5 *Roadmaps*

AS DESCRIBED in chapters 2 and 4, a planner plans a path from a particular start configuration to a particular goal configuration. If we knew that many paths were to be planned in the same environment, then it would make sense to construct a data structure once and then use that data structure to plan subsequent paths more quickly. This data structure is often called a *map,* and *mapping* is the task of generating models of robot environments from sensor data. Mapping is important when the robot does not have *a priori* information about its environment and must rely on its sensors to gather information to incrementally construct its map. In the context of indoor systems, three map concepts prevail: topological, geometric, and grids (see figure 5.1).

Topological representations aim at representing environments with graphlike structures, where nodes correspond to "something distinct" and edges represent an adjacency relationship between nodes. For example, places may be locations with specific distinguishing features, such as intersections and T-junctions in an office building, and edges may correspond to specific behaviors or motion commands that enable the robot to move from one location to another, such as wall-following. Recently, it has become popular to augment topological maps with metric information (e.g., relative distance, angle) to help disambiguate places that "look" the same [108, 250, 382, 418] or to use them for navigation [188, 213, 240, 339].

Geometric models use geometric primitives for representing the environment. Mapping then amounts to estimating the parameters of the primitives to best fit the sensor observations. In the past, different representations have been used with great success. Many researchers use line segments [27, 122, 169, 180, 334] to represent parts of the

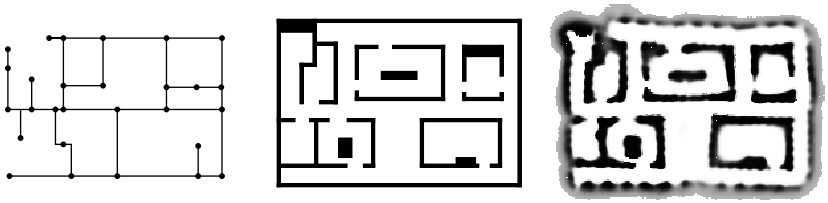

Figure 5.1 Different ways to represent an environment: topologically, geometrically, and using grids.

environment. Popular approaches also represent three-dimensional structures of the environment with triangle meshes [17, 161, 182, 416].

Finally occupancy grids are grid structures, similar as those described in chapter 4, where the value of each pixel corresponds to the likelihood that its corresponding portion of workspace or configuration space is occupied [142]. Occupancy grids were first introduced for mapping unknown spaces with wide-angle ultrasonic sensors; this topic is discussed in chapter 9.

This chapter focuses on a class of topological maps called *roadmaps* [91, 262]. A roadmap is embedded in the free space and hence the nodes and edges of a roadmap also carry physical meaning. For example, a roadmap node corresponds to a specific location and an edge corresponds to a path between neighboring locations. So, in addition to being a graph, a roadmap is a collection of one-dimensional manifolds that captures the salient topology of the free space.

Robots use roadmaps in much the same way people use highway systems. Instead of planning every possible side-street path to a destination, people usually plan their path to a network of highways, then along the highway system, and finally from the highway to their destination. The bulk of the motion occurs on the highway system, which brings the motorist from near the start to near the goal (figure 5.2).

Likewise, using a roadmap, the planner can construct a path between any two points in a connected component of the robot's free space by first finding a collision-free path onto the roadmap, traversing the roadmap to the vicinity of the goal, and then constructing a collision-free path from a point on the roadmap to the goal. The bulk of the motion occurs on the roadmap and thus searching does not occur in a multidimensional space, whether it be the workspace or the configuration space. If the robot knows the roadmap, then it in essence knows the environment. So one way a robot can explore an unknown environment is by relying on sensor data to construct a roadmap and then using that roadmap to plan future excursions into the environment. We now formally define the roadmap.

Figure 5.2 Los Angeles freeway system: Planning a path from Pasadena to the Manhattan Beach requires finding a path onto the 110, then to the 105 and 405, and finally from the 405 to the beach. Courtesy of Mapquest.

DEFINITION 5.0.2 (Roadmap) *A union of one-dimensional curves is a* ***roadmap*** RM *if for all* q_{start} *and* q_{goal} *in* $\mathcal{Q}_{\text{free}}$ *that can be connected by a path, the following properties hold:*

1. ***Accessibility****: there exists a path from* $q_{\text{start}} \in \mathcal{Q}_{\text{free}}$ *to some* $q'_{\text{start}} \in RM$,
2. ***Departability****: there exists a path from some* $q'_{\text{goal}} \in RM$ *to* $q_{\text{goal}} \in \mathcal{Q}_{\text{free}}$, *and*
3. ***Connectivity****: there exists a path in* RM *between* q'_{start} *and* q'_{goal}.

In this chapter, we consider five types of roadmaps: *visibility maps, deformation retracts, retract-like structures, piecewise retracts* and *silhouettes*. All of these roadmaps have a corresponding graph representation. Visibility maps tend to apply to configuration spaces with polygonal obstacles. Nodes of the map are the vertices of the polygons and for visibility maps we can use the terms node and vertex interchangeably. Two nodes of a visibility map share an edge if their corresponding vertices are within line of sight of each other. Deformation retractions are analogous to melting ice or burning grassland. As an arbitrary shaped piece of ice melts, a resulting "stick

figure" forms. The ice represents the robot's free space and since the stick figure captures the macroscopic properties of the piece of ice, it can be used for path planning in the robot's free space The representation used for silhouette methods is constructed by repeatedly projecting a shadow of the robot's multidimensional free space onto lower-dimensional spaces until a one-dimensional network is formed.

