Incremental Path Planning Algorithm via Topological Mapping with Metric Gluing

Aakriti Upadhyay[1], Boris Goldfarb[2] and Chinwe Ekenna[1]

***Abstract*— We present an incremental topology-based motion planner that, while planning paths in the configuration space, performs metric gluing on the constructed Vietoris-Rips simplicial complex of each sub-space (voxel). By incrementally capturing topological and geometric information in batches of voxel graphs, our algorithm avoids the time overhead of analyzing the properties of the entire configuration space. We theoretically prove in this paper that the simplices of all voxel graphs joined together are homotopy-equivalent to the union of the simplices in the configuration space. Experiments were carried out in seven different environments using various robots, including the articulated linkage robot, the Kuka YouBot, and the PR2 robot. In all environments, the results show that our algorithm achieves better convergence for path cost and computation time with a memory-efficient roadmap than state-of-the-art methods.**

I. Introduction

Sampling-based algorithms have shown tremendous success in solving complex high-dimensional robot motion planning problems with probabilistically complete guarantees, i.e., the planners guarantee to find a solution if one exists as the number of samples reaches infinity. These algorithms achieve asymptotic optimality by incrementally sampling the robot's configuration space to continually improve the shortest path in an anytime manner [1]. The two widely used sampling-based paradigms are Probabilistic RoadMaps (PRMs) [2] and Rapidly-exploring Random Trees (RRTs) [3]. PRM computes a roadmap by randomly generating configurations via a graph search to locally connect to nearby configurations, whereas RRT locally explores the space by expanding one or more trees in random directions until it either finds a path or reaches a computation limit. While such algorithms often compute a feasible path quickly, a roadmap generated by them does not optimize the sub-space information of the configuration space for self-reuse or as input to other planners. Despite the progress made by these planners to solve high-dimensional problems, little effort devotes to incrementally processing the sub-space information of the configuration space by capturing the topological or geometric features.

In our work [4], and [5], we combined the Topological Data Analysis (TDA) approach, i.e., Vietoris-Rips (VR) complex and discrete Morse theory, with sampling-based algorithms to construct a planner that extracts the topological and geometric representations of the configuration space for memory-efficient path planning. The methods generated near-optimal paths using only the extracted information from our roadmap. The work also showed the reusable capability of our roadmap by any PRM or RRT variants.

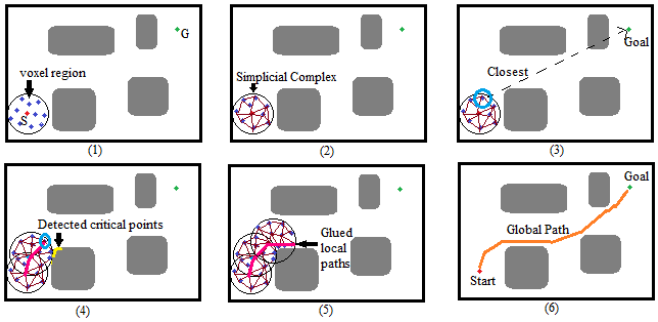


Fig. 1: A graph illustration of our algorithm in $\mathbb{R}^2$ space. (1) Our method first constructs the boundary of a sub-region with the start configuration as the center, and within this local voxel, the region generates collision-free configurations. (2) It uses topological collapses to build a skeleton of simplicial complexes from which topological information gets extracted. The part of the obstacles covered under each voxel polytope gets considered for geometric information extraction without pruning them from the sub-space, thus, preserving the topological significance with no information loss. (3) We call a distance metric method to find the closest configuration to the goal and use it as the center for the next voxel polytope. (4) The yellow bar on the obstacle indicates the detected critical points that identify configurations in its vicinity. (5) It plans a local path in each voxel graph and glues/joins with the local path of the next extending voxel polytope. These voxels get created until either the goal or a computational limit is reached. (6) Finally, it returns a global path that connects the start and goal configurations.

Contribution: In this work, we present a novel algorithm that incrementally glues the simplices of partially observable space while planning routes in the configuration space. We denote each sub-region as a voxel and theoretically/empirically prove that the VR complexes of all voxel graphs glued together are homotopy equivalent to the union of the VR complexes of the planned space. Our algorithm provides a sub-optimal solution by generating a near-optimal local path in each voxel and gluing them together to find a global route from the start to the goal positions of the robot. In this way, our algorithm incrementally applies the topological data analysis approach to each voxel block by preserving only necessary configurations and edges representing the unique property of the sub-space. This work contributes:

- An algorithm that creates voxel polytope [6] to represent the underlying sub-space of the simplicial complex.
- A topology-based path planner that incrementally extracts the topological and geometric information of the partially observable space for memory-efficient path planning.
- A proof of the geometric feasibility of gluing voxel graphs in different dimensions of configuration space.

[1]A.Upadhyay and [1]C.Ekenna are with the Department of Computer Science and [2]B.Goldfarb is with the Department of Mathematics and Statistics, University at Albany, SUNY. {aupadhyay,bgoldfarb,cekenna}@albany.edu

Our proposed algorithm achieves a better path cost and computation time convergence with a memory-efficient roadmap compared to other methods in all environments, as discussed in Section VI.

II. Related Work

In this section, we present the preliminary definitions for motion planning and discuss significant prior work relevant to our proposed algorithm.

