Decentralized Multi-Robot Planning in Dynamic 3D Workspaces

Arjav Desai and Nathan Michael *

The Robotics Institute, Carnegie Mellon University, Pittsburgh PA 15232, USA

Abstract. We consider the problem of decentralized multi-robot kinodynamic motion planning in dynamic workspaces. The proposed approach leverages offline precomputation on an invariant planning representation (invariant geometric tree) for low latency online planning and replanning amidst unpredictably moving dynamic obstacles to generate kinodynamically feasible and collision-free time-parameterized polynomial trajectories. Simulation results with up to 10 robots in dynamic workspaces composed of varying obstacle densities (up to 30 % by volume) and speeds (up to 2.5 m/s) suggest the use of the proposed methodology for real-time kinodynamic replanning in dynamic workspaces.

1 Introduction

Reliable and responsive multi-robot deployments in application domains such as search and rescue necessitate an online motion planning methodology to compute kinodynamically feasible and collision-free motion plans for each robot in the team. Additionally, the planning methodology must be robust to (1) unexpected changes in operator intent over the course of the deployment and (2) to the presence of dynamic obstacles (human operators, debris, displaced physical objects) sharing the workspace. In this work, we seek to develop a planning and coordination framework for multi-robot teams for generating kinodynamically feasible and safe motion plans for dynamic 3D workspaces.

This problem is challenging due to several reasons. *First*, kinodynamic planning in dynamic workspaces involves searching for a feasible sequence of states in a high-dimensional search space (challenge `C1`). The high dimensionality is attributed to the need to encode the higher-order kinematics and dynamics of the robot, as well as a time dimension to account for the spatiotemporal characteristics of the the dynamic obstacles in the workspace. *Second*, evolving mission conditions (e.g. online changes in goal locations) precludes the use of precomputed motion plans; therefore the multi-robot team must be able to replan online from potentially non-stationary initial states (challenge `C2`). This is challenging as robots cannot wait in place as in [11] in order to avoid conflicts. *Third*, long term trajectory prediction for stochastic dynamic obstacles is difficult due to

* Carnegie Mellon University, Pittsburgh, PA, USA, email: {`arjavdesai, nmichael`}`@cmu.edu`. We gratefully acknowledge support from industry.

compounding effects of modeling inaccuracies as well as state and motion uncertainty [1]. A short prediction horizon is thus favoured which in turn necessitates planning approach which can replan at high-rates while maintaining kinodynamic feasibility and safety (challenge C3).

Related Works: Several works address motion planning in dynamic workspaces. Hierarchical Cooperative A* (HCA*) proposed by Silver et al. [10] searches for geometric paths in the full space-time search space under the guidance of a lower dimensional heuristic. While HCA* guarantees the optimality of the solution, it is essentially an offline algorithm. Vemula et al. [12] sacrifice optimality for efficiency and extend the notion of adaptive dimensionality for path planning in dynamic workspaces. The dimensionality of the search space is selectively increased in regions of conflict and ignored everywhere else. While this approach provides higher success rates and lower computation times than HCA*, it does not explicitly consider kinodynamic constraints and assumes complete knowledge of the trajectories of dynamic obstacles. Phillips et al. in [9] propose a search-based approach for kinodynamic motion planning in unpredictably dynamic environments. The authors exploit the observation; in a dynamic environment, a particular robot configuration is collision-free for only a few time steps (a safe interval); to reduce the search space dimensionality. The success of this approach is predicated on the stationarity of the initial robot states. This assumption renders the approach in [9] impractical for online replanning scenarios. Sampling-based approaches such as Multipartite RRT (MP-RRT) proposed by Zucker et al. [14] proposes replanning amidst dynamic obstacles by biasing the sampling distribution and reusing sub-trees over the course of the search. RRT$^{\text{x}}$ proposed by Otte et al. [8] for single-robot replanning in dynamic environments address the limitations of MP-RRT; the search-tree in RRT$^{\text{x}}$ is rooted at the goal which eliminates the additional collision-checks and tree rewiring operations due to the motion of the robot. These algorithms work well for geometric planning tasks, however, they are not well suited for real-time planning in high-dimensional state spaces for agile systems like quadrotors due to the computational cost associated with solving a two-point boundary value problem for *each* sampled configuration and evaluating the corresponding solution for feasibility and safety. Optimization-based approaches such as the one proposed by Zhu et al. [13] also consider motion planning in dynamic environments. However, due to high computational complexity, these are typically restricted to sparsely cluttered environments.

Contributions: This work addresses the aforementioned challenges for multi-robot motion planning in dynamic workspaces. Our approach leverages offline reachability analysis on an invariant local planning representation (initially proposed in [3]) and decentralized planning to counter the *curse of dimensionality* (addressing C1) The proposed planning representation allows for replanning from non-stationary initial states (addressing C2). Additionally the representation allows for search in the lower dimensional geometric space and guarantees a kinodynamically feasible and safe solution if a geometric solution is found, without additional refinement. Finally, we propose a fast collision checking approach in dynamic environments that allows for low latency replanning (addressing C3).

