

Safe Bubble Cover for Motion Planning on Distance Fields

Ki Myung Brian Lee¹, Zhirui Dai¹, Cedric Le Gentil², Lan Wu², Nikolay Atanasov¹, and Teresa Vidal-Calleja²

¹ University of California San Diego, La Jolla, CA 92093, USA

² University of Technology Sydney, Ultimo NSW 2007, Australia

Abstract. We consider the problem of planning collision-free trajectories on distance fields. Our key observation is that querying a distance field at one configuration reveals a region of safe space whose radius is given by the distance value, obviating the need for additional collision checking within the safe region. We refer to such regions as safe bubbles, and show that safe bubbles can be obtained from any Lipschitz-continuous safety constraint. Inspired by sampling-based planning algorithms, we present three algorithms for constructing a safe bubble cover of free space, named bubble roadmap (BRM), rapidly exploring bubble graph (RBG), and expansive bubble graph (EBG). The bubble sampling algorithms are combined with a hierarchical planning method that first computes a discrete path of bubbles, followed by a continuous path within the bubbles computed via convex optimization. Experimental results show that the bubble-based methods yield up to 5–10 times cost reduction relative to conventional baselines while simultaneously reducing computational efforts by orders of magnitude.

1 Introduction

Motion planning is a foundational component of robot autonomy. It is important not only for operating in complex environments, but also as robots become increasingly physically capable. To fully harness the physical capabilities of modern robots in complex environments, it is necessary to develop planning algorithms that rapidly produce safe and dynamically feasible trajectories.

A significant bottleneck in conventional motion planning algorithms is collision checking. To ensure collision avoidance, a planning algorithm typically samples along a candidate trajectory to check for a potential collision. Although it is possible to accelerate collision queries [27], the computational efficiency of planning algorithms remains bounded by the number of collision checking queries.

We propose an approach to circumvent this fundamental limitation by sampling continuous regions of safe space instead of collision checking individual configurations. Our key insight is that querying a distance field representation of the environment yields the radius of collision-free space around the query point, since it represents the distance to the closest obstacle. See Fig. 1 for an illustration. We define such regions as ‘safe bubbles’, and show that they can be derived from any Lipschitz-continuous safety constraint, distance fields being a special case for which many perception algorithms are available [34,5,26].

To sample safe bubbles from a distance field representation, we first introduce three sampling algorithms, named bubble roadmap (BRM), rapidly-exploring bubble graph (RBG), and expansive bubble graph (EBG), inspired by conventional sampling-based planning algorithms. Given a bubble cover of free space, we present an efficient hierarchical planning method that first computes a sequence of intersecting bubbles minimizing an upper bound on the trajectory length, followed by continuous planning of a dynamically feasible trajectory via convex optimization.

We evaluate the practical utility of safe bubble planning through comparisons against popular sampling-based planning algorithms, PRM* and RRT*. The results show that safe bubble

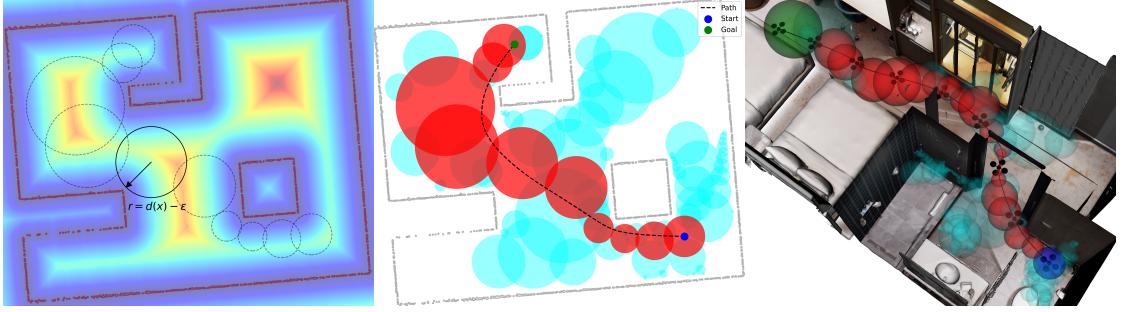


Fig. 1: Left: We construct ‘safe bubbles’ from a distance field representation of the environment, whose radii are given by the distance to obstacles. Middle: We present three algorithms for sampling safe bubbles (cyan), and a hierarchical planning method that first computes a bubble path (red) and then a continuous trajectory (dashed line) within the bubble path. Right: Our approach scales effectively to higher dimensions.

planning provides orders of magnitude improvements in computational efficiency along with 4-5 times improvement in trajectory cost. We also compare the three algorithms and find that the bubble-based methods can benefit from techniques for even spatial coverage in classical sampling-based planning. We view the main contribution of this work as establishing the foundation for a new class of planning algorithms for continuous implicit environment representations.

2 Related Work

Sampling-based motion planning methods plan collision-free paths by sampling safe configurations and connecting them with collision-free edges. Probabilistic roadmap (PRM) variants [10,3,9] draw samples from a uniform distribution, whereas rapidly exploring random tree (RRT) variants [14,9] promote even spatial coverage by “steering” the next sample toward a uniform random sample. For dynamic feasibility, a continuous trajectory optimization step is often employed to smooth the resulting path [37]. An in-depth treatise of discrete and continuous planning methods can be found in [13]. Our approach unifies continuous and discrete planning by sampling continuous regions of collision-free configurations, rather than individual configurations.

Although sampling-based methods are powerful for planning in complex, high-dimensional spaces, computation is often hindered by the cost of repeated *collision checking*. To this end, previous work attempts to reduce either the number [3,6] or computation time [27] of collision checks. We present an orthogonal approach that replaces collision checking with *safe space* construction.

The idea of safe space construction is popular especially for quadrotors [17,4,35]. These methods generate a convex partition of free space, using polytopes [17,4,35] or ellipsoids [8]. Given such partitions, convex optimization methods [17,18,4,35] can compute cost-optimal continuous-space trajectories. However, constructing a convex partition of free space is not readily compatible with environment representations obtained by perception and mapping methods. The locations and shapes of polytopes or ellipsoids are typically optimized within an occupancy grid or point cloud map of the environment.

In the perception community, an emerging trend is to use implicit representations of occupied space, in the form of distance fields [5,32,34,15,33] or radiance fields [20,11], which allow efficient and accurate reconstruction of an environment. Previous work on planning in these

representations explored differentiating through a scene representation for nonlinear trajectory optimization [1,34,37] or building a safe convex polytope from Gaussian splats [4]. Our approach generates a safe convex cover by simply querying a distance field representation [34], and shows that any Lipschitz-continuous representation [29,5] can be used.

Of particular relevance to our work are [23,12,24,28]. These approaches also consider spherical safe space representations, computed by keeping track of obstacle points [12,23,28] or a distance transformation of occupancy grid [24], coupled with specialized sampling strategies. We present a theoretical foundation for these "safe bubble" approaches, and explore their generality as a new class of sampling-based planning algorithms for implicit representations.

3 Problem Formulation

Consider a robot described by a nonlinear dynamical system:

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)), \quad (1)$$

where $\mathbf{x}(t) \in \mathcal{X} \subseteq \mathbb{R}^n$ is the state and $\mathbf{u}(t) \in \mathbb{R}^m$ is the control input. We assume that (1) is differentially flat.

