Roadmaps

Oren Salzman

Synonyms

Definitions

A *roadmap* $\mathcal{G}$ is a network or a graph embedded in the free space $\mathcal{C}_{\text{free}}$. Its vertices correspond to collision-free configurations and the edges correspond to one-dimensional curves in $\mathcal{C}_{\text{free}}$. The roadmap has the following properties:

- *Reachability*: There is a simple procedure to move from any collision-free configuration to a placement on the roadmap.
- *Connectivity*: The portion of the roadmap within each connected component of the free space is nonempty and connected.

Overview

In this essay we consider the problem of *complete* motion planning. Namely, we are interested in planners that are guaranteed to compute a solution, if one exists and to correctly report if no such solution exists otherwise. This is done by abstracting the continuous configuration space ($\mathcal{C}$-space) into a discrete graph-like data structure embedded in $\mathcal{C}_{\text{free}}$. This is needed because reasoning over the continuous structure of the free space is highly non trivial. Throughout this article, we assume that we have access to an explicit representation of the $\mathcal{C}$-space obstacles and that the $\mathcal{C}$-space is $\mathbb{R}^k$.

Oren Salzman (http://www.orensalzman.com/)
Carnegie Mellon University 5000 Forbes Ave, Pittsburgh, PA 15213 USA, e-mail: osalzman@andrew.cmu.edu

Roadmap-based methods are some of the first complete techniques for motion planning. As we will see, they are usually limited (from a practical point of view) to low dimensions, but can be very powerful if used correctly. Moreover, this approach inspired modern state-of-the-art tools for robot motion planning such as sampling-based algorithms.

Roadmaps are typically *precomputed* in an offline phase and then, given a query in the form of start and target collision-free configurations x_{start} and x_{goal}, respectively, the motion-planning problem can be solved by (i) computing a plan from x_{start} to a point on the roadmap $\mathcal{G}$ (ii) computing a plan on $\mathcal{G}$ and (iii) computing a plan to x_{goal} from a point on $\mathcal{G}$. Generally speaking, steps (i) and (iii) should be a straightforward task that involve little-to-no planning. Since the roadmap is embedded in $\mathcal{C}_{\text{free}}$, there is no need to account for collision detection when computing a plan on the roadmap. Thus, step (ii) may be seen as a pure graph-search step where standard path-finding algorithms such as Dijkstra (Dijkstra 1959), A* (Hart et al 1968) and their many variants can be used.

Roughly speaking, roadmaps differ with respect to the dimension of the underlying $\mathcal{C}$-space for which they can be used and the optimization metric that they consider. *Visibility graphs* and *Voronoi diagrams* are typically used for two-dimensional spaces. These roadmaps allow to compute paths of minimal Euclidean length (Visibility graphs) and maximal clearance (Voronoi diagrams). Finally, the *silhouette method* is a general-purpose approach to construct roadmaps that can be used for high-dimensional spaces but is typically considered too difficult to be implemented in practice.

For additional results on roadmaps, see (Latombe 1991, C. 4), (Choset et al 2005, C.5), (Halperin et al 2017) and (Berg et al 2008, C. 7,15).

Key Research Findings

Visibility maps

Let $\mathcal{O} = \{O_1, \ldots, O_m\}$ be a set of simple pairwise interior-disjoint polygons having n vertices in total representing the $\mathcal{C}$-space obstacles. Here, we consider the obstacles as open sets. Namely, $\mathcal{C}_{\text{free}}$ includes its boundaries. This means that the robot is allowed to "touch" or "graze" the obstacles, but it is not allowed to penetrate them. A *semi-free* path is defined as a path that is in the closure of $\mathcal{C}_{\text{free}}$.

The visibility graph $\mathcal{G}^{\text{vis}} = (\mathcal{V}^{\text{vis}}, \mathcal{E}^{\text{vis}})$ of $\mathcal{O}$ is a roadmap represented as an undirected graph embedded in $\mathcal{C}_{\text{free}}$ defined on the set of the obstacle vertices. Its set of edges consists of those pairs of vertices that are mutually visible. Two vertices are *mutually visible* if the straight-line segment connecting them does not intersect the interior of any of the polygons in $\mathcal{O}$. In this case, we call this segment a *visibility edge*.