2 Problem Formulation

2.1 Notation and Assumptions

$\mathbb{R}$ and $\mathbb{N}$ denote the set of real and natural numbers respectively. $\mathcal{S}$, $\mathbf{M}$, $\mathbf{v}$, and s denote sets, matrices, vectors, and scalars respectively. $|\mathcal{S}|$ denotes the set cardinality. The assumptions used in this work are as follows. First, the robots employed are differentially flat and jerk-controlled [6] quadrotor systems. The team composition is assumed to be homogeneous and each robot is physically modelled as a ball of of radius r denoted by $\mathcal{B}_r(\mathbf{x})$ where $\mathbf{x} \in \mathbb{R}^3$ denotes the cartesian coordinates of the centre of the ball. Second, the static portions of the 3-D workspace are known a priori. Third, the current states (position, velocity) and the bounding volumes of the dynamic obstacles are observable. The future trajectories are unknown and must be predicted. Fourth, robots can communicate without latency within a communication radius of r_{comm}.

2.2 Problem Statement

Consider a team of n robots with a p dimensional state space deployed in an uncertain dynamic workspace denoted by $\mathcal{W}$. The set of points corresponding to the known and static obstacles is given by $\mathcal{W}_{\text{s}}$ and the set of points occupied by the dynamic obstacles at time t are given by $\mathcal{W}_{\text{d}}^t$. At any time $t \in \mathbb{R}_{>0}$, the set of occupied points $\mathcal{W}_{\text{occ}}^t$ is the union set of $\mathcal{W}_{\text{s}}$ and $\mathcal{W}_{\text{d}}^t$ and the set of free points is given by $\mathcal{W}_{\text{free}}^t = \mathcal{W} \setminus \mathcal{W}_{\text{occ}}^t$. Let $\mathcal{I} \in \mathbb{R}^{n \times p}$ denote the set of initial robot states and $\mathcal{F} \in \mathbb{R}^{n \times p}$ denote the set of desired terminal states for n robots.

The objective of the motion planner is to generate a time-parameterized sequence of trajectories for each robot in the team that lead the robots from $\mathcal{I}$ terminate within an ϵ-ball, $\mathcal{B}_\epsilon$, of the terminal states $\mathcal{F}$ where $\epsilon \in \mathbb{R}^p$ are continuous up-to the second derivative of position, i.e., acceleration.

Let $\{\xi_i^{[t_0,t_1]}(t), \ldots, \xi_i^{[t_{k-1},t_k]}(t)\}$ denote the time-parameterized trajectories for robot i, and $\{\xi_j^{[t_0,t_1]}(t), \ldots, \xi_j^{[t_{k-1},t_k]}(t)\}$ denote the trajectories for robot j. At any time $t \in \mathbb{R}$ such that $t_{k-1} \leq t \leq t_k$, the corresponding states $\mathbf{x}_i^t \in \mathbb{R}^p$ and $\mathbf{x}_j^t \in \mathbb{R}^p$ satisfy the following constraints. First, each robot in the team must lie in the free space i.e. $\mathcal{B}_r(\text{pos}(\mathbf{x}_i^t)) \in \mathcal{W}_{\text{free}}^t$. Second, trajectories must satisfy the kinodyanmic constraints of the system. This corresponds to constraints on maximum acceleration and jerk. Third, robots must maintain clearances greater than $2r$ i.e. $||\mathbf{x}_i^t - \mathbf{x}_j^t|| > 2r$

3 Approach

This section is organized as follows. In Sect. 3.1 we discuss the static and dynamic workspace representation. Sect. 3.2 describes the kinodynamic planner, the dynamic obstacle avoidance procedure, the replanning algorithm, and the decentralized planning setup.

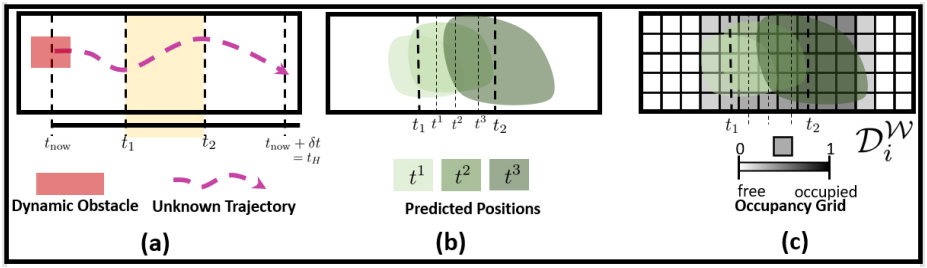

Fig. 1. Dynamic workspace representation. (a) The prediction horizon is split into k major intervals. (b) Each major interval is split into k' minor intervals and the position of the dynamic obstacles is predicted at each of the k' time instances. (c) An occupancy grid, $\mathcal{D}_i^{\mathcal{W}}$, with the collated occupancy values of k' minor intervals is generated for each major interval.

3.1 Workspace Representation

The dynamic workspace $\mathcal{W}_d^t$ is represented via two voxelgrids denoted by $\mathcal{S}$ and $\mathcal{D}$ that correspond to the static and dynamic components of the workspace.

Static Workspace Representation: Since the static obstacles are known *a priori*, the static voxelgrid $\mathcal{S}$ is computed offline. Additionally we precompute a sparse roadmap of the environment with respect to $\mathcal{S}$ using the `SPARS2` algorithm [5]. The sparse roadmap captures the free space topology and provides approximate cost-to-go estimates to the kinodynamic planner (refer Sect. 3.2).

