Improving Sparse Roadmap Spanners

Andrew Dobson and Kostas E. Bekris

***Abstract*— Roadmap spanners provide a way to acquire sparse data structures that efficiently answer motion planning queries with probabilistic completeness and asymptotic near-optimality. The current SPARS method provides these properties by building two graphs in parallel: a dense asymptotically-optimal roadmap based on PRM* and its spanner. This paper shows that it is possible to relax the conditions under which a sample is added to the spanner and provide guarantees, while not requiring the use of a dense graph. A key aspect of SPARS is that the probability of adding nodes to the roadmap goes to zero as iterations increase, which is maintained in the proposed extension. The paper describes the new algorithm, argues its theoretical properties and evaluates it against PRM* and the original SPARS algorithm. The experimental results show that the memory requirements of the method upon construction are dramatically reduced, while returning competitive quality paths with PRM*. There is a small sacrifice in the size of the final spanner relative to SPARS but the new method still returns graphs orders of magnitudes smaller than PRM*, leading to very efficient online query resolution.**

I. Introduction

An important challenge in motion planning is to compute compact representations for shortest paths in continuous configuration spaces (*C*-spaces) [1], such as the problem in Fig. 1. This work focuses on practical solutions that preprocess a *C*-space through sampling to (a) build small roadmaps that can quickly return solutions, and (b) provide theoretical guarantees regarding path quality. Previous efforts integrate asymptotically optimal planners [2] with graph spanners to provide *asymptotically near-optimal* solutions [3], [4]. Spanners are subgraphs where the shortest path between two nodes in the spanner is no longer than t times the shortest path in the original graph, where t is the stretch factor of the spanner. Spanners, however, only remove edges, and these methods have no obvious stopping criterion. The SPARS algorithm [5] provides asymptotic near-optimality and ensures the probability of adding new nodes to the roadmap converges to 0. This gives rise to a natural stopping criterion inspired by Visibility PRM [6]. SPARS builds in parallel an asymptotically optimal roadmap using PRM* that includes all *C*-space samples, incurring high memory costs, and limiting practicality.

This work extends SPARS and shows it is possible to build a sparse roadmap spanner without using an underlying dense roadmap. Fig. 2 shows the four cases that roadmap spanners need to consider for adding nodes. Samples added for coverage, connectivity, or for an "interface" do not need the dense graph, while shortcut nodes require *C*-space shortest path information. It is possible to relax the requirements for adding such nodes and provide the desired near-optimality guarantees *without the dense graph* so that the probability of adding new nodes to the spanner goes to zero. Local sampling and auxiliary information are utilized. This paper details this new algorithm and its properties. Experiments compare the method against PRM* and SPARS, and indicate that it uses significantly less memory during construction. The returned graph is similarly sparse compared to the SPARS solution, though slightly denser. SPARS2 returns paths competitive to those of PRM*, better than those of SPARS, and significantly better than the theoretical guarantees. Online query resolution times are very efficient on both roadmap spanners and significantly shorter than PRM*.

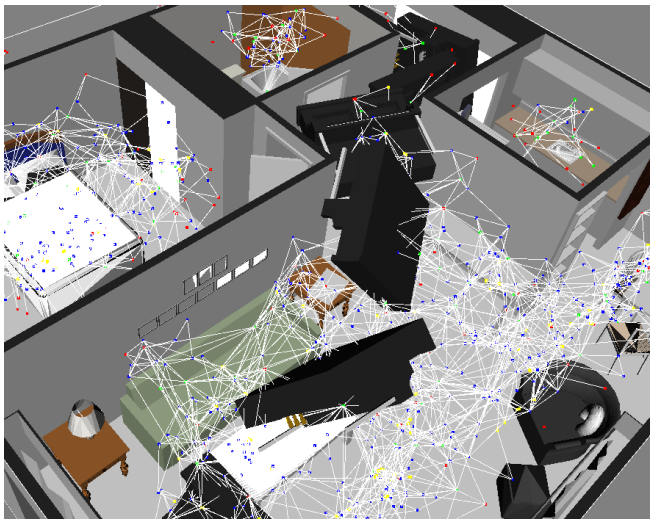

Fig. 1. A roadmap spanner covering OMPL's "apartment" environment [7], which corresponds to an $SE(3)$ challenge, and configurations for a piano moving along a solution path from the top of the figure to the bottom.

II. Related Work

Sampling-based Motion Planning The Probabilistic Roadmap Method (PRM) was the first approach that popularized the idea of building a *C*-space roadmap using sampling [8] and achieves probabilistic completeness [9]–[11]. The PRM and its extensions have been used to solve a variety of path planning problems, with some variations focusing on roadmap size, coverage and connectivity [12], [13]. For instance, the Visibility PRM rejects nodes unnecessary for coverage or connectivity [6]. The Useful Cycles criterion evaluates edges in terms of path quality [14]. Methods that reason about path homotopy provide solutions that can be smoothed to optimal ones [15], [16]. Hybridization graphs

Work by the authors has been supported by NSF CNS 0932423. Any conclusions expressed here are of the authors and do not reflect the views of the sponsors. The authors are with the Department of Computer Science, Rutgers University, Piscataway, NJ, 08854, USA, email: kostas.bekris at cs.rutgers.edu