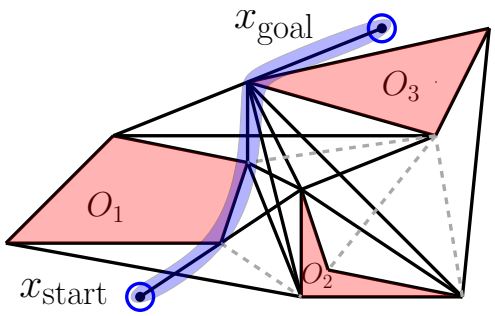

Fig. 1: Visibility graph defined over a set of three obstacles. The reduced visibility edges are depicted in black, the visibility edges which are not reduced are depicted in dashed gray. A shortest path computed using the visibility graph is highlighted in blue.

The visibility graph (consisting of the vertices and all the visibility edges) can be used to compute shortest paths amidst the $\mathcal{C}$-space obstacles (Berg et al 2008, C. 15). Each edge is given a weight equal to the Euclidean distance between its two end-vertices. To find a shortest path between a source x_{start} and a goal x_{goal}, one simply needs to add x_{start} and x_{goal} as vertices in $\mathcal{G}^{\text{vis}}$ together with all visibility edges incident to x_{start} and x_{goal}.

In fact, to compute shortest paths, it is sufficient to consider only the edges that are bitangent to the polygons they connect, namely edges that can be infinitesimally extended in both directions without penetrating any polygon. Such bitangent edges are called *reduced* visibility edges (see Fig. 1). We refer to the set of reduced visibility edges as $\tilde{\mathcal{E}}^{\text{vis}} \subseteq \mathcal{E}^{\text{vis}}$ and to the reduced visibility graph as $\tilde{\mathcal{G}}^{\text{vis}} = (\mathcal{V}^{\text{vis}}, \tilde{\mathcal{E}}^{\text{vis}})$. The path constructed using $\tilde{\mathcal{G}}^{\text{vis}}$ is the shortest path between x_{start} and x_{goal}.

The (reduced) visibility graph can be constructed in a straightforward manner by testing every line segment connecting two obstacle vertices if it is a visibility edge or not. There are $O(n^2)$ such candidate edges and each test can be done in $O(n)$ time by checking for collision with every obstacle edge yielding a running time of $O(n^3)$. The running time can be further improved and the (reduced) visibility graph can be computed in $O(|\mathcal{E}^{\text{vis}}| + n \log n)$ time (Ghosh and Mount 1991).

The visibility graph can be extended to the case where the obstacles include circular arcs. In this case, $\mathcal{G}^{\text{vis}}$ is referred to as the *generalized (reduced) visibility graph* and it consists of every visible bi-tangent of two circular arcs of the boundary of $\mathcal{O}$, every two mutually visible vertices and every visibility-edge between a vertex and a point tangent to a circular arc. Constructing the generalized (reduced) visibility graph can be done in $O(n^2 \log n)$ time (Wein et al 2007).

While the visibility graph can be extended to higher-dimensional settings, the approach is no longer guaranteed to compute shortest paths. Intuitively, this is because even in three-dimensional spaces shortest paths may touch obstacles along edges and *not* only along vertices. See, Fig. 2.

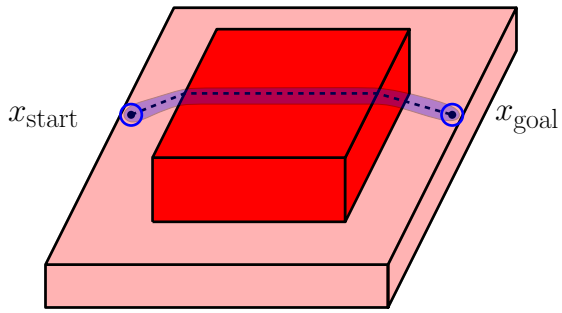

Fig. 2: Shortest path in a three-dimensional space does not necessarily pass through obstacle vertices.

Voronoi Diagrams

The path created using the visibility graph is the shortest path between the source and the target configurations. As such, the robot's *clearance* (namely the minimal distance from the obstacles while executing a path produced by a visibility diagram) can be zero in the general case. This may be undesirable in some applications, e.g., since a small deviation from the path may lead to collision.

