Abstract—Asymptotically optimal motion planning algorithms guarantee solutions that approach optimal as more iterations are performed. Nevertheless, roadmaps with this property can grow too large and unwieldy for fast online query resolution. In graph theory there are algorithms that produce subgraphs, known as spanners, which have guarantees about path quality. Applying such an algorithm to a dense, asymptotically optimal roadmap produces a sparse, asymptotically near optimal roadmap. Experiments performed on geometric problems in SE(3) show that a large reduction in roadmap edges can be achieved with a small increase in path length (Fig. 1). Online queries can be answered faster with similar results in terms of path quality. Additional time savings result from applying the spanner algorithm incrementally so edges that do not increase path quality will never be added to the roadmap and will not be checked for collisions.

I. INTRODUCTION

Roadmap planners [1] utilize an off-line phase to build up knowledge about the configuration space (C-space). PRM attempts to connect sampled configurations to either a fixed number, $k$, of nearest neighbors or all configurations within a fixed radius ball centered at the new configuration. Higher quality paths can result from constructing larger, denser roadmaps that better sample $C$ by investing more preprocessing time. However, it has been shown that a PRM using a fixed $k$ will not converge to the optimal path [2].

The $k$-PRM* algorithm minimizes the number of neighbors each new sample has to be connected to while increasing path quality over time (i.e., asymptotic optimality) by making $k$ a logarithmic function of the roadmap size. Roadmaps constructed with this variation will almost surely converge to an optimal solution [2]. While such roadmaps are desirable for their high path quality, their large size is problematic. Large roadmaps impose significant costs during storage, transmission, and query resolution. In order to relax optimality requirements while producing sparser roadmaps it is possible to use graph spanners, a tool from graph theory [3].

A graph spanner is a sparse subgraph. Given a weighted graph $G = (V, E)$, a subgraph $G_S = (V, E_S \subseteq E)$ is a $t$-spanner if, for all pairs of vertices $(v_1, v_2) \in V$, the shortest path between them in $G_S$ is no longer than $t$ times their shortest path in $G$. Because $t$ specifies the amount of additional length allowed, it is known as the stretch factor of the spanner. There are many algorithms in the literature for constructing graph spanners, each with their own properties.

This work presents two approaches for integrating $k$-PRM* with graph spanners. In the first approach, the randomized $(2k - 1)$-spanner algorithm [4] was chosen for application after a dense roadmap was constructed. This alternative provides optimal stretch along with reasonable size and linear time complexity. However, this algorithm operates on the entire graph and cannot be applied incrementally as the dense graph is being constructed. In the second approach, an algorithm with slightly worse time complexity is interleaved with roadmap construction so that the entire dense roadmap need not be held in memory all at once.

II. SEQUENTIAL APPROACH

The sequential approach [5] can be broken down into two parts. First, a dense roadmap is constructed using $k$-PRM* and then passed through the randomized $(2k - 1)$-spanner algorithm. This makes the roadmap sparse by removing edges that do not contribute much to path quality. Without the path quality guarantees provided by $k$-PRM*, there would be no guarantees on the path quality of the sparse roadmap.

Results for 100 random queries on roadmaps with 20,000 vertices show significant reduction in roadmap density and query resolution time with small increases in path quality.

1) Space Requirements: While each spanner contains the same vertices as the original roadmap, the space required for connectivity is reduced by up to 85%. Environments with a more connected free space had a larger reduction in the number of edges because fewer are needed for connectivity.

2) Path Quality: The length of the resulting paths increases as the number of edges in the spanner is reduced. For random start and goal configurations, the extra cost is smaller than the worst case guaranteed by the spanner algorithm.

Path quality degradation in the spanner roadmap is plotted in Fig. 3 as a function of the original path length and $k$. The worst degradation happens for short paths, where taking a detour of even a single vertex can increase the path length
This criterion is applied during times the weight of the proposed edge. If so, the edge is an existing path between two neighbors is no more than \[t\]. A length-limited graph search is used to determine if the expensive collision check step for this edge is avoided.

Before an edge is checked for collision, if an edge is filtered, the naıve greedy spanner algorithm \([3]\) and Useful Cycles \([7]\) were originally short. The mean increase is less than 25%.

III. INCREMENTAL APPROACH

The Incremental Roadmap Spanner (IRS) \([6]\) takes the idea of the sequential approach one step further by interleaving roadmap and spanner construction. When the algorithm adds an edge, the spanner algorithm can reject it before the edge is checked for collision and added to the roadmap.

The method used to filter edges is the same used by the the naıve greedy spanner algorithm \([3]\) and Useful Cycles \([7]\). A length-limited graph search is used to determine if an existing path between two neighbors is no more than \(t\) times the weight of the proposed edge. If so, the edge is rejected. This criterion is applied during \(k\)-PRM\(^*\) construction before an edge is checked for collision. If an edge is filtered, the expensive collision check step for this edge is avoided.

Otherwise, the \(k\)-PRM\(^*\) algorithm is unchanged. The density of the roadmap is reduced, resulting in lower space requirements and query resolution time. Additionally, construction time is reduced by up to 50 percent. Fig. 1 shows the result of applying this approach in one environment.

For each query, there was a simple attempt at path smoothing. Nonconsecutive vertices on the path that were near each other were tested for connectivity. If they could be connected, then the path is shortened by removing intervening vertices. This greedy and local method for smoothing brought the quality of paths on the sparse graph closer to that on the dense graph making the IRS approach even more favorable.

IV. DISCUSSION

This work shows that it is practical to compute sparse roadmaps in \(C\)-spaces that guarantee asymptotically near optimal paths. The experiments suggest that these roadmaps have considerably fewer edges than \(k\)-PRM\(^*\), while resulting in little degradation in path quality. The stretch factor parameter allows this trade-off to be tuned (Fig. 2).

Vertices that do not improve path quality could be removed, but previous work on this idea provides no guarantees on the resulting path quality \([8]\). Future work will investigate how to filter nodes while providing path quality guarantees.

Finally, it is important to study whether roadmap spanners can guarantee the preservation of the homotopic path classes \([9]\). Homotopic classes tend to be preserved by the spanner because their removal has significant effects in path quality.

REFERENCES