An Intention Guided Hierarchical Framework for Trajectory-based Teleoperation of Mobile Robots

Xuning Yang[1], Jasmine Cheng[2], and Nathan Michael[1]

***Abstract*— In human-in-the-loop navigation, the operator's intention is to locally avoid obstacles while planning long-horizon paths in order to complete the navigation task. We propose a hierarchical teleoperation framework that captures these characteristics of intention, and generates trajectories that are locally safe and follow the operator's global plan. The hierarchical teleoperation framework consists of 1) a global path which encapsulates the intended direction of the operator, 2) local trajectories that circumvent obstacles near the vehicle's vicinity while following the global path, and 3) safety monitoring to avoid possible imminent collisions. By removing the operator from providing dynamic-level control inputs and instead having inputs *inform* trajectory generation, we show a significant reduction of the operator's engagement while maintaining smooth performance.**

We showcase hierarchical teleoperation in navigation tasks in a random forest environment and a high-clutter warehouse characterized by narrow gaps and dense obstacles. With our method, we maintain consistent high speed throughout the task with smooth jerk profiles, decreased time to completion, and significantly reduced operator engagement.

I. Introduction

Teleoperation in unstructured environments for tasks such as navigation or exploration requires operators to 1) maneuver the vehicle with safety and dynamic feasibility to avoid collisions and 2) plan global paths that achieve the objective. In critical situations where the vehicle is traveling fast through dense obstacles, fast reactivity and high-frequency engagement is required to mitigate collisions. The operator must balance generating reactive motions that evade obstacles, and long-term path planning in order for task completion. Therefore, completing the task under these circumstances requires increased mental acuity and high-level engagement from the operator in order to stay safe.

For navigation tasks in unstructured environments, the operator's intention is characterized according to the above requirements: to locally avoid obstacles while generating dynamically safe motions, and planning long-horizon paths for task completion. In this work, we present a trajectory-based teleoperation framework that captures the hierarchical nature of operator's intention while removing the operator from the responsibilities of safety, reactivity and dynamic feasibility. The proposed framework continuously generates local trajectories that follows a global path, which, in the teleoperation context, encapsulates the operator's intended direction over longer time horizons as illustrated in Fig. 1. The hierarchical design of the proposed framework naturally

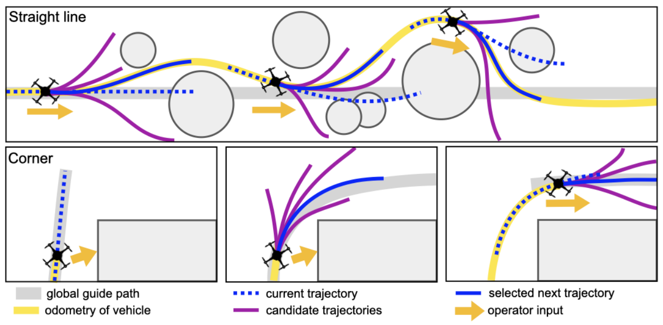


Fig. 1: Illustrations for the proposed hierarchical teleoperation framework in two example scenarios. Operator's intention, reflected in the global path, guides local trajectory generation. Top: the vehicle avoids obstacles following a linear motion. Bottom: The intention to round a corner is reflected in the global path, which triggers a local trajectory regeneration.

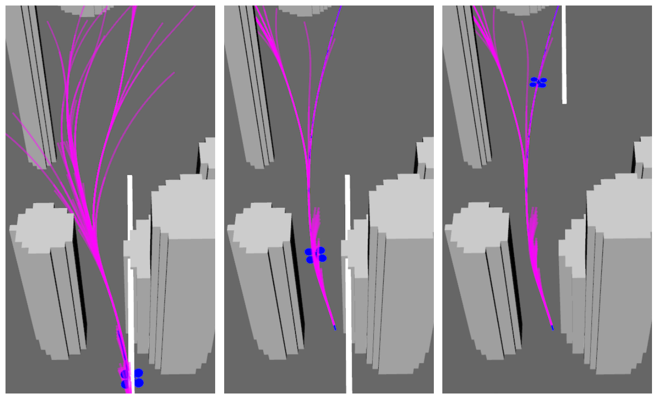


Fig. 2: A sequence of the hierarchical teleoperation framework in action in the random forest environment. In this snapshot, the global trajectory (in grey) represents a forward linear motion. The local trajectory generator generates candidate trajectories (in magenta). The vehicle (in blue) follows a chosen trajectory and successfully passes through a narrow gap and returns to follow the global path.

corresponds to the spectrum of intentions that would otherwise require increased focus and continuous engagement from the operator.