5.1 Visibility Maps: The Visibility Graph

The defining characteristics of a visibility map are that its nodes share an edge if they are within line of sight of each other, and that all points in the robot's free space are within line of sight of at least one node on the visibility map. This second statement implies that visibility maps, by definition, possess the properties of accessibility and departability. Connectivity must then be explicitly proved for each map for the structure to be a roadmap. In this section, we consider the simplest visibility map, called the *visibility graph* [262, 298].

5.1.1 Visibility Graph Definition

The standard visibility graph is defined in a two-dimensional polygonal configuration space (figure 5.3). The nodes v_i of the visibility graph include the start location, the goal location, and all the vertices of the configuration space obstacles. The graph edges e_{ij} are straight-line segments that connect two line-of-sight nodes v_i and v_j, i.e.,

$$e_{ij} \neq \emptyset \iff sv_i + (1-s)v_j \in \mathrm{cl}(\mathcal{Q}_{\mathrm{free}}) \ \ \forall s \in [0,1].$$

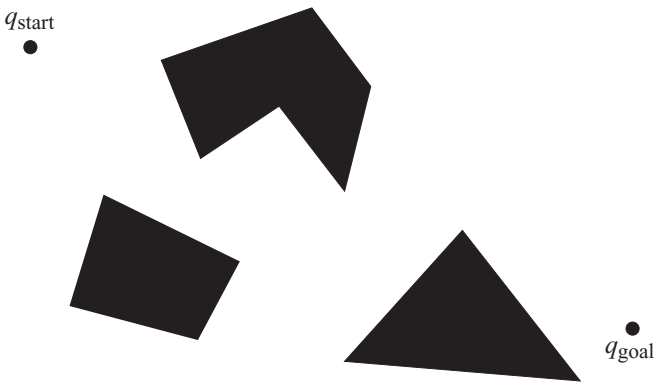

Figure 5.3 Polygonal configuration space with a start and goal.

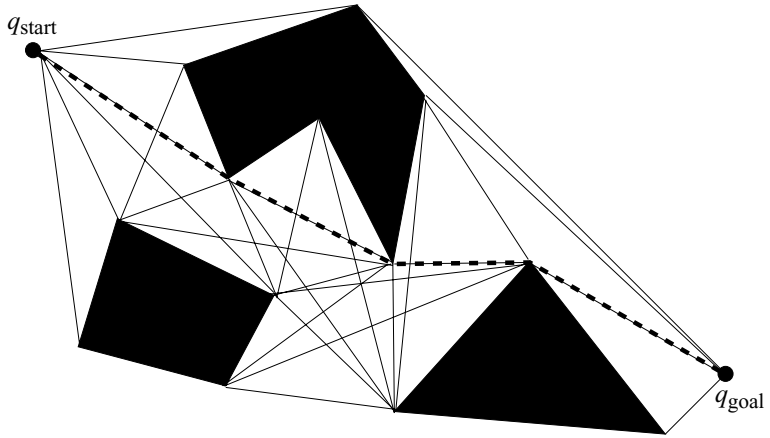

Figure 5.4 The thin solid lines delineate the edges of the visibility graph for the three obstacles represented as filled polygons. The thick dotted line represents the shortest path between the start and goal.

Note that we are embedding the nodes and edges in the free space and that edges of the polygonal obstacles also serve as edges in the visibility graph.

By definition, the visibility graph has the properties of accessibility and departability. We leave it to the reader as an exercise to prove the visibility graph is connected in a connected component of free space. Using the standard two-norm (Euclidean distance), the visibility graph can be searched for the shortest path (figure 5.4) [366]. The visibility graph can be defined for a three dimensional configuration space populated with polyhedral obstacles, but it does not necessarily contain the shortest paths in such a space.

Unfortunately, the visibility graph has many needless edges. The use of *supporting* and *separating* lines can reduce the number of edges. A supporting line is tangent to two obstacles such that both obstacles lie on the same side of the line. For nonsmooth obstacles, such as polygons, a supporting line l can be tangent at a vertex v_i if $\mathcal{B}_\epsilon(v_i) \bigcap l \bigcap \mathcal{QO}_i = v_i$. A separating line is tangent to two obstacles such that the obstacles lie on opposite sides of the line. See figure 5.5 for an example of supporting and separating lines.

The *reduced visibility graph* is soley constructed from supporting and separating lines. In other words, all edges of the original visibility graph that do not lie on a supporting or separating line are removed. Figure 5.6 contains the reduced visibility graph of the example in figure 5.4. The notion of separating and supporting lines can be used to generalize the visibility graph method for curved obstacles [294].