A. Motion Planning Primitives

A robot's placement, or configuration, can be uniquely described by a point $(x_1, x_2, ..., x_n)$ in an n dimensional space (x_i being the i^{th} degrees of freedom (DOF)). The space consisting of all possible robot configurations (feasible or not) is called configuration space ($\mathcal{C}_{space}$) [7]. The subset of all feasible configurations is the free space ($\mathcal{C}_{\text{free}}$), while the union of the unfeasible configurations is the obstacle space ($\mathcal{C}_{obst}$). $\mathcal{C}_{space}$ defines the planning space, consisting of the constraint-free region of a robot, i.e., $\mathcal{C}_{\text{free}}$ space, and the set of obstacles O (or $\mathcal{C}_{obst}$) in it. Given an initial configuration $s \in \mathcal{C}_{\text{free}}$ and a goal configuration $g \in \mathcal{C}_{\text{free}}$, the motion planning problem thus becomes finding a continuous valid path (e.g., collision-free and satisfying all joint limit and/or loop closure constraints) in $\mathcal{C}_{\text{free}}$ connecting them [1].

B. Sampling-based Motion Planners

Research work exists that combines the strengths of PRMs and RRTs into a planner to solve diverse motion planning problems. Sampling-based Roadmap of Trees (SRT) [8] combined PRM and RRT methodologies by growing RRTs from configurations generated by PRM. A motivation behind SRT was to exploit parallel processing. Similarly, SparkPRM [9] employed PRM to quickly cover large areas of the planning space while RRT locally explores narrow passages. We utilize a similar concept to incrementally compute our local topology roadmap by using the logic of PRM to sample configurations within each voxel block and RRT to connect/extend these blocks in this work.

Some other approaches utilized workspace decomposition to find narrow or difficult areas of the workspace to bias $\mathcal{C}_{space}$ sampling. Yershova et al. in [10] proposed a sampling framework- Dynamic-Domain RRT, which grows Voronoi-based regions in the dynamic environment. It biases the random node selection to be within a radius r of q_{near} to enable expansion. The radius is determined dynamically from the failed expansion attempts. Improving on this approach, Denny et al. in [11] proposed Dynamic Region-biased RRT, which biases growth based on dynamically moving regions using Reeb graphs. The algorithm performed workspace decomposition and presented the $\mathcal{C}_{obst}$ into holes in an embedded graph, which restricted its usage in non-uniform environments. An extension to the approach, [12] combined dynamic region sampling with PRM to connect multiple connected components for effectual solutions with complete coverage. While these methods reduce the total configurations required to compute a solution, they perform computationally-intensive geometric tests to accept samples into the roadmap, resulting in large planning times.

C. Incremental Motion Planners

Sampling-based algorithms for optimal motion planning sample the configuration space incrementally to determine shorter paths in an anytime manner [13], [14]. Karaman and Frazzoli proposed k-PRM* in [15] as an asymptotically-optimal variant of PRM that, with enough computation time, produces an optimal graph with probability 1. Another work in [16] proposed the LazyPRM* strategy to reduce the computational cost of collision checking on achieving an optimal solution. The method avoids checking edges with no chance of improving upon the current best path, thereby reducing the per-sample computational cost and accelerating convergence. Ichnowski et al. in [17] applied an incremental approach to refine the shortest path obtained from the multilevel-sparse roadmap for generating an asymptotically-optimal solution. A more recent work in [18] presented Guided Incremental Local Densification (GuILD), an incremental densification framework that leverages partial search information to focus sampling within the informed set. GuILD leverages the search tree to adapt the Local Subsets and converges to the optimal path with fewer samples. Similarly, in this work, we apply incremental planning to compute shorter paths for each sub-region while taking advantage of extracted topological and geometric information that becomes crucial in connecting feasible edges closer to the $\mathcal{C}_{obst}$.

III. Preliminaries

This section discusses some important mathematical concepts used in our proposed algorithm.

A. Space approximation using Vietoris-Rips complex

In our previous work [4], we applied the Vietoris-Rips complex to perform a memory-efficient path planning in a given $\mathcal{C}_{space}$ by generating a homotopy-equivalent topological map of the $\mathcal{C}_{\text{free}}$. A general definition of the Vietoris-Rips complex reads as below.

Definition 1: (Vietoris-Rips complex) Given a set S of points in a Euclidean space E, the Vietoris-Rips complex $R(S;\varepsilon)$ is the abstract simplicial complex whose k-simplices are the subsets of $k+1$ points in S of diameter at most ε.

In this prior work, a simplicial collapse removed redundant information to provide a space approximation of $\mathcal{C}_{\text{free}}$ as a pre-processing step.

B. Discrete Morse function in $\mathcal{C}_{space}$

We applied discrete Morse theory to the same simplicial complex to identify critical points on the boundary of $\mathcal{C}_{obst}$ and make the following formal definition.

Definition 2: Let D be the Euclidean distance function that measures the distance between the point $x \in \mathcal{C}_{\text{free}}$ and the nearest point y on the closest obstacle $O_i \in \mathcal{C}_{obst}$, that is, $D(x) = \min_{y \in O_i} \|x - y\|$.

Definition 3: Let $\Gamma(y, \varrho)$ be a density function where $\varrho > 0$ and y is the point on the obstacle. The function Γ counts all neighbors close to y in $R(S)$ within distance ϱ.

Definition 4: Let f be a discrete Morse function on $R(S)$ restricted to the vertices of the Vietoris-Rips complex. In our case, this is also the restriction of the Morse function formally defined at any point in $\mathcal{C}_{space}$ by

$$f(x) = D(x) \cdot \Gamma(y, \varrho). \tag{1}$$

Here, x represents a configuration in $R(S)$, y refers to a point on the closest obstacle to x, and ϱ is a constant. We formally define identified critical points and feasible critical points as below.