The hierarchical teleoperation framework consists of three components: 1) global path generation using a directional intention model informed by the operator's inputs; 2) a local trajectory generator that generates dynamically feasible snap-continuous trajectories that circumvent obstacles while following the global path; and lastly, 3) a safety monitoring system that continuously monitors imminent collisions. The inputs are processed such that linear motions such as yaw and stop are directly executed while other inputs (representing navigation) *inform* the trajectory generation process. This allows the operator to retain natural control of vehicle with while reducing necessary engagement from the operator in order to achieve the task while maintaining consistent high

[1]Xuning Yang and Nathan Michael are with the Robotics Institute, Carnegie Mellon University, Pittsburgh PA 15232, USA {xuning, nmichael}@cmu.edu

[2]Jasmine Cheng is with the School of Computer Science, Carnegie Mellon University, Pittsburgh PA 15232, USA jacheng@andrew.cmu.edu

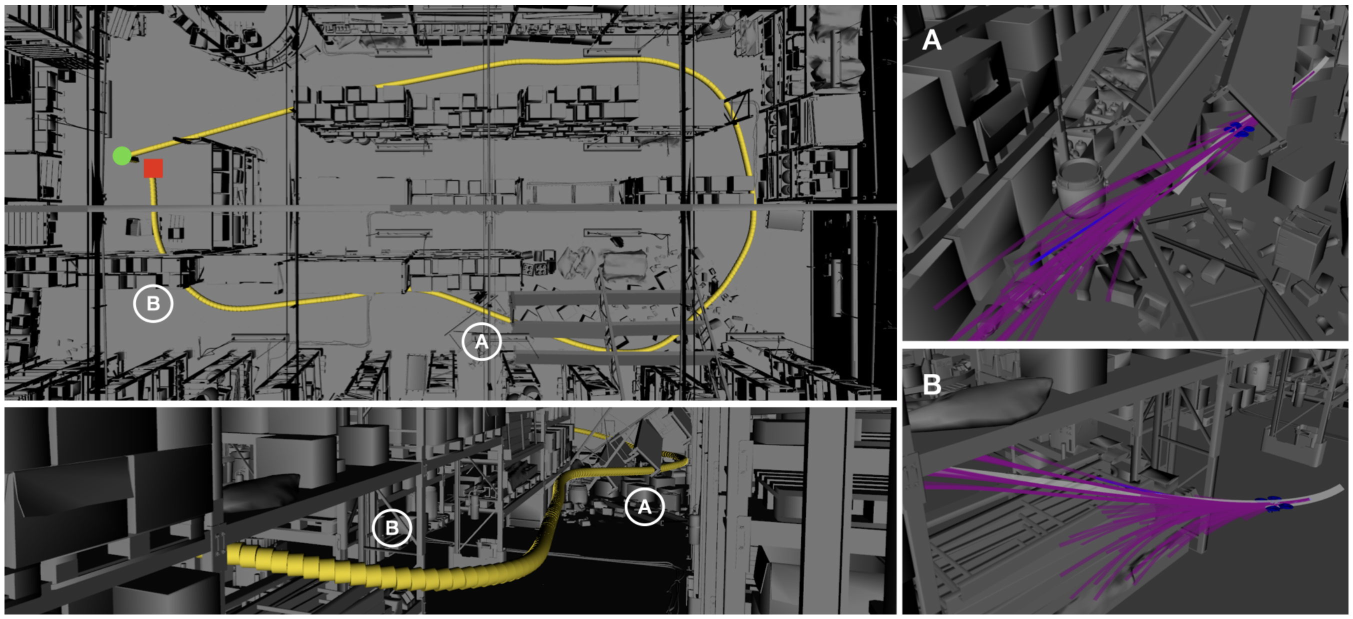


Fig. 3: Snapshot of the hierarchical teleoperation framework in action in the warehouse environment. Top left: An overhead view of the resulting odometry (in yellow) starting near the entryway of the warehouse and exploring clockwise. Bottom left: Highlighting a section of the trial where the vehicle encounters two difficult scenarios: (A) through a partially collapsed shelf and (B) return to the origin by going through two shelves with low clearance. In those two scenarios, the global guiding path (in grey) allows generation of candidate local trajectories (in magenta). The selected trajectory (in blue) successfully leads the vehicle through narrow gaps in the highlighted scenarios.

speeds, especially during critical navigation scenarios, as shown in Fig. 2-3.

We test the proposed teleoperation frameworks in two tasks: Navigating a quadrotor in various density random forest environments where the operator is asked to follow a straight line, and navigating in a cluttered warehouse characterized by narrow gaps and irregular free space formed by collapsed shelves. With the proposed method, the operators are able to complete the task with a significant reduction in engagement while maintaining consistent high speed and smoother jerk profiles.

II. Related Works

1) Trajectory-based teleoperation: In traditional multirotor UAV teleoperation, the operator typically issues low-level inputs to the vehicle according to the vehicle dynamics, requiring experience in teleoperation to maintain stability [1]. Previous works exploit the differentially flat properties of multirotors to allow direct teleoperation in the state space with reactive collision avoidance [2]. However, reactive methods still rely on the operator to navigate vehicle around obstacles. Recent trajectory-based teleoperation methods generate trajectories that locally circumvent obstacles, reducing operator engagement around immediate obstacles [3]. In these works, the global intention is not considered and thus they require continuous operator engagement to complete the task.

2) Hierarchical planning: Global-local planning architectures have been deployed in autonomous systems for global navigation and local collision avoidance for both UAVs [4, 5] and ground robots [6–9]. These methods usually depend on a pre-specified goal location and select trajectories that most closely follow the global path. In navigation-based teleoperation tasks where an operator is directly controlling the vehicle, the goal may not be specified a priori. Therefore, the global path in the teleoperation context can be interpreted as a representation of the operator's intended path, and local trajectories can be interpreted as a set of sequential actions to follow the intended path.

III. Preliminaries

Multirotors are differentially flat systems [10]. Exact control inputs can be computed such that the vehicle follows a specified trajectory in the flat outputs x, y, z, and yaw θ. Define the state to be $\mathbf{x}=[x, y, z, \theta]$. A trajectory is a time-parameterized function $\mathit{\Gamma}(t)$, defined over a time interval $t \in [0, T]$ that maps a given time t to a state $\mathbf{x}_t$.