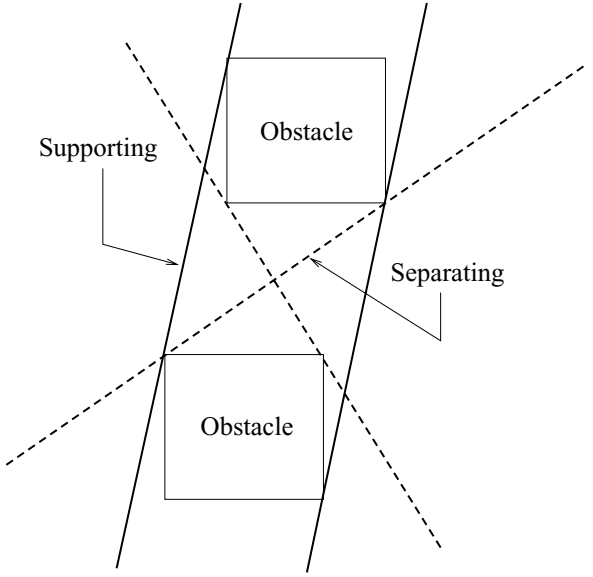

Figure 5.5 Supporting and separating line segments. Note that for polygonal obstacles, we use a nonsmooth notion of tangency.

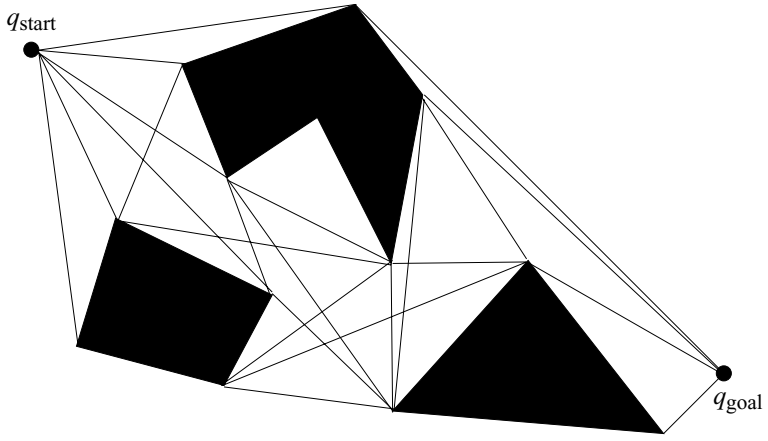

Figure 5.6 Reduced visibility graph.

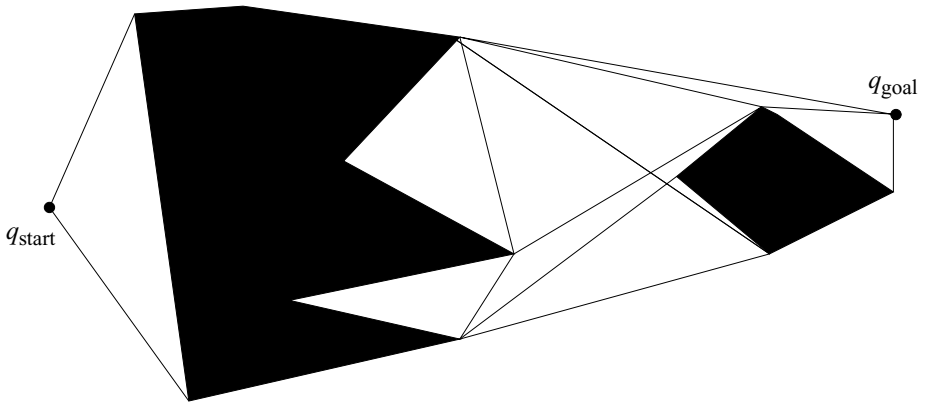

Figure 5.7 Reduced visibility graph with nonconvex obstacles.

At first, the definitions of the supporting and separating lines may seem to only apply to convex obstacles. However, this definition applies to nonconvex shapes as well. Here, we use the notion of local convexity. Recall that convex sets in the plane have the property that for all points on their boundary, there exists a line orthogonal to the surface normal that separates the convex set. This means that the set lies entirely on one side of the line. A set is *locally convex* at a point $\bar{c}$ if the hyperplane tangent to $\bar{c}$ separates the points in a neighborhood of $\bar{c}$ on the boundary of the convex set $\mathcal{QO}_i$. In other words, when N is the surface normal at $\bar{c}$, $\mathcal{QO}_i$ is locally convex at $\bar{c}$ if for all $c \in \left(\mathcal{QO}_i \bigcap \text{nbhd}\,(\bar{c})\right)$, $(c - \bar{c}) \cdot N \geq 0$ or $(c - \bar{c}) \cdot N \leq 0$. Convex obstacles are locally convex everywhere on the boundary of the set. Figure 5.7 contains a reduced visibility graph for a configuration space with nonconvex obstacles. The reduced visibility graph is beneficial because it has fewer edges making the search for the shortest path more efficient.

5.1.2 Visibility Graph Construction

Let $V = \{v_1, \ldots, v_n\}$ be the set of vertices of the polygons in the configuration space as well as the start and goal configurations. To construct the visibility graph, for each $v \in V$ we must determine which other vertices are visible to v. The most obvious way to make this determination is to test all line segments $\overline{vv_i}$, $v \neq v_i$ to see if they intersect an edge of any polygon. For a particular $\overline{vv_i}$, there are $O(n)$ intersections to check because there are $O(n)$ edges from the obstacles. Now, there are $O(n)$ potential segments emanating from v, so for a particular v, there are $O(n^2)$ tests to determine which vertices are indeed visible from v. This must be done for all $v \in V$ and thus the construction of the visibility graph would have complexity $O(n^3)$.