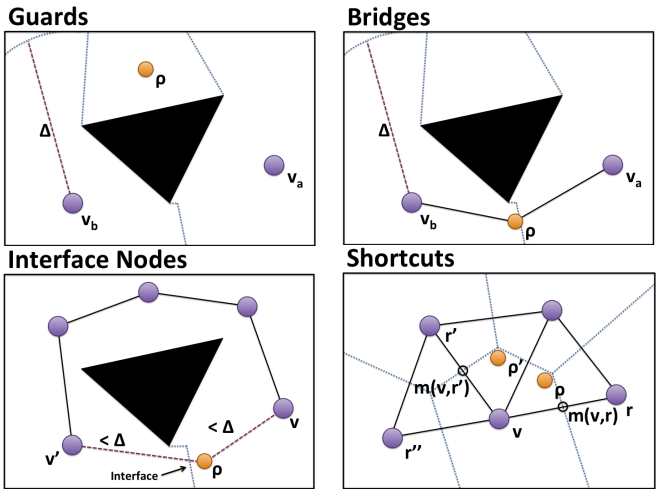

Fig. 2. The four types of samples $\rho \in C$ that roadmap spanners consider for addition ro G_S. Nodes may be added for coverage (guards), connectivity (bridges), to connect nodes which share an "interface" (interface nodes), or to satisfy near-optimality constraints when efficient C-space paths (ρ to ρ') are found (shortcuts).

can be seen as a smoothing process that combines multiple solutions into a single, better quality one [17]. Tree-based planners, such as RRT [18], can also address problems with dynamics and already return sparse graphs. RRT has been shown, however, to converge to a suboptimal solution [2]. Tree-based planners can speed up their exploration using C-space distance oracles, such as those provided by roadmap spanners [19].

Asymptotic (Near-)Optimality Recent work introduced asymptotically optimal sampling-based planners [2], such as PRM*, which requires each sample to be tested for connection to a logarithmic function of the number of nodes in the roadmap. Applying an efficient graph spanner [20] on the output of PRM* reduces the number of edges, but not nodes, of the roadmap and provides asymptotic near-optimality. An incremental integration of spanners with PRM* provides even better results [4]. The more recent SPArse Roadmap Spanner (SPARS) algorithm [5] provides: i) probabilistic completeness, ii) asymptotic near-optimality [for any cl-clearance optimum path with cost c^* in C_{free} the method returns a path of length $t \cdot c^* + 4 \cdot \Delta$ where t and Δ are user provided] and iii) the probability of adding new nodes and edges to the roadmap spanner converges to 0. The method uses an asymptotically optimal, dense roadmap to build the spanner. C-space samples are included if they improve path quality relative to paths on the dense graph. The method provides efficient online query resolution with path quality significantly better than the theoretical bounds. Unfortunately, the memory requirements of the approach during construction are significant.

III. Problem Formulation

A robot operates in a d-dimensional configuration space (C-space), where a point is denoted as q and C_{free} is the space's collision-free subset. This work focuses on planar and rigid body configurations ($SE(2)$, $SE(3)$), but is applicable if appropriate metric and sampling functions exist.

Definition 1 (The Path Planning Problem): Given the set of free configurations $C_{\text{free}} \subset C$, initial and goal points $q_{\text{init}}, q_{\text{goal}} \in C_{\text{free}}$, find a continuous path $\pi \in \Pi = \{q | q : [0,1] \rightarrow C_{\text{free}}\}$, $\pi(0) = q_{\text{init}}$ and $\pi(1) = q_{\text{goal}}$.

Definition 2 (Robust Feasible Paths): A path planning instance (C_{free}, q_{init}, q_{goal}) is robustly feasible if a cl-robust path exists that solves it, for clearance $cl > 0$. A path $\pi \in \Pi$ is cl-robust, if π lies entirely in the cl-interior of C_{free}.

A $cl-$robust *shortest path* in C_{free} is denoted as π^*_{cl}. The following primitives are also used in this work:

Local Planner: Given $q, q' \in C$, a *local planner* computes a *local path* $\mathtt{L}(q,q')$ connecting q and q' ignoring obstacles. A straight line between q and q' in C is often sufficient.

Distance function: The space C is endowed with a *distance function* $d(q,q') \rightarrow \mathbb{R}$ that returns distances between configurations in the absence of obstacles.

The objective is to compute a compact roadmap for answering queries with high-quality paths. This requires finding which C-space samples can be omitted. An implicit, exhaustive graph $G(V,E)$ over C_{free} can be defined by taking all the elements of C_{free} as nodes and collision free paths between them as edges. Then, a "roadmap spanner" is a subgraph $G_S(V_S \subset V, E_S \subset E)$ of G with the following properties:

i) All G_S nodes are connected in C_{free} with a node on G.
ii) G_S has the same number of connected components as G (connectivity).
iii) All shortest paths on G_S are no longer than t times the corresponding shortest paths in G.

The above properties guarantee that if a query has a solution, then G_S will provide asymptotically near-optimal paths for them. The shortest path between two configurations computed on G_S is denoted as π_S. Specifically, the following variant of asymptotic near-optimality is provided:

Definition 3 (Asympt. Near-Optimality w. Additive Cost): An algorithm is asymptotically near-optimal with additive cost if, for a path planning problem (C_{free}, q_{init}, q_{goal}) and cost function $c : \Pi \rightarrow \mathbb{R}^{\geq 0}$ with a cl-robust optimal path of finite cost c^*, the probability it will find a path with cost $c \leq t \cdot c^* + \varepsilon$, for a stretch factor $t \geq 1$ and additive error $\varepsilon \geq 0$, converges to 1 as the iterations approach infinity.