A *maximum clearance roadmap* tries to keep the robot as far from the obstacles as possible. For a point robot this means moving along the bisectors of the *Voronoi Diagram* (Fig. 3): Similar to the previous setting, set $\mathcal{O} = \{O_1, \ldots, O_m\}$ to be a set of m obstacles. We refer to the m obstacles as *Voronoi sites*. The *Voronoi diagram* $\text{Vor}(\mathcal{O})$ is the partition of space into maximally connected cells where each cell consists of the points that are closest to one particular site (or a set of sites). These cells are referred to as *Voronoi cells*. The *bisector* $B(O_i, O_j)$ of two Voronoi sites O_i and O_j is the locus of points that have an equal distance to both sites. That is

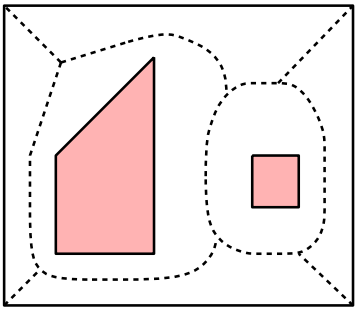

Fig. 3: Voronoi diagram (dashed edged) of two obstacles (bounding square is considered as an additional obstacle).

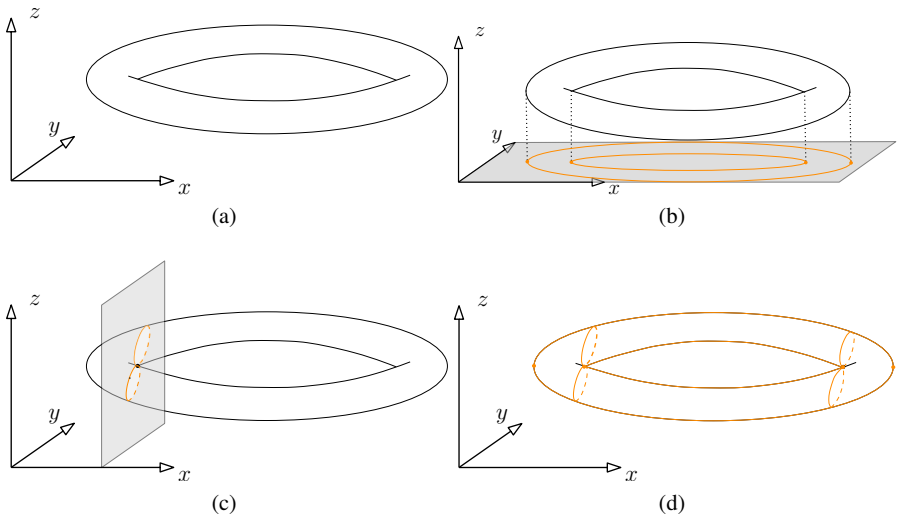

Fig. 4: Visualization of the silhouette method. (a) Free space represented by a solid torus in $\mathbb{R}^3$. (b) Projection of the torus to the x,y- plane to compute the silhouette curves (orange curves) together with the critical values (orange dots). (c) Recursive projection to the y,z-plane at one of the critical values. (d) Resultant roadmap.

$$B(O_i,O_j) = \{p \mid \|O_i,p\| = \|O_j,p\|\}.$$

An edge in the Voronoi diagram is thus part of a bisector of two sites such that there does not exist a third site closer to any point on the edge.

The Voronoi diagram can be used to compute paths of maximal clearance from the obstacles. The source and target configurations are added to the roadmap by connecting them to the closest bisector. It can be shown that the total complexity of a planar Voronoi diagram is $O(n)$ where n is the total number of vertices in $\mathcal{O}$ and that it can be constructed in $O(n\log n)$ time (see e.g., (Aurenhammer and Klein 1999; Yap 1987)).

Note: Voronoi diagrams are a specific instance of a genereal appraoch to compute roadmaps called *retraction methods*. For further details, see (Halperin et al 2017), (Latombe 1991, C. 4) and (Choset et al 2005, C.5).

Silhouette method (Canny's roadmap algorithm)

Canny (1988; 1993) introduced the first method to construct a roadmap for a general high-dimensional $\mathcal{C}$-space. In contrast to the previous methods where we assumed that we represented obstacles as polygons, here we represent them using the more general model of a *semi algebraic set* $\mathcal{F}$. A semi algebraic set is a subset of $\mathbb{R}^k$

defined by a Boolean combination of polynomial equalities and inequalities in the k coordinates. See (Basu and Mishra 2017) and (Latombe 1991, C. 3).