There is a more efficient way to compute the set of vertices that are visible from v. Imagine a rotating beam of light emanating from a lighthouse beacon. At any moment, the beam illuminates the object that is closest to the lighthouse. Furthermore, as the beam rotates, the obstacle that is illuminated changes only at a finite number of orientations of the beam. If the obstacles in the space are polygons, these orientations occur when the beam is incident on a vertex of some polygon. This insight motivates a class of algorithms known in the computational geometry literature as *plane sweep algorithms*.

A plane sweep algorithm solves a problem by sweeping a line, called the *sweep line,* across the plane, pausing at each of the vertices of the obstacles. At each vertex, the algorithm updates a partial solution to the problem. Plane sweep algorithms are used to efficiently compute the intersections of a set of line segments in the plane, to compute intersections of polygons, and to solve many other computational geometry problems.

For the problem of computing the set of vertices visible from v, we will let the sweep line, l, be a half-line emanating from v, and we will use a rotational sweep, rotating l from 0 to 2π. The key to this algorithm is to incrementally maintain the set of edges that intersect l, sorted in order of increasing distance from v. If a vertex v_i is visible to v, then it should be added to the visibility graph (algorithm 5). It is

Algorithm 5 Rotational Plane Sweep Algorithm

Input: A set of vertices $\{v_i\}$ (whose edges do not intersect) and a vertex v
Output: A subset of vertices from $\{v_i\}$ that are within line of sight of v

1: For each vertex v_i, calculate α_i, the angle from the horizontal axis to the line segment $\overline{vv_i}$.
2: Create the vertex list $\mathcal{E}$, containing the α_i's sorted in increasing order.
3: Create the active list $\mathcal{S}$, containing the sorted list of edges that intersect the horizontal half-line emanating from v.
4: **for all** α_i **do**
5: **if** v_i is visible to v **then**
6: Add the edge (v, v_i) to the visibility graph.
7: **end if**
8: **if** v_i is the beginning of an edge, E, not in $\mathcal{S}$ **then**
9: Insert the E into $\mathcal{S}$.
10: **end if**
11: **if** v_i is the end of an edge in $\mathcal{S}$ **then**
12: Delete the edge from $\mathcal{S}$.
13: **end if**
14: **end for**

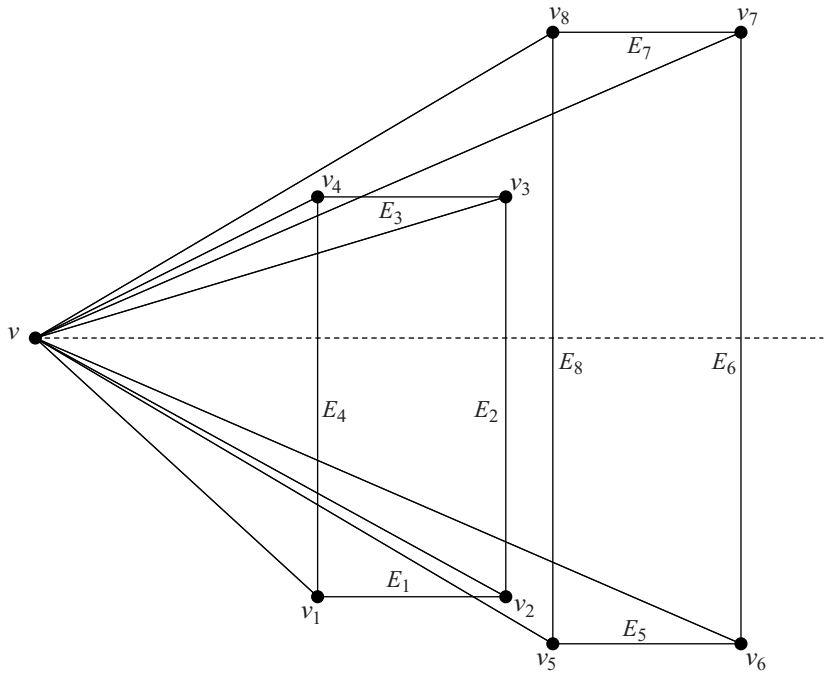

Figure 5.8 An example of the sweep line algorithm at work for an environment containing two rectangular obstacles.

straightforward to determine if v_i is visible to v. Let $\mathcal{S}$ be the sorted list of edges that intersects the half-line emanating from v; the set $\mathcal{S}$ is incrementally constructed as the algorithm runs. If the line segment $\overline{vv_i}$ does not intersect the closest edge in $\mathcal{S}$, and if l does not lie between the two edges incident on v (the sweep line does not intersect the interior of the obstacle at v), then v_i is visible from v.