Definition 1 ([30, Ch. 2]). *A nonlinear dynamical system (1) is differentially flat if there exists a flat output $\mathbf{y}(t) \in \mathbb{R}^m$ such that:*

- $\mathbf{y}(t)$ is expressed by a smooth function of $\mathbf{x}(t)$, $\mathbf{u}(t)$, and the first $r \in \mathbb{N}$ derivatives of $\mathbf{u}(t)$:

$$\mathbf{y}(t) = h(\mathbf{x}(t), \mathbf{u}(t), \dot{\mathbf{u}}(t), \dots, \mathbf{u}^{(r)}(t)),$$

- $\mathbf{x}(t)$ and $\mathbf{u}(t)$ can be expressed as smooth functions of $\mathbf{y}(t)$ and its derivatives:

$$\begin{aligned} \mathbf{x}(t) &= \alpha(\mathbf{y}(t), \dot{\mathbf{y}}(t), \dots, \mathbf{y}^{(r-1)}(t)), \\ \mathbf{u}(t) &= \beta(\mathbf{y}(t), \dot{\mathbf{y}}(t), \dots, \mathbf{y}^{(r)}(t)), \end{aligned}$$

- there exists no differential relation among the output derivatives such as $\eta(\mathbf{y}, \dot{\mathbf{y}}, \dots) = 0$.

Many robot systems are differentially flat, including quadrotors [21,19], fully actuated Lagrangian systems [22], and some non-holonomic systems [22].

Our objective is to plan a sufficiently smooth output trajectory $\mathbf{y}(t)$ minimizing a convex cost function $c : \mathbb{R}^m \rightarrow \mathbb{R}$ subject to constraints $\mathbf{y}(t) \notin \Omega \subset \mathbb{R}^m$. We assume that the distance function of the unsafe set Ω is known:

$$d_\Omega(\mathbf{y}) = \min_{\mathbf{q} \in \partial\Omega} \|\mathbf{y} - \mathbf{q}\|, \quad (2)$$

which maybe constructed online from sensor data [34,25,26]. The problem of planning a collision-free output trajectory can then be formulated as follows:

$$\begin{aligned} \min_{\mathbf{y} \in C^r} \quad & \int_0^T c(\mathbf{y}(t)) dt, \\ \text{s.t. (start and goal point)} \quad & \mathbf{y}(0) = \mathbf{y}_s, \mathbf{y}(T) = \mathbf{y}_g, \\ \text{(collision avoidance)} \quad & d_\Omega(\mathbf{y}(t)) \geq \epsilon, \quad \forall t \in [0, T], \end{aligned} \quad (3)$$

where \mathbf{y}_s and \mathbf{y}_g are given start and goal, $\epsilon > 0$ is a parameter, T is the planning horizon, and $\mathbf{y} \in C^r$ means that $\mathbf{y} : [0, T] \rightarrow \mathbb{R}^m$ has a continuous r -th derivative.

4 Constructing a Bubble Cover

We present algorithms for efficiently approximating the collision-avoidance constraint in (3) by sampling a *safe bubble cover*. We define and explain *safe bubbles* in Sec. 4.1, and present three randomized algorithms for constructing such bubbles inspired by well-known sampling-based planning algorithms in Sec. 4.2, 4.3, 4.4.

4.1 Bubbles for Safe Space Representation

Our core insight for solving (3) efficiently is that the distance field d_Ω can be used to build a set of *bubbles* covering the safe space. Intuitively, since the distance field $d_\Omega(\mathbf{y})$ yields the closest distance to the unsafe set Ω , there can be no obstacle within radius $d_\Omega(\mathbf{y})$ of a query point \mathbf{y} . Otherwise, $d_\Omega(\mathbf{y})$ must be smaller to reflect the presence of an obstacle within the ball.

More generally, we can define such bubbles whenever the safety constraint in (3) is specified by a *Lipschitz continuous* function, which may be constructed from data using methods such as [29,5]. We define a *safe bubble* and discuss its properties next.

Theorem 1 (Safe bubble). *Consider a constraint $l(\mathbf{y}) \geq 0$, where $l : \mathbb{R}^m \rightarrow \mathbb{R}$ is Lipschitz continuous with constant L . Then, for any \mathbf{y} such that $l(\mathbf{y}) \geq 0$, all points \mathbf{y}' in ball $B(\mathbf{y}, \frac{l(\mathbf{y})}{L})$ centered at \mathbf{y} with radius $\frac{l(\mathbf{y})}{L}$ also satisfy $l(\mathbf{y}') \geq 0$.*

Proof. With Lipschitz continuity, we have $|l(\mathbf{y}) - l(\mathbf{y}')| \leq L\|\mathbf{y} - \mathbf{y}'\|$ for any $\mathbf{y}, \mathbf{y}' \in \mathbb{R}^m$. For $\mathbf{y}' \in B(\mathbf{y}, \frac{l(\mathbf{y})}{L})$, we have $\|\mathbf{y} - \mathbf{y}'\| \leq \frac{l(\mathbf{y})}{L}$. Combining the two inequalities yields $l(\mathbf{y}') \geq 0$ (i.e., arbitrary points \mathbf{y}' in $B(\mathbf{y}, \frac{l(\mathbf{y})}{L})$ are feasible). \square

Theorem 1 includes the distance field constraint in (3) as a special case with $l(\mathbf{y}) = d_\Omega(\mathbf{y}) - \epsilon$ because the distance function d_Ω of any set Ω is Lipschitz continuous with constant $L = 1$. Importantly, Theorem 1 implies that querying the distance field at a point where $d_\Omega(\mathbf{y}) \geq \epsilon$ readily yields a safe bubble around the point. Then, an important consideration is *how to sample* such query points at which to generate safe bubbles to cover the safe space. Next, we take inspiration from sampling-based planning methods to generate safe bubbles.

4.2 Bubble Roadmap

Inspired by PRM [10], the simplest method we propose is the *bubble roadmap* (BRM), described in Alg 1. Just as PRM samples random configurations, BRM samples random bubble centers from a uniform distribution. In doing so, a minimum radius requirement is enforced, which speeds up subsequent planning inside the bubble cover, discussed in Sec. 5. This is achieved by first sampling possible centers for the bubbles, computing the radii as per Theorem 1, and keeping only those large enough. Results with varying numbers of samples are visualized in Fig. 2.

Algorithm 1 Bubble Roadmap Algorithm

Parameters: No. of samples N_{sample} , minimum radius r_{\min} , footprint radius ϵ .

```

1:  $\mathcal{C} \leftarrow \text{SAMPLE}(N_{\text{sample}})$                                  $\triangleright$  Sample  $N$  random centers.
2:  $\mathcal{B} = \{\}$                                                         $\triangleright$  Initialize bubble cover as an empty set.
3: for  $\mathbf{y} \in \mathcal{C}$  do
4:   if  $d_\Omega(\mathbf{y}) - \epsilon > r_{\min}$  then                                $\triangleright$  If radius greater than  $r_{\min}$ ,
5:      $\mathcal{B} \leftarrow \mathcal{B} \cup \{B_{\text{new}} = (\mathbf{y}, d_\Omega(\mathbf{y}) - \epsilon)\}$             $\triangleright$  add to set of bubbles.
return  $\mathcal{B}$ 

```

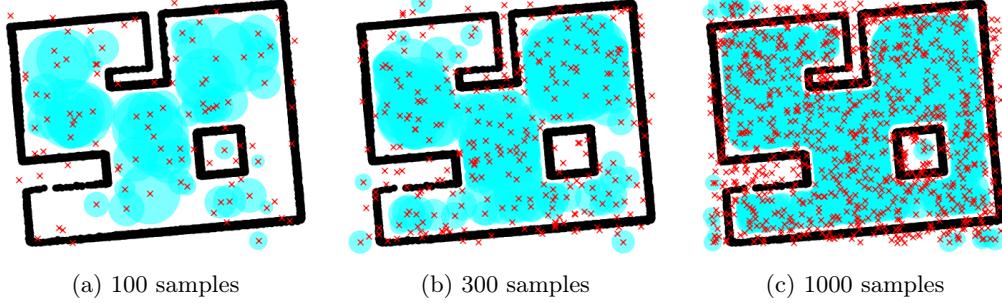


Fig. 2: Illustration of the BRM algorithm with varying number of samples (red crosses) and bubbles (cyan). Not all random samples lead to a valid bubble due to footprint and minimum radius requirements. The free space is filled as the number of samples increases (from (a) to (c)).