Definition 5: (Critical points) The set of critical points is defined as the set of non-degenerate points on the convex hull of each obstacle in $\mathcal{C}_{obst}$ when the given discrete Morse function f reaches its extreme values, i.e., local minima or maxima.

Definition 6: (Feasible critical points) This set is defined as all vertices in $R(S)$ at a radial clearance of ϱ from a critical point of $\mathcal{C}_{obst}$. In other words, it is the union of intersections of vertices in $R(S)$ within the metric balls of radius ϱ centered at some critical point.

The feasible critical points computation is independent of a specific calculation of ϱ, so the value of ϱ can be of any choice. Here, we calculate the value of ϱ defined in [5].

IV. Methodology

We propose an incremental topology path planner that uses the concept of voxel polyhedron [19] to consider partially observable $\mathcal{C}_{space}$ into sub-regions and perform metric gluing to explore and cover the underlying space.

A. Voxel Polytope

Different from existing incremental work on roadmap construction, whose incremental block is in the form of samples, our proposed incremental approach is in batches, and our blocks are in the form of simplices. The batched step-by-step process generates local roadmaps of the subspace of $\mathcal{C}_{\text{free}}$ decomposed as voxel polytope. The increase in the number of voxels glued together is proportional to the increase in the $\mathcal{C}_{\text{free}}$ area covered by the robot resulting in a higher probability of finding a solution if one exists. For each voxel block/window, we consider the spherical boundary of radius λ given as

$$\lambda = n * d, \tag{2}$$

where n is the dimensionality of $\mathcal{C}_{space}$ and d is the diameter of the circumscribed circle of the robot. So, the length of the global path becomes equal to the total number of glued voxel blocks times λ, i.e., gluing the local paths from each connecting voxel. The connection between the configurations in each voxel window is formed within the connection length γ. We compute the value of γ as

$$\gamma = min(\log(|V_i|)/|V_i|, \eta); i \geq 0, \tag{3}$$

where $|V|$ is the number of the vertices generated in each voxel window i to meet the sampling condition from [4]. η is the constant value provided by the user. Here, the boundary of each voxel window gets considered as the criteria for satisfying the sampling condition. In the subsequent section, we discuss the theoretical proof of metric gluing our local topology maps together.

B. Metric gluing of simplicial complexes

The work [20] provides mathematical proof showing that the Vietoris-Rips complex of two metric graphs glued together along a sufficiently small sub-complex is homotopy equivalent to the union of the Vietoris-Rips complexes.

In general, the term metric space refers to a set X equipped with a distance function $m\colon X^2 \to X$ satisfying the customary axioms: non-negativity, symmetry, the triangle inequality, and the property that the value 0 is achieved exactly on the pairs of the type (x, x). Let us assume two metric spaces X_1 and X_2 with a set-theoretic intersection $X_1 \bigcap X_2$. We say the intersection is a well-defined on metric space if the two distance functions m_1 and m_2 agree on $(X_1 \bigcap X_2)^2$. Notice that in general, even in this case, the distance function on the set-theoretic union $X_1 \cup X_2$ is not well-defined in a natural way, and so, there can be more than one metric on $X_1 \cup X_2$ which restricts to the given metrics on X_1 and X_2, but if X_1 and X_2 are subsets of a larger metric space X and the metrics are the restrictions of an ambient metric m on X then we can conclude that m gives a unique natural extension of both m_1 and m_2 to the restriction of m on $X_1 \bigcup X_2$. When we take another perspective where m_1 and m_2 are path metrics as usually defined in graphs, the path metric on the union of graphs is, in fact, a well-defined extension. Accordingly, our metric spaces are either sub-regions of $\mathcal{C}_{\text{free}}$ or sub-graphs of the ambient graph obtained as the 1-dimensional skeleton of the Vietoris-Rips complex from samples on $\mathcal{C}_{\text{free}}$.

A $\mathcal{C}_{space}$ is a topological space, so we apply the properties of metric gluing in it to join the simplices of our voxel graphs. Suppose D_A and D_B represent the simplicial complex obtained after the collapse on metric spaces A and B respectively, and H is a sub-complex on the vertices common to both A and B metric spaces. The function $diam$ refers to the circle diameter that circumscribes the metric space. For a subset Q of a metric space, we define the diameter $diam(Q)$ as the supremum of values $m(x, y)$ over all choices of x, $y \in Q$. Theorem 8, as stated in [20] by Adamaszek et al., reads as follows.

Theorem 1: Let A and B be metric spaces with $A \bigcap B = H$, where H is a closed subspace of A and B , and let $r > 0$. Consider $A \bigcup_H B$, the metric gluing of A and B along the intersection H. Suppose that if $diam(D_A \bigcup D_B) \leq r$ for some $\Phi \neq D_A \subseteq A \backslash H$ and $\Phi \neq D_B \subseteq B \backslash H$, then there is a unique maximal nonempty subset $\sigma \subseteq H$ such that $diam(D_A \bigcup D_B \bigcup \sigma) \leq r$. Then $R(A \bigcup_H B; r) \simeq R(A; r) \bigcup_{R(H;r)} R(B; r)$. Hence if $R(H; r)$ is contractible, then $R(A \bigcup_H B; r) \simeq R(A; r) \vee R(B; r)$.