IV. A Practical Sparse Roadmap Spanner

The proposed SPARS2 method (Algorithm 1) generates samples $\rho \in C_{\text{free}}$ and decides if they must be added to the graph G_S. If M consecutive samples are not added, where M is an input parameter, the algorithm returns G_S.

The "coverage" criterion checks whether ρ is in a part of C_{free} not covered by nodes $V_S \in G_S$ (Algorithm 1 lines 4-5). SPARS2 adds ρ to G_S as a new "guard" (see Figure 2a), if there are no nodes $v \in V_S$ within a "visibility" distance, Δ, so that $\mathtt{L}(\rho, v) \in C_{\text{free}}$. The "connectivity" criterion is tested in lines 8-12. Among the set N of spanner nodes which can connect to ρ, the algorithm identifies those in different connected components. If there are disconnected nodes, ρ is useful as a "bridge" and is added together with the edges to the disconnected nodes.

Algorithm 1: SPARS2$(M, t, \Delta, \delta, k)$

1 $failures \leftarrow 0$; $V_S \leftarrow \emptyset$; $E_S \leftarrow \emptyset$; $G_S \leftarrow (V_S, E_S)$;
2 **while** $failures < M$ **do**
3 $\rho \leftarrow$ Uniform_Random_Sample();
4 $N \leftarrow$ Visible_Guards(ρ, V_S, Δ);
5 **if** $N == \emptyset$ **then**
6 $V_S \leftarrow V_S \cup \rho$;
7 **else**
8 $v \leftarrow \arg\min_{n \in N} d(\rho, n)$;
9 **if** *any two* $n \in N$ *not connected* **then**
10 $V_S \leftarrow V_S \cup \rho$;
11 **for** *all such* $n \in N$ *not connected* **do**
12 $E_S \leftarrow E_S \cup$ L(ρ, n);
13 **else**
14 $v_1 \leftarrow \arg\min_{n \in N} d(\rho, n)$;
15 $v_2 \leftarrow \arg\min_{n \in N, n \neq v_1} d(\rho, n)$;
16 **if** L(ρ, v_1), L$(\rho, v_2) \in C_{free} \wedge$ L$(v_1, v_2) \notin E_S$ **then**
17 Close_Interface$(\rho, \mathrm{v}_1, \mathrm{v}_2, G_S)$;
18 **if** $\rho \notin V_S$ **then**
19 $(\Sigma, R) \leftarrow$ Get_Close_Reps$(\rho, \mathrm{v}, G_S, \Delta, \delta, \mathrm{k})$;
20 **if** $R \neq \emptyset$ **then**
21 **for** *each* $r \in R$ *and* $\sigma \in \Sigma$ **do**
22 Update_Points$(\rho, \sigma, \mathrm{v}, \mathrm{r}, G_S)$;
23 Update_Points$(\sigma, \rho, \mathrm{r}, \mathrm{v}, G_S)$;
24 Test_Add_Path(v, G_S);
25 **for** *each* $r \in R$ **do**
26 Test_Add_Path(r, G_S);
27 **if** *no change in* (V_S, E_S) **then**
28 *failures*++;
29 **return** (V_S, E_S);

The "interface" criterion checks if ρ can connect to its two closest guards, $v_1, v_2 \in V_S$, which share an interface but no edge (lines 14-17). An *interface* $i(v_1, v_2)$ between two spanner nodes v_1 and v_2 is the shared boundary of their visibility regions (see Figure 2 (bottom left)). These samples must be included so that shortest paths $\pi^*_{cl} \in C_{\text{free}}$ will be "covered", illustrated in Figure 3. This is necessary to argue about the lengths of paths $\pi^*_{cl}(\rho_0, \rho_m)$ and $\pi_S(\rho_0, \rho_m)$. The algorithm employs Close_Interface, which attempts to directly add the edge L(v_1, v_2) and if this is not possible, then the sample ρ is added together with the edges L(ρ, v_1) and L(ρ, v_2).

Lines 18-26 of the algorithm serve to uphold the spanner property for G_S. In relation to Figure 3, SPARS2 has to make sure that for each shortest path $\pi^*_{cl}(\rho_i, \rho_{i+1})$ between points on the interfaces, the corresponding "midpoint" path M_i satisfies the spanner property. The algorithm reasons over shortest paths between samples supporting interfaces. Every sample, ρ, is checked to determine if it does so via Get_Close_Reps (Algorithm 2), which generates k random

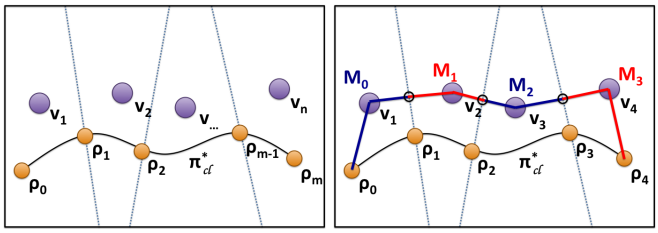

Fig. 3. (left) All configurations along $\pi^*_{cl}(\rho_0, \rho_m)$ will eventually be covered by a node in G_S. (right) A path between all spanner nodes covering $\pi^*_{cl}(\rho_0, \rho_m)$ will be created, decomposing path $\pi_S(\rho_0, \rho_m)$ into "midpoint paths" M_{i-1} that exist only in a single visibility region. Each M_{i-1} covers the path $\pi^*_{cl}(\rho_i, \rho_{i+1})$.