A potential concern with BRM is that some samples may be redundant because some bubbles may contain other. However, we show that the event of one bubble containing another is of probability zero.

Theorem 2 (Random bubbles do not contain each other). *Let $\mathbf{y}_1, \mathbf{y}_2$ be samples from a distribution such that $\|\mathbf{y}_1 - \mathbf{y}_2\| = \delta$ is probability zero $\forall \delta \geq 0$. Let $B_1 = (\mathbf{y}_1, r_1)$ and $B_2 = (\mathbf{y}_2, r_2)$ be safe bubbles for a Lipschitz constraint $l(\mathbf{y}) \geq 0$ as per Theorem 1. Then, the probability of $B_1 \subseteq B_2$ is zero.*

Proof. First, notice $B_1 \subseteq B_2$ iff $\|\mathbf{y}_1 - \mathbf{y}_2\| \leq |r_1 - r_2|$. With Theorem 1, we have $|r_1 - r_2| = |\frac{l(\mathbf{y}_1) - l(\mathbf{y}_2)}{L}|$. Meanwhile, by Lipschitz continuity, $|\frac{l(\mathbf{y}_1) - l(\mathbf{y}_2)}{L}| \leq \|\mathbf{y}_1 - \mathbf{y}_2\|$. Thus, $B_1 \subseteq B_2$ is only possible when $\|\mathbf{y}_1 - \mathbf{y}_2\| = |r_1 - r_2|$, which is probability zero as per assumption. \square

Theorem 2 applies to most practical sampling distributions, including uniform. However, the problem of redundancy remains, because Theorem 2 only applies to the case of one bubble wholly containing another. It is still possible and empirically frequent that the union of multiple bubbles contains another bubble. Next, we consider sampling strategies that improve on BRM in terms of overlap redundancy.

4.3 Rapidly Exploring Bubble Graph

Although BRM samples centers from a uniform distribution, the resulting bubbles can exhibit uneven coverage of space because their radii vary over space. Inspired by RRT [14], we propose the *rapidly exploring bubble graph* (RBG) algorithm that improves coverage by expanding bubbles toward random samples.

The RBG algorithm is described in Alg. 2, and illustrated in Fig. 3. A set of bubbles is initialized with a bubble at a seed point \mathbf{y}_s , which may be the start point in (3) (line 1). Similar to RRT, a random point is sampled (line 3), and the nearest bubble is identified (line 3). In doing so, we use the distance to the *boundary* of the existing bubbles:

$$d(\mathbf{y}, B) = \|\mathbf{y} - \mathbf{c}\| - r, \quad (4)$$

where \mathbf{c} and r are the center and radius of $B = (\mathbf{c}, r)$ respectively.

Algorithm 2 Rapidly-exploring Bubble Graph

Parameters: Starting location \mathbf{y}_s , minimum radius r_{\min} , footprint radius ϵ

```

1:  $\mathcal{B} \leftarrow \{B(\mathbf{y}_s, d_{\Omega}(\mathbf{y}_s) - \epsilon)\}$                                 ▷ Initialize with bubble at starting location
2: while termination condition is not met do
3:    $\mathbf{y}_{\text{rand}} \leftarrow \text{SAMPLE\_OUTSIDE}(\mathcal{B})$                                 ▷ Sample outside current bubbles
4:    $B_{\text{near}} = (\mathbf{y}_{\text{near}}, r_{\text{near}}) \leftarrow \min_{B \in \mathcal{B}} d(\mathbf{y}_{\text{rand}}, B)$     ▷ Find nearest bubble (see (4))
5:    $\mathbf{y}_{\text{new}} \leftarrow \text{STEER}(\mathbf{y}_{\text{rand}}, B_{\text{near}})$                                 ▷ Steer to perimeter of nearest bubble (see (5))
6:   if  $r_{\text{new}} = d(\mathbf{y}_{\text{new}}) - \epsilon > r_{\min}$  then
7:      $\mathcal{B} \leftarrow \mathcal{B} \cup \{B(\mathbf{y}_{\text{new}}, r_{\text{new}})\}$                                 ▷ Add new bubble if size requirement holds

```

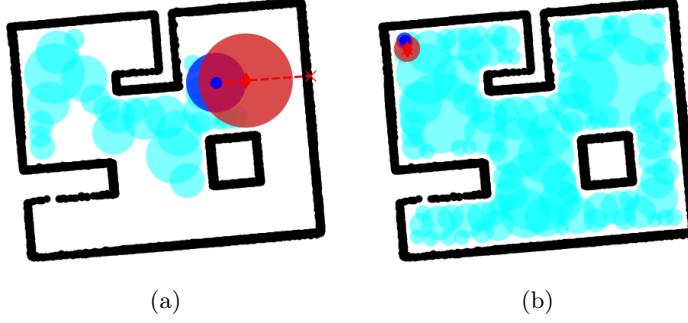


Fig. 3: Illustration of the RBG algorithm. Similar to RRT, RBG promotes even spatial coverage by sampling and steering towards random points (red crosses). The center of a new bubble (red diamond) is set at the intersection between the boundary of the nearest bubble and the straight line between the random point and the nearest bubble center (line 5). Repeating this process from a) to b), we obtain a safe bubble cover of the free space with limited variation in radii.

Steering is achieved by setting the new center at the intersection between the boundary of the nearest bubble and the straight line between the random point and the nearest bubble center (line 5). In other words, the new center is steered toward the random sample, up to the *perimeter* of the nearest bubble:

$$\mathbf{y}_{\text{new}} = \mathbf{y}_{\text{near}} + r_{\text{near}} \frac{\mathbf{y}_{\text{rand}} - \mathbf{y}_{\text{near}}}{\|\mathbf{y}_{\text{rand}} - \mathbf{y}_{\text{near}}\|}. \quad (5)$$

The main difference between RBG and RRT is that the random samples \mathbf{y}_{rand} must be outside the union of existing bubbles (line 3). This is because, otherwise, the new center after steering (5) will be contained inside an existing bubble, slowing down the planning progress. Sampling outside existing bubbles can be achieved with rejection sampling in the simplest case, though more sophisticated and efficient methods may be possible. With rejection sampling, we found it beneficial to inflate the support of the random samples \mathbf{y}_{rand} , especially in closed obstacles. This leaves room for random samples to still be drawn from outside existing bubbles even after expansion. Suitable termination conditions for RBG include checking if a certain number of bubbles have been drawn, or if a given target point is reached.

4.4 Expansive Bubble Graph

Another pertinent idea to achieve even bubble coverage is to consider the local density of existing samples, as is done in the EST algorithm [7]. We propose the *expansive bubble graph* (EBG) algorithm, which aims to achieve even bubble coverage by limiting overlap with existing bubbles.