The theorem proves that gluing Vietoris-Rips complexes of finite metric spaces A and B provides a simplex homotopy equivalent to the Vietoris-Rips complex of the $A \bigcup B$ metric space glued along the closed subspace H. A conclusion from corollary 9 in [20] is as follows.

Corollary 1: Let A and B be metric spaces with $A \bigcap B = H$, where H is a closed subspace of A and B, and $A \bigcup_H B$ is their metric gluing along H. Let $r > 0$, and suppose $diam(H) \leq r$. Then $R(A \bigcup_H B; r) \simeq R(A; r) \vee R(B; r)$.

Further, the conclusion from theorem 1 extends to show for metric graphs, from theorem 10 in [20].

Theorem 2: Let $G = G_A \bigcup_{G_H} G_B$ be a metric graph, where $G_H = G_A \bigcap G_B$ is a closed metric subgraph of the metric graphs G_A and G_B. Suppose furthermore that G_H is a path, and that each vertex of G_H besides the two endpoints has degree 2 not only as a vertex in G_H, but also as a vertex in G. Suppose the length of G_H is at most $l/3$, where any

simple loop in G that goes through G_H has length at least l. Let $A \subseteq G_A$ and $B \subseteq G_B$ be arbitrary subsets such that $A \subseteq G_A = B \subseteq G_B = A \bigcap B := H$. Then $R(A \bigcup_H B; r) \simeq R(A; r) \bigcup_{R(H;r)} R(B; r)$ for all $r > 0$. Hence if $R(H; r)$ is contractible, then $R(A \bigcup_H B; r) \simeq R(A; r) \vee R(B; r)$.

In this work, we extend this theory to high-dimensional spaces with paths greater than degree 2, i.e., in $\mathcal{C}_{space}$.

Proposition 1: Let G_A and G_B be metric/voxel graphs with $G_A \bigcap G_B = G_H$, where G_H is a closed sub-graph of G_A and G_B with vertices of degree greater than 2, and $G_v = G_A \bigcup_{G_H} G_B$ is the metric gluing of two metric graphs along G_H. Let $r \geq 2 * \lambda > 0$, the voxel radius from Eq. 2, and suppose the $diam(G_H) \leq r$. Then $R(A \bigcup_H B; r) \simeq R(A; r) \vee R(B; r)$.

Proof: Let G_v, G_A, G_B, G_H, A, B, and H satisfy the same hypotheses as in the statement of Theorem 2. Let z be a vertex in G_A that extends to the voxel graph G_B, such that $z \in G_A \bigcap G_B$ and voxel B is centered at z. We claim that no point $p \in G_v \setminus G_H$ is within distance r of G_H. Indeed, if there were such a point $p \in G_v \setminus G_H$ satisfying $d(p, u) \leq r$ and $d(p, v) \leq r$ where d is the Euclidean distance function and $u, v \in G_H$, then we could produce a homotopically non-trivial loop through the gluing simplex in G_H with local path shorter than λ, giving contradiction to our assumption. It follows that if the maximal non-empty set of G_A and G_B has $diam(\sigma_A \bigcap \sigma_B) \leq r$. Then there exists a unique maximal clique σ_H connecting the simplicial complexes $R(A)$ and $R(B)$. Hence, from theorem 2 we can conclude that $R(A \bigcup_H B; r) \simeq R(A; r) \vee R(B; r)$ can extend for configurations of degree greater than 2. Figure 2 shows an example of gluing two voxel complexes. ■

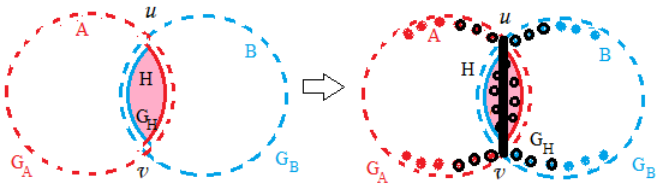


Fig. 2: Illustration of voxel graph gluing for G_A and G_B

Our work utilizes the metric gluing property of the VR complex to connect local paths contributing toward the formation of a global pathway from the start to the goal position. The incremental approach first constructs the local VR complex that is topologically equivalent to a sampled sub-space and then glues different pieces together to cover larger sections of the sampled $\mathcal{C}_{\text{free}}$ space.

C. Incremental Path Planner

Algorithm 1 creates voxel polytopes with each extending configuration taken as its center until the robot reaches/connects the goal configuration. The bounding radius of the voxel gets computed from Eq. 2. At each voxel, the algorithm generates a topology map of the local region from steps in [4], [5] and plans the local path from start to an intermediate configuration, and so forth until it connects to the goal configuration. It constructs the VR complex using method *ConstructComplex()* and perform simplicial collapse to obtain 1-dimensional skeleton of simplex using *TopologicalCollapse()* method. The generated local topological map is used to extract critical points information and the configurations (or feasible critical points) at proximity to $\mathcal{C}_{obst}$ using methods *IdentifyCriticalPoints()* and *GetValidConfigurations()*, respectively. In line 9, the algorithm considers the vertices of the simplicial complex and feasible critical points to plan a memory-efficient route at a ϱ-clearance from obstacles in method *PlanPath()*.