samples, σ, within a δ ball of ρ, where k and δ are input parameters to SPARS2. If σ and L(ρ, σ) are collision free, the method finds the representative guard $v_\sigma \in V_S$ of σ. If there is no such guard, then σ is a new guard that needs to be added to G_S. Otherwise, if $v_\sigma \neq v$, where v is ρ's representative, the two samples ρ and σ support interface $i(v, v_\sigma)$. Function Get_Close_Reps returns the sets Σ of samples, and R, their guards, which revealed that ρ lies on an interface of v.

Algorithm 2: Get_Close_Reps $(\rho, v, G_S, \Delta, \delta, k)$

1 $(\Sigma, R) \leftarrow (\emptyset, \emptyset)$;
2 **for** k *iterations* **do**
3 $\sigma \leftarrow$ Sample_Near(ρ, δ);
4 **if** L$(\rho, \sigma) \in C_{free}$ **then**
5 $N_\sigma \leftarrow$ Visible_Guards(σ, V_S, Δ);
6 $v_\sigma \leftarrow \arg\min_{n \in N_\sigma} d(\sigma, n)$;
7 **if** v_σ *does not exist* **then**
8 $V_S \leftarrow V_S \cup \sigma$;
9 **return** $(\emptyset, \emptyset)$;
10 **else if** $v \neq v_\sigma$ **then**
11 $\Sigma \leftarrow \Sigma \cup \sigma$;
12 $R \leftarrow R \cup v_\sigma$;
13 **return** (Σ, R);

If R and Σ are not empty, then SPARS2 checks if ρ reveals a short path between two interfaces of v. This is done through functions Update_Points and Test_Add_Path. In Figure 2 (bottom right), sample ρ lies on the interface of nodes v and r, and may reveal that G_S is not satisfying the spanner property with regards to the shortest path $\pi^*_{cl}(\rho, \rho')$.

Algorithm 3: Update_Points $(\rho, \sigma, \mathrm{v}, r, G_S)$

1 **for** $r' \in V_S \setminus r$ *so that* L$(\mathrm{v}, r') \in E_S$ *and* L$(r, r') \notin E_S$ **do**
2 $(p, p') \leftarrow P_{\mathrm{v}}(r, r')$;
3 $(\xi, \xi') \leftarrow S_{\mathrm{v}}(r, r')$;
4 **if** $d(\rho, p') < d(p, p')$ **then**
5 $P_{\mathrm{v}}(r, r') = (\rho, p')$;
6 $S_{\mathrm{v}}(r, r') = (\sigma, \xi')$;

Update_Points maintains two pairs of configuration sam-

ples for each pair of interfaces $i(v,r), i(v,r')$ of a spanner node v so that $\mathtt{L}(r,r') \notin E_S$ (see Figure 2 (bottom right)). The pair $P_{\mathrm{v}}(r,r')$ maintains the samples in the visibility region of v which have the shortest distance among all samples generated on the interfaces $i(v,r)$ and $i(v,r')$. If ρ reveals a distance between $i(v,r)$ and $i(v,r')$ that is shorter than the previous best, the information is updated.

Algorithm 4: `Test_Add_Path` (v, G_S)

```
1  success ← false;
2  for all r ∈ V_S so that L(v,r') ∈ E_G do
3      for all r' ∈ V_S \ r so that L(v,r') ∈ E_G and
       L(r,r') ∉ E_G do
4          (ρ,ρ') ← P_v(r,r');
5          (σ,σ') ← S_v(r,r');
6          d_{r,r'} ← d(ρ,ρ');
7          Π_S ← {π_S(m(v,r), m(v,r'))};
8          for r'' : L(v,r''), L(r',r'') ∈ E_S ∧ L(r,r'') ∉ E_S do
9              Π_S ← Π_S ∪ {π_{G_S}(m(v,r), m(v,r''))};
10         π_S = argmax_{∀π∈Π_S} |π|;
11         if (|π_S| > t * d_{r,r'}) then
12             if L(r,r') ∈ C_free then
13                 E_S ← E_S ∪ L(r,r');
14             else
15                 π_add ← Smooth_Path(
                   {L(σ,ρ), L(ρ,v), L(v,ρ'), L(ρ',σ')});
16                 E_S ← E_S ∪ L(r,σ) ∪ L(σ',r');
17                 Add_Path(G_S, π_add);
18             success ← true;
19 return success;
```

`Test_Add_Path` (Algorithm 4) uses this information to check if a shortest path in C_{free} violates the spanner. For the node currently tested, v, the method checks all pairs of neighbors r, r', so that $\mathtt{L}(r,r') \notin G_S$, finding the path through G_S that connects $m(v,r)$ and $m(v,r')$ (line 7). For reasons detailed in Section V (Lemma 3 - case two), it is necessary to consider all paths between midpoints $m(v,r)$ and $m(v,r'')$, where r'' is a neighbor of v not connected to r but connected to r'. Among all such paths, Π_S, the longest, π_S, is tested if it violates the spanner property relative to $d(\rho,\rho')$ (line 11). If so, the spanner is extended. If a shortcut from r to r' exists in C_{free} (line 12), then only this edge needs to be added to G_S. Otherwise, the algorithm employs a smoothing operation over a path from the sample σ along interface $i(v,r)$ to ρ, then to v, ρ' and eventually σ' along interface $i(v,r')$ (lines 14-17).

