Online Planning for Quadrotor Teams in 3-D Workspaces via Reachability Analysis On Invariant Geometric Trees

Arjav Desai and Nathan Michael

***Abstract*— We consider the kinodynamic multi-robot planning problem in cluttered 3-D workspaces. Reachability analysis on position invariant geometric trees is leveraged to find kinodynamically feasible trajectories for the multi-robot team from potentially non-stationary initial states. The key contribution of our approach is that a collision-free geometric solution guarantees a kinodynamically feasible, safe solution without additional refinement. Simulation results with up-to 40 robots and hardware results with 5 robots suggest the viability of the proposed approach for online planning and replanning for large teams of aerial robots in cluttered 3-D workspaces.**

I. Introduction

A. Motivation

Safe and responsive real-world multi-robot deployments in application domains such as coverage [13], target search and tracking [23], search and rescue [1], [28], construction [16] as well as building clearance (Fig 1) necessitate an online kinodynamic planning and replanning framework that can efficiently cater to changes in operator intent (e.g. assignment of new goals over the course of the deployment) and spatiotemporal changes in the workspace (e.g. due to the presence of dynamic obstacles [20]). In this work, we seek to develop a scalable and responsive planning framework for generating dynamically feasible and collision-free motion plans for large teams of quadrotors in known and static workspaces in response to online changes in goal locations.

B. Related Works

Two primary challenges are associated with the online kinodynamic multi-robot planning problem. First, the high dimensionality of the composite search space [26], [27] restricts direct applicability of standard search [7] or sampling-based [12] methods. State-of-the-art methods [11], [29] typically search for a geometric solution followed by an iterative refinement procedure that converts the geometric paths into dynamically feasible, time-parameterized polynomial trajectories. The iterative refinement procedure is a computational bottleneck for large team sizes due to an increase in the number of dynamic feasibility evaluations and collision checks. This restricts the use of these approaches in scalable, and responsive online planning frameworks.

Second, replanning for agile systems like quadrotors involves planning from non-stationary initial states making the motion planning and coordination sub-problems additionally challenging since paths in collision cannot be naively assigned time-offsets as in [29] to resolve conflicts. Several recent works have addressed the problem of motion planning and coordination from non-stationary initial states. Approaches include MAPF [5], [25] guided sampling-based [14], [15] methods, optimization-based [10], [19], [22], [24] methods, and search-based methods using motion-primitives [17], [30] and state lattices [4], [21]. We restrict our discussion to search- and sampling-based techniques due to the high computational complexity of optimization-based methods. Sampling-based approaches, although probabilistically complete, are not well suited for online multi-robot planning frameworks primarily due to the cost associated with the large number of sample evaluations for dynamic feasibility and safety. Search-based methods using motion-primitives [17], [30] have received significant attention in the context of single robot replanning; however, reasoning over induced state-space discretization (as obtained using motion-primitives) can be less beneficial in the context of coordination due to reasons illustrated in Fig. 2. In this work, we argue that maintaining a distribution of higher-order derivatives over a fine geometric lattice (or an invariant geometric tree) improves coordination while maintaining dynamic feasibility and avoiding the computationally expensive refinement procedures common in geometric methods.

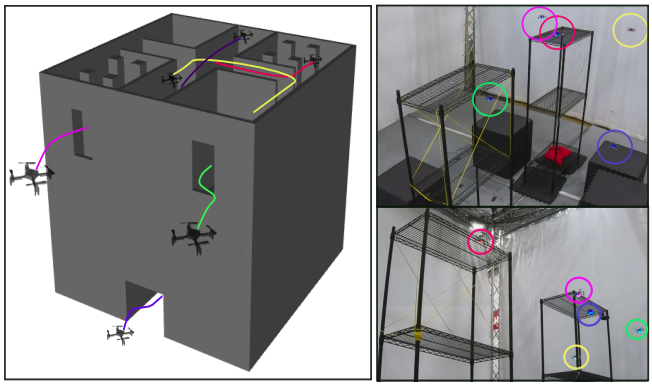

Fig. 1: (Left) Building clearance as a motivating application domain where teams of robots operate and coordinate in close proximity to assess and identify threats in a cluttered 3-D space. (Right) Online planning with 5 crazyflie quadrotors in a cluttered workspace.

C. Contributions

The contributions of this paper are as follows:

- Existing geometric methods, **(A)** do not guarantee a kinodynamically feasible solution if a geometric solution is found [17] and **(B)** computationally expensive

The authors are affiliated with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA. {arjavdesai, nmichael}@cmu.edu. We gratefully acknowledge support from industry.

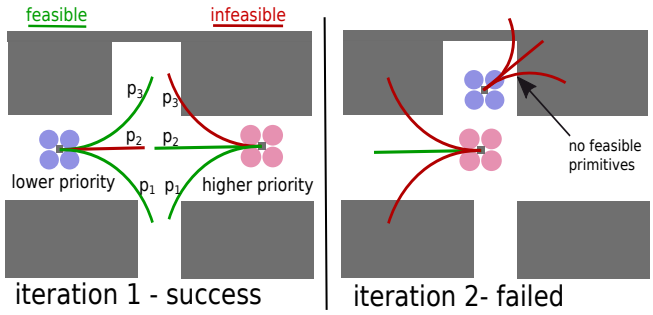

Fig. 2: Example of limitations of motion primitive based approaches in the context of coordination of differentially-constrained systems. Two robots need to swap positions. Lower priority robot has no feasible motion-primitives in the second iteration and cannot stay in position due to non-stationary underlying state i.e. the robot reaches an inevitable collision state.

refinement is needed for safe hardware execution [29]. Our approach eliminates dependence on refinement by leveraging *offline* reachability analysis on position invariant geometric trees (Sec. III-B) which guarantees a kinodynamically feasible and collision-free solution if a collision-free geometric solution is found.

