaction or establish hardware performance for all modes.

Table 2. Simulation only: performance on 20 unseen tunnels. Values are mean ± standard deviation across three training seeds. Clearance violations are steps with a sensor distance below 0.15 m.

Controller	Completion (%)	Steps Below Clearance Margin (%)
TD3, no STL shaping	33 ± 58	4.7 ± 3.6
TD3 + STL, safety	75 ± 39	2.2 ± 2.0
TD3 + STL, centering	100 ± 0	0.7 ± 0.3
TD3 + STL, alignment	100 ± 0	0.1 ± 0.0

7. Conclusions and Future Work

This paper presented an STL-guided reinforcement learning framework for nano-quadrotor tunnel navigation. The TD3 reward combines instantaneous navigation terms with a windowed log-sum-exp approximation of predicate robustness and an adaptive penalty. Three selectable predicates express wall-clearance, centering, or heading-alignment requirements. In a limited simulation comparison, centering and alignment shaping both achieved 100% completion across the evaluated tunnels and training seeds, while alignment had the lowest mean clearance-violation fraction. The comparison does not isolate the effects of the temporal window or adaptive penalty. The centering policy was deployed through the Crazyflie velocity-control interface and completed a physical tunnel course over 17 m long and approximately 0.6 m wide. In the representative flight, lateral and vertical range differences were typically within approximately ±0.15 m and ±0.1 m, respectively. These are range-difference measurements rather than direct centerline-position errors. The approach provides specification-guided reward shaping without a formal satisfaction guarantee. The hardware results demonstrate deployment feasibility; systematic comparisons with alternative controllers and broader robustness studies remain future work.

Future research will investigate an end-to-end dynamics model that includes motor and low-level controller dynamics to better represent actuation and latency during training. Additional priorities include comparisons with conventional and learning-based controllers, larger multi-seed ablations with longer training budgets, sensitivity analysis of the temporal window and temperature, and controlled tests of noise, disturbances, latency, model mismatch, and unseen tunnel geometries on hardware. Further physical trials are needed to characterize reliability beyond the demonstrated conditions.

Statement of the Use of Generative AI and AI-Assisted Technologies in the Writing Process

During the preparation of this manuscript, the authors used Claude (Anthropic PBC, San Francisco, CA, USA) in order to assist with the writing of the manuscript. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

Acknowledgments

The authors thank the Ericsson 5G Autonomous Drone Lab at Carleton University for access to the physical tunnel facility used in the flight experiments.

Author Contributions

Conceptualization, M.H.H.; Methodology, M.H.H. and M.F.K.; Software, M.H.H.; Validation, M.H.H.; Formal Analysis, M.H.H.; Investigation, M.H.H.; Resources, H.S.; Data Curation, M.H.H.; Writing—Original Draft Preparation, M.H.H.; Writing—Review & Editing, H.S. and M.F.K.; Visualization, M.H.H.; Supervision, H.S.; Project Administration, H.S.; Funding Acquisition, H.S. All authors have read and agreed to the published version of the manuscript.

Ethics Statement

Not applicable.

Informed Consent Statement

Not applicable.

Data Availability Statement

The flight logs supporting the findings of this study are available from the corresponding author upon reasonable request.

Funding

This research was funded by Ericsson and the Natural Sciences and Engineering Research Council of Canada (NSERC).