V. Properties of SPARS2

SPARS2 and SPARS are equivalent to Visibility PRM in terms of coverage and connectivity; thus, it is possible to argue [5], [6] that SPARS2 is probabilistically complete:

Theorem 1 (Probabilistic Completeness): For all $\rho \in C_{\text{free}} : \exists \mathrm{v} \in V_S$ *so that* $\mathtt{L}(\rho,\mathrm{v}) \in C_{\text{free}}$, and For all $\mathrm{v}, \mathrm{v}' \in V_S$ with a collision-free path in C_{free}, $\exists \pi_S(\mathrm{v},\mathrm{v}')$ which connects them on G_S with probability 1 as M goes to infinity in SPARS2.

The method also makes two assumptions:

Assumption 1: For $v, v' \in C_{\text{free}}$, if the set of configurations q for which $\mathtt{L}(q,v) \in C_{\text{free}}$, $\mathtt{L}(q,v') \in C_{\text{free}}$, $d(q,v) < \Delta$ and $d(q,v') < \Delta$ is non-empty, then it has non-zero measure.

Assumption 2: No sample ρ is within distance cl from obstacles. No $v \in V_S$ is within cl-distance from obstacles.

SPARS and SPARS2 deviate on (a) how they detect pairs of nodes that share an interface (for criterion 3) and (b) estimating shortest paths in C_{free} between interfaces (for criterion 4). Note that interfaces often have complex shape due to obstacles and are hard to explicitly represent.

SPARS2 ensures every two spanner nodes that share an interface also eventually share an edge in lines 14-16:

Theorem 2 (Connected Interfaces): For all $v_1, v_2 \in V_S$ which share an interface, then $\mathtt{L}(v_1,v_2) \in E_S$ with probability 1 as M goes to infinity in SPARS2.

The above theorem is straightforward to prove, and the core of this proof relies on the fact that there is a non-zero measure set of configurations for any pair of vertices, v_1, v_2, which share an interface, such that all points in this set have v_1 and v_2 as their closest guards and are also visible. This implies samples will eventually be generated in this set, and the algorithm will bridge the interface.

SPARS2 relies on sampling rather than the dense graph to pinpoint interfaces, necessary for the last criterion. This increases computation time for construction, which will be exacerbated in high-dimensional spaces, though not maintaining the dense graph helps balance this cost. The following lemmas argue that optimal paths in C_{free} will be represented by G_S, which satisfies the spanner property.

Lemma 1 (Coverage of Optimal Paths by G_S): For optimal path $\pi^*_{cl}(\rho_0,\rho_m)$, the probability of having a sequence of nodes, $V_\pi = (\mathrm{v}_1, \mathrm{v}_2, ..., \mathrm{v}_n) \in G_S$ with the following properties goes to 1 as M goes to infinity:

- $\forall \rho \in \pi^*_{cl}(\rho_0,\rho_m),\ \exists\ v \in V_\pi :\ \mathtt{L}(\rho,v) \in C_{\text{free}}$
- $\mathtt{L}(\rho_0,\mathrm{v}_1) \in C_{\text{free}}$ and $\mathtt{L}(\rho_m,\mathrm{v}_n) \in C_{\text{free}}$
- $\forall \mathrm{v}_i, \mathrm{v}_{i+1} \in V_\pi, \mathtt{L}(\mathrm{v}_i,\mathrm{v}_{i+1}) \in E_S$.

The above lemma follows from Theorems 1 and 2. Consider a decomposition of path $\pi_S(\rho_0,\rho_m)$ in G_S through V_π into sub-paths $\{M_0, M_1, \ldots, M_{m-1}\}$ (Figure 3(right)), where M_i is the path between $m(v_i,v_{i+1})$ and $m(v_{i+1},v_{i+2})$. M_0 connects ρ_0 to v_1, and $m(v_1,v_2)$ and M_{m-1} is the corresponding last segment. The following can be easily shown:

Lemma 2 (Additive Connection Cost): The length of paths M_0 and M_{m-1} for a path through G_S connecting ρ_0 to ρ_m is upper bounded by $4 \cdot \Delta$.

The important property, however, of segments M_i is:

Lemma 3 (Spanner Property of G_S over C_{free}): All M_i have length bounded by $t \cdot |\pi^*_{cl}(\rho_{i-1},\rho_i)|$, where ρ_i lies at the intersection of $\pi^*_{cl}(\rho_0,\rho_m)$ with interface $i(\mathrm{v}_i,\mathrm{v}_{i+1})$.