Dynamic Workspace Representation: The dynamic occupancy grid $\mathcal{D}^{\mathcal{W}}$ is a four dimensional grid (cartesian coordinates x, y, and z and time t) that encodes the spatiotemporal occupancy of the dynamic obstacles in the 3-D workspace over a time horizon t_H. Each cell in $\mathcal{D}$ is assigned a value, $c \in \mathbb{R};\ 0 \leq c \leq 1$, which denotes the occupancy probability of that cell. Let t_{now} denote the current time at which the state of the dynamic obstacles is observed and t_H denote the time at the prediction horizon. The time-period $[t_{\text{now}}, t_H]$ is split into k major time-intervals and a 3D occupancy grid is maintained corresponding to each time-interval (Fig. 1). Each time-interval is further discretized into k' minor time instances and the state of the dynamic obstacles is predicted for each of these time instances (Fig. 1b). The predicted states of the dynamic obstacles are used to update the 3D occupancy grid corresponding to that interval (Fig. 1c). Thus, each 3D occupancy grid in $\mathcal{D}^{\mathcal{W}}$ represents the *collated* spatiotemporal occupancy of the dynamic obstacles in that time interval. When multiple occupancy probabilities are possible for a voxel (e.g. due to multiple dynamic obstacles), the highest occupancy probability is assigned to that voxel.

Dynamic Obstacle Modelling and Prediction: The motion of the dynamic obstacles is predicted via the Linear Velocity Polyhedron (`LVP`) model proposed in [7]. Let $\mathcal{H}$ denote a halfspace in $\mathbb{R}^m$, i.e., $\mathcal{H} = \{\mathbf{p} \mid \mathbf{a}^{\mathrm{T}}\mathbf{p} \leq b,\ \mathbf{p} \in \mathbb{R}^m\}$. A convex polyhedron, $\mathcal{C}$, is defined as an intersection set of k halfspaces. Let the jth halfspace at time instance $t = 0$ be defined as $\mathcal{H}_j^{t_0} = \{\mathbf{p} \mid \mathbf{a}_{\{j,t_0\}}^{\mathrm{T}}\mathbf{p} \leq b_{\{j,t_0\}}\}$. Further, let the linear velocity of the points in $\mathcal{C}$ be $\mathbf{v}_{\mathcal{C}}$. The jth halfspace of $\mathcal{C}$ at $t = t_1$ is given by $\mathbf{a}_j(t_1) = \mathbf{a}_{\{j,t_0\}}$, and $b_j(t_1) = b_{\{j,t_0\}} + \mathbf{a}_{\{j,t_0\}}^{\mathrm{T}}\mathbf{v}_{\mathcal{C}}\Delta_t$. Here, Δ_t is the time difference $t_1 - t_0$. Equations (1-2) do not account for uncertainty in

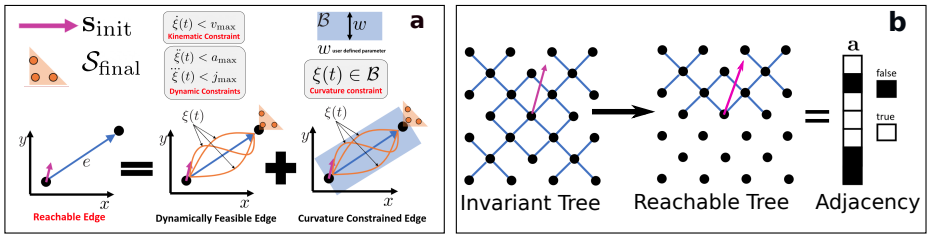

Fig. 2. Figure shows (a) reachable edges that are dynamically feasible and curvature constrained (b) cartesian projection of the reachable tree given the initial higher-order state at the root and its alternate representation via a boolean valued adjacency vector.

the predicted motion of the dynamic obstacles. To do so, the obstacle's geometry is inflated over time by a constant inflation factor $\mathbf{v}_e$. While the equation for $\mathbf{a}_j(t_1)$ remains the same, $b_j(t_1)$ is updated as follows:

$$b_j(t_1) \;=\; b_{\{j,t_0\}} + (\mathbf{a}^{\mathrm{T}}_{\{j,t_0\}}\mathbf{v}_{\mathcal{C}} \;+\; ||\mathbf{a}_{\{j,t_0\}}||\mathbf{v}_e)\Delta_t \tag{1}$$

We note that the primary contribution of this work is development and analysis of a local planning representation capable to efficient motion-planning in uncertain dynamic workspaces and thus, the `LVP` predictor can be replaced with state-of-the-art prediction methodologies. We refer the reader [13] and the references therein for a detailed exposition of various prediction mechanisms.

3.2 Kinodynamic Planner

Local Planning Representation: We summarize the key properties of the planning representation initially proposed in our prior work [4]. The planning representation is an invariant geometric tree, T, constructed by propagating a geometric lattice in an obstacle-free workspace for multiple time steps. The tree is defined by an invariant vertex set $\mathcal{V}_T$ and a set of directed edges $\mathcal{E}_T$. The initial and terminal points of a tree edge $e \in \mathcal{E}_T$ are represented by $e(0)$ and $e(1)$ respectively. Kinodynamic constraints are encoded in the definition of edges.

- A *dynamically feasible edge* is one such that for some initial higher-order derivative $\mathbf{s}_{\text{init}}$ at $e(0)$, a non-empty set of higher-order derivatives exists $\mathcal{S}_{\text{final}}$ at $e(1)$ such that the fixed duration time-parameterized polynomial trajectories [6] connecting $\mathbf{s}_{\text{init}}$ to $\mathcal{S}_{\text{final}}$ along e do not violate the kinodynamic constraints of the platform (Fig. 2a).
- A *curvature constrained edge* is one such that for some initial higher-order derivative $\mathbf{s}_{\text{init}}$ at $e(0)$, there exists a non-empty set of higher-order derivatives $\mathcal{S}_{\text{final}}$ at $e(1)$ such that the trajectories connecting $\mathbf{s}_{\text{init}}$ to $\mathcal{S}_{\text{final}}$ are entirely contained within a cuboidal bounding box, $\mathcal{B}$, of length $l(e)$ and a fixed width and height, oriented along the edge e (Fig. 2a).
- A *reachable edge* is dynamically-feasible *and* curvature constrained.