- A collision checking method (Sec. III-C) which exploits the *invariance* of geometric trees to efficiently remove colliding edges (6×10^4 edges in the order of ms).
- Simulation results with teams of up-to 40 robots and hardware results with 5 robots in cluttered 3-D workspaces (Sec. IV).

II. Problem Formulation

A. Notation and Assumptions

$\mathbb{R}$ and $\mathbb{N}$ denote the set of real and natural numbers respectively. $\mathcal{S}$ represents a set and $\mathbf{M}$, $\mathbf{v}$, and s denote matrices, vectors, and scalars respectively. $|\mathcal{S}|$ refers to the cardinality of the set. The robots in this work are differentially flat [18], jerk-controlled [9] quadrotor systems. The 3-D workspace is static and known *a priori*.

B. Problem Statement

Let $\mathcal{W}$ be the set of points describing the 3-D workspace. The set of occupied points is given by $\mathcal{W}_{\text{occ}}$ and the set of free points is given by $\mathcal{W}_{\text{free}}$ i.e. $\mathcal{W} := \mathcal{W}_{\text{free}} \cup \mathcal{W}_{\text{occ}}$. The set of initial and final states are given by $\mathcal{I} \in \mathbb{R}^{p \times n}$ and $\mathcal{F} \in \mathbb{R}^{p \times n}$ respectively. Here, p denotes the dimensionality of the robot state and n represents the number of robots in the team. We seek to generate time-parameterized polynomial trajectories [18] for each robot in the team that are dynamically feasible with accelerations and jerks below specified limits, collision-free, and terminate within the goal region. Additional requirements include *scalability* (large teams of robots) and *responsiveness* (low online planning times) in cluttered 3-D workspaces.

III. Approach

This section is organized as follows:

- Sect. III-A discusses the environment representation.
- Sect. III-B, discusses the proposed approach for offline reachability analysis on position invariant geometric trees and the procedure for *online inference* of the k-step reachability of arbitrary initial states.
- Sect. III-C discusses the collision checking approach.
- Sect. III-D presents the single robot planning algorithm that is extended to the multi-robot case in III-E.

A. Environment Representation

We use a voxel grid representation of the environment. A sparse roadmap of the environment is computed during the offline preprocessing step using the `SPARS2` algorithm [8]. The sparse roadmap is used in the single (Sect. III-D) and multi-robot planners (Sect. III-E) for cost-to-go evaluation. The `SPARS2` algorithm is chosen due to its low memory requirements and competitive solution quality compared to the traditional roadmap based methods (`PRM` [3]).

B. Invariant Geometric Tree Reachability Analysis

Definition 1. *An **edge** e is defined as a line segment connecting two geometric points in $\mathbb{R}^3$. The initial and terminal points of the edge are represented by $e(0)$ and $e(1)$ respectively. The length of the edge is denoted by $l(e)$.*

Definition 2. *A **dynamically feasible edge** is defined as an edge such that for some initial higher-order derivative $\mathbf{s}_{init}$ at $e(0)$, there exists a non-empty set of higher-order derivatives $\mathcal{S}_{final}$ at $e(1)$ and the trajectories connecting $\mathbf{s}_{init}$ to $\mathcal{S}_{final}$ along e do not violate differential constraints.*

Definition 3. *A **curvature constrained edge** is defined as an edge such that for some initial higher-order derivative $\mathbf{s}_{init}$ at $e(0)$, there exists a non-empty set of higher-order derivatives $\mathcal{S}_{final}$ at $e(1)$ such that the trajectories connecting $\mathbf{s}_{init}$ to $\mathcal{S}_{final}$ along e 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.*

Definition 4. *A **reachable edge** is defined as an edge that is dynamically feasible and curvature constrained.*

Definition 5. *A **lattice** $\mathcal{E}$ is defined as a set of edges that start at the origin.*

Definition 6. *A **k-step tree** T is obtained by propagating a lattice in an obstacle-free workspace for k steps. Since each node in a tree has a single parent, the tree can be defined by the tuple $T = (\mathcal{V}_T, \mathbf{a}_T, \mathbf{c}_T)$, where $\mathcal{V}_T$ is the set of vertices in $\mathbb{R}^3$, $\mathbf{a}_T \in \{0,1\}^{|\mathcal{V}_T|}$ is the boolean valued adjacency (0 denotes unreachable) vector, and $\mathbf{c}_T \in \mathbb{R}^{|\mathcal{V}_T|}$ is the cost vector where $\mathbf{c}_T(i)$ is the the cost-to-come to the ith vertex.*

Definition 7. *A **reachable k-step tree** is defined as a tree such that for some initial higher-order derivative $\mathbf{s}_{init}$ at the root state, if $\mathbf{a}_T(i) = 1$, the path from the root to the ith vertex is composed of reachable edges.*

Proposition 1. *Let $\mathcal{S}$ be a distribution of higher-order derivatives at the root vertex of an invariant k-step tree given by $T = (V, \mathbf{a}, \mathbf{c})$. Let $T_{\mathbf{s}_i} = (V, \mathbf{a}_{\mathbf{s}_i}, \mathbf{c})$ represent the reachable k-step tree of $\mathbf{s}_i \in \mathcal{S}$. The reachable k-step tree*