Proof Figures 4 and 5 support this proof. Given Lemma 1, the edges $\mathtt{L}(\mathrm{v}_{i-1},\mathrm{v}_i)$ and $\mathtt{L}(\mathrm{v}_i,\mathrm{v}_{i+1})$ are in E_S, at least as M goes to infinity. Assuming π^*_{cl} travels through the region of node $\mathrm{v}_i \in V_\pi$, there are three possible cases: (a)

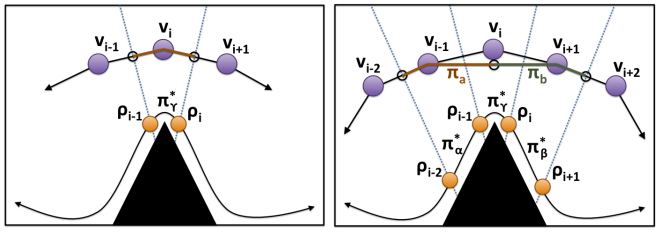

Fig. 4. (left) The path from $i(\mathrm{v}_{i-1},\mathrm{v}_i)$ to $i(\mathrm{v}_i,\mathrm{v}_{i+1})$ via G_S must satisfy the spanner property for path π^*_γ. (right) If there is a path between v_i's neighbors, then the paths $\pi_a = \pi(m(\mathrm{v}_{i-2},\mathrm{v}_{i-1}),\mathrm{v},m(\mathrm{v}_{i-1},\mathrm{v}_{i+1}))$ and $\pi_b = \pi(m(\mathrm{v}_{i+2},\mathrm{v}_{i+1}),\mathrm{v},m(\mathrm{v}_{i+1},\mathrm{v}_{i-1}))$ must be checked against π^*_α and π^*_β.

$\mathrm{L}(\mathrm{v}_{i-1},\mathrm{v}_{i+1}) \notin E_S$ (Figure 4(left)), (b) $\mathrm{L}(\mathrm{v}_{i-1},\mathrm{v}_{i+1}) \in E_S$ (Figure 4(right)), and (c) $\mathrm{L}(\mathrm{v}_{i-1},\mathrm{v}_{i+1}) \in E_S$ and $\mathrm{L}(\mathrm{v}_{i-2},\mathrm{v}_i) \in E_S$ or $\mathrm{L}(\mathrm{v}_i,\mathrm{v}_{i+2}) \in E_S$ (Figure 5).

For case one, SPARS2 checks $|M_i|$ against $|\pi^*_{cl}(\rho_{i-1},\rho_i)|$. SPARS asymptotically approximates the optimal path $\pi^*_{cl}(\rho_{i-1},\rho_i)$ via the dense graph; however, SPARS2 applies a conservative approximation by checking the distance $d(\rho_{i-1},\rho_i)$ (Algorithm 4 lines 4-11) utilizing the information stored in $P_{\mathrm{v}}(r,r')$. If $|M_i| > t \cdot d(\rho_{i-1},\rho_i)$, Algorithm 4 adds a shortcut path: either $\mathrm{L}(\mathrm{v}_{i-1},\mathrm{v}_{i+1})$ leading to proof case two, or a path including σ and σ', which would change the representative sequence V_π.

The second case addresses the issue that $|\pi^*_{cl}(\rho_{i-1},\rho_i)|$ could be arbitrarily close to 0. In lines 8-9 of Algorithm 4, the method compares $d(\rho_{i-1},\rho_i)$ not only with the segment M_i, but also considers all spanner paths from $i(v_{i-1},v_i)$ to the interfaces of all neighbors of v_i, which are not connected to v_{i-1} but share an interface with v_{i+1}. In the context of Figure 4(right), this checks whether path π_a satisfies the spanner property for path π^*_α and that path π_b satisfies π^*_β. If they do, then it does not matter if segment M_{i-1} satisfies the spanner property for path π^*_γ, because π_a and π_b cover all three consecutive subpaths of the optimum path.

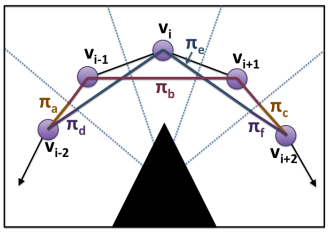

Fig. 5. In the case that v_i depends on its neighbors for checking the spanner property which in turn rely on v_i, it must be that $|\pi_e| < |\pi_b|$.

The last case, illustrated in Figure 5, has two subcases: the solution returned from v_{i-2} to v_{i+2} does so through path $\pi_{a,b,c}$ or through path $\pi_{d,e,f}$. Return path $\pi_{a,b,c}$ is handled by case two of this proof. For path $\pi_{d,e,f}$, it is necessary to show that $|\pi_e| \le t \cdot (|\pi^*_\alpha| + |\pi^*_\beta| + |\pi^*_\gamma|)$. It is known that $|\pi_b| \le t \cdot |\pi^*_\alpha| + |\pi^*_\beta|$ from case two, and that $|\pi_{d,e,f}| \le |\pi_{a,b,c}|$ or it would not have been returned by SPARS2. Furthermore, the construction of these segments enforces $|\pi_a| \le |\pi_d|$ and $|\pi_c| \le |\pi_f|$, as the endpoints are the intersections with $i(\mathrm{v}_{i-2},\mathrm{v}_{i-1})$ for π_a and π_d, and $i(\mathrm{v}_{i+1},\mathrm{v}_{i+2})$ for π_c and π_f. Combining these inequalities yields $|\pi_e| \le t \cdot (|\pi^*_\alpha| + |\pi^*_\beta|) \le t \cdot (|\pi^*_\alpha| + |\pi^*_\beta| + |\pi^*_\gamma|)$. □

Combining the above lemmas yields:

Theorem 3 (Asymptotic Near-Optimality w/Additive Cost): As M goes to infinity in SPARS2: $\forall \rho_0, \rho_m \in C_{\text{free}}$: $|\pi_S(\rho_0,\rho_m)| < t \cdot |\pi^*_{cl}(\rho_0,\rho_m)| + 4 \cdot \Delta$.