Figure 5.8 shows an example configuration space containing two obstacles with vertices $v_1, \ldots, v_8$. Table 5.1 shows how the data structures are updated as the algorithm proceeds from initialization to termination. Step 1 of the algorithm determines the angles, α_i's, at which the line l will pause; such angles correspond to the vertices of the obstacles. In step 2 of the algorithm, these angles are used to construct the vertex list, $\mathcal{E}$, and in step 3 the active list $\mathcal{S}$ is initialized. After initialization, $\mathcal{E}$ and $\mathcal{S}$ are the sorted lists:

$\mathcal{E} = \{\alpha_3, \alpha_7, \alpha_4, \alpha_8, \alpha_1, \alpha_5, \alpha_2, \alpha_6, \}$,

$\mathcal{S} = \{E_4, E_2, E_8, E_6\}$.

Vertex	New $\mathcal{S}$	Actions
Initialization	$\{E_4, E_2, E_8, E_6\}$	Sort edges intersecting horizontal half-line
α_3	$\{E_4, E_3, E_8, E_6\}$	Delete E_2 from $\mathcal{S}$. Add E_3 to $\mathcal{S}$.
α_7	$\{E_4, E_3, E_8, E_7\}$	Delete E_6 from $\mathcal{S}$. Add E_7 to $\mathcal{S}$.
α_4	$\{E_8, E_7\}$	Delete E_3 from $\mathcal{S}$. Delete E_4 from $\mathcal{S}$. ADD (v, v_4) to visibility graph
α_8	$\{\}$	Delete E_7 from $\mathcal{S}$. Delete E_8 from $\mathcal{S}$. ADD (v, v_8) to visibility graph
α_1	$\{E_1, E_4\}$	Add E_4 to $\mathcal{S}$. Add E_1 to $\mathcal{S}$. ADD (v, v_1) to visibility graph
α_5	$\{E_4, E_1, E_8, E_5\}$	Add E_8 to $\mathcal{S}$. Add E_5 to $\mathcal{S}$.
α_2	$\{E_4, E_2, E_8, E_5\}$	Delete E_1 from $\mathcal{S}$. Add E_2 to $\mathcal{S}$.
α_6	$\{E_4, E_2, E_8, E_6\}$	Delete E_5 from $\mathcal{S}$. Add E_6 to $\mathcal{S}$.
Termination		

Table 5.1 Table showing the progress of the rotational plane sweep algorithm for the environment of figure 5.8.

At termination, the algorithm has added three new edges to the visibility graph: (v, v_4), (v, v_8), and (v, v_1).

The complexity of algorithm 5 is $O(n^2 \log n)$. The time required by step 1 is $O(n)$, since each vertex must be visited exactly once. For step 2, the required time is $O(n \log n)$, since this is the time required to sort a list of n elements. For step 3, the set of active edges can be computed in $O(n)$ time by merely testing each edge to see if it intersects the horizontal axis. In the worst case, if every edge were to intersect the horizontal axis, this set could be sorted in time $O(n \log n)$. The main loop of the program (step 4) iterates n times (once for each vertex). At each iteration, the algorithm must perform basic bookkeeping operations (insert or delete), but these can be done in time $O(\log n)$ if an appropriate data structure, such as a balanced tree, is used to maintain $\mathcal{S}$ Thus, the time required by step 4 is $O(n \log n)$, and therefore the total time complexity of the algorithm is $O(n^2 \log n)$.

Finally, we have not considered here the case when l may simultaneously intersect multiple vertices. In order for this to occur, three vertices must be collinear. When this does occur, the problem can be resolved by slightly perturbing the position of one of

the three vertices. When no three vertices are collinear, we say that the polygons are in *general position,* and the general position assumption is common for computational geometry algorithms. It is also possible to modify the visibility test to account for nongeneral configurations, and this is addressed in [124].

5.2 Deformation Retracts: Generalized Voronoi Diagram

The generalized Voronoi diagram (GVD) is the set of points where the distance to the two closest obstacles is the same. Figure 5.9(d) displays an example of the GVD. Path planning is achieved by moving away from the closest point until reaching the GVD, then along the double equidistant GVD to the vicinity of the goal, and then from the GVD to the goal. Since the GVD is defined in terms of distance, one can expect