*of the entire distribution, $T_{\mathcal{S}}$, is given by $T_{\mathcal{S}} = (V, \mathbf{a}_{\mathcal{S}}, \mathbf{c})$ where $\mathbf{a}_{\mathcal{S}} = \sum_{\forall i,\ bitwise} \mathbf{a}_{\mathbf{s}_i}$. We refer to this as the **merge-tree** operation. For the ith vertex, 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.*

Here, we discuss generation of reachable k-step trees for arbitrary states in the robot's state-space. Exploiting position invariance of geometric trees, we define the state-space of the robot to be the space of higher-order derivatives defined by the set $\mathcal{S}$ where each element of $\mathcal{S}$ is a six dimensional vector $[v_x, v_y, v_z, a_x, a_y, a_z]^{\mathrm{T}}$ Here, v denotes velocity and a denotes acceleration. Our approach relies on *offline* reachability analysis of (a) a finely discretized *axes decoupled* state space on axis aligned edges of various lengths (Sect. III-B.1) and, (b) coarsely discretized *axes coupled* state space on invariant geometric k-step trees (Sect. III-B.3).

1) Decoupled Reachability Analysis: Since trajectory generation for quadrotors is decoupled in the cardinal directions x, y, and z [9], we construct axis decoupled sets $\mathcal{S}_x$, $\mathcal{S}_y$, and $\mathcal{S}_z$, where each element of these sets is a two dimensional vector of velocity and acceleration in the respective cardinal direction. We sample edges of various lengths $l_i \in \mathcal{L}$ oriented along the positive x axis Fig. 3a.

We evaluate the reachability of the axis aligned edges in $\mathcal{L}$ (Def. 4) for all pairs of initial and final states drawn from $\mathcal{S}_x$, $\mathcal{S}_y$, and $\mathcal{S}_z$. If the time-parameterized, single axis polynomial trajectory resulting from a pair of sampled states is dynamically feasible and curvature constrained (Fig. 3b), the cost, i.e. total trajectory jerk as in [9], is stored in the cost matrix corresponding to the cardinal direction (cost is infinity if the edge is unreachable).

For each $l_i \in \mathcal{L}$, cost matrices $\mathbf{C}_x \in \mathbb{R}^{\mathcal{S}_x \times \mathcal{S}_x}$, $\mathbf{C}_y \in \mathbb{R}^{\mathcal{S}_y \times \mathcal{S}_y}$ and $\mathbf{C}_z \in \mathbb{R}^{\mathcal{S}_z \times \mathcal{S}_z}$ are stored. Note that reachability along the y, and z cardinal directions is independent of the edge length as illustrated in Fig. 3c. Thus, cost matrices $\mathbf{C}_y$, and $\mathbf{C}_z$ need to be computed only once.

The edges in $\mathcal{L}$ are oriented in 3-D space to create a geometric lattice (Def. 5) and this lattice is propagated for k-steps to construct a geometric tree, T.

2) Reachability Inference for Oriented Edges: The decoupled reachability information is used to infer the reachability of arbitrarily oriented edges *without explicit evaluation*. Given an edge e of length l at an arbitrary orientation in 3-D space, let $\mathbf{R}_{\text{ab}}, \mathbf{R}_{\text{ba}} \in \mathbb{R}^{3\times3}$ be the rotation matrices to align e with the positive x-axis and back, respectively. Further, let $s_{\text{init}} \in \mathbb{R}^6$ be the higher-order derivatives at $e(0)$. The reachability (Def. 4) of the edge can be inferred without explicit evaluation using the following procedure:

1) Rotate the velocity and acceleration components of $\mathbf{s}_{\text{init}}$ using $\mathbf{R}_{\text{ab}}$ and find the closest representative states in $\mathcal{S}_x$, $\mathcal{S}_y$, and $\mathcal{S}_z$.
2) Using the representative states, query the cost matrices corresponding to l to obtain the set of terminal states along each cardinal axis i.e. $\mathcal{S}_x^{\text{final}}$, $\mathcal{S}_y^{\text{final}}$, and $\mathcal{S}_z^{\text{final}}$
3) If $\mathcal{S}_x^{\text{final}}$, $\mathcal{S}_y^{\text{final}}$, and $\mathcal{S}_z^{\text{final}}$ are non-empty sets, the edge is deemed reachable (Def. 4).
4) The terminal states are obtained by taking a cartesian product $\mathcal{S}^{\text{final}} = \mathcal{S}_x^{\text{final}} \times \mathcal{S}_y^{\text{final}} \times \mathcal{S}_z^{\text{final}}$ and applying the $\mathbf{R}_{\text{ba}}$ rotation matrix to the velocity and acceleration components of $\mathcal{S}^{\text{final}}$.