Algorithm 1 Incremental Path Planner

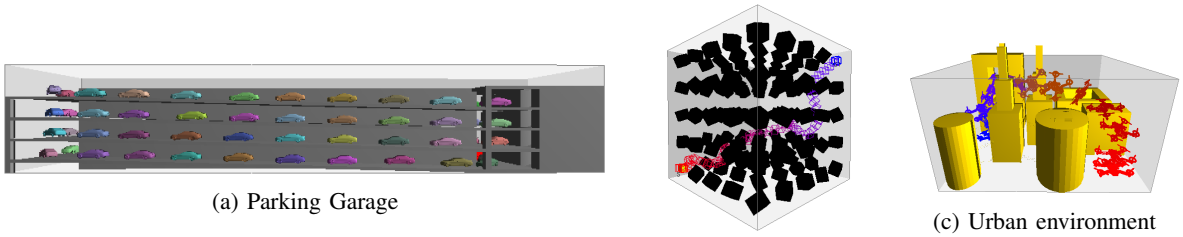
Input: G: A dense graph comprising of vertex set V and edge set E where $G = \{V, E\}$, Q: A query to be solved from a start to a goal position, S: Simplicial Complex, Z: Voxel Polytope, λ: Radius of voxel polytope, p: Local path set, P: Global path set, M: Glued voxel graphs, N: Nodes set.

1: Let $P \leftarrow null$, $p \leftarrow null$, q_0 = start configuration.
2: Create a voxel polytope Z(q_0, λ).
3: **while** Q not solved **do**
4: **if** Boundary(Z) not in collision with the robot **then**
5: $S \leftarrow ConstructComplex(G, Z)$; ◁ Refer [4]
6: $TopologicalCollapse(S)$; ◁ Refer [4]
7: $C \leftarrow IdentifyCriticalPoints(S)$; ◁ Refer [5]
8: $F \leftarrow GetValidConfigurations(S, C)$; ◁ Refer [5]
9: p = PlanPath(Z, S, F)
10: **if** goal $\notin$ P **then**
11: $N = ClosestToGoal(ConvexHull(S))$
12: **while** N is not empty **do**
13: $q = SelectRandomNode(N)$
14: **if** q expands Z and $q \neq q_0$ **then**
15: Create a voxel polytope $Z_q(q, \lambda)$.
16: **else**
17: $N = N \setminus q$
18: Z $\leftarrow Z_q$
19: $q_0 = q$
20: $Z_q \leftarrow null$
21: **else**
22: $\lambda = 2 * \lambda$
23: Create a voxel polytope Z(q_0, λ).
24: $P = P \bigcup p$
25: $M = M \bigcup S$
26: **return** $\{P, M\}$

The algorithm extends voxels by considering nodes from the convex hull of S of the immediate voxel polytope and narrows its selection to only nodes in the semicircle of the convex hull that are facing or are closest to the goal position, in line 11. It randomly selects a node from N for expansion until it succeeds and discards failed configurations to avoid redundancy, lines 12-17. It keeps performing these steps for each extended voxel and incrementally glues simplices or the planned local path of the voxel until the query is solved or it reaches a computation limit. As a result, the algorithm returns a global pathway P connecting the start and goal configurations and the roadmap M.

V. Experimental Setup

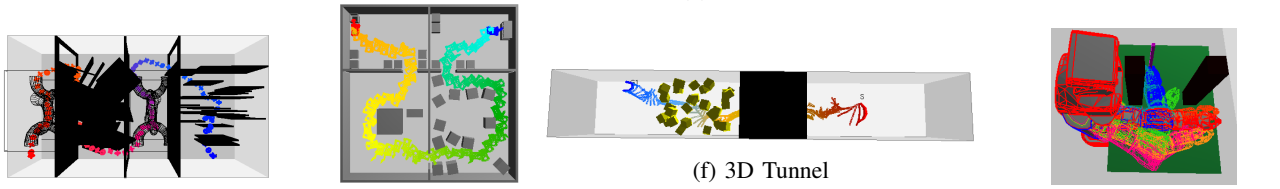
We executed experiments on a Dell Optiplex 7040 desktop machine running OpenSUSE operating system and implemented algorithms using the C++ motion planning library [21]. We used the RAPID [22] collision detection method during the sampling, connection, and query stages and used the brute force k-closest neighbor finding technique [23], the Euclidean distance metric method, and a straight line local planner for sampling and connection stages. We perform experiments in seven different environments with robots ranging from 3 DOF to 14 DOF, as discussion follows.



(a) Parking Garage

(b) Cluttered environment

(c) Urban environment



(d) Heterogeneous 3D

(e) Kuka YouBot environment

(f) 3D Tunnel

(g) PR2 robot environment

Fig. 3: Environments Studied

- **Parking Garage**: The 3D environment has a vehicle parking garage structure, as shown in Figure 3a. The robot is a planar car with 3 DOF.
- **Cluttered environment**: Obstacles are cluttered around the room as shown in Figure 3b. The robot has to traverse through these obstacles successfully to reach its goal. The robot is a 6 DOF cube.
- **Urban environment**: This environment consists of buildings as obstacles in the city-like structure, as shown in Figure 3c. The robot is a 6 DOF drone.
- **Heterogeneous 3D**: A 3D maze environment with walls and narrow passages between the walls. A robot has to pass through maze-like tunnels to reach the goal, as shown in Figure 3d. The robot is a 6 DOF toroidal plus.
- **Kuka YouBot environment**: An 8 DOF robot in an environment with four different rooms, see Figure 3e. The robot moves through these rooms within narrow passages and arrives at its destination, where it performs an action (grasps or puts an object down). This robot is a simulation replica of Kuka YouBot [24].
- **3D Tunnel**: This environment comprises a narrow passage tunnel and a set of cluttered cubes within it, as shown in Figure 3f. The robot is a 9 DOF articulated linkage chain that has to pass through a hole in the wall and encounter obstacles at the other end of the wall.
- **PR2 robot environment**: The environment has two pillars and a table as the obstacles where the robot is required to bypass through the pillars to grasp the stick kept on the other side, as shown in Figure 3g. The robot is a simulation replica of the PR2 robot [25] with only the right-hand arm (14 DOF) and fixed base.