Canny's approach constructs the roadmap recursively, as follows. We projects $\mathcal{F}$ onto some generic 2-plane, and compute the *silhouette* of $\mathcal{F}$ under this projection (namely, the extremal points traced out by the plane when swept across $\mathcal{F}$). Next, the *critical values* of the projection of the silhouette on some line are found (points where the connectivity of the silhouette curves change). The roadmap is then constructed recursively within each slice of $\mathcal{F}$ at each of these critical values. The resulting "sub-roadmaps" are then merged with the silhouette, to obtain the desired roadmap. The running in time of the algorithm is $O\left(n^k \cdot (\log n) \cdot d^{O(k^4)}\right)$ where k is the dimension of the $\mathcal{C}$-space (number of degrees of freedom of the robot), d is the maximum algebraic degree of any polynomial function in $\mathcal{F}$ and n is the number of polynomial functions in $\mathcal{F}$.

The approach requires assumptions on the structure of the semi algebraic set (general-position assumption) as well as tools from Differential Geometry (stratifications) and Elimination Theory (multivariate resultant). While the exact details are out of the scope of this entry, we visualize it using one illustrative example (Fig. 4) adapted from (Latombe 1991).

Examples of Applications

Motion planning for a polygonal robot

When we described roadmaps for low-dimensional spaces, we assumed that we had an explicit representation of obstacles in the $\mathcal{C}$-space (or that we are planning for a point robot). To extend these tools from point robots to translating polygonal robots (such as Roombas (iRobot 2019)) we will use *Minkowski sums* (seep Halperin et al (2017) and references within). Given two sets $P, Q \in \mathbb{R}^2$, their Minkowski sum, denoted $P \oplus Q$, is their point-wise vector sum, namely the set $P \oplus Q = \{p + q \mid p \in P, q \in Q\}$. Let R be a polygonal robot such that the origin lies in R and O be a polygonal obstacle. The robot intersects the obstacle if the origin lies inside the Minkowski sum of the obstacle and a reflection of R through the origin, namely in $M = -R \oplus O$. Thus, given a set of obstacles $\mathcal{O}$, the set of placements of the robot that will cause a collision with an obstacle is simply $\bigcup_{O \in \mathcal{O}} O \oplus -R$ (see, e.g., (LaValle 2006, C. 4)). We can now construct a roadmap (Viisibility graph or Voronoi diagram) over this set to compute minimal length or maximum clearance paths.

Visibility maps have been used to construct minimal-length path for a polygonal robot anchored to a fixed base point by a finite tether translating among polygonal obstacles in the plane. Tethers are often used to provide energy to robots or as a communication link for tele-operation. Examples of scenarios where such tethered robots are used can be found in underwater or disaster recovery missions (Christ and Wernli 2013; Pratt et al 2008). The finite length of the tether, as well as its position,

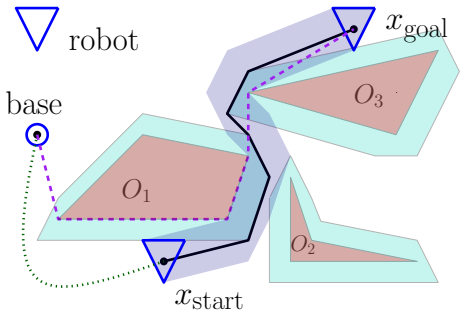

Fig. 5: Shortest path for a tethered triangular robot R. Obstacles are depicted in red, inflated obstacles ($\bigcup_{O\in\mathcal{O}} O \oplus -R$) are depicted in turquoise, initial and final placements of the tether are drawn in dotted green and dashed purple, respectively. Finally, the path traversed by the center of the robot is drawn in black while the area swept by the robot along this path is depicted in blue. Notice the difference in the paths traversed by the center of robot and by the tether.

need to be accounted for when planning a path for the robot. Figure 5 visualizes a shortest path computed for a tethered robot using visibility maps (Salzman and Halperin 2015).

Planning high-clearance short paths

Recall that the visibility map and the Voronoi diagram allow to compute minimal-length paths and maximal-clearance paths, respectively. However, in practical applications we wish to compute a short path between two points while also maintaining a high clearance. To this end, Wein et al (2008) considered the problem of computing shortest paths where every point has clearance at least δ for some parameter $\delta > 0$. Such paths are computed using a new roadmap called the *Visibility-Voronoi diagram* for clearance δ which is a hybrid between the visibility graph and the Voronoi diagram of polygons in the plane. A set of polygonal obstacles can be preprocessed into a data structure called the *Visibility-Voronoi complex* which encodes Visibility-Voronoi diagram for all possible clearance values. This allows to efficiently plan paths for any start and goal configuration and any clearance value δ. The preprocessing time is $O(n^2 \log n)$, where n is the total number of obstacle vertices. The Visibility-Voronoi complex has applications beyond planning for polygonal robots. For example, Karamouzas et al. (2009) suggest to use the Visibility-Voronoi complex to perform crowd simulation.