Algorithm 3 Expansive Bubble Graph

Parameters: No. of directions N_{explore} , overlap factor k_{overlap} , min. radius r_{\min} , footprint radius ϵ ,
 1: $\mathcal{Q} \leftarrow \{B(\mathbf{y}_s, d_\Omega(\mathbf{y}_s) - \epsilon)\}$ ▷ Priority queue in descending order of radius
 2: $\mathcal{B} \leftarrow \{\}$
 3: **while** \mathcal{Q} is not empty and termination condition not met **do**
 4: $B_{\text{current}} = (\mathbf{y}_{\text{current}}, r_{\text{current}}) \leftarrow \text{POP}(\mathcal{Q})$
 ▷ Skip if overlap with existing bubbles
 5: **if** $\min_{B \in \mathcal{B}} d(\mathbf{y}_{\text{current}}, B) < -k_{\text{overlap}} r_{\text{current}}$ **then continue**
 6: $\mathcal{B} \leftarrow \mathcal{B} \cup \{B_{\text{current}}\}$ ▷ Otherwise, append and expand
 7: **for** $i \in [1, N_{\text{explore}}^m]$ **do**
 8: $\mathbf{y}_{\text{new}} \leftarrow \mathbf{y}_{\text{current}} + r_{\text{current}} \hat{\mathbf{e}}_i$ ▷ Expand in random or uniform directions.
 9: **if** $r_{\text{new}} = d(\mathbf{y}_{\text{new}}) - \epsilon > r_{\min}$ **then** ▷ Enqueue if large enough
 10: $\mathcal{Q} \leftarrow \text{PUSH}(\mathcal{Q}, B(\mathbf{y}_{\text{new}}, r_{\text{new}}))$

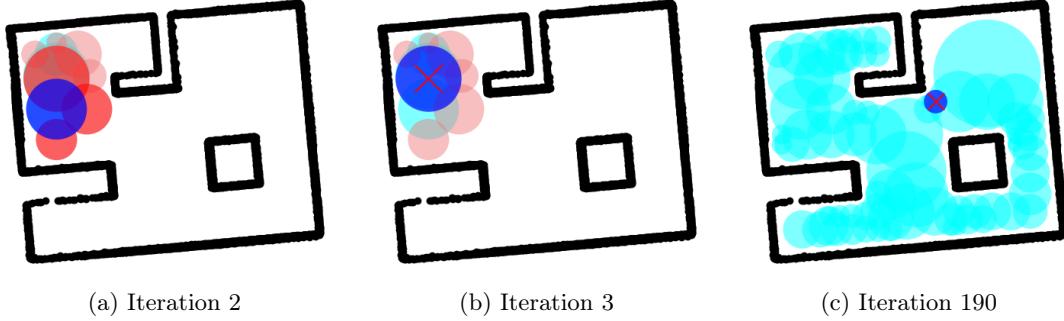


Fig. 4: Illustration of EBG with $N_{\text{explore}} = 4$. a) New bubbles (red) are expanded from current (blue). b) A new bubble from a) is considered, but discarded because of overlap (red cross). c) Repeating this process fills the free space with confirmed bubbles (cyan).

Pseudocode for the EBG algorithm is presented in Alg. 3, and is illustrated in Fig. 4. The EBG algorithm expands new bubbles in N_{explore} different directions per dimension, and keeps ones that have limited overlap with existing bubbles and are larger than r_{\min} . The EBG algorithm starts by initializing a priority queue \mathcal{Q} with a bubble at the robot's starting location (line 1). The queue \mathcal{Q} is in descending order of radii so that larger bubbles appear first.

During iteration, the largest bubble in the queue is popped and added to the cover if there is a limited overlap with existing bubbles (line 5). Overlap is measured using the ratio k_{overlap} of a bubble's own radius r_{current} to the signed distance of the center to the union of existing bubbles: $d(\mathbf{y}, \cup_{B_e \in \mathcal{B}} B_e) = \min_{B_e \in \mathcal{B}} d(\mathbf{y}, B_e)$, where $d(\mathbf{y}, B_e)$ is given by (4). Setting $k_{\text{overlap}} = 0$ discards bubbles whose center is on the perimeter of an existing bubble, whereas $k_{\text{overlap}} = 1$ only discards bubbles that are contained in an existing bubble.

Expansion is performed in a total of N_{explore}^m directions, $\hat{\mathbf{e}}_i$, which are unit vectors sampled in uniform or random angular increments. The expanded bubbles are enqueued for subsequent iterations if the minimum radius requirement r_{\min} holds (line 10). The algorithm terminates if the queue is empty, or if an early termination condition holds. Suitable early termination conditions for EBG include a certain number of bubbles being reached, or a desired end point being contained in the current bubble.

5 Planning in Bubble Cover

The bubble cover \mathcal{B} generated by the sampling algorithms in Sec. 4 enables a hierarchical discrete-continuous planning method. Using \mathcal{B} as an approximation of the safe space, the original planning problem (3) can be approximately recast as:

$$\begin{aligned} \min_{\mathbf{y} \in C^r} \quad & \int_0^T c(\mathbf{y}(t))dt, \\ \text{s.t.} \quad & \mathbf{y}(0) = \mathbf{y}_s, \mathbf{y}(T) = \mathbf{y}_g, \\ & \mathbf{y}(t) \in \bigcup_{B \in \mathcal{B}} B, \quad \forall t \in [0, T], \end{aligned} \tag{6}$$

where the collision-free constraint has been replaced by containment in the bubble cover. This conservative reformulation of (3) is substantially simpler than the original problem because the nonlinear feasible set has been replaced by a union of simple convex sets. However, the disjunctive containment constraint remains non-convex.

We develop an efficient hierarchical planning method that utilizes the convexity of the bubbles. We first build an intersection graph of bubbles to generate a discrete *bubble path*, followed by using the bubbles along the path as convex constraints to generate a continuous trajectory.

5.1 Discrete Planning of Bubble Path

The bubble path should be so chosen to ensure a) feasibility and b) approximate optimality of the continuous trajectory within the bubbles. Feasibility can be ensured by constructing an intersection graph of the bubble cover. An intersection graph $\mathcal{G} = (\mathcal{B}, \mathcal{E})$ is an undirected graph where each node is a bubble, and an edge $(i, j) \in \mathcal{E}$ indicates an overlap between two bubbles $B_i \cap B_j \neq \emptyset$. Intuitively, where two bubbles are connected by an edge, it is feasible for the robot to continuously move between the two bubbles through the overlap.

With the intersection graph constructed, approximate optimality of the bubble path can be ensured by considering the ‘worst-case optimal cost’ $\hat{c}(B_i, B_j)$ between two bubbles B_i and B_j , defined as:

$$\begin{aligned} \hat{c}(B_i, B_j) \equiv \max_{\mathbf{y}_{ij}^s} \min_{\mathbf{y}_{ij} \in C^r} \int_0^{T_{ij}} c(\mathbf{y}_{ij}(t))dt \\ \text{s.t.} \quad \mathbf{y}_{ij}(0) = \mathbf{y}_{ij}^s, \\ \mathbf{y}_{ij}(t) \in B_i, \quad \forall t \in [0, T_{ij}), \quad \mathbf{y}(T_{ij}) \in B_j. \end{aligned} \tag{7}$$

Here, the duration T_{ij} of potential trajectory $\mathbf{y}_{ij}(t)$ is chosen arbitrarily, e.g., as $T_{ij} = \frac{r_i}{V_0}$ for some nominal speed V_0 and r_i being the radius of bubble B_i . The intuition for this cost is as follows. Between bubbles B_i and B_j , consider picking an optimal trajectory that is contained within B_i and reaches B_j given a start point $\mathbf{y}_{ij}^s \in B_i$, so that the terminal point $\mathbf{y}_{ij}(T_{ij})$ is chosen greedily. Since the terminal point is the start point of the next bubble, (7) captures the case when such greedy choice of terminal point is adversarial, so that the terminal point from B_i is the worst start point for B_j . Since the true optimum of (6) is a special case with \mathbf{y}_{ij}^s and T_{ij} being selected optimally across all bubbles, the sum of maximum optimal cost (7) along a path forms an upper bound on the true optimal cost of (6).