For teleoperation, the operator provides inputs with respect to a level frame $\mathcal{C}$, which is defined as a world-z-aligned frame rigidly attached to the vehicle's origin. The inputs are given in the form of $\mathbf{a}=[v_x, \omega, v_z]^\top$ via a joystick, where v_x is the linear velocity along the level frame $x^{\mathcal{C}}$-axis, ω is the angular velocity about the level frame $z^{\mathcal{C}}$-axis, and v_z is the velocity along the level frame $z^{\mathcal{C}}$-axis.

A motion primitive $\gamma(t)$ is a parameterized trajectory function which generates a unique sequence of states in the world frame given an initial state $\mathbf{x}_0 \in \mathcal{X}$ and an input $\mathbf{a} \in \mathcal{A}$ according to specific dynamics:

$$\gamma_{\mathbf{a},T} : [0, T] \rightarrow \mathcal{X} \qquad \mathbf{a} \in \mathcal{A},\ T \in [0, \infty) \tag{1}$$

A choice of motion primitive is to use the unicycle model to evolve the state [11]:

$$\dot{\mathbf{f}}_{\mathbf{a}}(t) = [v_x \cos(\omega t) \quad v_x \sin(\omega t) \quad v_z \quad \omega]^\top, t \in [0, T] \tag{2}$$

However, this model does not provide higher order references. Since attitude references can be directly computed from higher order derivatives of the flat outputs, we compute an eighth order polynomial with continuity between subsequent primitives up to snap. Further, the references generated using Eq. (2) with input $\mathbf{a}$ would need to be transformed from the level frame $\mathcal{C}$ to the appropriate frame of control, e.g., the world frame $\mathcal{W}$. Therefore, the motion primitive γ is generated as follows:

Given an initial state $\mathbf{x}_0$ and its higher order derivatives $\dot{\mathbf{x}}_0, \ddot{\mathbf{x}}_0, \dddot{\mathbf{x}}_0, \mathbf{x}_0^{(4)}$, the polynomial is generated with velocity endpoints constrained according to the unicycle model Eq. (2):

$$\gamma_{\mathbf{a},T}(t) = \sum_{i=0}^{8} \mathbf{c}_i t^i \tag{3}$$

$$\begin{aligned} \text{s.t. } \gamma^{(j)}(0) &= \mathbf{x}_0^{(j)}(t) \quad \text{for } j = 0,1,2,3,4 \\ \dot{\gamma}(T) &= R_{t\mathcal{C}}^{\mathcal{W}} \dot{\mathbf{f}}_{\mathbf{a}}(T) \\ \gamma^{(j)}(T) &= 0 \quad \text{for } j = 2,3,4 \end{aligned}$$

where $\{\cdot\}^{(j)}$ specifies the j^{th} time derivative and $R_{t\mathcal{C}}^{\mathcal{W}}$ is the transformation from $\mathcal{C}$ to $\mathcal{W}$ at time t. This is a fully constrained system of linear equations, and thus the coefficients $\mathbf{c}_i$ can be computed in closed form.

A sequence of N motion primitives is given by:

$$\xi = (\gamma_1, \dots, \gamma_N) = (\gamma_i)_{i=1}^{N} \tag{4}$$

The total duration of the motion primitive sequence is given by $\mathbf{T} = \sum_{i=1}^{N} T_i$, where T_i is the duration of the ith primitive. The trajectory function for a sequence of motion primitives is defined as:

$$\mathit{\Gamma}(t) = \gamma_i(t - \tau_{i-1}) \qquad t \in [\tau_{i-1}, \tau_i) \tag{5}$$

where τ_i is the cumulative duration up to primitive i.

IV. Method

The hierarchical teleoperation architecture is shown in Fig. 5. The three components, global path generation, local trajectory generation and safety monitoring, run in parallel with increasing frequencies, and trigger regeneration according to the logic flow in Fig. 6.

The current architecture assumes the availability of a map for the purposes of collision checking. For this paper, we utilize a KD-Tree local map representation [2]. The existence of various map representations can be readily incorporated into this framework as discussed in Sect. VI.

A. User Input Processing

Inputs from the operator are received as a continuous stream of joystick values around 200Hz. A *novel input*

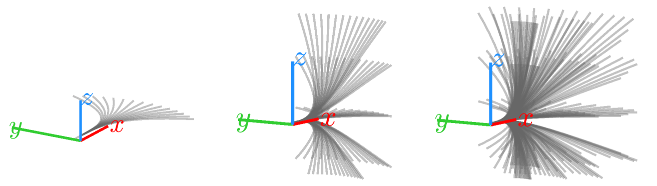


Fig. 4: Motion Primitive Library constructed with forward arc primitives. The variations in angular velocity, z velocity, and linear velocity are added incrementally for maximum clarity.