A *reachable tree* is then a tree that is composed of reachable edges. For any higher-order state at the root node $\mathbf{s}_{\text{init}}$, the corresponding reachable tree is

then defined by the tuple $T = (\mathcal{V}, \mathbf{p}, \mathbf{a}, \mathbf{c})$, where $\mathcal{V}$ is the vertex set in $\mathbb{R}^3$, $\mathbf{p} \in \mathbb{N}^{|\mathcal{V}_T|}$ stores the parent ids of each vertex in the tree, $\mathbf{c} \in \mathbb{R}^{|\mathcal{V}_T|}$ is a cost vector that stores the the cost-to-come to the ith vertex, and $\mathbf{a} \in \{0,1\}^{|\mathcal{V}_T|}$ is a boolean valued adjacency (0 denotes unreachable) vector that encodes the set of reachable edges subject to the *underlying planning context* such as higher-order derivatives at the root vertex, or collisions with obstacles (Fig. 2b).

Proposition 1 Let $\mathcal{S}$ be a higher-order derivative set at the root vertex of an invariant k-step tree. Let $T_{\mathbf{s}_i} = (V, \mathbf{p}, \mathbf{a}_{\mathbf{s}_i}, \mathbf{c})$ represent the reachable k-step tree of $\mathbf{s}_i \in \mathcal{S}$. The reachable k-step tree of the entire set, $T_{\mathcal{S}}$, is given by $\mathbf{a}_{\mathcal{S}} = \sum_{\forall i,\ \text{bitwise}} \mathbf{a}_{\mathbf{s}_i}$. We refer to this as the **merge-tree** operation. If $\mathbf{a}_{\mathcal{S}}(i) = 1$, the path from the root to the ith vertex is composed of reachable edges and there exists at least one $\mathbf{s}_i \in \mathcal{S}$ for which this path is reachable. The proof follows from the fact that each vertex of a tree can have only one parent. □

Let $\mathbf{a}_T^{\text{kino}}$ denote the kinodynamically feasible tree corresponding to higher-order derivatives at the tree root, $\mathbf{a}_T^{\text{static}}$, the collision-free tree w.r.t the static obstacles (edges in $\mathbf{a}_T^{\text{static}}$ may be kinodynamically infeasible) and let $\mathbf{a}_T^{\text{dynamic}}$ denote the collision-free tree with respect to the dynamic obstacles. The reachable tree $\mathbf{a}_T$ with kinodynamic and collision constraints is obtained by.

$$\mathbf{a}_T = \mathbf{a}_T^{\text{kino}} \odot \mathbf{a}_T^{\text{static}} \odot \mathbf{a}_T^{\text{dynamic}} \tag{2}$$

Here, $\odot$ refers to the elementwise multiplication operation. We refer the reader to our prior work [4] for details on construction of $\mathbf{a}_T^{\text{kino}}$ and $\mathbf{a}_T^{\text{static}}$. Construction of $\mathbf{a}_T^{\text{dynamic}}$ is discussed in the subsection on dynamic obstacle avoidance.

Online Planning Procedure: The single-robot planner constructs trajectories from the start to the goal via a two step process that decouples geometric search (step 1) and higher-order derivative assignment to the intermediate vertices of the geometric path (step 2). The geometric search proceeds by incrementally translating the invariant tree towards the goal via:

- **Infer (Merge) Reachable Tree**: In each iteration, a reachable tree T is inferred given the underlying higher-order state at the root node (Fig. 3a). The $\mathbf{a}_T$ vector is updated such that the unreachable vertices and edges are pruned from the tree as in Eqn. (2). After the first iteration, a *set* of higher-order states may feasibly exist at the root node. This is due to the association of multiple kinodynamically feasible trajectories with each geometric edge (Fig. 2a). The reachable tree corresponding to the entire set is generated via the merge-tree operation (Prop. 1).
- **Select Intermediate Goal**: An intermediate goal is selected from the reachable vertices (using cost-to-go estimates from the `SPARS` roadmap) and the root of the tree is translated to the intermediate goal (Fig. 3b).
- **Update Derivative Graph**: A directed graph structure called the derivative graph $\mathcal{G}_{\text{der}}$ updated in each iteration with the time indexed distribution of higher-order derivatives that can *feasibly* exist at the intermediate vertices of the geometric path constructed *thus far* (Fig. 3c and inset).

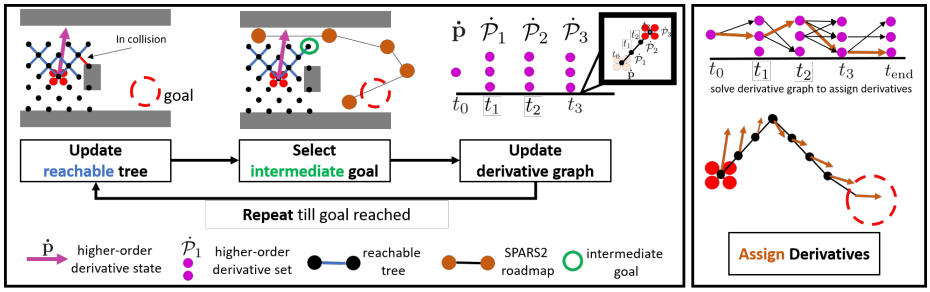

Fig. 3. Online planning procedure with invariant geometric trees

This process continues till the root node of the tree reaches the goal after which, higher-order states are assigned to the intermediate geometric vertices by searching for the least cost (minimum jerk) path in the derivative graph Fig. 3d.