VI. Result Analysis

We compare our proposed method with optimal planners k-PRM* [15] and LazyPRM* [16], with cell decomposition methods Dynamic Domain RRT [10] and Dynamic Region-based RRT/PRM [11], [12], and with Sampling-based RoadMap Trees (SRT) planner SparkPRM [9] which combines a PRM and RRT appraoch. All results were averaged over ten randomized seeds with an evaluation of 500 trials. The value of $\eta = 1$ in Eq. 3. We used the PRM strategy to uniformly sample configurations within each voxel.

A. *Optimal Planners*

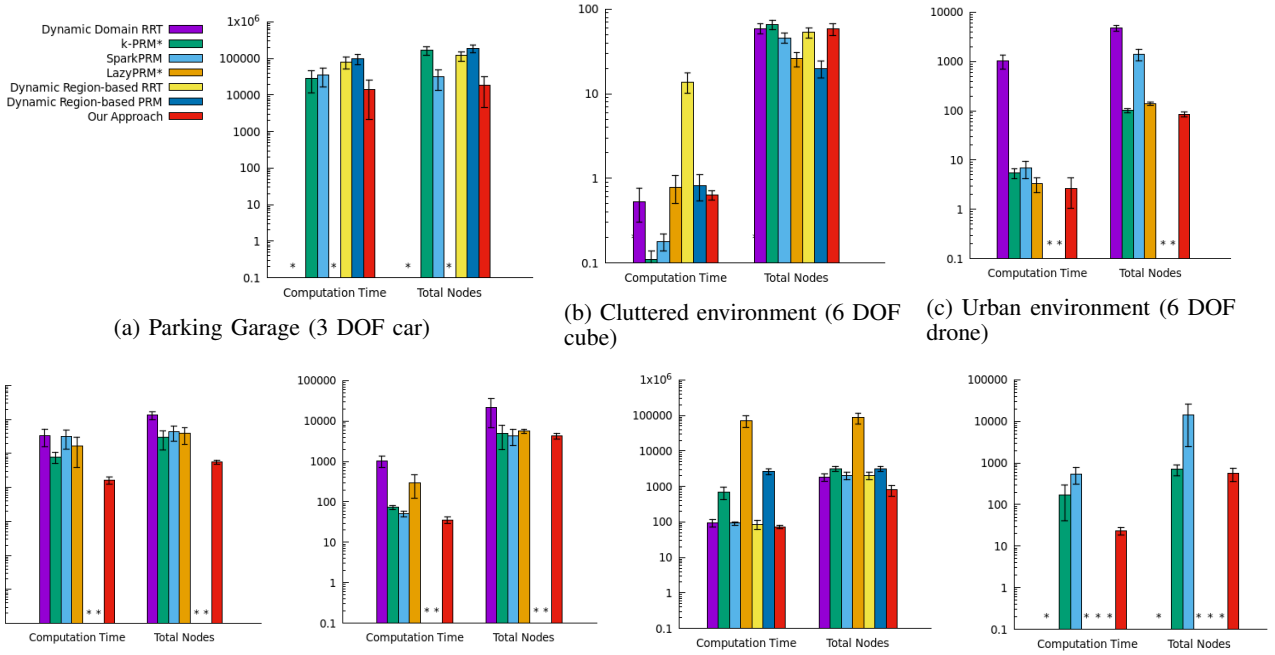
We consider the value of k = 15 for k-PRM* as derived in Karaman et al. paper on optimal path planning [15] and the connection radius as the function of random configuration nodes in $\mathcal{C}_{space}$ for both k-PRM* and LazyPRM* methods.

a) Computation Time and Total Nodes: We plot our results considering the time and total nodes needed to solve the query as seen in Figure 4 and report that our method performs faster than the optimal planners and needs fewer nodes in all the environments except the Cluttered environment (Figure 4b). k-PRM* achieves the lowest computation time, and LazyPRM* requires the least number of nodes in the roadmap. The minimal overhead of our planner in the Cluttered environment is due to the time consumed to process topological and geometric information at each voxel. As our results have shown, our planner improves as the DOF of the robot increases, and the Cluttered environment comprises a simple box robot in a uniformly distributed space.

b) Average Path Cost: Table I show the average path cost with standard deviation (superscript) for all methods. Compared to the extra time taken by our planner to process the environments, it still produced the shortest path in all environments. LazyPRM* failed to finish finding a path in the Parking Garage and PR2 robot environments within 72 hrs. Since LazyPRM* continually increases the density of its approximation, the graph search eventually becomes too expensive and fails to find solutions in complex environments.