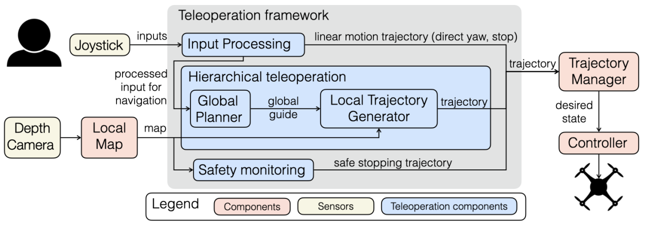
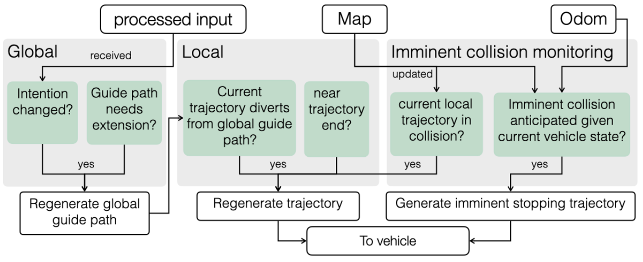


Fig. 5: System diagram of the proposed framework. The teleoperation framework takes in a continuous stream of inputs from an user-operated joystick, and generates a trajectory to be sent to the controller. Inputs are processed and sent to the planning pipeline.



processed input
Map
Odom
Global
received
Local
Imminent collision monitoring
updated
Intention changed?
Guide path needs extension?
Current trajectory diverts from global guide path?
near trajectory end?
current local trajectory in collision?
Imminent collision anticipated given current vehicle state?
yes
yes
yes
yes
Regenerate global guide path
Regenerate trajectory
Generate imminent stopping trajectory
To vehicle

Fig. 6: Logic flow diagram for the proposed hierarchical teleoperation framework during navigation tasks.

is defined as an input that remains constant for at least 100ms. Only novel inputs are retained and the rest discarded; hence the term "novel inputs" is used interchangeably with "operator inputs" or "inputs". In order to allow operators to remain in control, inputs are categorized as pure yaw, zero input, or navigation. For pure yaw and zero inputs, a direct linear trajectory is sent to the controller bypassing the rest of the system, as they do not correspond to the task of navigation. The navigation inputs are passed onto the hierarchical planning framework.

B. Global Planning

To reflect the long horizon motion that the operator intends to achieve, we utilize a simplified intention model based on previous navigational inputs as follows: First, the navigational inputs are filtered to produce a likely global input $\mathbf{a}_{Gt}$ at time t given current input $\mathbf{a}_t$:

$$\mathbf{a}_{Gt} = \lambda \mathbf{a}_{Gt-1} + (1-\lambda)\mathbf{a}_t \qquad 0 < \lambda < 1 \tag{6}$$

Then, a global trajectory $\mathit{\Gamma}_G$ is generated according to Eq. (2) with a duration of $\mathbf{T}$. As the global path is only used for guidance, any higher order dynamics of the trajectory can be safely ignored. For our experiments, we choose a horizon of $\mathbf{T}$=10s and λ=0.8.

C. Local Trajectory Generation

1) Trajectory generation: The operator's input is first parameterized as a single-step motion primitive. If the motion primitive is in collision, we utilize a trajectory generation method to circumvent the immediate obstacle. We generate a candidate set of trajectories by constructing a motion primitive tree using Biased Incremental Action Sampling (BIAS) [3] given a local map representation. BIAS iteratively builds a tree of sequential motion primitives that minimizes an objective. The objective cost function for teleoperation is a weighted combination of the local direction and behavior

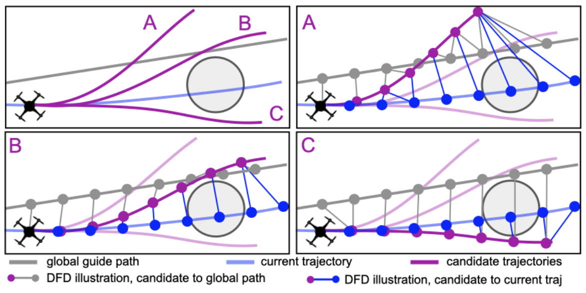

A
B
C
A
B
C
global guide path
current trajectory
candidate trajectories
DFD illustration, candidate to global path
DFD illustration, candidate to current traj

Fig. 7: Illustration of new trajectory selection based on weighted discrete Fréchet distance (DFD). Each candidate trajectory is sampled and DFD is computed between the candidate and global path and local trajectory. In this example, candidate B scores the lowest DFD and is sent to the controller.

cost functions. The local direction cost function evaluates the dot product between a candidate trajectory and the ideal primitive:

$$C_{\text{input}_{\mathbf{a}}}(\xi) = \|1 - \mathbf{p} \cdot \mathbf{p}^*\| \tag{7}$$

$$\mathbf{p} = \frac{\Gamma(\mathbf{T}) - \Gamma(0)}{\|\Gamma(\mathbf{T}) - \Gamma(0)\|} \qquad \mathbf{p}^* = \frac{\gamma_{\mathbf{a}}(\mathbf{T}) - \gamma_{\mathbf{a}}(0)}{\|\gamma_{\mathbf{a}}(\mathbf{T}) - \gamma_{\mathbf{a}}(0)\|} \tag{8}$$

where $\Gamma(\tau)$ is the multi-step motion primitive trajectory evaluated at time τ, and $\gamma_{\mathbf{a}}(\tau)$ is the single-step motion primitive parameterized by the operator's given input $\mathbf{a}$ evaluated at time τ. Therefore, γ is exactly the motion primitive in collision as generated in the previous section, so as to maximize adherance to the operator's intended input. The behavior cost functions and the BIAS algorithm are discussed in [3].