Dynamic Obstacle Avoidance: Here, we describe the procedure to identify the edges in the tree with a collision probability greater than p_{coll} and accordingly update the $\mathbf{a}_T^{\text{dynamic}}$ vector. While this can be done via explicit evaluation of edges for trees with a low cardinality, operating in dense multi-robot scenarios typically requires a high cardinality tree as higher cardinality corresponds to greater maneuverability. For efficient collision detection, we exploit the invariance of the planning representation and propose a two-stage offline-online procedure for obstacle avoidance (Fig. 4).

Offline Stage: In an obstacle free workspace, each edge of the tree T is discretized and a time index is associated with each discrete point. We only consider the time dimension up to t_H seconds (the prediction horizon). Similar to the construction of $\mathcal{W}^{\mathcal{D}}$, we split this time horizon into k intervals. For each interval, we construct a 3D voxelgrid where voxels marked occupied intersect with the tree edges spatially *and* in time. Thus, as in $\mathcal{D}^{\mathcal{W}}$, this gives rise to k three-dimensional voxelgrids (Fig. 4a). Let $\mathcal{D}^T$ denote the spatiotemporal tree voxelgrid. In addition to constructing $\mathcal{D}^T$, with each occupied voxel in $\mathcal{D}^T$, we associate the *minimal* set of edges that intersect with it i.e. if edges i, j, and k intersect with a voxel and if i is the parent of j and k, only edge i is associated with that voxel (Fig. 4b). This spatiotemporal voxelgrid representation and the voxel-to-edge mapping is precomputed and used during the online stage.

Online Stage: In the *online* stage (Fig. 4c), all voxels in $\mathcal{D}^{\mathcal{W}}$ with occupancy values greater than or equal to $1-p_{\text{coll}}$ are marked as occupied i.e. the occupancy values of these voxels are set to 1. All other voxels in $\mathcal{D}^{\mathcal{W}'}$ with occupancy values less than $1-p$ are marked as free. For each of the k voxelgrids in $\mathcal{D}^{\mathcal{W}'}$, we search for the set of voxels, $\mathcal{V}_{\text{common}}$, that are occupied by *both* the tree edges in $\mathcal{D}^T$ and by the dynamic obstacles $\mathcal{D}^{\mathcal{W}'}$ given the current position of the robot and the state of $\mathcal{D}$. The edges that intersect with $\mathcal{V}_{\text{common}}$ i.e. $\text{occ}(\mathcal{D}^{\mathcal{W}'}) \cap \text{occ}(\mathcal{D}^T)$ are queried from the voxel-to-edge mapping and the corresponding indices in $\mathbf{a}_T^{\text{dynamic}}$ are marked unreachable. The set of unreachable vertices $\mathbf{a}_T^{\text{dynamic}}$ are obtained by running Dijkstra's algorithm from the root. All reachable edges in the resulting invariant tree have a probability of collision less than p_{coll}.

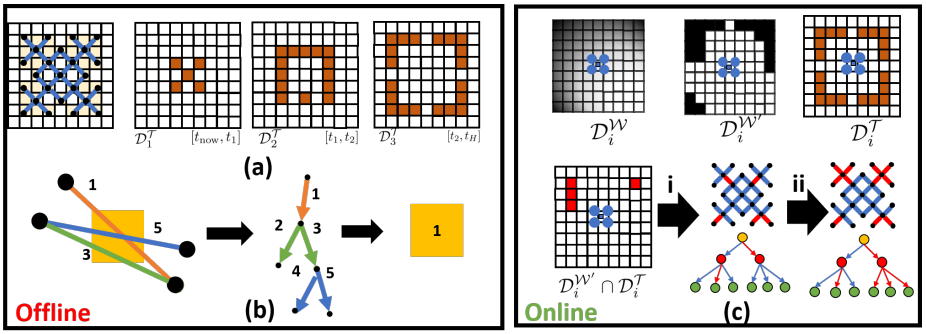

Fig. 4. Offline preprocessing of invariant geometric trees to construct (a) spatiotemporal voxel grids and (b) voxel-to-edge mapping. (c) During online operation, given the ith slice of the dynamic workspace grid $\mathcal{D}_i^{\mathcal{W}}$, the occupied voxels that also intersect with the ith slice of the spatiotemporal tree voxel grid $\mathcal{D}_i^{\mathcal{T}}$ are found. The minimal set of edges corresponding to the common voxels are queried from the voxel-to-edge mapping (process i) and Dijkstra's algorithm is run from the root to obtain the true set of reachable edges (process ii).

