From: Eli Case
Date: October, 12, 2023
Subject: MECH 450 – Homework 3

---

1. (a) By inspection, the set F/S has points with the lowest ϵ fraction. Within F/S, one can estimate the lowest ϵ fraction as 0.15-good. This value can be found by computing the ratio $vol(F/S)/vol(S)$.

$$\begin{aligned}\frac{vol(F/S)}{vol(S)} &= \frac{(0.1)^2 + (0.1)(0.9) + (0.5)(0.1)(0.9)}{1.0} \\ &\approx 0.15\end{aligned}$$

Thus, we conclude that the entire space is 0.15-good.

(b) This is not true due to the configuration of S. The 0.5-lookout of S is approximately the square given by side lengths a. Using this area to compute α, we obtain $\alpha \approx 0.16$. Thus, we have that the 0.5-lookout of S is a 0.16 fraction of S.

(c) In a space with low $(\epsilon, \alpha, \beta)$-expansiveness, visibility-based sampling would be effective. Visibility based sampling generates samples that belong to a new roadmap component or connect two existing roadmap components. In a space with low $(\epsilon, \alpha, \beta)$ expansiveness, there are potentially many roadmap components that need to be connected, and visibility-based sampling samples in a manner that should sample in as many roadmap components as possible and generate valid connections between them. Additionally, a bridge-based sampling technique would also be effective for a space with low $(\epsilon, \alpha, \beta)$ expansiveness. This is because bridge-based sampling would generate samples in narrow passageways that could connect roadmap components that have low visibility to each other.

(d) If this space is extruded to 3 dimensions, the $(\epsilon, \alpha, \beta)$ values will remain unchanged. The expansiveness values in each slice are unchanged since the $(\epsilon, \alpha, \beta)$ parameters will be the same across each 2D slice. Additionally, since the parameters are defined relative to the volume of the entire space, the uniform expansiveness properties in the extruded direction will yield the same value with respect to the entire volume of the space.

2. (a) Given a roadmap R' in F', we can "push" the nodes into F by randomly selecting a node in F and then selecting the k nearest neighbors in F' to move toward the randomly selected node in F. For each selection of the k nearest neighbors in F', we can step them towards the randomly selected node in F at a desired resolution and apply a transformation to contract the nodes together by a predetermined amount at each step. Once all the k nearest neighbors in F' are in F, we repeat the process for another randomly selected node in F and the k nearest neighbors in F'. We can repeat this process until all the nodes originally in F now lie in F'.

(b) We can push the edges into F by retaining the original connectivity in between nodes in F'. Another method could be to reconnect the nodes from F' once they are moved into F. We could have a list of nodes without edges that correspond to nodes originally in F'. We could connect these nodes together based on the spatial proximity to each other, while checking for collision. We can these choose nodes at random to connect to the original roadmap, for a desired number of iterations.

(c) This idea could be applied to spaces with low expansiveness. One major issue of generating a roadmap in low expansive space is special sampling techniques are require to generate a roadmap with sufficient coverage and connectivity. If the space is dilated by some δ_k successively, a k could be found so that the expansiveness of the space corresponds to a roadmap of sufficient coverage and connectivity. At this level of k, a roadmap can be generated with a standard uniform sampling method and then push back into the original space. This approach would decrease the problem difficulty of selecting a correct sampler in a space with low expansiveness, but could potentially have a higher time and memory complexity to execute. Thus, in the case of a computation on a system without many hardware constraints, this approach could be effective to reduce the complexity of sampling.

3. Using PRM, we use a sampling algorithm to sample nodes in the configuration-space and then do a collision checking procedure to draw edges between a node and its k nearest neighbors. With the visibility graph method, the nodes of the graph correspond to obstacle vertices, and the graph edges are generated by successively connecting nodes that are visible from a given node.

Generally, the visibility graph is more dependent on the environment than PRM. Notably, the visibility graph is also more dependent on the types of obstacles present in the environment, as it requires that there be well-defined vertices and edges on the obstacles, which may require additional processing, i.e. creating convex hulls for obstacles. Additionally, a visibility graph likely requires more post-processing one a shortest path is found, since the graph nodes are obstacle corners and cannot be easily used in the actual path output by the planner.

Contrarily, in expansive spaces, a visibility graph can be more efficient since it will only place nodes at the obstacle vertices, meaning a visibility graph will have less nodes than a graph generated via random sampling with PRM in this situation.

From the above discussion, a high-dimensional space with narrow passageways between obstacles would be more efficient conquered with PRM rather than a visibility graph. This is because it is likely that in this space, there is poor visibility between obstacle edges, meaning one needs more points between obstacles to generate a roadmap with sufficient connectivity. Contrarily, in a low-dimensional space that has high expansiveness, a visibility graph will likely generate a graph that has less states than PRM, but has comparable coverage and connectivity.

4. (a) In this case, we have a high-dimensional configuration space with static objects. Thus, we could make use of an SRT planner to efficiently build a roadmap in the high-dimensional space that can then be used to solve the multiple queries given to it with the initially constructed roadmap, since the environment is static. The SRT planner will be particularly effective for efficient sampling in a high-dimensional space, and connecting the various trees will save having to generate a large number of samples.

(b) We can use a RRT planner with Voronoi bias for the base, as this planner will ensure that the base has sufficient clearance in order for the manipulator to move effectively. For the manipulator we can use Lazy PRM to construct a roadmap efficiently in a high-dimensional space. Once we have the base in a position where the manipulator can move, we can form collision checking. Constructing a roadmap for the manipulator will allow for multiple start-to-goal queries to be quickly answered as the base moves to different locations in space specified by its planner.

(c) In this case, a planner like RRT with Voronoi bias to guide tree growth would likely be efficient. Voronoi bias biases towards paths that maximizes clearance between all obstacles within a given visibility radius. Each path request could generate a tree from start to goal, and using Voronoi bias would ensure a large path clearance between obstacles as the tree grows from the start to goal point.

(d) Since the environment is static, we could use PRM to construct a roadmap. Since it is a car-like robot that is not necessarily high-dimensional, PRM should provide sufficiently quick sampling to create a roadmap that can then be use to solve multiple start-to-goal queries.