2) Trajectory selection: Given a set of candidate trajectories, we select the trajectory to be followed by the vehicle. We balance two objectives: maximize smoothness in transitioning from the current vehicle trajectory, and minimize its distance to the global guiding path. To do so, we introduce a selection cost function that evaluates each trajectory by its closeness to both the current local trajectory and the global path by evaluating the *discrete Fréchet distance* δ_{dF} [12, 13]. We provide a brief definition below.

Consider a discrete sampling of two continous functions f and g that forms two polygonal curves $\mathcal{P} = \{f_1, f_2, ..., f_n\}$ and $\mathcal{Q} = \{g_1, g_2, ..., g_m\}$ which are sequences of n and m discrete points, respectively. An *order-preserving, complete correspondence* between $\mathcal{P}$ and $\mathcal{Q}$ is a pair (α, β) of *discrete monotone reparameterizations* [1] of α from $\{1, ..., k\}$ to $\{1, ..., n\}$ and of β from $\{1, ..., k\}$ to $\{1, ..., m\}$. The discrete Fréchet distance of $\mathcal{P}$ and $\mathcal{Q}$ is given by:

$$\delta_{dF}(f, g) := \min_{(\alpha,\beta)} \max_{i \in [1,k]} d(f_{\alpha(i)}, g_{\beta(i)}) \tag{9}$$

where (α, β) ranges over all order-preserving complete correspondences between $\mathcal{P}$ and $\mathcal{Q}$. Therefore, the Fréchet distance for a pair of time parameterized trajectories Γ, Φ is given by

$$\delta_{dF}(\Gamma, \Phi) := \min_{(\alpha,\beta)} \max_{i \in [1,k]} d(\Gamma_{\alpha(i)}, \Phi_{\beta(i)}) \tag{10}$$

where $\Gamma_i = \Gamma(i \cdot \Delta t)$ for a fixed Δt sampling of Γ.

The trajectory selection is then as follows: Given the current local trajectory Γ_L and the guiding global trajectory Γ_G, and a candidate set of trajectories $\{\Gamma_i\}_{i=1}^N$,

$$\Gamma^* = \min_{\Gamma \in \{\Gamma_i\}_{i=1}^N} w_L \delta_{dF}(\Gamma, \Gamma_L) + w_G \delta_{dF}(\Gamma, \Gamma_G) \tag{11}$$

We select w_L=w_G=1. An illustration of the trajectory selection process is shown in Fig. 7.

D. Safety monitoring

1) Trajectory Safety: The size of the local map is dependent on the accuracy range of the sensors in exploring unknown environments. As such, this range is sometimes limited. Therefore, the unknown space is treated as free space during trajectory generation with the trajectories possibly extending beyond the size of the local map. As the vehicle moves, the local map is being updated on a rolling basis with incoming new sensor scans. Therefore, the current trajectory is continually checked against the updated map for collisions.

2) Imminent collision monitoring: The imminent collision checking system continuously monitors vehicle safety given its current velocity and environment. The motion primitives for teleoperation, by design, end in non-zero velocity; therefore, if map updates render the newly observed area unsafe and a new trajectory is unable to be generated in time according to the navigation objective, a safe, dynamically feasible stop trajectory is generated that brings the vehicle from in-motion to at-rest.

To determine whether an imminent collision will occur, we first find the set of possible collision points nearby by evaluating the normalized vector projection between a set of closest obstacle locations given by the map and the vehicle velocity:

$$\text{proj}(\mathbf{x}, \mathbf{p}) = \left\langle \frac{\dot{\mathbf{x}}}{\|\dot{\mathbf{x}}\|}, \frac{\mathbf{p} - \mathbf{x}}{\|\mathbf{p} - \mathbf{x}\|} \right\rangle = \left\langle \frac{\dot{\mathbf{x}}}{\|\dot{\mathbf{x}}\|}, \frac{\mathbf{r}}{\|\mathbf{r}\|} \right\rangle$$

Where $\mathbf{x}$, $\dot{\mathbf{x}}$ is the vehicle position and velocity respectively, and the vector to an obstacle point $\mathbf{p}$ is $\mathbf{r} = \mathbf{p} - \mathbf{x}$. The points away from the direction of motion, e.g., $\text{proj}(\mathbf{x}, \mathbf{p})$<0 are discarded. Then, a combined stop criteria is computed for the remaining points $\{\mathbf{p}\}$ using distance to the obstacle, speed, and the angle offset:

$$C_{\text{stop}}(\mathbf{x}, \mathbf{p}) = w_1 \|\mathbf{r}\| - w_2 \|\dot{\mathbf{x}}\| + w_3 \arccos(\text{proj}(\mathbf{x}, \mathbf{p})) \qquad \text{if } \text{proj}(\mathbf{x}, \mathbf{p}) >= 0$$

An imminent stop trajectory is issued if for any obstacle point $\mathbf{p}$, $C_{\text{stop}}(\mathbf{x}, \mathbf{p})$<0. In our experiments, w_1, w_2 and w_3 are chosen to be 0.5, 0.3, 1.2 respectively. To generate an imminent stop trajectory, an initial group of escape points $\{\mathbf{e}\}$ is generated using a uniform grid, then sampled along the direction of motion using stratified sampling for points near the vehicle's original heading via the following cost:

$$C_{\text{escape}}(\mathbf{x}, \mathbf{e}) = w_1 q + w_2 d$$

[1]A *discrete monotone reparameterization* α from $\{1, ..., k\}$ to $\{1, ..., l\}$ is defined as a non-decreasing function $\alpha : \{1, ..., k\} \rightarrow \{1, ..., l\}$ for integers $k \geq l \geq 1$, with $\alpha(1) = 1$, $\alpha(k) = l$, and $\alpha(i + 1) \leq \alpha(i) + 1 \; \forall \; i = 1, ..., k - 1$.