Unfortunately, the Visibility-Voronoi diagram does not quantify the tradeoff between the length and the clearance of a path, and optimal paths may be very long. Wein et al. (2008) suggested a cost function that balances between minimizing the path length and maximizing its clearance. Specifically, optimal paths minimize the reciprocal of the clearance integrated over the length of the path. Intuitively, such

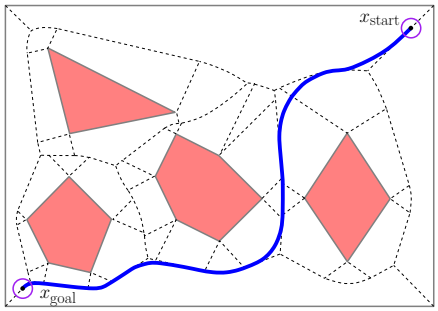

Fig. 6: Balancing length and clearance. Optimal path for the cost function suggested by Wein et al. (2008) (blue) among polygonal obstacles (red). The Voronoi diagram of the obstacles is depicted by dashed black lines.

paths lie on the Voronoi diagram but take "shortcuts" when possible. See Fig. 6 and (Wein et al 2008; Agarwal et al 2018) for further details.

Beyond complete planners

While the practical application of roadmaps may be somewhat limited, the theoretical guarantees that they exhibit are often used to analyze alternative approaches or to approximate the structure of a high-dimensional $\mathcal{C}$-space.

Hoffmann et al. (2008) use the visibility graph to generate a 2D initial guess to navigate an unmanned aerial vehicle (UAV) within a polygonal environment. Kuwata & How (2004) use the visibility graph to perform limited-horizon planning for UAVs and Nuske *et al*. (2015) construct an approximate 3D visibility graph as a heuristic to guide UAVs. Finally, Bullo *et al*. (2011) employ Voronoi diagrams as a tool for dynamic vehicle routing where we are interested in planning optimal multi-vehicle routes to perform different tasks that are generated over time.

From a theoretic point of view, Voronoi diagrams are often used to explain the behavior of sampling-based roadmaps such as RRT (LaValle and Kuffner 2001) and its many variants (see, e.g., (Karaman and Frazzoli 2011; Solovey et al 2016)). Roughly speaking, the sizes of cells in the Voronoi diagram of a sampling-based roadmap are used to argue about the probability of extending a certain node in the roadmap.

Future Directions for Research

Roadmaps provide a powerful tool for motion planning as they allow to devise *complete* algorithms. They have been studied for decades and efficient algorithms exist

for constructing many different roadmaps. However, their practical applications are limited to low-dimensional, relatively simple $\mathcal{C}$-spaces.

A key question for future research is

How can we apply roadmap methods to complex robotic systems?

One possible approach is to combine the aforementioned geometric methods for exact and complete analysis of low-dimensional $\mathcal{C}$-spaces, together with practical, considerably simpler sampling-based approaches that are appropriate for higher dimensions (see, e.g., (Salzman et al 2013, 2015)).

Another possible approach is to extend the low-dimensional roadmpas to low-dimensional complex systems such as systems with curvature constraints or systems with dynamics.

Cross-References

Graph Representations/Graph Representations; Mobile Ground Robots/Navigation of Mobile Robots

References

Agarwal PK, Fox K, Salzman O (2018) An efficient algorithm for computing high-quality paths amid polygonal obstacles. ACM Trans Algorithms 14(4):46:1–46:21

Aurenhammer F, Klein R (1999) Handbook of Computational Geometry, Elsevier Publishing House, chap 5, pp 201–290

Basu S, Mishra B (2017) Computational and quantitative real algebraic geometry. In: Csaba D Toth JEG Joseph O'Rourke (ed) Handbook of Discrete and Computational Geometry, 3rd edn, CRC Press, Inc., chap 37, pp 969–1002