An important goal of SPARS2 is to create a sparse data structure, which is supported by the following theorem:

Theorem 4 (Cessation of Node Addition): As the number of iterations of SPARS2 increases, the probability of adding a node to G_S goes to 0.

There are four methods for adding nodes to G_S; thus, all must be shown to have no reason to add nodes to G_S. Nodes for coverage and connectivity will stop being added for reasons identical to SPARS. Theorem 2 argues that SPARS2 stops adding nodes for bridging interfaces. Nodes for ensuring path quality will stop being added due to a check in Algorithm 4, line 12.

VI. Simulations

Simulations were run in the Open Motion Planning Library (OMPL) [7], using the "2D Maze" (SE2) and "Bug-trap"(SE3) environments, over parameters δ = 0.5, Δ = 15, and t=(2,3,5,9), recording the maximum consecutive failures, M. Data points were taken at 1, 2, 4, 8, 16, and 32 minutes and compared with PRM*.

Figure 6 compares path quality of SPARS and SPARS2 relative to the best PRM* solution computed after 32 minutes. Path quality improves over construction time, as expected, but moreover, SPARS2 produces very high-quality paths early on, even for large stretch factors. Furthermore, paths produced by SPARS2 have lower cost than SPARS. Figure 7 (left) measures memory requirements of the final roadmap spanner. SPARS2 results in slightly larger spanners than SPARS, but are orders of magnitude smaller than PRM*. It also shows the average path degradation is significantly smaller than the theoretical guarantees.

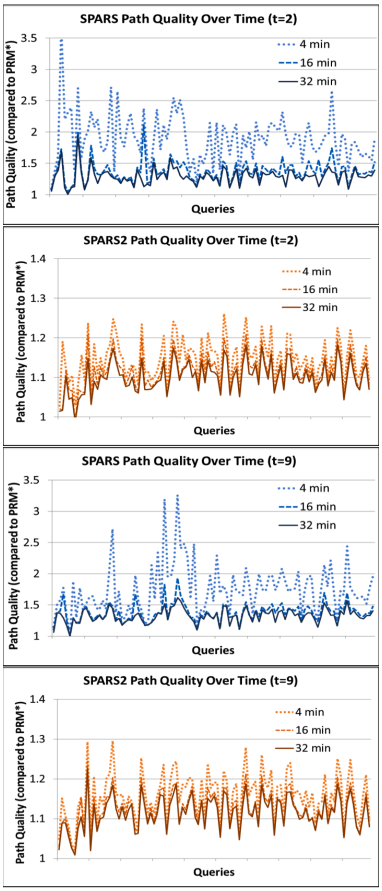

Fig. 6. Performance of SPARS and SPARS2 in terms of path quality relative to the best PRM* path for 100 queries. Stretch factor varies between t = 2(top) to 9(bottom). Construction time varies from 4mins to 32 mins.

The maximum consecutive failures to add nodes through time is shown in Figure 7 (right). This value increases, which is critical for the stopping criterion. Figure 8 details memory requirements for SPARS2 in comparison to SPARS

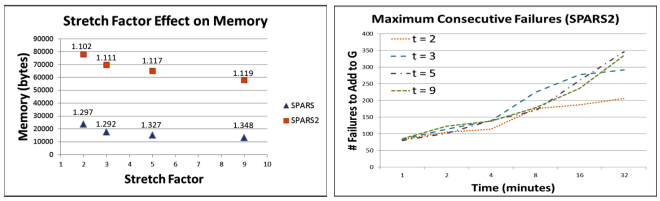

Fig. 7. (left) Effects of the stretch factor on the size of the final roadmap spanner after 32 minutes of construction. Each data point is labeled with the average path quality for the experiment after 32 minutes. The size of the roadmap returned by PRM* was 6,667,616 bytes. (right) Maximum consecutive failures reached.

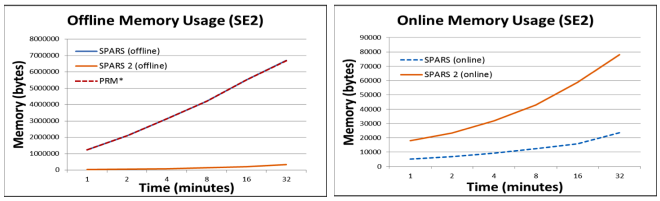

Fig. 8. Memory usage over time in SE2. (right) offline memory usage and (left) online memory requirements. Offline memory usage for PRM* and SPARS almost coincides with SPARS using slightly more.

and PRM*. SPARS2, has a great advantage over the other methods in terms of memory requirements for the offline preprocessing, using close to two orders of magnitude less memory. For online query resolution, the algorithms need only the final planning structure to answer queries, with memory requirements shown in Figure 8 (right).

Figure 9 reports path quality and query resolution times for different construction times in SE(3). Overall, path quality for both SPARS and SPARS2 are shown to improve over time, though SPARS2 returns paths of higher quality than SPARS. The query times for the algorithms correlate to the size of the graph, giving both spanner methods a significant advantage.