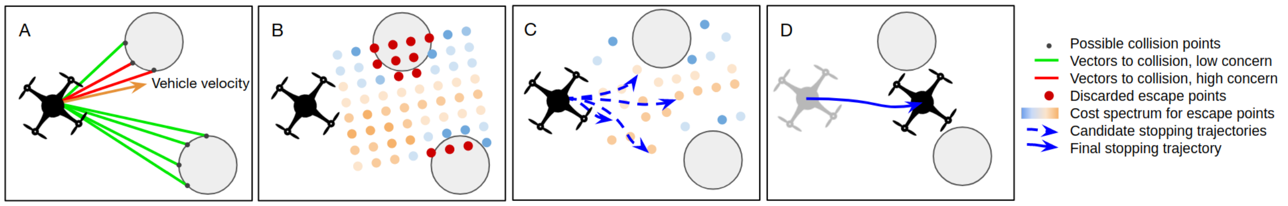

A
Vehicle velocity
B
C
D
Possible collision points
Vectors to collision, low concern
Vectors to collision, high concern
Discarded escape points
Cost spectrum for escape points
Candidate stopping trajectories
Final stopping trajectory

Fig. 8: Illustration of imminent collision monitoring and safe stop. (A) Initial assessment: possible collision points are obtained from nearby obstacles by querying the map and assessed using C_{stop}. If the stop criterion is met, stop trajectory generation begins. (B) Initial grid of escape points with costs computed and colliding points discarded (C) A stratified sampling method is used to downsample the possible escape points, with candidate trajectories generated. Dynamically infeasible candidate trajectories are discarded. (D) Lowest cost, dynamically feasible trajectory is selected and sent to the controller.

Where $q = ||(\mathbf{x} - \mathbf{e} - ((\mathbf{x} - \mathbf{e}) \cdot \hat{\dot{\mathbf{x}}})\hat{\dot{\mathbf{x}}}||$ is the shortest distance from the escape point $\mathbf{e}$ to the line $l(s) = \hat{\dot{\mathbf{x}}}s + \mathbf{x}$ along the velocity vector, $\hat{\dot{\mathbf{x}}}$ is the normalized velocity vector, and d is the distance from the the escape point to its nearest obstacle. The stratified method guarantees a set of points with both high and low costs such that in the event low cost trajectories are not dynamically feasible, we are able to sacrifice cost for dynamic feasibility. Stopping trajectories are then generated starting with the lowest C_{escape} and checked for safety and dynamic feasibility via an acceleration bound along the trajectory, i.e., $\ddot{\mathbf{x}} < \bar{\alpha}$. For our experiments, we select $\bar{\alpha} = 10\text{m/s}^2$. An illustration of this process is shown in Fig. 8.

V. Experiments and Results

We evaluate our method in a simulated random forest environment and in a dense realistic warehouse environment, where the operator is asked to perform a navigation task by following a global path described to the operator. The operator is given a third-person follower view of the vehicle. Each task is repeated with five trials in each environment. The simulated vehicle is a high fidelity quadrotor model with an effective diameter of 30.6cm. The simulated experiments are performed on a CPU (Intel Core 2.20GHz i7-8750H CPU), with 16GB of RAM.

The proposed method is compared to direct motion primitive teleoperation (MP) [2], which is a one-step motion primitive, as well as multi-step trajectory generation via motion primitive trees (MPT) [3]. All three methods are equipped with the imminent collision monitoring system. An example trial of the collision monitoring and prevention in the warehouse scenario is shown in Fig. 9. The stopping trajectory quicky stops vehicle before possible collisions and allows the vehicle to escape.

A. Evaluation Criteria

The following criteria are used: 1) time to completion, 2) smoothness by evaluation of the jerk integral, and 3) operator engagement by evaluation of the number of novel inputs over the fixed task. Further, for the warehouse scenario, we additionally evaluate 4) the average velocity. The key metric is operator engagement. If the number of new inputs is high, this indicates that operators feel the need to control the vehicle in order to correct its course. Alternatively, if the number of new inputs is low, it implies that the vehicle is following course on its intended trajectory and does not require correction to its motion.

B. Random Forest

The random forest environment contains 120 various-height pillars in a 60m×30m×10m volume. The operator is asked to navigate through the random forest environment following a straight line path that passes through many pillars. The vehicle begins on one side of the random forest, and the task is marked complete as soon as the vehicle reaches the other side of the random forest.

The results are tabulated in Table I. Overall, the tasks are complete within reasonable time. We observe that the proposed method maintains a jerk integral of 25m^2/s^3 for all three density environments, whereas previous methods show an increase in jerkiness depending on the density. We also observe a significant decrease in the number of joystick inputs to complete the trial: the proposed method shows a 89%, 82%, 92% reduction from the baseline motion primitive method for the sparse, medium and dense environments respectively. We also observe that the number of inputs required to complete the random forest environment remains constant, regardless of the density of the environment as shown in Fig. 11. For the single-step method, the number of inputs increases significantly in the dense environment. However, for trajectory-based methods, this is reduced as the trajectories allows the vehicle to maneuver through high clutter areas without frequent operator engagement.