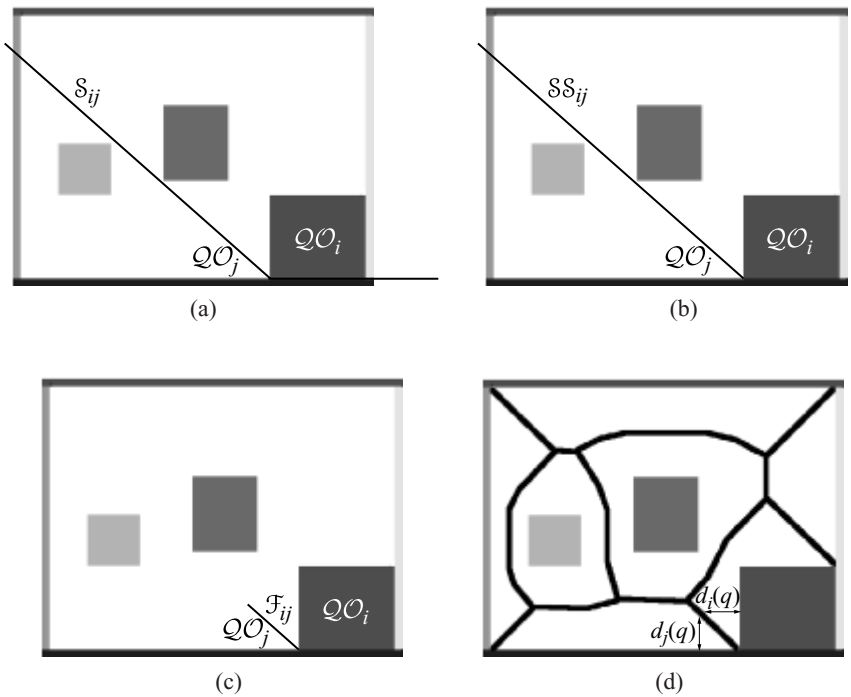

Figure 5.9 (a) The set $\mathcal{S}_{ij}$ contains points equidistant to two obstacles $\mathcal{QO}_i$ and $\mathcal{QO}_j$. (b) The set $\mathcal{SS}_{ij}$ contains equidistant points with distinct gradients; note that there is no $\mathcal{SS}_{ij}$ structure to the right of the obstacles. We delay discussion of this structure for a moment. (c) The set $\mathcal{F}_{ij}$ has the closest pair of obstacles. (d) The GVD is the union of all such sets.

that a robot equipped with range sensors can incrementally construct the GVD in an unknown space. Once the GVD is constructed, the robot has essentially explored the space because the robot can use the GVD to plan paths in the free space with the GVD.

We show that the GVD is a roadmap because the GVD is a type of *deformation retract*. Deformation retracts are best described by an analogy. Imagine a doughnut-shaped candy: a candy with a hole in the middle of it. As the candy dissolves, eventually a ring remains. This ring captures the topological structure of the candy even though it is significantly smaller than the original. Every point on the ring serves as the center of a corresponding planar disk orthogonal to the ring; each disk is shrunk to a point. In this analogy, the original candy represents the robot's free space and the resulting ring corresponds to a geometric structure called a deformation retract. The function that represents this shrinking process, i.e., the function that maps the filled torus[1] onto a ring, is called a *deformation retraction.*

First in section 5.2.1, we define the GVD and then in section 5.2.2, we show it has the properties of accessibility, connectivity, and departability. In section 5.2.2, we rely on the fact that the GVD is indeed a deformation retract to assure it has the roadmap properties and in section 5.2.3 we describe in more detail as to how the GVD is a deformation retract. Next, in section 5.2.4, we prove that the GVD is indeed one-dimensional. Here, we review the preimage theorem to assert the dimensionality property of the GVD. Finally, in section 5.2.5, we describe three methods to construct the GVD.

5.2.1 GVD Definition

The *Voronoi diagram* is defined for a set of points called *sites* [31]. A *Voronoi region* is the set of points closest to a particular site [31]. The Voronoi diagram is then the set of points equidistant to two sites; it sections off the free space into regions that are closest to a particular site. Points on the Voronoi diagram have two closest sites. In the planar case, the Voronoi diagram is a collection of line segments.

For the purposes of path planning, we can think of the point sites as obstacles, but obstacles are not simple points. Therefore, the definition of a Voronoi region is extended to the *generalized Voronoi region,* $\mathcal{F}_i$, which is the closure of the set of points closest to $\mathcal{QO}_i$. In other words,

$$\mathcal{F}_i = \{q \in \mathcal{Q}_{\text{free}} \mid d_i(q) \leq d_h(q) \quad \forall h \neq i\}, \tag{5.1}$$

1. A torus is two-dimensional structure, and the filled torus is a three-dimensional version, i.e., the convex hull of a torus embedded in $\mathbb{R}^3$.

where $d_i(q)$ is the distance to an obstacle $\mathcal{QO}_i$ from q, i.e., $d_i(q) = \min_{c \in \mathcal{QO}_i} d(q, c)$ (chapter 4, equation (4.6)).

The basic building block of the GVD is the set of points equidistant to two sets $\mathcal{QO}_i$ and $\mathcal{QO}_j$, which we term a *two-equidistant surface* denoted by $\mathcal{S}_{ij} = \{x \in \mathcal{Q} \mid (d_i(q) - d_j(q)) = 0\}$. Note that $d_i(q) - d_j(q) = 0$ is an equivalent way to state $d_i(q) = d_j(q)$ (figure 5.9). A two-equidistant surface pierces obstacles, so we restrict it to the set of points that are both equidistant to $\mathcal{QO}_i$ and $\mathcal{QO}_j$ and have $\mathcal{QO}_i$ and $\mathcal{QO}_j$ as their closest obstacles. This restricted structure is the *two-equidistant face*, which could be denoted by $\mathcal{F}_{ij} = \{q \in \mathcal{S}_{ij} \mid d_i(q) \leq d_h(q) \ \ \forall h\}$[2]. We refine this definition shortly. The union of the two-equidistant faces forms the GVD, i.e.,

$$\text{GVD} = \bigcup_i \bigcup_j \mathcal{F}_{ij}.$$

This definition of the GVD applies to any dimensional spaces. One can see that the GVD partitions the free space into regions $\mathcal{F}_i$ such that points in the interior of one $\mathcal{F}_i$ are closer to $\mathcal{QO}_i$ than to any other obstacle. Points on the GVD have two or more closest obstacles. In the planar case, we term the $\mathcal{F}_{ij}$ as *GVD edges* and they terminate at either *meet points,* the set of points equidistant to three or more obstacles ($\mathcal{F}_{ijk}$), or *boundary points,* the set of points whose distance to the closest obstacle is zero. Boundary points are the endpoints of "spokes" of the GVD.