VII. Discussion

SPARS2 is presented as an approach for solving path planning problems in continuous configuration spaces, which provides asymptotic near-optimality and greatly reduces memory requirements upon construction versus previous methods while maintaining theoretical guarantees on the rate of node addition. It removes a dependence on a dense graph representation of the space, instead relying on properties of visibility utilized by localized sampling and smoothing processeses. The resulting graph spanner provides high quality paths, which are better than theoretical bounds suggest, even early during its execution.

Open questions in this line of work include: (i) Extending the given properties to graphs with directed edges; a necessary requirement for motion-constrained systems. The current work relies on a BVP solver, which is often unavailable for interesting systems. (ii) Showing G_S converges to a finite structure, as the given properties cannot guarantee this. (iii) It would be exciting to compute a confidence value of attaining near-optimal paths in finite time. (iv) It is relevant to know whether the method finds important homotopic classes compared to other methods which focus

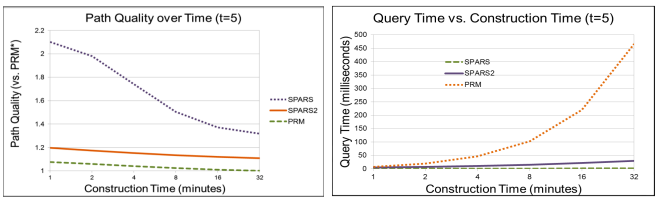

Fig. 9. Path quality at various times during execution(left) and query resolution times (right) in SE(3) for commonly solved queries.

on this issue [15]. (v) The selection of t and Δ to tune results still requires investigation, as the dependence on these parameters as well as δ is still unclear. (vi) Overall optimization of the method to reduce collision-checking calls and other expensive operations will speed up the method.

References

[1] P. Agarwal, "Compact Representations for Shortest-Path Queries," September 2011, appeared at the IROS 2012 Workshop on Progress and Open Problems in Motion Planning.

[2] S. Karaman and E. Frazzoli, "Sampling-based Algorithms for Optimal Motion Planning," *IJRR*, vol. 30, no. 7, pp. 846–894, June 2011.

[3] J. D. Marble and K. E. Bekris, "Computing Spanners of Asymptotically Optimal Probabilistic Roadmaps," in *IEEE/RSJ IROS*, San Francisco, CA, September 2011.

[4] ——, "Asymptotically Near-Optimal is Good Enough for Motion Planning," in *ISRR*, Flagstaff, AZ, August 2011.

[5] A. Dobson, T. D. Krontiris, and K. E. Bekris, "Sparse Roadmap Spanners," in *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, Cambridge, MA, June 2012.

[6] T. Simeon, J.-P. Laumond, and C. Nissoux, "Visibility-based Probabilistic Roadmaps for Motion Planning," *Advanced Robotics Journal*, vol. 41, no. 6, pp. 477–494, 2000.

[7] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012.

[8] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces," *IEEE TRA*, vol. 12, no. 4, pp. 566–580, 1996.

[9] L. E. Kavraki, M. N. Kolountzakis, and J.-C. Latombe, "Analysis of Probabilistic Roadmaps for Path Planning," *IEEE TRA*, vol. 14, no. 1, pp. 166–171, 1998.

[10] D. Hsu, L. Kavraki, J.-C. Latombe, R. Motwani, and S. Sorkin, "On Finding Narrow Passages with Probabilistic Roadmap Planners," in *WAFR*, Houston, TX, 1998.

[11] A. M. Ladd and L. E. Kavraki, "Measure Theoretic Analysis of Probabilistic Path Planning," *IEEE TRA*, vol. 20, no. 2, pp. 229–242, April 2004.

[12] D. Xie, M. Morales, R. Pearce, S. Thomas, J.-L. Lien, and N. M. Amato, "Incremental Map Generation (IMG)," in *WAFR*, New York City, NY, July 2006.

[13] G. Varadhan and D. Manocha, "Star-shaped Roadmaps: A Deterministic Sampling Approach for Complete Motion Planning," *IJRR*, 2007.

[14] D. Nieuwenhuisen and M. H. Overmars, "Using Cycles in Probabilistic Roadmap Graphs," in *IEEE ICRA*, 2004, pp. 446–452.

[15] L. Jaillet and T. Simeon, "Path Deformation Roadmaps," in *WAFR*, New York City, NY, July 2006.

[16] E. Schmitzberger, J. L. Bouchet, M. Dufaut, D. Wolf, and R. Husson, "Capture of Homotopy Classes with Probabilistic Roadmap," in *IEEE/RSJ IROS*, 2002, pp. 2317–2322.

[17] B. Raveh, A. Enosh, and D. Halperin, "A Little More, a Lot Better: Improving Path Quality by a Path-Merging Algorithm," *IEEE TRO*, vol. 27, no. 2, pp. 365–370, 2011.

[18] S. M. LaValle and J. J. Kuffner, "Randomized Kinodynamic Planning," *IJRR*, vol. 20, pp. 378–400, May 2001.

[19] Y. Li and K. E. Bekris, "Learning Approximate Cost-to-Go Metrics To Improve Sampling-based Motion Planning," in *IEEE ICRA*, Shanghai, China, 9-13 May 2011.

[20] S. Baswana and S. Sen, "A simple and linear time randomized algorithm for computing sparse spanners in weighted graphs," *Random Structures and Algorithms*, vol. 30, no. 4, pp. 532–563, Jul. 2007.