TABLE I: Path Cost computed for Optimal Planners

Environments	Our Approach	k-PRM*	LazyPRM*
Parking Garage	**11139.5**$^{\pm 105.5}$	11879$^{\pm 118.9}$	DNF
Cluttered environment	**402**$^{\pm 12.1}$	425$^{\pm 26.1}$	463$^{\pm 17.4}$
Urban environment	**1750**$^{\pm 4.8}$	1823$^{\pm 36.2}$	1758$^{\pm 14.8}$
Heterogeneous 3D	**2341.4**$^{\pm 18.5}$	3640$^{\pm 19.3}$	3611$^{\pm 9.6}$
3D Tunnel	**293.3**$^{\pm 3.9}$	337$^{\pm 6.5}$	407$^{\pm 17.2}$
Kuka YouBot environment	**3601.3**$^{\pm 24.5}$	9405$^{\pm 46.7}$	9211$^{\pm 20.9}$
PR2 robot environment	**10**$^{\pm 0.8}$	11$^{\pm 1.3}$	DNF



(a) Parking Garage (3 DOF car)

(b) Cluttered environment (6 DOF cube)

(c) Urban environment (6 DOF drone)

(d) Heterogeneous 3D (6 DOF toroidal plus)

(e) Kuka YouBot environment (8 DOF)

(f) 3D Tunnel (9 DOF serial chain)

(g) PR2 robot environment (14 DOF)

Fig. 4: The plots for each environment show the total computation time (in seconds), and the number of nodes in a roadmap recorded for each planner averaged over 10 random runs. The error bars show the standard deviation. "*" indicates no result data available for the respective planner or the planner failed to finish within 72 hrs.

B. *Cell Decomposition Methods*

We compare our method with two different cell decomposition strategies that also analyze the properties of $\mathcal{C}_{space}$. We combined the dynamic region sampling strategy from [11], [12] with RRT and PRM methods and performed experiments in all seven environments.

a) Computation Time and Total Nodes: Our method extracts the topological and geometric representation of the $\mathcal{C}_{space}$ at each voxel and metrically glues them to maintain the topological connectivity between the voxel graphs. We can deduce from Figure 4 that the overhead of approximating the $\mathcal{C}_{\text{free}}$ information becomes negligible as the complexity of $\mathcal{C}_{space}$ increases. Hence, we observe a boost in the performance of our planner with the lowest computation time and least number of nodes in six of our seven environments studied. Via Figure 4b, we report that Dynamic Domain RRT uses less time than other cell decomposition methods, and Dynamic Region-based PRM requires fewer nodes to plan a path in the Cluttered environment. However, Dynamic Region-based RRT/PRM were unsuccessful in solving queries for the Urban environment, Heterogeneous 3D, and Kuka YouBot environment. All three baseline methods failed to search a route in the PR2 robot environment, and Dynamic Domain RRT could not finish finding pathways in the Parking Garage within 72 hrs.

b) Average Path Cost: Table II shows the average path cost attained by the cell decomposition methods. We observe that using the topological and geometric information preserved at each voxel does not affect the homotopy-equivalence representation of our roadmaps, and our approach was still able to obtain edges closer to the $\mathcal{C}_{obst}$ generating the shortest path in six of our seven environments compared to other baseline methods. Dynamic Region-based PRM finds the shortest route compared to other planners in the Cluttered environment.

As explained in Section IV-C, our method performs validity checks and convex hull computations within sub-regions and the distance metric test to find the closest configuration to the goal for the next voxel polytope. Overall our method avoids heavy geometric evaluation that requires Voronoi cell computation, sweep-line validity check, or pruning objects into holes. These computational methods start degrading in their performance when dealing with high DOF robots or cluttered/non-uniform environments. Hence, our planner becomes more successful than the baseline methods.

C. *Sampling-based RoadMap Trees (SRT) Planners*

We compare our approach with SparkPRM as it combines the capability of PRM and RRT methods to find a solution in an environment similar to ours.

a) Computation Time and Total Nodes: From Figure 4f, we observe a slight difference in computation time between SparkPRM and our method, which indicates an improvement in the efficiency of our planner as it extracts the features of $\mathcal{C}_{space}$. The performance of our method improves as the complexity of the $\mathcal{C}_{space}$ increases with importance placed on the preserved topology information, which influenced the generation of configurations in proximity to $\mathcal{C}_{obst}$, thus creating a better path. Our method uses less time and fewer nodes to solve the query than SparkPRM in six of the seven environments. In the Cluttered environment, SparkPRM performs better than our approach in computation time and

TABLE II: Path Cost computed for Cell Decomposition methods

Environments	Our Approach	Dynamic Domain RRT	Dynamic Region-based RRT	Dynamic Region-based PRM
Parking Garage	**11139.5**$^{\pm 105.5}$	DNF	13647$^{\pm 19.1}$	16982$^{\pm 130.3}$
Cluttered environment	402$^{\pm 12.1}$	419.89$^{\pm 4.5}$	433.66$^{\pm 5.8}$	**364**$^{\pm 16.1}$
Urban environment	**1750**$^{\pm 4.8}$	1761.5$^{\pm 12.9}$	N/A	N/A
Heterogeneous 3D	**2341.4**$^{\pm 18.5}$	3483$^{\pm 22}$	N/A	N/A
3D Tunnel	**293.3**$^{\pm 3.9}$	336.04$^{\pm 6}$	337.69$^{\pm 6.1}$	396$^{\pm 7.9}$
Kuka YouBot environment	**3601.3**$^{\pm 24.5}$	3623$^{\pm 25}$	N/A	N/A
PR2 robot environment	**10**$^{\pm 0.8}$	N/A	N/A	N/A

generates fewer nodes to solve a query, as shown in Figure 4b.

b) Average Path Cost: Although SparkPRM finds the shortest path in the Cluttered environment, our method attains convergence to low path cost in the remaining six environments due to the preserved topological and geometric information of the $\mathcal{C}_{space}$ that influences the formation of edges closer to the $\mathcal{C}_{obst}$, in Table III.