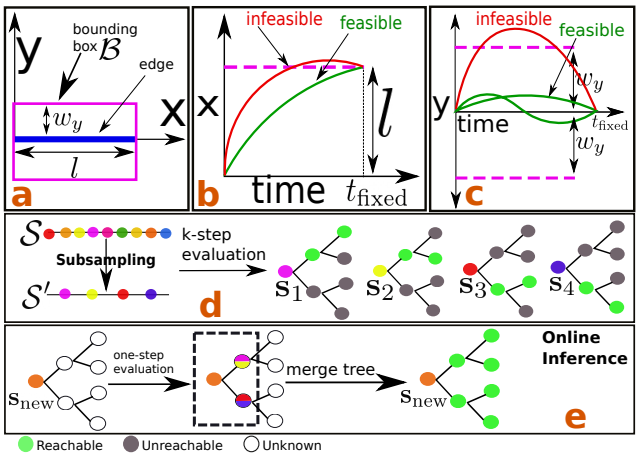

Fig. 3: (a) An edge of length l (in blue) aligned with the positive x axis. The pink rectangle represents the 2-D projection of the bounding box $\mathcal{B}$ of length l and width w_y. (b,c) Reachability evaluation for decoupled axes. (d) Offline coupled k-step evaluation of subsampled states and use in (e) online k-step tree inference for arbitrary states (e.g. $\mathbf{s}_{\text{new}}$) drawn from robot's feasible state space.

3) Coupled Reachability Analysis on k-Step Trees: Let $\mathcal{S}'$ be a set of higher-order derivatives constructed by taking a cartesian product of subsampled sets $\mathcal{S}'_x$, $\mathcal{S}'_y$, and $\mathcal{S}'_z$. We note that $|\mathcal{S}'_x \times \mathcal{S}'_y \times \mathcal{S}'_z| \ll |\mathcal{S}_x \times \mathcal{S}_y \times \mathcal{S}_z|$ (Fig. 3d). In practice, $\mathcal{S}'$ is constructed via uniform sampling of the velocity space (Sect. IV). The reachability of an *invariant* k-step tree (Def. 6) is evaluated for each $\mathbf{s}_i \in \mathcal{S}'$ using the decoupled reachability information in Sect. III-B.1 and the inference procedure described in Sect. III-B.2. The boolean valued adjacency vectors $\mathbf{a}_i$ (Def. 5) corresponding to each $\mathbf{s}_i \in \mathcal{S}'$ are precomputed and stored.

4) Online Inference of Reachable k-step Trees: Let $\mathbf{s}_{\text{new}} \in \mathbb{R}^6$) denote an arbitrary state in the robot's feasible state space, $\mathcal{S}$. The k-step reachability of $\mathbf{s}_{\text{new}}$ is inferred (Fig. 3e) using the following procedure:

1) If $\mathbf{s}_{\text{new}}$ lies in the set $\mathcal{S}'$ (constructed in Sect. III-B.3), reuse precomputed k-step tree.
2) If not, apply the reachability inference method in Sect. III-B.2 to identify the reachable edges and the corresponding terminal states in the one step lattice, $\mathcal{E}$, (Def. 5) used for constructing the k-step tree.
3) For each reachable edge, identify the subset of terminal states which lie in $\mathcal{S}'$ (defined in Sect. III-B.3) and merge the corresponding $k-1$ step trees using the merge-tree operation (Prop. 1).

The subsequent subsection describes the proposed collision checking procedure.

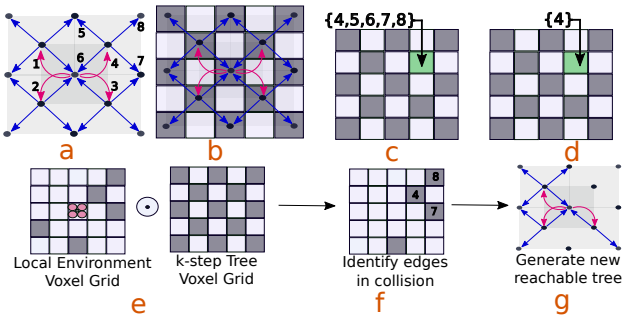

Fig. 4: (a) Cartesian projection of 2-step geometric tree where the pink and blue edges represent steps 1 and 2 respectively. (b) Voxel grid representation of tree. (c,d) Associating the minimal set of edges with each occupied voxel. (e) Element-wise multiplication and (f) identification of tree voxels in collision with the environment. (g) Removing conflict edges from the tree and generating true reachable set of tree vertices using Dijkstra's algorithm [7].

C. Collision Checking

Let $T_{\mathbf{s}} = (\mathcal{V}, \mathbf{a_s}, \mathbf{c})$ (Def. 6) denote the reachable k-step tree for initial higher-order derivatives $\mathbf{s}$, with the root vertex at position $\mathbf{p} \in \mathcal{W}_{\text{free}}$. While all reachable edges of $T_{\mathbf{s}}$ are dynamically feasible (Def. 2) and curvature constrained (Def. 3), some edges might be in collision with $\mathcal{W}_{\text{occ}}$. We seek to modify $\mathbf{a_s}$ to yield an adjacency vector which corresponds to a tree with dynamically feasible, curvature constrained, *and* collision-free edges.

1) Offline Processing of Geometric Trees: We exploit the invariance of the k-step tree to generate a voxel grid representation of the geometric tree with the same voxel resolution as that of the environment. Voxels that intersect with the edges of the tree and the bounding box $\mathcal{B}$ around the edge (as in Def. 3) are marked as occupied as shown in Fig. 4(a-d). Additionally, with each occupied voxel we associate the *minimal* set of edges that intersect with it, e.g., if edges leading to the ith and jth tree vertex intersect with a specific voxel and the jth vertex is a child of the ith vertex, only the ith vertex (or the edge leading to the ith vertex) is associated with the voxel. This voxel to edge mapping is stored as a dictionary data structure.