Replanning Procedure: Let t_{now} be the current time, ξ_i be the current trajectory of the ith robot in the team and $\mathcal{D}^{\mathcal{W}}$ be the most recent estimate of the dynamic voxelgrid. The trajectory ξ_i is evaluated for collisions with respect to $\mathcal{D}^{\mathcal{W}}$ in the time interval $[t_{\text{curr}}, t_H]$ where t_H is the time at the prediction horizon of $\mathcal{D}^{\mathcal{W}}$. If a collision is detected either with a dynamic obstacle or with the trajectory ξ_j of any robot in the team such that $i \neq j$, replanning is triggered. The replanning procedure typically requires a finite amount of time to execute. Thus, replanning from the current state of the robot $\xi_i(t_{\text{curr}})$ can introduce discontinuities in position as well as higher-order derivatives of the robot state that can lead to catastrophic failures. In order to avoid such a scenario, a state $\mathbf{x}_{\text{start}} \in \mathbb{R}^p$ that is $t_{\text{lookahead}}$ in the future is extracted from ξ_i and the robot replans from that state. If the actual execution time is greater than $t_{\text{lookahead}}$, the computed plan is rejected and a new replanning instance is initiated. While replanning, the single-robot planner with dynamic obstacle avoidance is executed. However, the dynamic obstacle avoidance is only considered until t_H seconds, after which, the dynamic obstacles are ignored. After successful replanning, each robot communicates its updated motion plans to its neighbors in $\mathcal{G}_{\text{comm}}$.

Decentralized Planning Architecture: Our decentralized planning architecture consists of a *commander* node (e.g. a human operator) that assigns labelled goal states, $\mathcal{F}$, to a subset of robots, $n' \leq n$, in the team. A *workspace observer* tracks the dynamic obstacles and broadcasts the timestamped obstacle states (position, velocity) to each robot in the team. On-board, each robot maintains and updates the dynamic voxelgrid $\mathcal{D}^{\mathcal{W}}$. All robots plan their trajectories *synchronously* and communicate the computed trajectories to their neighbors in the *communication graph* $\mathcal{G}_{\text{comm}}$ i.e. an undirected graph structure that encodes the time-varying communication topology of the multi-robot team. Two robots communicate if the distance between them is less than a predefined communication

radius r_{comm}. At each time step, each robot avoids the trajectories computed by all the other robots in the previous time step thus guaranteeing safety.

4 Evaluation

4.1 Implementation Details and Experiment Design

The experimental evaluation was conducted using the Julia programming language on a Lenovo Thinkpad with an Intel 4-Core i7 CPU and 16 GB RAM. The proposed methodology is evaluated via three studies. All studies employ robots of radius 0.1 m with a 2.6 m/s velocity limit and a 6.8 m/s^2 acceleration limit.
Study 1: Comparison of the collision-checking method with lazy evaluation [2].
Study 2: Comparison of the local planning representation with motion-primitive trees commonly employed in state-of-the-art local planners [7].
Study 3: Decentralized planning architecture evaluation in dynamic 3D workspaces.

4.2 Results

Study 1: Evaluation of Collision-Avoidance Methodology: We compare the proposed dynamic collision-checking methodology with lazy evaluation [2] on six geometric trees of cardinalities $|T_1| = 70,491$ and $|T_2| = 109,893$ and spatial coverage volumes of $4 \times 4 \times 4$ m^3, $6 \times 6 \times 6$ m^3 and $8 \times 8 \times 8$ m^3. We conduct 100 trials for each tree and in each trial a random dynamic voxelgrid $\mathcal{D}^{\mathcal{W}}$ is generated. The objective of each trial is to select a sequence of edges in the tree that do not collide with the occupied voxels in $\mathcal{D}^{\mathcal{W}}$ and lead the robot towards a predefined goal, i.e., a feasible path selection problem. The lazy collision checking approach (abbrv. `LazyEval`) employs the `Forward` edge selector [2] in order to fully exploit the tree structure of the planning representation and avoid redundant edge evaluations. Table 1 reports the mean and standard deviation of the time required to find a collision-free path using both approaches. The proposed collision-checking approach achieves significantly lower computation times (upto 3.1 times for T_1 and 3.5 times for T_2) for all six geometric trees compared to `LazyEval`. The low computation times (approx. 50 Hz for T_2 with a coverage volume of $8 \times 8 \times 8$ m^3) suggest viability of the proposed approach for use in high-rate local planners typically used in dynamic workspaces. As the coverage volume increases, the time required by the proposed method approaches that of

Table 1. Mean and S.D. of the time required (in s) to search for collision-free sequence of edges in geometric trees of varying cardinalites and coverage volumes for 100 trials in random dynamic voxelgrids $\mathcal{D}$ with 0.1 m resolution. Multipliers are shown in blue.

Cardinality	$\lvert T_1\rvert = 70,491$			$\lvert T_2\rvert = 109,893$		
Coverage	$4 \times 4 \times 4$ m^3	$6 \times 6 \times 6$ m^3	$8 \times 8 \times 8$ m^3	$4 \times 4 \times 4$ m^3	$6 \times 6 \times 6$ m^3	$8 \times 8 \times 8$ m^3
Proposed	**0.007** $\pm$ 0.002	**0.017** $\pm$ 0.008	**0.047** $\pm$ 0.034	**0.009** $\pm$ 0.010	**0.019** $\pm$ 0.009	**0.050** $\pm$ 0.06
LazyEval (Forward)	0.022 $\pm$ 0.015 **(3.142x)**	0.042 $\pm$ 0.046 **(2.470x)**	0.079 $\pm$ 0.085 **(1.680x)**	0.032 $\pm$ 0.023 **(3.555x)**	0.048 $\pm$ 0.056 **(2.526x)**	0.102 $\pm$ 0.414 **(2.040x)**

Table 2. Comparison of the proposed planning representation (`IGTree`) with a state-of-the-art planning representation `MPTree` in nine control environments. Table reports the success rate (`SR`) and average response times (`RT`) for 900 trials. $|T|$ and $|U|$ denote the cardinality of `IGTree` and `MPTree` respectively and d denotes the depth.