When the given cost function is length (i.e., $c(\mathbf{y}(t)) = \|\dot{\mathbf{y}}(t)\|$), the worst-case optimal cost (7) reduces to the single-sided Hausdorff distance:

$$d_H(B_i, B_j) \equiv \sup_{\mathbf{y}_i \in B_i} d(\mathbf{y}_i, B_j) = \|\mathbf{c}_i - \mathbf{c}_j\| + r_i - r_j, \tag{8}$$

which holds for two overlapping bubbles $B_i = (\mathbf{c}_i, r_i)$ and $B_j = (\mathbf{c}_j, r_j)$.

For simplicity, we adopt the Hausdorff distance (8) as an approximation of the worst-case optimal cost (7), with the expectation that shorter length trajectories incur less cost. With this approximation, the bubble path \mathcal{P} can be found by solving a graph shortest path problem:

$$\begin{aligned} \min_{\mathcal{P}=(B_p)} \quad & \sum_p d_H(B_p, B_{p+1}), \\ \text{s.t.} \quad & (B_p, B_{p+1}) \in \mathcal{E}, \\ & B_1 \in \mathcal{B}(\mathbf{y}_s), B_{|\mathcal{P}|} \in \mathcal{B}(\mathbf{y}_t), \end{aligned} \quad (9)$$

where $\mathcal{B}(\mathbf{y}_s)$ and $\mathcal{B}(\mathbf{y}_t)$ denote the bubbles that contain the start position \mathbf{y}_s and target position \mathbf{y}_t respectively. There may be multiple bubbles that contain either start or target position. Because the Hausdorff distance (8) is non-negative, Dijkstra's algorithm can be used to quickly solve for the shortest path.

5.2 Continuous Planning

With a discrete bubble path $\mathcal{P} = \{B_p\}_p$ given, we compute a sequence of $|\mathcal{P}|$ continuous segments $\mathbf{y}_p(s) : [0, T_p] \rightarrow \mathbb{R}^N$, one for each bubble in the discrete path P . This can be formulated as:

$$\begin{aligned} \min_{\mathbf{y}_1(t), \dots, \mathbf{y}_{|\mathcal{P}|}(t)} \quad & \sum_p \int_0^{T_p} c(\mathbf{y}_p(t)) dt, \\ \text{s.t. (bubble containment)} \quad & \mathbf{y}_p(t) \in B_p, \quad \forall p, t, \\ \text{(start/end points)} \quad & \mathbf{y}_1(0) = \mathbf{y}_s, \mathbf{y}_{|\mathcal{P}|}(1) = \mathbf{y}_g, \\ \text{(continuous derivatives)} \quad & \mathbf{y}_p^{(d)}(T_p) = \mathbf{y}_{p+1}^{(d)}(0), \quad \forall d \in [0, r], \forall p. \end{aligned} \quad (10)$$

Conveniently, the continuous problem (10) can be solved as a convex program by parameterizing the trajectories $\mathbf{y}_p(t)$ as *Bezier curves* given by:

$$\mathbf{y}_p(t) = \sum_{k=0}^K b_k \left(\frac{t}{T_p} \right) \mathbf{b}_k^p, \quad (11)$$

where $b_k(s) = \binom{K}{k} s^k (1-s)^{K-k}$ are the Bernstein basis polynomials, $\mathbf{b}_k^p \in \mathbb{R}^m$ are the *control points*, T_p is the duration of each trajectory, and K is the order of the Bezier curve. As noted in [18], Bezier curves have several useful properties that allow parameterizing (10) as a convex program of control points. The start and end of each Bezier curve are given by $\mathbf{y}_p(0) = \mathbf{b}_0^p$, $\mathbf{y}_p(T_p) = \mathbf{b}_K^p$, and the derivatives are another Bezier curve with control points given by the finite difference $\Delta[\mathbf{b}_k^p] = \mathbf{b}_{k+1}^p - \mathbf{b}_k^p$ of control points as $\dot{\mathbf{y}}_p(t) = \sum_{k=0}^{K-1} b_k \left(\frac{t}{T_p} \right) \frac{K}{T_p} \Delta[\mathbf{b}_k^p]$. These two properties show that the derivative continuity and start/end points constraints in (10) are affine in the control points \mathbf{b}_k^p .

More importantly, each curve \mathbf{y}_p is entirely contained in the convex hull of control points. Therefore, a *sufficient* relaxation of the bubble containment constraint is to ensure that all control points are contained in respective bubbles, i.e., $\mathbf{b}_k^p \in B_p, \forall k, p$. Moreover, the cost function remains a convex function of control points since the trajectory is an affine function of the control points at each t , and can be upper bounded as $\int_0^{T_p} c(\mathbf{y}_p(t)) dt \leq \frac{1}{K+1} \sum_k c(\mathbf{b}_k^p)$ [18]. Furthermore, common cost functions, such as the time integral of squared norm of n -th order derivatives of Bezier curves can be represented as a positive semidefinite quadratic form of \mathbf{b}_k^p [19].

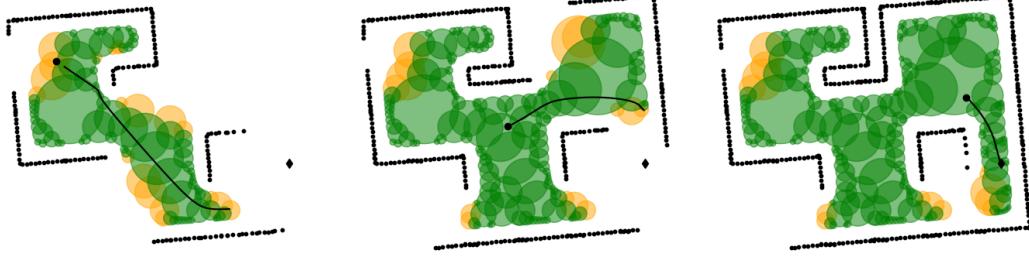


Fig. 5: An illustration of EBG modified for static unknown environments. Before the goal (black diamond) is in fully expanded bubbles (green), the planned trajectory (black solid line) points to the closest fully expanded bubble. As frontier bubbles (yellow) become visible, more fully expanded bubbles (green) appear.

Combining these properties, the continuous problem (10) can be written in terms of control points $\mathbf{b} = \{\mathbf{b}_k^p\}$ in the following form:

$$\begin{aligned} & \min_{\mathbf{b}} \sum_{p,k} c(\mathbf{b}_k^p), \\ \text{s.t. (bubble containment)} \quad & \mathbf{b}_k^p \in B_p, \quad \forall p, k, \\ \text{(start/end points)} \quad & \mathbf{b}_0^0 = \mathbf{y}_s, \mathbf{b}_K^{|\mathcal{P}|} = \mathbf{y}_t, \\ \text{(continuous derivatives)} \quad & (\Delta)^d [\mathbf{b}_{K-d}^p] = (\Delta)^d [\mathbf{b}_0^{p+1}], \quad \forall d \in [0, r], \forall p, \end{aligned} \quad (12)$$

where Δ^d is the d -th finite difference. The bubble containment constraint is quadratic, while the other constraints are affine. As the cost is convex, the overall problem is a convex program. In the special case when the cost is the squared norm of d -th derivatives, the cost is quadratic, and the overall problem is a quadratically-constrained quadratic program.

5.3 Extension to Unknown Environments

An important aspect of motion planning algorithms is operation in an unknown environment. We briefly outline an approach to planning in static unknown environments inspired by the concept of frontiers in occupancy grids [36]. The main idea is to verify whether a safe bubble had been fully *visible* from the robot at a particular pose. Those that had not been fully visible are the *frontier* bubbles. The queue \mathcal{Q} in Alg. 3 is modified, so that frontier bubbles are skipped, and only those fully visible are expanded, until only frontier bubbles remain. Upon receiving new sensor data, the visibility information is updated, and the expansion loop is repeated. To plan a path towards a goal in this incomplete cover, we simply pick the closest bubble to the goal, with an additional terminal cost of distance to the goal. An example result is illustrated in Fig. 5.