2) Online Collision Checking: A 3-D slice from the environment voxel grid centered around the position of the robot is extracted. The dimensions of this 3-D slice are the same as that of the the voxel-grid of the k-step tree. We perform an elementwise multiplication operation between the two grids and identify all voxels that are *both* in intersection with the geometric tree and occupied by the workspace obstacles. The minimal set of edges corresponding to each occupied voxel are queried (Fig. 4: e-g) from the dictionary data structure created in Sect. III-C.1 and the corresponding vertices are marked as unreachable in $\mathbf{a_s}$. The true set of all reachable vertices is obtained by running Dijkstra's algorithm [7] from the root of the tree. In multi-robot planning, a time dimension is added to the voxel grid of the invariant k-step tree for collision checking with geometric paths of other robots in the team.

D. Single-Robot Planner

The proposed single-robot planning strategy (Alg. 1) incrementally constructs a collision-free geometric path (Alg. 1, lines 8, 13) and maintains the distribution of possible higher-order derivatives at the intermediate geometric vertices in the form of a directed graph which we call the *derivative graph*, denoted $G_{\mathcal{S}}$ (Alg. 1, lines 9, 13). Once a geometric path is found to the goal, higher-order derivatives are assigned to the geometric vertices by searching for the shortest path in the *derivative graph* (Alg. 1, line 17). The shortest path in the derivative graph corresponds to an assignment of derivatives which minimizes the total trajectory jerk.

Algorithm 1 Find a feasible and safe trajectory ξ, given the initial and final positions, $\mathbf{p}_{\text{init}}$ and $\mathbf{p}_{\text{final}}$ and derivatives, $\mathbf{s}_{\text{init}}$ and $\mathbf{s}_{\text{final}}$ respectively.

```
1:  procedure SingleRobotPlanner(p_init, p_final, s_init, s_final)
2:      construct k-step tree, T from s_init and p_init (III-B.4)
3:      S_active ← s_init, P ← ∅, G_S ← ∅
4:      while iter < max_iter do
5:          incorporate workspace constraints in T using (III-C)
6:          goal_found ← is goal state (p_final, s_final) in T
7:          if goal_found then
8:              append k-step path, P_k , to goal to P
9:              update G_S by propagating S_active along P_k
10:             break
11:         select intermediate goal in T using roadmap
12:         append k-step path, P_k, to intermediate goal to P
13:         update G_S by propagating S_active along P_k
14:         update S_active with derivative set at P_k(k)
15:         construct T from S_active (Prop. 1), set p_init ← P_k(k)
16:     if goal_found then
17:         assign derivatives by finding shortest path in G_S
18:         return trajectory, ξ [9].
```

E. Multi-Robot Planner

A prioritized (hence, incomplete), cooperative planning strategy (similar to [25]) is used for the multi-robot case. In each iteration, a priority ordering $\mathcal{O}$ is computed for the robot team and k-step paths are computed sequentially for each robot in a manner similar to the single-robot planner described in Algorithm 1 (lines 5-16). Once collision-free geometric paths are found, derivatives are assigned to the geometric vertices and trajectories are computed for each robot in the team. We refer the reader to [6] for additional details on the single and multi-robot planners.

Algorithm 2 Find a feasible and safe polynomial trajectory set, Ξ, given the initial and final states $\mathcal{I}$ and $\mathcal{F}$ respectively.

```
1:  procedure MultiRobotPlanner(I, F)
2:      while iter < max_iter do
3:          O ← assign priority ordering
4:          for ∀i ∈ O do
5:              Algorithm 1: lines 5-15
6:      if all robots at goal then
7:          assign derivatives to each robot's geometric path
8:          return trajectory set, Ξ
```

IV. Evaluation

A. Implementation Details and Experiment Design

All algorithms are implemented in Julia [2] and evaluated on a Lenovo Thinkpad with a Intel 4-Core i7 CPU and 16 GB RAM. The proposed approach is evaluated by means of four studies. Studies use a geometric tree of depth 2 and 63253 edges. We use a subsampled set of higher-order derivatives $\mathcal{S}'$ (defined in Sect. III-B.3) with $|\mathcal{S}'| = 729$. For constructing $\mathcal{S}'$, we sample velocities along each cardinal direction (velocity limits $\{-2, 2\}$ m/s) with a resolution of 0.5 m/s. The acceleration along the three cardinal directions for these samples is set to 0 m/s^2. The offline reachability data structures (cost matrices, k-step trees of states in $\mathcal{S}'$) utilize 1.5 gigabytes of memory.

- Sect. IV-B experimentally verifies the dynamic feasibility and safety of our approach.
- Sect. IV-C compares our planning representation (invariant geometric trees) to motion-primitives using computation times, success rates, and solution quality as the comparison metrics.
- Sect. IV-D studies the effect of map size and obstacle density on the success rates and planning times of our approach and Sect. IV-E evaluates the computational complexity.
- Sect. IV-F verifies the viability of our approach for hardware execution.

B. Dynamic Feasibility and Safety

We conduct 20 multi-robot planning experiments (randomly generated start and goal states) for team sizes of 1, 10, 30, and 40 robots in a cluttered 3-D workspace (voxel resolution 0.25 m). For each experiment, if a set of collision-free *geometric* paths are found, we evaluate the resulting trajectories for violation of differential and collision constraints. The distribution of maximum accelerations and minimum clearance distances for each experiment are shown in Fig. 5. Results verify that the proposed approach generates motion plans that are dynamically feasible and collision-free.