TABLE III: Path Cost computed for SRT planners

Environments	Our Approach	SparkPRM
Parking Garage	**11139.5**$^{\pm 105.5}$	14915$^{\pm 138.6}$
Cluttered environment	402$^{\pm 12.1}$	**336**$^{\pm 8.3}$
Urban environment	**1750**$^{\pm 4.8}$	1964$^{\pm 30.8}$
Heterogeneous 3D	**2341.4**$^{\pm 18.5}$	2486$^{\pm 24.9}$
3D Tunnel	**293.3**$^{\pm 3.9}$	331$^{\pm 5.6}$
Kuka YouBot environment	**3601.3**$^{\pm 24.5}$	4648$^{\pm 34.8}$
PR2 robot environment	**10**$^{\pm 0.8}$	17.9$^{\pm 1.1}$

Hence, we can conclude that using the topological and geometric information of $\mathcal{C}_{space}$, our method converges to a better solution than the tested baseline methods. We also observe that our approach successfully finds the route from start to goal position by metric gluing the voxel blocks without losing the topological significance.

VII. Conclusion

The paper presented an incremental path planner that constructs a simplicial complex at each sub-region of $\mathcal{C}_{\text{free}}$ and extracts the topological and geometric features locally. It metrically glues these complexes together while adapting the PRM and RRT capabilities to solve a query on growing voxels in a goal-biased direction. The results compared in three strategy categories have shown that, with a small overhead, our algorithm performed better than the discussed sampling-based methods. Extensions of this work will investigate its application to kinodynamic systems and physical robots in unknown tangent space.

References

[1] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Cambridge, MA: MIT Press, June 2005.

[2] L. E. Kavraki, P. Švestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Automat.*, vol. 12, no. 4, pp. 566–580, August 1996.

[3] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 1999, pp. 473–479.

[4] A. Upadhyay, W. Wang, and C. Ekenna, "Approximating c free space topology by constructing vietoris-rips complex," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 2517–2523.

[5] A. Upadhyay, B. Goldfarb, W. Wang, and C. Ekenna, "A new application of discrete morse theory to optimizing safe motion planning paths," in *15th International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2022.

[6] W. MathWorld, "Polytope," https://mathworld.wolfram.com/Polytope.html, 2022, accessed: 2022-05-05.

[7] T. Lozano-Perez, "Spatial planning: A configuration space approach," *Computers, IEEE Transactions on*, vol. 100, no. 2, pp. 108–120, 1983.

[8] E. Plaku and L. E. Kavraki, "Distributed sampling-based roadmap of trees for large-scale motion planning," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, 2005, pp. 3868–3873.

[9] K. Shi, J. Denny, and N. M. Amato, "Spark prm: Using rrts within prms to efficiently explore narrow passages," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 4659–4666.

[10] A. Yershova, L. Jaillet, T. Simeon, and S. M. Lavalle, "Dynamic-domain RRTs: Efficient exploration by controlling the sampling domain," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, April 2005, pp. 3856–3861.

[11] J. Denny, R. Sandström, A. Bregger, and N. M. Amato, "Dynamic region-biased rapidly-exploring random trees," in *Twelfth International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2016.

[12] R. Sandström, D. Uwacu, J. Denny, and N. M. Amato, "Topology-guided roadmap construction with dynamic region sampling," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6161–6168, 2020.

[13] S. Karaman and E. Frazzoli, "Incremental sampling-based optimal motion planning," in *Robotics: Science and Systems*, 2010.

[14] S. Karaman and E. Frazzoli, "Sampling-based optimal motion planning for non-holonomic dynamical systems," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 5041–5047.

[15] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[16] K. Hauser, "Lazy collision checking in asymptotically-optimal motion planning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 2951–2957.

[17] J. Ichnowski and R. Alterovitz, "Multilevel incremental roadmap spanners for reactive motion planning," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 1504–1509.

[18] A. Mandalika, R. Scalise, B. Hou, S. Choudhury, and S. S. Srinivasa, "Guided incremental local densification for accelerated sampling-based motion planning," *arXiv preprint arXiv:2104.05037*, 2021.

[19] J. Ryde, V. Dhiman, and R. Platt, "Voxel planes: Rapid visualization and meshification of point cloud ensembles," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 3731–3737.

[20] M. Adamaszek, H. Adams, E. Gasparovic, M. Gommel, E. Purvine, R. Sazdanovic, B. Wang, Y. Wang, and L. Ziegelmeier, "Vietoris-rips and cech complexes of metric gluings," in *34th International Symposium on Computational Geometry (SoCG 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[21] P. Lab, "Parasol planning library," https://github.com/parasol-ppl/ppl, 2022.

[22] S. Gottschalk, M. Lin, and D. Manocha, "Obb-tree: A hierarchical structure for rapid interference detection," in *Proc. ACM SIGGRAPH*, 1996, pp. 171–180.

[23] T. McMahon, S. Jacobs, B. Boyd, L. Tapia, and N. M. Amato, "Evaluation of the k-closest neighbor selection strategy for prm construction," Texas A&M, College Station Tx., Technical Report TR12-002, 2011.

[24] KUKA Robotics Corporation, "Kukayoubot," https://cyberbotics.com/doc/guide/youbot, accessed: January 7, 2022.

[25] WillowGarage, "PR2 Robot," www.willowgarage.com, Palo Alto, CA, 2016.