6 Evaluation

We evaluate the performance of the safe bubble cover algorithms in three benchmark environments, shown in Fig. 6. There are two 2D environments, namely the Gazebo Room [34] and the

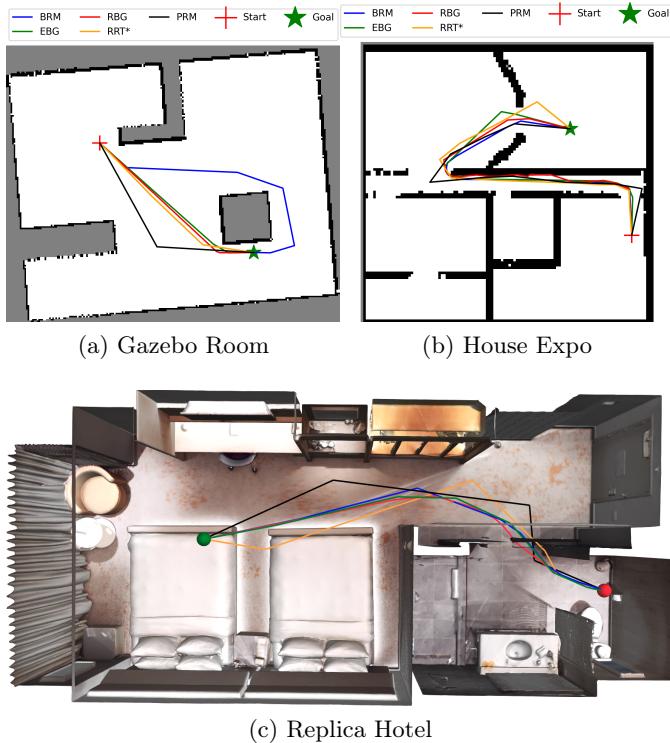


Fig. 6: Qualitative comparison of shortest-distance paths computed by different algorithms in three benchmarks. Paths shown correspond with the lowest computational effort recorded.

House Expo [31], which are representative of indoor environments. The House Expo, in particular, features the challenge of a relatively narrow corridor. We also consider a 3D environment, the Replica Hotel [2], which is widely used as a navigation benchmark, and represents an indoor environment cluttered with objects and a narrow entrance to the bathroom area. For all environments, the distance function is built from simulated LiDAR data using the Log-GPIS algorithm [34], which allows querying distance values at arbitrary continuous query points.

6.1 Planning Shortest Distance Paths

We first compare the planning performance of BRM, RBG, and EBG against two classical sampling-based planning algorithms, namely PRM* [10,9] and RRT* [14,9], in planning shortest distance paths. PRM* and RRT* perform collision checking by querying the distance field along each edge with a resolution of 0.05 m, which was chosen empirically to ensure correct collision checking.

For each of the three environments, 100 random start/goal pairs were chosen. For each start/goal pair, we repeated each planning algorithm with five different random seeds. For each run, we evaluated the number of SDF query positions, success rate of finding a path, and the path cost of successful runs.

The results are shown in Fig. 7-9. Fig. 7 shows the number of unique SDF query positions over the maximum number of iterations or samples, which is representative of the computation time in an optimized implementation. It can be seen that the classical sampling-based algorithms

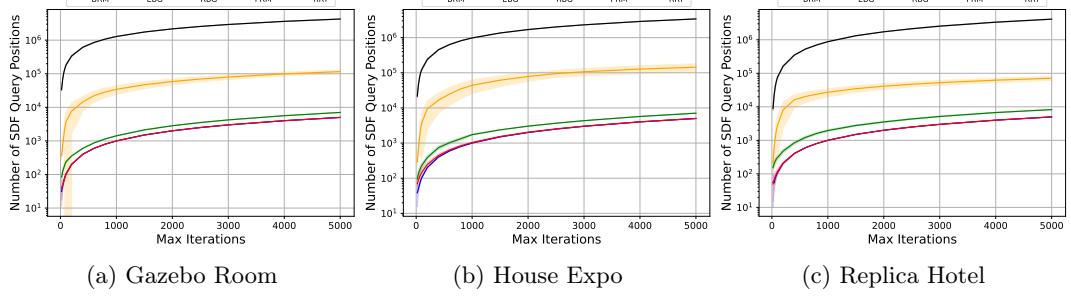


Fig. 7: Comparison of number of SDF query positions over maximum allowed iterations. The number of SDF queries signifies the computational effort required. Bubble-based methods incur 1-2 orders of magnitude less computational effort than conventional baselines. The results are averaged over 100 random start/end pairs, each with 5 different random seeds.

(RRT* and PRM*) make 1 - 3 orders of magnitude more SDF queries than the bubble cover algorithms (BRM, RBG and EBG). This shows that the bubble cover algorithms are 1 - 3 orders of magnitude more computationally efficient per iteration.

In Figs. 8 and 9, we compare the success rate and path cost relative to computation effort represented by the number of SDF query positions, because each iteration incurs varying computational effort between different algorithms. For comparison, we normalize the path cost by the worst run in Fig. 9, since start/goal pairs vary. In Fig. 8, it can be seen that RRT* and PRM* require at least four times more SDF queries than bubble-based algorithms to achieve 90% success rate across all environments. The distribution of normalized path cost and number of SDF query in Fig. 9 shows that PRM* generally produces the highest cost paths. In comparison, BRM, RBG and EBG are distributed near the bottom left corner, with up to 10-fold reduction in cost and computation simultaneously, with RBG and EBG showing advantage over BRM in 3D. The qualitative comparison in Fig. 6 shows that, even with limited compute, the bubble-based methods produce shorter paths that cut corners. This is because the bubble-based methods naturally incorporate continuous trajectory optimization.

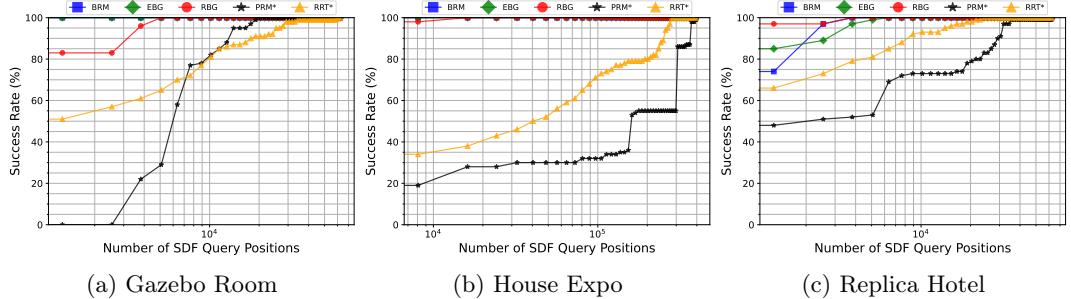


Fig. 8: Comparison of success rate over number of SDF query positions. Bubble-based methods reach 90% success rate with at least four times less computational effort (in Replica Hotel). The evaluation is performed over 100 random start/end pairs, each with 5 different random seeds.