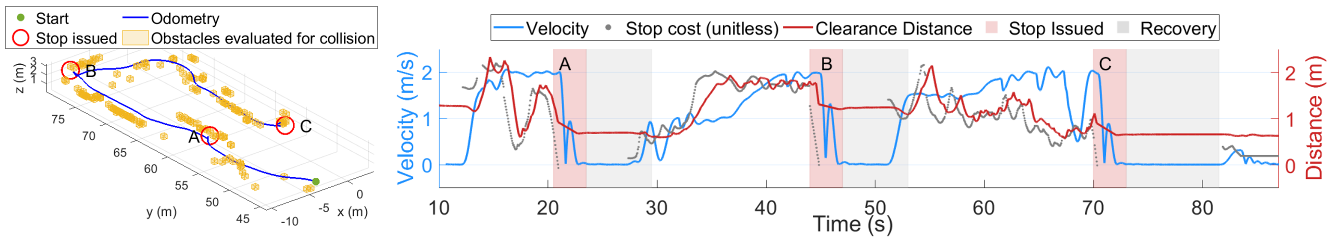


Fig. 9: Trial run in the warehouse environment with imminent collision checking enabled. Left: Vehicle odometry with queried obstacles. Locations A, B, C indicate where stopping trajectories were issued corresponding with the left plot. Right: Stopping trajectories are issued when stop cost drops below a threshold, ensuring that the vehicle is safe. The operator recovers the vehicle with an in-place yaw before normal flight is resumed.

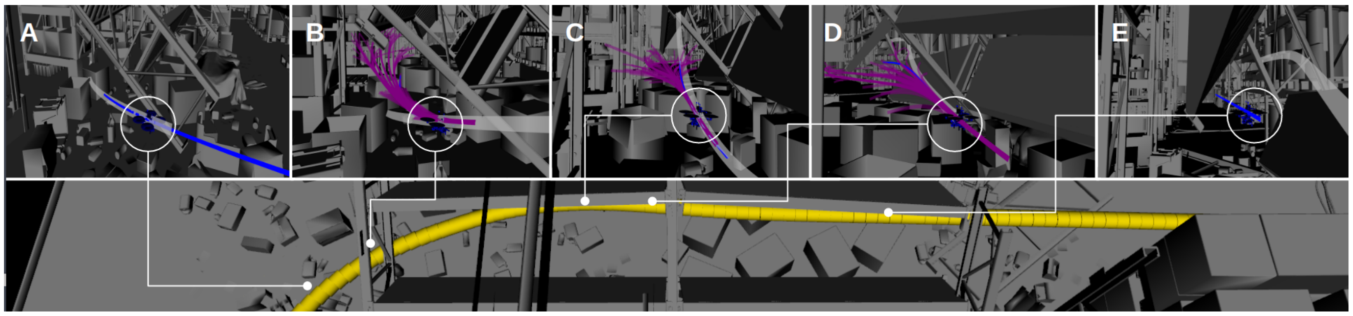


Fig. 10: Sequence of the passage through the collapsed shelf, from left to right. Trajectory currently following (in blue) leads to a collision, and trajectory generation is triggered resulting in candidate trajectories (in magenta) being generated. Light grey trajectory highlights the global path that guides the trajectory generation process, which adapts over time as the vehicle moves.

TABLE I: Results for three different density random forest environments

Approach	Sparse	Medium	Dense
Time to completion (s)			
MP (single-step)	38.84 ± 1.08	37.06 ± 0.50	37.95 ± 1.91
MPT (multi-step)	39.45 ± 1.43	36.40 ± 1.99	37.55 ± 2.34
Hierarchical	$\mathbf{38.10 \pm 3.96}$	$\mathbf{33.4 \pm 1.44}$	$\mathbf{34.73 \pm 1.35}$
Jerk Integral (m^2/s^3)			
MP (single-step)	28.41 ± 2.88	26.72 ± 4.30	50.14 ± 7.38
MPT (multi-step)	18.96 ± 5.05	25.32 ± 4.32	33.75 ± 2.77
Hierarchical	$\mathbf{25.10 \pm 18.10}$	$\mathbf{23.12 \pm 10.42}$	$\mathbf{25.94 \pm 7.96}$
Number of operator inputs to complete the trial			
MP (single-step)	192 ± 27	147 ± 45	421 ± 67
MPT (multi-step)	44 ± 12	38 ± 9	58 ± 10
Hierarchical	$\mathbf{21 \pm 12}$	$\mathbf{27 \pm 12}$	$\mathbf{35 \pm 10}$

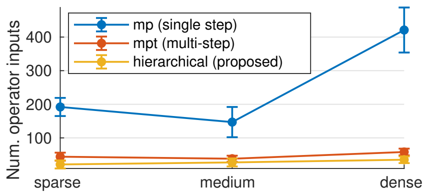


Fig. 11: Number of operator inputs as a function of the environment density for the random forest scenario, visualized. The number of inputs required to complete the task remains constant for the proposed method regardless of the density of the environment.