5.2.2 GVD Roadmap Properties

In $\mathbb{R}^m$, the GVD has the properties of accessibility, connectivity, and departability. In the plane, the GVD is a roadmap because it has these properties and is one-dimensional. We show that the planar GVD is one-dimensional in the next subsection and the properties of accessibility, connectivity, and departability here. The robot achieves accessibility by moving away from the closest obstacle; it performs gradient ascent of distance D to the closest obstacle, i.e.,

$$\frac{dc(t)}{dt} = \nabla D(c(t)) \quad \text{where} \quad c(0) = q_{\text{start}}, \tag{5.2}$$

until it reaches a point on the GVD.

Equation (5.2) is a first order differential equation implicitly defining the the path $c : [0, 1] \rightarrow \mathcal{Q}_{\text{free}}$. At any point $c(t) \in \mathcal{Q}_{\text{free}}$, the tangent to the path is defined by the gradient of distance to the closest obstacle. The gradient $\nabla D(q)$ points in the direction that maximally increases distance. The tangent of the curve $\frac{dc(t)}{dt}$ is "set"

2. Note that we could have written $d_i(q) = d_j(q) \leq d_h(q)$, but the "$= d_j(q)$" is already implied by the $q \in \mathcal{S}_{ij}$.

to the gradient of distance. By constantly following the distance gradient, a path is traced that maximally increases the distance.

LEMMA 5.2.1 (Accessibility of the GVD) *In an obstacle-bounded environment, gradient ascent of D traces a path from any point in the free space to the GVD.*

Proof Assume the robot starts at a point q that is not on the GVD. Let $\mathcal{QO}_i$ be the closest obstacle to q. Hence $d_i(q) = D(q)$ and $d_h(q) > d_i(q)$ for all h. The robot traces a path $\frac{dc(t)}{dt} = \nabla d_i(c(t))$ where $c(0) = q$. Since the environment is bounded, continuity of the distance function guarantees that there exists a $\bar{t} \in \mathbb{R}$ and a $\mathcal{QO}_j$, such that $d_i(c(\bar{t})) = d_j(c(\bar{t}))$. ■

We use the fact that the GVD is a *deformation retract* to ensure connectivity of the GVD. A deformation retract is the image of a continuous function called a *deformation retraction* RM such that

$$RM(q) = q, \quad \text{for all } q \text{ in the GVD,}$$
$$RM(q) = q', \quad \text{for any } q \in \mathcal{Q}_{\text{free}} \text{ and } q' \in \text{GVD.}$$

We more formally define the deformation retraction in the next section.

For the GVD, the gradient ascent accessibility procedure *implicitly* defines the deformation retraction without explicitly doing so [340]. In other words, if q is on the GVD, then the image of q is q, i.e., $RM(q) = q$. If q is in the free space but not in the GVD, then the image q is the q' in the GVD that is obtained by moving away from the closest point on the closest obstacle until encountering the GVD, i.e., $RM(q) = q'$.

Connectivity of the GVD is then a consequence of continuity of the RM function. In other words, since RM is continuous, for each connected component of the free space there is a connected component of the GVD. Therefore, there exists a path that connects q_{start} and q_{goal} if and only if there exists a path in the GVD that connects q'_{start} and q'_{goal} where $q'_{\text{start}} = RM(q_{\text{start}})$ and $q'_{\text{goal}} = RM(q_{\text{goal}})$.

Departability is simply accessibility in reverse. However, there are other ways to achieve departability. It can be shown that all points in free space have at least one point on the GVD within line of sight, i.e.,

$$\forall\ q \in \mathcal{Q}_{\text{free}},\ \ \exists q' \in \text{ GVD such that } sq + (1-s)q' \in \mathcal{Q}_{\text{free}}\ \ \forall s \in [0,1].$$

This means that if the robot comes within line of sight of the goal, the robot can drive straight toward it. This approach to departability only makes sense if the robot can detect the goal using its on-board sensors.

5.2.3 Deformation Retract Definition

Before defining the deformation retract, we define a weaker structure called a *retract*. For a manifold X, a *retraction* is a continuous function $f : X \to A$ such that $A \subset X$, and $f(a) = a$ for all $a \in A$ [410]. The subset A is the retract. Typically, the dimension of A is less than the dimension of X.

The set of deformation retracts is a subset of the set of retracts and hence the GVD is a retract also. However, the properties of a retract are not sufficient to guarantee that the GVD is a roadmap. It is the fact that that GVD is indeed a deformation retract that makes it a roadmap. Essentially, a deformation retract inherits many topological properties from its ambient space, whereas a retract may not. One important property is that the number of "types" of closed paths in the free space is equal to the number of "types" of closed paths in the deformation retract of the free space.