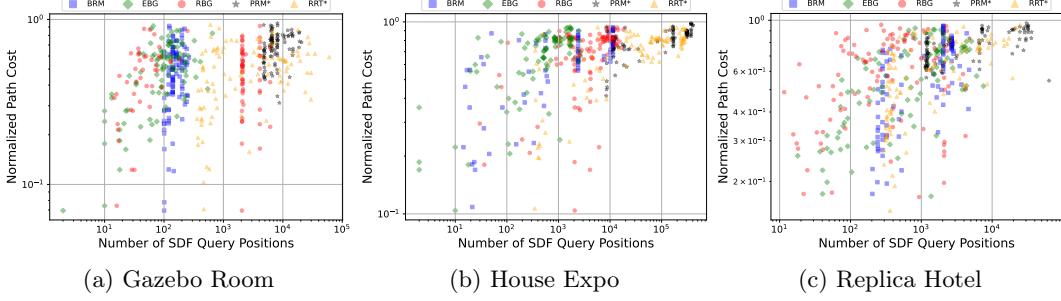


Fig. 9: Comparison of normalized trajectory length relative to worst, over number of SDF query positions. The bubble-based methods return shorter paths in less time. The results show 100 random start/goal pairs, each with 5 different random seeds.

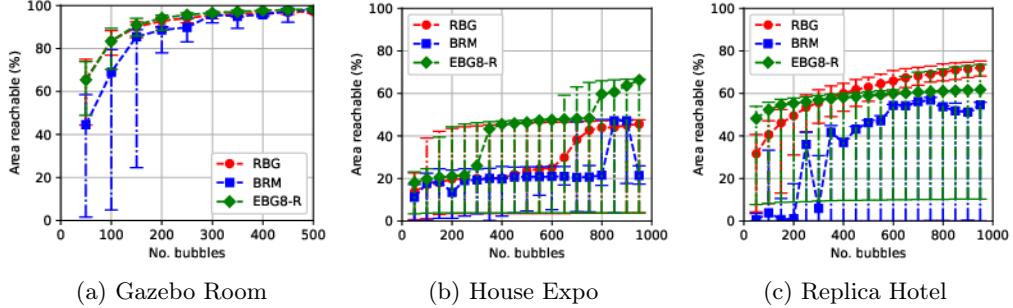


Fig. 10: Comparison of reachable area over iterations. Trend lines: median, error bars: 10% and 90% quantiles. RBG and EBG perform almost equally well in coverage (except in House Expo), while BRM performs worse with large variability. EBG used with 8 random directions.

6.2 Comparison of Bubble Sampling Algorithms

To evaluate the effectiveness of the proposed sampling methods, we task BRM, RBG and EBG to plan for a smooth, minimum snap trajectory [19], and compare the cost and the area of reachable space of the safe bubble covers. The planning is done in the same setting as Sec. 6.1, except for the cost function for continuous planning. The resulting trajectories can be used to control a quadrotor as illustrated in Fig. 1, by recovering the thrust and angular velocity inputs using differential flatness methods from [21,16].

To compute the reachable area, we generate safe bubble covers from 200 randomly initialized seed locations in the free space in the benchmark environments. We record the area of each bubble cover every 50 iterations, approximated using the Monte Carlo method with 100000 random samples from free space. Since BRM is not iterative, we consider a varying number of samples instead of iterations. Moreover, because BRM does not guarantee the bubble cover to be connected, we only consider the area of the connected component from the same starting location as RBG and EBG, to faithfully represent the utility in a planning scenario. The results are shown in Fig. 10. It can be seen that in the Gazebo Room (Fig. 10a), all methods eventually cover nearly 100% of the free space (Fig. 10a). RBG and EBG cover the space faster than BRM, which is attributed to RBG and EBG having respective means to promote even spatial coverage.

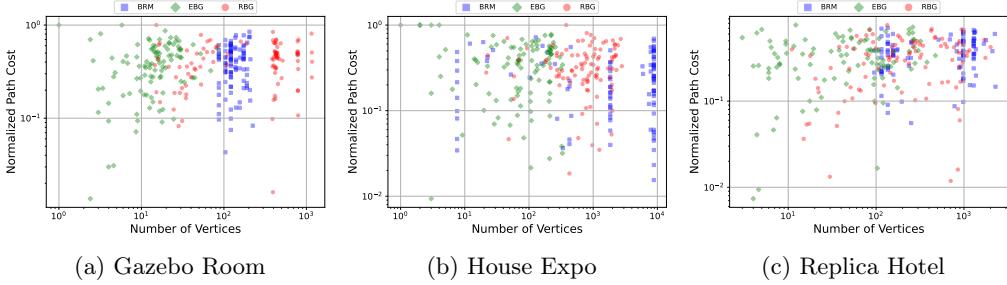


Fig. 11: Comparison of minimum snap trajectory cost over number of bubbles. Bubbles from EBG generally find the lowest-cost paths with fewer bubbles, followed by RBG and BRM.

The 10% and 90% quantiles of RBG and EBG are also narrower than BRM, which shows that RBG and EBG perform more reliably than BRM in covering the safe space.

A similar pattern is observed in the House Expo environment in Fig. 10b, which is the most challenging. Only up to 75% of the environment is covered in the best case, due to the narrow corridor. In this setting, RBG performs similarly to BRM in median, albeit with a higher 10% quantile. EBG performs the best, because it continues to make progress in the narrow corridor by expanding from current bubbles, whereas RBG and BRM rely on random samples.

The Replica Hotel (Fig. 6c) is also challenging, with all methods covering up to 90% of the free space (Fig. 10c). It can be seen that RBG and EBG generally cover safe space faster than BRM at all number of iterations in the Replica Hotel environment. This is consistent with the observation from the Gazebo Room environment, with a greater gap. The greater gap between BRM and other methods shows that ensuring even spatial coverage is more important with greater environment complexity and higher number of dimensions. Moreover, RBG outperforms EBG in median reachable area except at the very initial and final stages, with consistently narrower quantiles. We thus conclude that a) RBG exhibits better spatial coverage than EBG in 3D, and hypothesize that RBG will scale better to higher dimensions than EBG, and b) EBG is more suited for environments with narrow corridors.

The cost distribution plot in Fig. 11 suggests that, unlike for shortest length objective, EBG generally performs the best, followed by RBG and BRM. We attribute this to two factors: a) there is a disparity between the discrete planning objective (Hausdorff distance as an upper bound on the length) and the continuous planning objective (minimum snap); and b) EBG has more overlapping areas than RBG, because RBG only expands outwards. In this case, continuous planning can exploit the overlaps in EBG to better resolve the disparity in cost functions used.

7 Conclusion

We presented hybrid discrete-continuous sampling-based planning methods based on the idea of safe bubbles. We showed that safe bubbles can be defined for any Lipschitz-continuous safety constraint, with distance function as a special case. We introduced three sampling algorithms for safe bubble cover construction, namely BRM, RBG and EBG, drawing inspirations from PRM, RRT, and EST respectively, and developed a hierarchical method for planning continuous trajectories in the safe bubble cover. Our evaluations show that bubble-based methods yield trajectories with an order of magnitude lower cost, while being an order of magnitude more computationally efficient owing to the lack of explicit collision checking.

We anticipate that our results will lead to a new class of sampling-based planning algorithms for implicit representations that side-step collision checking and efficiently generate continuous trajectories. We hope that the proposed sampling techniques inspire other sampling methods for safe bubbles drawing upon decades of research in sampling-based planning. We plan to support this direction through theoretical analysis of the bubble cover methods and applications to multi-rigid-body robots in future work.

Acknowledgement This work was supported by the Ministry of Trade, Industry, and Energy (MOTIE), Korea, under the Strategic Technology Development Program supervised by the Korea Institute for Advancement of Technology (KIAT) [Grant No. P0026052]. Cedric and Teresa are supported by the Australian Research Council Discovery Project under Grant DP210101336.