Berg Md, Cheong O, Kreveld Mv, Overmars M (2008) Computational Geometry: Algorithms and Applications, 3rd edn. Springer

Bullo F, Frazzoli E, Pavone M, Savla K, Smith SL (2011) Dynamic vehicle routing for robotic systems. Proceedings of the IEEE 99(9):1482–1504

Canny J (1988) The complexity of robot motion planning

Canny J (1993) Computing roadmaps of general semi-algebraic sets. The Computer Journal 36(5):504–514

Choset H, Lynch KM, Hutchinson S, Kantor G, Burgard W, Kavraki LE, Thrun S (2005) Principles of Robot Motion: Theory, Algorithms, and Implementation. MIT Press

Christ R, Wernli R (2013) The ROV Manual: A User Guide for Remotely Operated Vehicles. Elsevier Science

Dijkstra EW (1959) A Note on Two Problems in Connexion with Graphs. Numerische Mathematik 1(1):269–271

Ghosh SK, Mount DM (1991) An output-sensitive algorithm for computing visibility graphs. SIAM Journal on Computing 20(5):888–910

Halperin D, Salzman O, Sharir M (2017) Algorithmic motion planning. In: Csaba D Toth JEG Joseph O'Rourke (ed) Handbook of Discrete and Computational Geometry, 3rd edn, CRC Press, Inc., chap 50, pp 1307–1338

Hart PE, Nilsson NJ, Raphael B (1968) A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems, Science, and Cybernetics 4(2):100–107

Hoffmann G, Waslander S, Tomlin C (2008) Quadrotor helicopter trajectory tracking control. In: AIAA guidance, navigation and control conference and exhibit, p 7410

iRobot (2019) Roomba vacum cleaning robot. `https://www.irobot.com/for-the-home/vacuuming/roomba`, accessed: 2019-02-07

Karaman S, Frazzoli E (2011) Sampling-based algorithms for optimal motion planning. Int J Robotics Research (IJRR) 30(7):846–894

Karamouzas I, Geraerts R, Overmars MH (2009) Indicative routes for path planning and crowd simulation. In: Int. Joint Conf. on Foundations of Digital Games (FDG), pp 113–120

Kuwata Y, How J (2004) Three dimensional receding horizon control for UAVs. In: AIAA Guidance, Navigation, and Control Conference and Exhibit, pp 2100–2113

Latombe JC (1991) Robot Motion Planning. Kluwer Academic Publishers

LaValle SM (2006) Planning Algorithms. Cambridge University Press

LaValle SM, Kuffner JJ (2001) Randomized kinodynamic planning. Int J Robotics Research (IJRR) 20(5):378–400

Nuske S, Choudhury S, Jain S, Chambers A, Yoder L, Scherer S, Chamberlain L, Cover H, Singh S (2015) Autonomous exploration and motion planning for an unmanned aerial vehicle navigating rivers. J Field Robotics 32(8):1141–1162

Pratt KS, Murphy RR, Burke JL, Craighead J, Griffin C, Stover S (2008) Use of tethered small unmanned aerial system at Berkman plaza II collapse. In: Safety, Security and Rescue Robotics, pp 134–139

Salzman O, Halperin D (2015) Optimal motion planning for a tethered robot: Efficient preprocessing for fast shortest paths queries. In: IEEE Int. Conf. Robotics and Automation (ICRA), pp 4161–4166

Salzman O, Hemmer M, Raveh B, Halperin D (2013) Motion planning via manifold samples. Algorithmica 67(4):547–565

Salzman O, Hemmer M, Halperin D (2015) On the power of manifold samples in exploring configuration spaces and the dimensionality of narrow passages. IEEE Transactions on Automation Science and Engineering 12(2):529–538

Solovey K, Salzman O, Halperin D (2016) Finding a needle in an exponential haystack: Discrete RRT for exploration of implicit roadmaps in multi-robot motion planning. I J Robotics Res 35(5):501–513

Wein R, van den Berg J, Halperin D (2007) The visibility-Voronoi complex and its applications. Computational Geometry: Theory and Applications 36(1):66–78

Wein R, van den Berg JP, Halperin D (2008) Planning high-quality paths and corridors amidst obstacles. Int J Robotics Research (IJRR) 27(11-12):1213–1231

Yap CK (1987) An $O(n \log n)$ algorithm for the voronoi diagram of a set of simple curve segments. Discrete & Computational Geometry 2:365–393