C. *Warehouse*

The warehouse environment is a large room with dimensions of 44.81m×22.17m×11m. The room contains three rows of industrial shelves, with a large shelf partially collapsed over and scattered objects surrounding the collapsed shelf, as highlighted in Fig. 3. The operator is asked to navigate in the warehouse following a rectangular path and through the collapsed shelf, which has a maximum clearance of 0.6m, and return to approximately close to the origin, which would require a narrow pass through between two horizontal shelves. The task is marked complete as soon as the vehicle completes the desired pathway and reaches within 1m of the origin. The difficulty of this experiments is highlighted in the high density clutter near the shelves and changes in direction, as shown in Fig. 10.

The results are tabulated in Table II. We highlight that while all three methods allow the operator to safely complete the task, the proposed hierarchical teleoperation method is able to do so with 63% and 17% less number of inputs, which is a significant reduction of operator engagement. The single-step and multi-step requires on avg. 175 and 77 inputs respectively, whereas the proposed method requires only 64. While the multi-step method is comparable, the multi-step method takes 15 seconds longer to complete the task than the proposed method for the approximately same distance travelled with a reduced average velocity of 1.48m/s, compared to the proposed method's 1.85m/s. This is due to the fact that the multi-step optimizes for finite-horizon navigation such that a rapid direction change would require the vehicle to slow down significantly to turn the corner, then speed up. The single-step method performs comparatively to the proposed method in terms of task completion duration and average velocity. However, we are able to achieve the same level of performance with a 63% reduction in the number of inputs required.

VI. Conclusion and Future Work

This paper presents a hierarchical teleoperation framework for mobile robot navigation in unstructured environments, informed by the operator's intention. We show our method in teleoperation tasks navigating in densely cluttered random forest and warehouse environments characterized by narrow gaps and unstructured free space. The proposed method completed the navigation task with consistently high average speeds while requiring the least operator engagement.

The hierarchical teleoperation framework can be readily extended in a few ways. The hierarchical formulation of the global and local planning allows a natural extension to interface with global and local map representations, such that the global plan incorporates environment features. Further, various types of local map representations can be adapted. Lastly, a more sophisticated operator intention model can be used to inform the trajectory generation process.

TABLE II: Results for the warehouse environment

Approach	Distance (m)	Duration (s)	Avg. Vel. (m/s)	Jerk integral (m^2/s^3)	Num. Inputs
MP (single-step)	82.126 ± 0.92	47.778 ± 0.503	1.772 ± 0.041	47.475 ± 21.328	175 ± 30
MPT (multi-step)	85.07 ± 3.95	59.73 ± 3.23	1.48 ± 0.08	78.21 ± 11.72	77 ± 19
Hierarchical (proposed)	$\mathbf{80.67 \pm 4.63}$	$\mathbf{44.85 \pm 0.93}$	$\mathbf{1.85 \pm 0.04}$	$\mathbf{31.19 \pm 6.81}$	$\mathbf{64 \pm 6}$

References

[1] M. Nieuwenhuisen, D. Droeschel, J. Schneider, D. Holz, T. Läbe, and S. Behnke, "Multimodal obstacle detection and collision avoidance for micro aerial vehicles," in *2013 European Conference on Mobile Robots*. IEEE, 2013.

[2] A. Spitzer, X. Yang, J. Yao, A. Dhawale, K. Goel, M. Dabhi, M. Collins, C. Boirum, and N. Michael, "Fast and agile vision-based flight with teleoperation and collision avoidance on a multirotor," in *Int. Sym. on Exp. Robot. (ISER)*. Springer, 2018.

[3] X. Yang and N. Michael, "Assisted mobile robot teleoperation with intent-aligned trajectories via biased incremental action sampling," *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[4] M. Hwangbo, J. Kuffner, and T. Kanade, "Efficient two-phase 3d motion planning for small fixed-wing uavs," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, 2007, pp. 1035–1041.

[5] M. Nieuwenhuisen and S. Behnke, "Hierarchical planning with 3d local multiresolution obstacle avoidance for micro aerial vehicles," in *ISR/Robotik 2014; 41st International Symposium on Robotics*. VDE, 2014, pp. 1–7.

[6] A. R. Diéguez, R. Sanz, and J. Lopez, "Deliberative online local path planning for autonomous mobile robots," *Journal of Intelligent and Robotic Systems*, vol. 37, no. 1, pp. 1–19, 2003.

[7] P. Sermanet, R. Hadsell, M. Scoffier, M. Grimes, J. Ben, A. Erkan, C. Crudele, U. Miller, and Y. LeCun, "A multirange architecture for collision-free off-road robot navigation," *Journal of Field Robotics*, vol. 26, no. 1, pp. 52–87, 2009.

[8] D. Ferguson, T. M. Howard, and M. Likhachev, "Motion planning in urban environments," in *The DARPA Urban Challenge*. Springer, 2009, pp. 61–89.

[9] A. Stentz and M. Hebert, "A navigation system for goal acquisition in unknown environments," in *Intelligent Unmanned Ground Vehicles*. Springer, 1997, pp. 277–306.

[10] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 2520–2525.

[11] M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Differentially constrained mobile robot motion planning in state lattices," *J. Field Robot.*, 2009.

[12] T. Eiter and H. Mannila, "Computing discrete fréchet distance," Citeseer, Tech. Rep., 1994.

[13] B. Aronov, S. Har-Peled, C. Knauer, Y. Wang, and C. Wenk, "Fréchet distance for curves, revisited," in *European Symposium on Algorithms*. Springer, 2006, pp. 52–63.