References

1. Adamkiewicz, M., Chen, T., Caccavale, A., Gardner, R., Culbertson, P., Bohg, J., Schwager, M.: Vision-only robot navigation in a neural radiance world. *IEEE Robotics and Automation Letters* **7**(2), 4606–4613 (2022)
2. et al, J.S.: The Replica dataset: A digital replica of indoor spaces. arXiv preprint arXiv:1906.05797 (2019)
3. Bohlin, R., Kavraki, L.: Path planning using lazy PRM. In: IEEE Intl. Conf. on Robotics and Automation (ICRA). vol. 1, pp. 521–528 (2000)
4. Chen, T., Shorinwa, O., Zeng, W., Bruno, J., Dames, P., Schwager, M.: Splat-nav: Safe real-time robot navigation in Gaussian splatting maps. arXiv preprint arXiv:2403.02751 (2024)
5. Coiffier, G., Bethune, L.: 1-Lipschitz neural distance fields. arXiv preprint arXiv:2407.09505 (2024)
6. Hou, B., Choudhury, S., Lee, G., Mandalika, A., Srinivasa, S.S.: Posterior sampling for anytime motion planning on graphs with expensive-to-evaluate edges. In: IEEE Intl. Conf. on Robotics and Automation (ICRA). pp. 4266–4272. IEEE (2020)
7. Hsu, D., Latombe, J.c., Motwani, R.: Path planning in expansive configuration spaces. *Int. J. of Computational Geometry and Applications* **09**, 495–512 (03 1999)
8. Huh, J., Arslan, Ö., Lee, D.D.: Probabilistically safe corridors to guide sampling-based motion planning. In: Intl. Symp. of Robotics Research (ISRR). pp. 311–327. Springer International Publishing, Cham (2022)
9. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. *Intl. J. of Robotics Research* **30**(7), 846–894 (2011)
10. Kavraki, L., Svestka, P., Latombe, J.C., Overmars, M.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Automat.* **12**(4), 566–580 (1996)
11. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3D Gaussian splatting for real-time radiance field rendering. *ACM Trans. on Graphics (TOG)* **42**(4) (2023)
12. Kim, D., Kwon, Y., Yoon, S.E.: Dancing PRM*: Simultaneous planning of sampling and optimization with configuration free space approximation. In: IEEE Intl. Conf. on Robotics and Automation (ICRA). pp. 7071–7078 (2018)
13. LaValle, S.M.: Planning Algorithms. Cambridge University Press (2006)
14. LaValle, S.M., James J. Kuffner, J.: Randomized kinodynamic planning. *Intl. J. of Robotics Research* **20**(5), 378–400 (2001)
15. Le Gentil, C., Ouabi, O.L., Wu, L., Pradalier, C., Vidal-Calleja, T.: Accurate gaussian-process-based distance fields with applications to echolocation and mapping. *IEEE Robotics and Automation Letters* **9**(2), 1365–1372 (2024)
16. Liu, S., Mohta, K., Atanasov, N., Kumar, V.: Search-based motion planning for aggressive flight in SE(3). *IEEE Robotics and Automation Letters* **3**(3), 2439–2446 (2018). <https://doi.org/10.1109/LRA.2018.2795654>

17. Liu, S., Watterson, M., Mohta, K., Sun, K., Bhattacharya, S., Taylor, C.J., Kumar, V.: Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3D complex environments. *IEEE Robotics and Automation Letters* **2**(3), 1688–1695 (2017)
18. Marcucci, T., Petersen, M., von Wrangel, D., Tedrake, R.: Motion planning around obstacles with convex optimization. *Science Robotics* **8**(84) (2023)
19. Mellinger, D., Kumar, V.: Minimum snap trajectory generation and control for quadrotors. In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. pp. 2520–2525 (2011)
20. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: NeRF: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM* **65**(1), 99–106 (2021)
21. Morrell, B., Rigter, M., Merewether, G., Reid, R., Thakker, R., Tzanetos, T., Rajur, V., Chamitoff, G.: Differential flatness transformations for aggressive quadrotor flight. In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. pp. 5204–5210 (2018)
22. Murray, R.M., Rathinam, M., Sluis, W.: Differential flatness of mechanical control systems: A catalog of prototype systems. In: *ASME Intl. Mechanical Engineering Congress and Exposition*. pp. 349–357 (1995)
23. Musil, T., Petrlík, M., Saska, M.: SphereMap: Dynamic multi-layer graph structure for rapid safety-aware UAV planning. *IEEE Robotics and Automation Letters* **7**(4), 11007–11014 (2022)
24. Noël, T., Lehuger, A., Marchand, E., Chaumette, F.: Skeleton disk-graph roadmap: A sparse deterministic roadmap for safe 2D navigation and exploration. *IEEE Robotics and Automation Letters* **9**(1), 555–562 (2024)
25. Oleynikova, H., Taylor, Z., Fehr, M., Siegwart, R., Nieto, J.: Voxblox: Incremental 3D Euclidean signed distance fields for on-board MAV planning. In: *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)* (2017)
26. Ortiz, J., Clegg, A., Dong, J., Sucar, E., Novotny, D., Zollhoefer, M., Mukadam, M.: iSDF: Real-time neural signed distance fields for robot perception. In: *Robotics: Science and Systems (RSS)* (2022)
27. Ramsey, C.W., Kingston, Z., Thomason, W., Kavraki, L.E.: Collision-affording point trees: SIMD-amenable nearest neighbors for fast collision checking. In: *Robotics: Science and Systems (RSS)* (2024), to appear.
28. Ren, Y., Zhu, F., Liu, W., Wang, Z., Lin, Y., Gao, F., Zhang, F.: Bubble Planner: Planning high-speed smooth quadrotor trajectories using receding corridors. In: *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. pp. 6332–6339 (2022)
29. Revay, M., Wang, R., Manchester, I.R.: Lipschitz bounded equilibrium networks. arXiv preprint arXiv:2010.01732 (2020)
30. Rigatos, G.G.: Nonlinear control and filtering using differential flatness approaches: Applications to electromechanical systems, vol. 25. Springer Verlag (2015)
31. Tingguang, L., Danny, H., Chenming, L., Delong, Z., Chaoqun, W., Meng, M.Q.H.: Houseexpo: A large-scale 2d indoor layout dataset for learning-based algorithms on mobile robots. arXiv preprint arXiv:1903.09845 (2019)
32. Ueda, I., Fukuhara, Y., Kataoka, H., Aizawa, H., Shishido, H., Kitahara, I.: Neural density-distance fields. In: *European Conf. on Computer Vision (ECCV)* (2022)
33. Wu, L., Le Gentil, C., Vidal-Calleja, T.: Vdb-gpdf: Online gaussian process distance field with vdb structure (2024), <https://arxiv.org/abs/2407.09649>
34. Wu, L., Lee, K.M.B., Le Gentil, C., Vidal-Calleja, T.: Log-GPIS-MOP: A unified representation for mapping, odometry, and planning. *IEEE Trans. Robotics* **39**(5), 4078–4094 (2023)
35. Wu, Y., Spasojevic, I., Chaudhari, P., Kumar, V.: Optimal convex cover as collision-free space approximation for trajectory generation. arXiv preprint arXiv:2406.09631 (06 2024)
36. Yamauchi, B.: A frontier-based approach for autonomous exploration. In: *IEEE Intl. Symp. on Computational Intelligence in Robotics and Automation (CIRA)*. pp. 146–151 (1997)
37. Zucker, M., Ratliff, N., Dragan, A.D., Pivtoraiko, M., Klingensmith, M., Dellin, C.M., Bagnell, J.A., Srinivasa, S.S.: CHOMP: Covariant Hamiltonian optimization for motion planning. *Intl. J. of Robotics Research* **32**(9-10), 1164–1193 (2013)