C. Comparison with Motion-Primitive Based Approaches

We compare the performance of the our planning representation, i.e., invariant geometric trees, to motion-primitives, which are used in state-of-the art single-robot kinodynamic planning and replanning approaches like [17], [30]. 20 experiments are conducted for team sizes of 1, 10, 30, and 40 robots in the random forest environment like the one shown in Fig. 6. Success rate, computation times, and solution cost are used as the comparison metrics. We use the motion-primitive based single-robot planner as proposed in Zhou et al. [30] (with 729 primitives per node expansion) in our multi-robot formulation (Alg. 2) (abbreviated as `MP` in Table I). For a fair comparison, we do not rely on the `SPARS2` roadmap for cost-to-go evaluation for intermediate goal selection but instead use a heuristic function (Euclidean distance to goal). The results are summarized in Table I. The motion-primitive based approach outperforms our approach

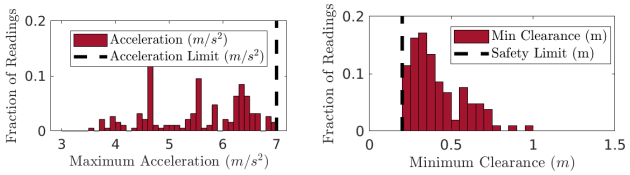

Fig. 5: Tests employ robots of radius 0.1 m and an acceleration limit of 7 m/s^2. Sub-figures show the distribution of maximum accelerations and minimum inter-robot clearance distances. Results verify the claim that if collision-free geometric paths are found, the resulting trajectories are dynamically feasible and collision-free

TABLE I: Success rate, computation time, and solution quality of our planning representation compared to motion-primitives (discretized control-input space) for various team sizes (n). Proposed representation achieves higher success rates and comparable computation times compared to motion-primitives.

# Robots		1		10		30		40	
Approach		Our	MP	Our	MP	Our	MP	Our	MP
Success %		**100**	**100**	**100**	70	**85**	25	**75**	20
Cost (m^2s^{-3})	**Mean**	28.8	**15.1**	269.5	**155.9**	807.2	**475.0**	1082.0	**626.9**
	Std. Dev	5.8	3.2	20.4	10.8	40.6	12.5	65.9	17.1
Time (s)	**Mean**	0.25	**0.1**	2.1	**1.5**	7.7	**6.3**	12.0	**10.6**
	Std. Dev	0.14	0.1	0.2	0.5	1.2	1.4	3.0	3.1

in the single-robot case. However, in the multi-robot case, our approach achieves significantly higher success rates. The computation times are comparable to that of `MP`. Our approach incurs a higher cost (approximately twice) than that of motion-primitive based approach since the trajectories are constrained to remain within spatial lattices. The high success rates of our approach is primarily attributed to availability of (1) multi-step geometric paths—preventing deadlocks and reciprocal dances common in myopic planning strategies—and (2) reasoning over the distribution of higher-order vertices at the intermediate geometric vertices (Prop. 1) that gives rise to diverse set of spatial options that is not possible in motion-primitive based approaches where candidate trajectories typically remain within a dynamically feasible cone (Fig. 2).

D. Effect of Map Size and Obstacle Density

We study the effect of map size and obstacle density on the success rate and computation times of the our approach. Tables II and III summarize the results. Increasing the map size (keeping obstacle density constant) primarily affects the computation times due to the increase in the number of planning iterations. Increasing the obstacle density decreases the success rate due to the availability of a fewer number of geometric edges during the geometric search step. However, it should be noted that the proposed approach is capable of successfully computing motion plans in high congestion scenarios (e.g. 75% success rate with 30 robots in a $10 \times 10 \times 10$ m^3 cluttered workspace).

E. Computational Complexity

The complexity of the approach is found to be greater than $\mathcal{O}(n)$ but less than $\mathcal{O}(n^2)$ for up-to 40 robots (Fig. 7). n refers to the team size. The scaling of computation times

per iteration for collision checking are shown in Table IV. Low planning times suggest viability of the approach for online use in application domains mentioned in Sect. I-A.

TABLE II: Effect of map size on the success rate and computation times of the proposed approach. Map size primarily affects the computation time due to increase in number of planning iterations.

	Small Size $10 \times 10 \times 10$ m^3		Medium Size $20 \times 10 \times 10$ m^3		Large Size $50 \times 10 \times 10$ m^3	
# Robots	Success %	Time (s)	Success %	Time (s)	Success %	Time (s)
1	100	0.1	100	0.2	100	0.6
10	100	0.6	95	1.4	95	5.5
30	75	2.9	75	7.5	70	20.1
40	40	4.8	50	11.1	55	37.6

TABLE III: Effect of obstacle density on the success rate and computation times. Obstacle density adversely affects the success rate due to fewer available geometric edges. The computation times increase since clutter forces robots to deviate from straight-line paths leading to an increase in the number of iterations.