Obst. Density	A (4.6 %)						B (7.5 %)						C (9.0 %)					
Obst. Speed	Low (3.0 m/s)		Medium (4.0 m/s)		High (5.0 m/s)		Low (3.0 m/s)		Medium (4.0 m/s)		High (5.0 m/s)		Low (3.0 m/s)		Medium (4.0 m/s)		High (5.0 m/s)	
Metrics	SR %	RT (ms)	SR %	RT (ms)	SR %	RT (ms)	SR %	RT (ms)	SR %	RT (ms)	SR %	RT (ms)	SR %	RT (ms)	SR %	RT (ms)	SR %	RT (ms)
IGTree (Our) $\lvert T\rvert$ = 14,763	**96.2**	**5.9**	**96.2**	**5.5**	**96.2**	**5.5**	**100.0**	**8.5**	**100.0**	**18.0**	**97.3**	**8.0**	**100.0**	**9.5**	**88.5**	**8.3**	**87.3**	**8.2**
MPTree (d=1) $\lvert U\rvert$ = 5,329	70.2	21.2	67.3	21.2	67.1	21.3	68.4	24.3	67.4	24.2	65.5	23.5	37.0	27.5	30.6	25.2	28.4	26.1
MPTree (d=1) $\lvert U\rvert$ = 14,641	72.1	60.4	69.9	59.9	69.1	81.3	69.1	98.7	68.0	77.4	66.9	94.4	36.5	97.0	29.5	100.6	27.6	84.5
MPTree (d=2) $\lvert U\rvert$ = 14,763	84.0	58.6	83.5	59.8	83.8	59.7	84.5	66.0	84.4	68.7	84.3	71.8	83.5	79.0	72.5	79.1	67.4	78.1

`LazyEval` but the key takeaway is that here is for volumes resembling local map dimensions, leveraging invariant structures is beneficial as offline preprocessing can significantly expedite online search.

Study 2: Evaluation of Local Planning Representation: We compare the proposed representation, i.e., invariant geometric trees (abbrev. `IGTree`) with control-input discretized motion-primitive trees (abbrev. `MPTree`) commonly employed by several state-of-the-art local planners [7]. The performance is evaluated by studying their behavior in nine 2D environments composed of varying obstacle densities and maximum obstacle speeds. In this study, 900 trials are conducted, and in each trial, the initial velocity of the robot $[v_x, v_y]^T$ is randomly selected from the range $v_x, v_y \in [-2.5, 2.5]$. The initial position of the robot is fixed at $[0.0, 0.0]^T$ and the projected or intermediate goal is fixed at $[4.0, 0.0]^T$. A trial is deemed successful if the local planner can compose kinodynamically-feasible and collision-free trajectories for a prediction horizon of 2.0 s. For this study, success rate (abbrev. `SR`), response time (abbrev. `RT`), and solution cost (total jerk for t_H seconds) are used as comparison metrics. From Table 2 we conclude that the proposed representation can safely tackle a wider range of dynamic local planning scenarios with a higher success rate and lower response times than the `MPTree` representation. This is because `IGTree` reasons over a *set* of higher-order states at each geometric vertex as opposed to committing to a singular state at each time-step as in the `MPTree` representation. In terms of the solution cost (Fig. 5-e), the `MPTree` representation outperforms the proposed representation

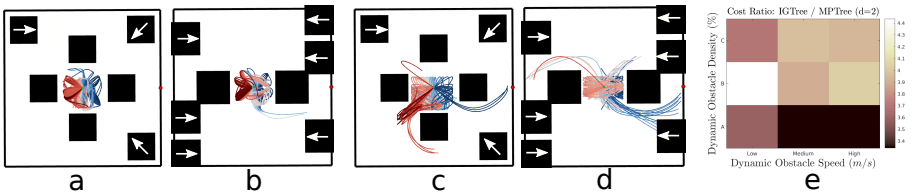

Fig. 5. Qualitative single-horizon solutions for `IGTree` (**a-b**) and `MPTree` (**c-d**) with $d = 2$ in 2D maps of varying obstacle densities.The black boxes indicate the obstacles and the white arrows indicate the direction of motion of the obstacles. (**e**) Heatmap with ratios of solution cost of `IGTree` compared to `MPTree`.