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. U.S. Bureau of Labor Statistics. Fatal Occupational Injuries Involving Confined Spaces, 2011–2018. Available online: https://www.bls.gov/iif/factsheets/fatal-occupational-injuries-confined-spaces-2011-19.htm (accessed on 16 September 2025).
2. SureHire. Confined Space Awareness. Available online: https://surehire.com/confined-space-awareness/ (accessed on 25 October 2026).
3. Fishwick T. The hazards of confined space operations. *Loss Prev. Bull.* **2015**, *244*, 4–8. Available online: https://www.icheme.org/media/2149/lpb244_p04.pdf (accessed on 19 November 2026).
4. ScoutDI. The Next-Generation Confined-Space Inspection Drone System. Available online: https://www.scoutdi.com/ (accessed on 16 September 2025).
5. Flyability. Confined Space Inspection Drones: A Safer, Faster, and More Cost-Effective Solution. Available online: https://www.flyability.com/ (accessed on 16 September 2025).
6. Elmokadem T, Savkin AV. A method for autonomous collision-free navigation of a quadrotor UAV in unknown tunnel-like environments. *Robotica* **2022**, *40*, 835–861. DOI:10.1017/S0263574721000849
7. Wang L, Ning Y, Chen H, Liu P, Xu Y, Xu H, et al. Autonomous flights inside narrow tunnels. *IEEE Trans. Robot.* **2025**, *41*, 2230–2250. DOI:10.1109/TRO.2025.3548525
8. Bauersfeld L, Scaramuzza D. Low-latency event-based velocimetry for quadrotor control in a narrow pipe. *IEEE Trans. Robot.* **2026**, *42*, 1–19. DOI:10.1109/TRO.2026.3654764

9. Gallego G, Delbrück T, Orchard G, Bartolozzi C, Taba B, Censi A, et al. Event-based vision: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *44*, 154–180. DOI:10.1109/TPAMI.2020.3008413
10. Zhang R, Hao G, Zhang K, Li Z. Unmanned aerial vehicle navigation in underground structure inspection: A review. *Geol. J.* **2023**, *58*, 2454–2472. DOI:10.1002/gj.4763
11. Tranzatto M, Mascarich F, Bernreiter L, Godinho C, Camurri M, Khattak S, et al. CERBERUS: Autonomous legged and aerial robotic exploration in the tunnel and urban circuits of the DARPA Subterranean Challenge. *arXiv* **2022**, arXiv:2201.07067. DOI:10.48550/arXiv.2201.07067
12. Kominiak D, Mansouri SS, Kanellakis C, Nikolakopoulos G. MAV development towards navigation in unknown and dark mining tunnels. *arXiv* **2020**, arXiv:2005.14433. DOI:10.48550/arXiv.2005.14433
13. Hwangbo J, Sa I, Siegwart R, Hutter M. Control of a quadrotor with reinforcement learning. *IEEE Robot. Autom. Lett.* **2017**, *2*, 2096–2103. DOI:10.1109/LRA.2017.2720851
14. Hezaveh MH, Foroozan Koodiyani M, Schwartz H, Lambadaris I. Trajectory tracking of autonomous vehicles using data-driven modeling and neural predictive controller (NPC). *IEEE Trans. Control Syst. Technol.* **2026**, *34*, 1625–1632. DOI:10.1109/TCST.2026.3660313
15. Eschmann J. Reward function design in reinforcement learning. In *Reinforcement Learning Algorithms: Analysis and Applications*; Belousov B, Abdulsamad H, Klink P, Parisi S, Peters J, Eds.; Springer International Publishing: Cham, Switzerland, 2021; pp. 25–33. DOI:10.1007/978-3-030-41188-6_3
16. Adams S, Cody T, Beling PA. A survey of inverse reinforcement learning. *Artif. Intell. Rev.* **2022**, *55*, 4307–4346. DOI:10.1007/s10462-021-10108-x
17. Retzlaff CO, Das S, Wayllace C, Mousavi P, Afshari M, Yang T, et al. Human-in-the-loop reinforcement learning: A survey and position on requirements, challenges, and opportunities. *J. Artif. Intell. Res.* **2024**, *79*, 359–415. DOI:10.1613/jair.1.15348
18. Cai M, Vasile CI. Safety-critical learning of robot control with temporal logic specifications. *IEEE Trans. Autom. Control* **2025**, *70*, 5553–5560. DOI:10.1109/TAC.2025.3550850
19. Sun D, Chen J, Mitra S, Fan C. Multi-agent motion planning from signal temporal logic specifications. *IEEE Robot. Autom. Lett.* **2022**, *7*, 3451–3458. DOI:10.1109/LRA.2022.3146951
20. Hamilton N, Robinette PK, Johnson TT. Training agents to satisfy timed and untimed signal temporal logic specifications with reinforcement learning. In Proceedings of the 20th International Conference on Software Engineering and Formal Methods (SEFM), Berlin, Germany, 26–30 September 2022; pp. 190–206.
21. Li X, Vasile CI, Belta C. Reinforcement learning with temporal logic rewards. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 3834–3839. DOI:10.1109/IROS.2017.8206234
22. Balakrishnan A, Deshmukh JV. Structured reward shaping using signal temporal logic specifications. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 3–8 November 2019; pp. 3481–3486. DOI:10.1109/IROS40897.2019.8968254
23. Mehdipour N, Vasile CI, Belta C. Specifying user preferences using weighted signal temporal logic. *IEEE Control Syst. Lett.* **2021**, *5*, 2006–2011. DOI:10.1109/LCSYS.2020.3047362
24. Leung K, Aréchiga N, Pavone M. Back-propagation through signal temporal logic specifications: Infusing logical structure into gradient-based methods. In *Algorithmic Foundations of Robotics XIV*; Springer International Publishing: Cham, Switzerland, 2021; pp. 432–449.
25. Zhang N, Liu W, Belta C. Distributed control using reinforcement learning with temporal-logic-based reward shaping. In Proceedings of the Conference on Learning for Dynamics and Control (L4DC), Stanford, CA, USA, 23–24 June 2022; pp. 751–762.
26. Jiang Y, Bharadwaj S, Wu B, Shah R, Topcu U, Stone P. Temporal-logic-based reward shaping for continuing reinforcement learning tasks. *Proc. AAAI Conf. Artif. Intell.* **2021**, *35*, 7995–8003. DOI:10.1609/aaai.v35i9.16975
27. Saxena N, Gorantla S, Jagtap P. Funnel-based reward shaping for signal temporal logic tasks in reinforcement learning. *IEEE Robot. Autom. Lett.* **2024**, *9*, 1373–1379. DOI:10.1109/LRA.2023.3341775
28. García J, Fernández F. A comprehensive survey on safe reinforcement learning. *J. Mach. Learn. Res.* **2015**, *16*, 1437–1480. Available online: https://jmlr.org/papers/v16/garcia15a.html (accessed on 19 November 2026).
29. Achiam J, Held D, Tamar A, Abbeel P. Constrained policy optimization. In Proceedings of the 34th International Conference on Machine Learning (ICML), Sydney, Australia, 6–11 August 2017; pp. 22–31.
30. Alshiekh M, Bloem R, Ehlers R, Könighofer B, Niekum S, Topcu U. Safe reinforcement learning via shielding. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; pp. 2669–2678.