	Low Density 8 Obstacles		Medium Density 16 Obstacles		High Density 24 Obstacles	
# Robots	Success %	Time (s)	Success %	Time (s)	Success %	Time (s)
1	100	0.2	100	0.3	100	0.3
10	95	1.4	100	2.5	90	2.6
30	75	7.5	70	8.0	70	11.3
40	75	11.1	45	13.1	40	18.28

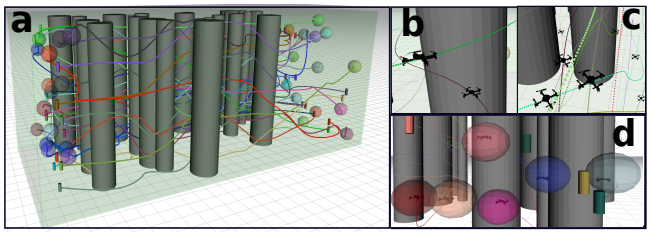

Fig. 6: (a) Final trajectories of 30 robots in a dense random forest environment (high density environment Tab. III). Colored cylinders represent start states, and spheres represent goal regions. (b) Robots avoiding static clutter in the environment. (c) Collision avoidance in high congestion regions. (d) Trajectories terminate in goal regions.

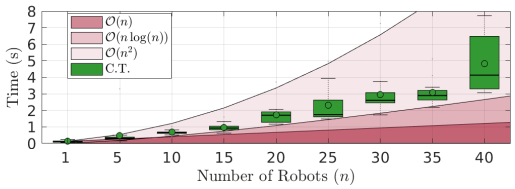

Fig. 7: Scaling of computation times with number of robots in a cluttered $10 \times 10 \times 10$ m^3 workspace. The computational complexity of the proposed approach is higher than $\mathcal{O}(n)$ but less than $\mathcal{O}(n^2)$.

TABLE IV: Scaling of computation time per iteration for collision checking components using approach proposed in Sect. III-C with team size n. Computation times reported in seconds. Tests use a tree with 63253 edges spanning a $12 \times 12 \times 12$ m^3 cubic volume centered at the robot position. Collision checking with static obstacles (voxel resolution 0.25 m) scales linearly with n. Inter-robot collision checking scales (approx.) quadratically with n.

Component	$n = \mathbf{1}$	$n = \mathbf{10}$	$n = \mathbf{30}$	$n = \mathbf{40}$
Collision check (static)	$0.007 \pm .002$	0.09 ± 0.008	0.29 ± 0.039	0.40 ± 0.069
Collision check (inter-robot)	-	0.38 ± 0.093	2.17 ± 1.34	3.51 ± 1.63

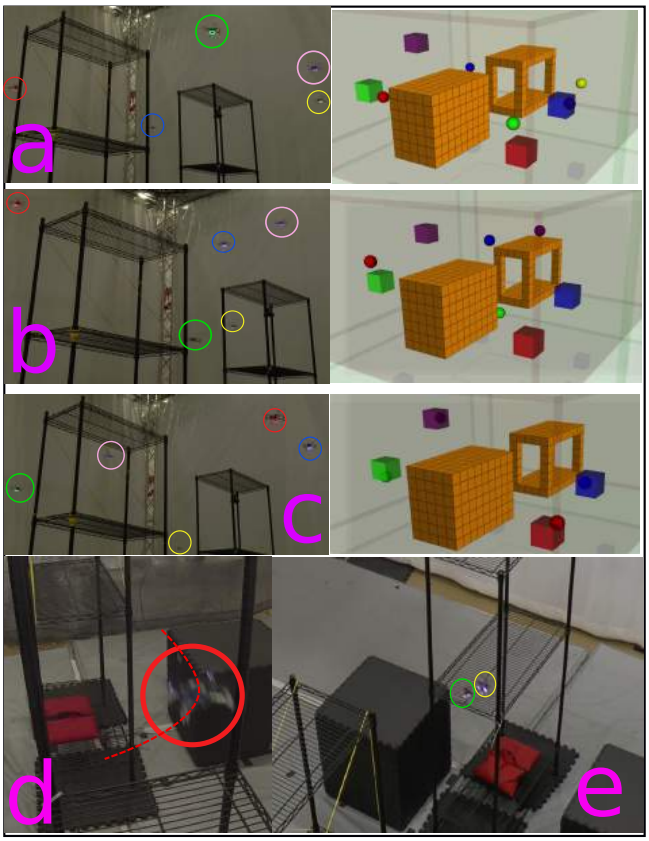

Fig. 8: Online planning with 5 crazyflie quadrotors in a cluttered 3-D environment in response to online changes in goal locations. The goal locations are selected to be random points in $\mathcal{W}_{\text{free}}$. Subfigures **a**, **b**, and **c** show one online planning trial. Cubes indicate goal locations and spheres indicate the current robot positions. 10 online planning trials were conducted. Across the 10 trials, the maximum acceleration was found to be 5 m/s^2 and minimum clearance distance was found to be 0.22 m thus satisfying constraints of 7 m/s^2 (acceleration) and 0.2 m (clearance). Subfigure **d** shows paths being computed through tight spaces and subfigure **e** shows robots operating in close proximity while avoiding collisions.

F. Hardware Results

We evaluate our approach with 5 crazyflie quadrotors in a cluttered 3-D workspace in a motion capture arena. Results (Fig. 8) verify that proposed approach generates plans that can be safely executed by differentially constrained systems.

V. Conclusion and Future Work

This work presented an approach for kinodynamic multi-robot trajectory planning and coordination in 3-D workspaces via leveraging offline reachability analysis on invariant geometric trees. Low planning times (approximately 1 second for 10 robots) and high success rates in high congestion scenarios suggest viability of the proposed approach in centralized online planning frameworks for critical real-world applications such as building clearance (Fig 1). As future work, we intend to use this approach within a distributed framework with additional considerations such as state uncertainty and communication delays. Additionally, we intend to extend the planning approach to consider dynamic workspaces.