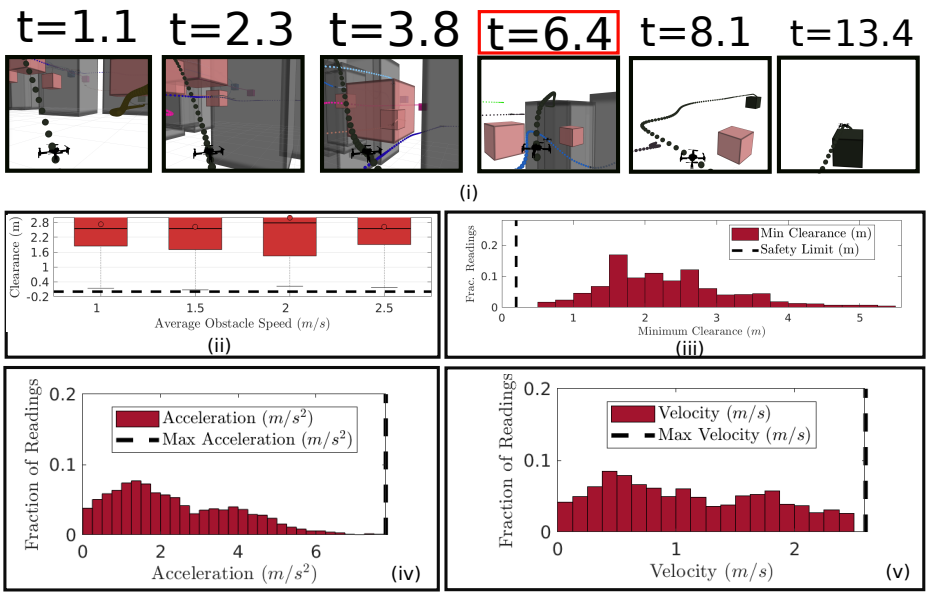

Fig. 6. **(i)** Snapshots of a single robot (dark green) at different time instances (in seconds) over the course of one entire planning experiment. The robot avoids the dynamic obstacles in the workspace and terminates in the goal region (dark green cube). At time 6.4 seconds, the robot replans a new trajectory from a non-stationary state to avoid collision with one of the dynamic obstacles. **(ii)** Figure shows the adjusted robot-obstacle clearance distances (observed clearance - safety limit) for different average obstacle speeds ranging from 0.5 m/s to 2.5 m/s (negative indicates collision). **(iii)-(v)** Bar plots shows distribution of inter-robot clearance distances, velocities, and accelerations across all experiments. Trajectories are dynamically feasible and safe.

as the trajectories obtained using our representation are constrained to remain within fixed geometric lattices thus incurring a higher control effort.

Study 3: Decentralized System Evaluation in 3D Workspaces The control environments used in this study are $10 \times 10 \times 5$ m^3 workspaces consisting of two dynamic obstacle configurations—a *pillar* ($2 \times 2 \times 5$ m^3) that translates in 2-D and a *box* ($1 \times 1 \times 1$ m^3) that translates in 3-D. Forty environments with varying obstacle density and speeds are used. The obstacle density varies from 2.2% to 30.8% (by volume) and the obstacle speed varies from 1.0 m/s to 2.5 m/s. Ten planning experiments were conducted for team sizes of $n = 1$ to $n = 10$ each in the given control environments (Fig. 6-i). The resulting trajectories are evaluated for (1) robot-obstacle collisions (Fig. 6-ii), (2) inter-robot collisions (Fig. 6-iii), (3) kinodynamic feasibility (Fig. 6-iv,v), and response times. The clearance plots show that the robots maintain safe inter-robot clearance distances and on average, maintain a clearance distance of 2.5m from the dynamic obstacles. Further, based on the distribution of velocities and acceleration across all the computed motion plans, we conclude that the motion plans adhere to to the specified kinodynamic constraints. Representative experiment videos are available at https://bit.ly/2VXn3in.

5 Conclusion and Future Work

This work presents a decentralized planner and a planning representation for multi-robot navigation in uncertain 3D workspaces. The invariant nature of the kinodynamic planning representation allows for offline preprocessing thus enabling low-latency generation of kinodynamically feasible and collision-free trajectories online in dynamic 3D workspaces. As part of the future research, we intend to conceive a distributed heirarchical coordination framework composed of long-term deliberative and short-term reactive planning processes.

References

1. Georges S Aoude, Brandon D Luders, Joshua M Joseph, Nicholas Roy, and Jonathan P How. Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns. *Autonomous Robots*, 35(1):51–76, 2013.
2. Christopher M Dellin and Siddhartha S Srinivasa. A unifying formalism for shortest path problems with expensive edge evaluations via lazy best-first search over paths with edge selectors. In *Twenty-Sixth International Conference on Automated Planning and Scheduling*, 2016.
3. Arjav Desai, Matthew Collins, and Nathan Michael. Efficient kinodynamic multi-robot replanning in known workspaces. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1021–1027. IEEE, 2019.
4. Arjav Desai and Nathan Michael. Online planning for quadrotor teams in 3-d workspaces via reachability analysis on invariant geometric trees. In *2020 International Conference on Robotics and Automation (ICRA)*. IEEE, 2020.
5. Andrew Dobson and Kostas E Bekris. Improving sparse roadmap spanners. In *2013 IEEE International Conference on Robotics and Automation*, pages 4106–4111. IEEE, 2013.
6. Markus Hehn and Raffaello D'Andrea. Quadrocopter trajectory generation and control. *IFAC proceedings Volumes*, 44(1):1485–1491, 2011.
7. Sikang Liu. Motion planning for micro aerial vehicles. 2018.
8. Michael Otte and Emilio Frazzoli. Rrtx: Asymptotically optimal single-query sampling-based motion planning with quick replanning. *The International Journal of Robotics Research*, 35(7):797–822, 2016.
9. Mike Phillips and Maxim Likhachev. Sipp: Safe interval path planning for dynamic environments. In *2011 IEEE International Conference on Robotics and Automation*, pages 5628–5635. IEEE, 2011.
10. David Silver. Cooperative pathfinding. *AIIDE*, 1:117–122, 2005.
11. Matthew Turpin, Kartik Mohta, Nathan Michael, and Vijay Kumar. Goal assignment and trajectory planning for large teams of interchangeable robots. *Autonomous Robots*, 37(4):401–415, 2014.
12. Anirudh Vemula, Katharina Muelling, and Jean Oh. Path planning in dynamic environments with adaptive dimensionality. In *Ninth Annual Symposium on Combinatorial Search*, 2016.
13. Hai Zhu and Javier Alonso-Mora. Chance-constrained collision avoidance for mavs in dynamic environments. *IEEE Robotics and Automation Letters*, 4(2):776–783, 2019.
14. Matt Zucker, James Kuffner, and Michael Branicky. Multipartite rrts for rapid replanning in dynamic environments. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 1603–1609. IEEE, 2007.