31. Gilpin Y, Kurtz V, Lin H. A smooth robustness measure of signal temporal logic for symbolic control. *IEEE Control Syst. Lett.* **2021**, *5*, 241–246. DOI:10.1109/LCSYS.2020.3001875
32. Li W, Wen C, Shi M, Qin K, Shi L. On asynchronous bipartite consensus over quasi-structurally balanced signed networks. *IEEE Trans. Control Netw. Syst.* **2026**, 1–12, *early access*. DOI:10.1109/TCNS.2026.3722797
33. Li W, Yan S, Shi L, Yue J, Shi M, Lin B, et al. Multiagent consensus tracking control over asynchronous cooperation–competition networks. *IEEE Trans. Cybern.* **2025**, *55*, 4347–4360. DOI:10.1109/TCYB.2025.3583387
34. Hezaveh MH, Ramezanlou MT, Schwartz H, Lambadaris I. Enhancing UAV altitude control: PID tuning with reinforcement learning and genetic algorithm. In Proceedings of the 2025 IEEE International Systems Conference (SysCon), Montreal, QC, Canada, 21–24 April 2025; pp. 1–6. DOI:10.1109/SysCon64521.2025.11014809
35. Fujimoto S, van Hoof H, Meger D. Addressing function approximation error in actor-critic methods. In Proceedings of the International Conference on Machine Learning (ICML), Stockholm, Sweden, 10–15 July 2018; pp. 1587–1596.
36. Bitcraze AB. Crazyflie 2.1 Brushless. Available online: https://store.bitcraze.io/products/crazyflie-2-1-brushless (accessed on 16 September 2025).