References

[1] J. L. Baxter, E. K. Burke, J. M. Garibaldi, and M. Norman. *Multi-Robot Search and Rescue: A Potential Field Based Approach*, pages 9–16. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[2] Jeff Bezanson, Stefan Karpinski, Viral B Shah, and Alan Edelman. Julia: A fast dynamic language for technical computing. *arXiv preprint arXiv:1209.5145*, 2012.

[3] Valérie Boor, Mark H Overmars, and A Frank Van Der Stappen. The gaussian sampling strategy for probabilistic roadmap planners. In *ICRA*, pages 1018–1023, 1999.

[4] Marcello Cirillo, Tansel Uras, and Sven Koenig. A lattice-based approach to multi-robot motion planning for non-holonomic vehicles. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 232–239. IEEE, 2014.

[5] Liron Cohen, Tansel Uras, TK Satish Kumar, Hong Xu, Nora Ayanian, and Sven Koenig. Improved solvers for bounded-suboptimal multi-agent path finding. In *IJCAI*, pages 3067–3074, 2016.

[6] 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.

[7] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[8] Andrew Dobson and Kostas E Bekris. Improving sparse roadmap spanners. In *2013 IEEE International Conference on Robotics and Automation*, pages 4106–4111. IEEE, 2013.

[9] Markus Hehn and Raffaello D'Andrea. Quadrocopter trajectory generation and control. *IFAC proceedings Volumes*, 44(1):1485–1491, 2011.

[10] Wolfgang Hönig, TK Satish Kumar, Liron Cohen, Hang Ma, Hong Xu, Nora Ayanian, and Sven Koenig. Multi-agent path finding with kinematic constraints. In *ICAPS*, pages 477–485, 2016.

[11] Wolfgang Hönig, James A Preiss, TK Satish Kumar, Gaurav S Sukhatme, and Nora Ayanian. Trajectory planning for quadrotor swarms. *IEEE Transactions on Robotics*, 34(4):856–869, 2018.

[12] Sertac Karaman, Matthew R Walter, Alejandro Perez, Emilio Frazzoli, and Seth Teller. Anytime motion planning using the rrt. Institute of Electrical and Electronics Engineers, 2011.

[13] Tushar Kusnur, Shohin Mukherjee, Dhruv Mauria Saxena, Tomoya Fukami, Takayuki Koyama, Oren Salzman, and Maxim Likhachev. A planning framework for persistent, multi-uav coverage with global deconfliction. *arXiv preprint arXiv:1908.09236*, 2019.

[14] Duong Le and Erion Plaku. Multi-robot motion planning with dynamics guided by multi-agent search. In *IJCAI*, pages 5314–5318, 2018.

[15] Duong Le and Erion Plaku. Multi-robot motion planning with dynamics via coordinated sampling-based expansion guided by multi-agent search. *IEEE Robotics and Automation Letters*, 4(2):1868–1875, 2019.

[16] Quentin Lindsey, Daniel Mellinger, and Vijay Kumar. Construction of cubic structures with quadrotor teams. *Proc. Robotics: Science & Systems VII*, 2011.

[17] Sikang Liu, Nikolay Atanasov, Kartik Mohta, and Vijay Kumar. Search-based motion planning for quadrotors using linear quadratic minimum time control. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 2872–2879. IEEE, 2017.

[18] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2520–2525. IEEE, 2011.

[19] Daniel Mellinger, Alex Kushleyev, and Vijay Kumar. Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 477–483. IEEE, 2012.

[20] Derek Mitchell and Nathan Michael. Persistent multi-robot mapping in an uncertain environment. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 4552–4558. IEEE, 2019.

[21] Mihail Pivtoraiko, Ross A Knepper, and Alonzo Kelly. Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics*, 26(3):308–333, 2009.

[22] James A Preiss, Wolfgang Hönig, Nora Ayanian, and Gaurav S Sukhatme. Downwash-aware trajectory planning for large quadrotor teams. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 250–257. IEEE, 2017.

[23] Cyril Robin and Simon Lacroix. Multi-robot target detection and tracking: taxonomy and survey. *Autonomous Robots*, 40(4):729–760, 2016.

[24] D Reed Robinson, Robert T Mar, Katia Estabridis, and Gary Hewer. An efficient algorithm for optimal trajectory generation for heterogeneous multi-agent systems in non-convex environments. *IEEE Robotics and Automation Letters*, 3(2):1215–1222, 2018.

[25] David Silver. Cooperative pathfinding. *AIIDE*, 1:117–122, 2005.

[26] Kiril Solovey and Dan Halperin. On the hardness of unlabeled multi-robot motion planning. *The International Journal of Robotics Research*, 35(14):1750–1759, 2016.

[27] Kiril Solovey, Oren Salzman, and Dan Halperin. Finding a needle in an exponential haystack: Discrete rrt for exploration of implicit roadmaps in multi-robot motion planning. In *Algorithmic Foundations of Robotics XI*, pages 591–607. Springer, 2015.

[28] Yulun Tian, Katherine Liu, Kyel Ok, Loc Tran, Danette Allen, Nicholas Roy, and Jonathan P How. Search and rescue under the forest canopy using multiple uavs. *arXiv preprint arXiv:1908.10541*, 2019.

[29] 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.

[30] Boyu Zhou, Fei Gao, Luqi Wang, Chuhao Liu, and Shaojie Shen. Robust and efficient quadrotor trajectory generation for fast autonomous flight. *IEEE Robotics and Automation Letters*, 4(4):3529–3536, 2019.