Let's return to the candy example from the beginning of this section. Although a retract can be a ring, it could also be a single point, a two-dimensional disk orthogonal to the ring, etc. We need to enforce additional properties on the retract so as to guarantee that it captures the topology of its free space and is still one-dimensional. Recall from chapter 3, section 3.4.1 that global diffeomorphisms are mappings that relate spaces that are "topologically similar." Diffeomorphic spaces must have same dimension. Now, we consider spaces that are similar, but of different dimensions.

Let $f : U \to V$ and $g : U \to V$ where U and V are manifolds. A *homotopy* is a continuous function $H : U \times [0, 1] \to V$ such that $H(x, 0) = f(x)$ and $H(x, 1) = g(x)$. An example of H is $H(x, t) = (1 - t)f(x) + tg(x)$. If there exists such a continuous mapping that "deforms" f to g, then f and g are *homotopic,* and the resulting equivalence relation is denoted $f \sim g$. We can also say that two paths f and g are path-homotopic, i.e., $f \sim g$, if they can be continuously deformed into one another. This relation allows for the classification of functions into equivalence classes termed *path-homotopy classes* and are denoted as

$$[c] = \{\bar{c} \in C^0 \mid \bar{c} \sim c\}.$$

where c is a representative element of the class.

Let $A \subset X$ and let $f : X \to A$ be a retraction. A *deformation retraction* is a homotopy $H : X \times [0, 1] \to X$ such that

- $H(x, 0) = x$
- $H(x, 1) \in A$
- $H(a, t) = a$ for $a \in A$ and $t \in [0, 1]$

In other words, H is a homotopy between a retraction and the identity map[3]. Note that all retractions are not necessarily homotopic to the identity map. The retract is now called a *deformation retract.*

We use deformation retractions to smoothly deform, without tearing or pasting X onto a lower, preferably one-dimensional subset A of X. So, as t varies from 0 to 1, a point in X continuously moves through X to a point in A. Moreover, a point y in a neighborhood of x also continuously moves through X to a point in A such that $H(x, t)$ and $H(y, t)$ are close to each other as t varies from 0 to 1. Thefore, the deformation retraction preserves many topological properties of the free space. Thus, while a diffeomorphism preserves the structure of two spaces of the same dimension, a deformation retraction preserves the structure of two spaces of different dimension.

One of the key topological properties of deformation retracts is that they preserve the number of homotopically equivalent closed loops from the ambient space. The number of homotopy equivalence classes of closed loops is called the *first fundamental group,* and is denoted as $\pi_1(X, x_0)$ for loops in X passing through x_0. Since this is a group, it has a group operator ($\star$) that simply concatenates paths. A set X is simply connected if the fundamental group associated with the set, $\pi_1(X, x_0)$, contains only the identity element (e.g., the group only contains one element). If f is a deformation retraction with A as its deformation retract of X, then $\pi_1(X, x_0) = \pi_1(A, f(x_0))$. In other words, the ambient space X and the deformation retract A have the same number of homotopically equivalent closed loops.

Deformation retracts have the properties of connectivity, accessibility, and departability. For each connected component of X, A is a connected set because the image of a connected set under a continuous mapping is a connected set [9]. The deformation retraction determines a path from the start to the deformation retract, as well as a path from the goal to the retract. Let H be the deformation retraction and $H(x, 0) = q_{\text{start}}$. The path to the deformation retract is then defined by $H(x, \cdot) : [0, 1] \rightarrow \mathcal{Q}_{\text{free}}$ where $H(x, 1)$ is an element of the deformation retract. Departability is shown in the same manner. Since the deformation retract is connected, there is a path between the retracted start and retracted goal configurations along the deformation retract. Hence, one-dimensional deformation retracts are roadmaps.

The GVD is a retract because the RM (equation (5.2.2)) has been shown to be continuous and maps all points on the GVD to the GVD. Since RM is continuous, the GVD is connected in a connected component of the free space because the image of connected set under a continuous function is a connected set. The GVD is a deformation retract because RM has been shown to be homotopic to the indentity

3. Sometimes, a deformation retraction is defined as a retraction that is homotopic to the identity map [207] as opposed to the homotopy.

map. Therefore, RM smoothly deforms the free space onto a one-dimensional subset and defines the accessibility and departability criteria. Finally, since the GVD is a deformation retract, the number of closed-loop path equivalent classes in the GVD equals the number of closed-loop path equivalent classes in the free space because RM preserves the cardinality of the first fundamental group. This makes the GVD a concise representation of the free space.

5.2.4 GVD Dimension: The Preimage Theorem and Critical Points

A key property of a roadmap is that it is one-dimensional. Actually, we show that in the plane, the GVD consists of one-dimensional manifolds. Before we can demonstrate this, we have to take a more careful look at the definition of the GVD. Recall that we are using the distance function d_i to define the GVD, but this function assumes that the obstacles are convex, which is unrealistic in most situations.

At first, it seems to make sense to decompose nonconvex obstacles into convex pieces. This causes problems because there are many ways to construct such a decomposition, thereby resulting in different representations of the free space. Consider the obstacle in figure 5.10. Both decompositions are valid, but unfortunately they give rise to two different definitions of $\mathcal{S}_{ij}$, the set of points equidistant to two obstacles $\mathcal{QO}_i$ and $\mathcal{QO}_j$. There are infinitely many ways to decompose a nonconvex obstacle and hence the possibility for infinitely many representations.

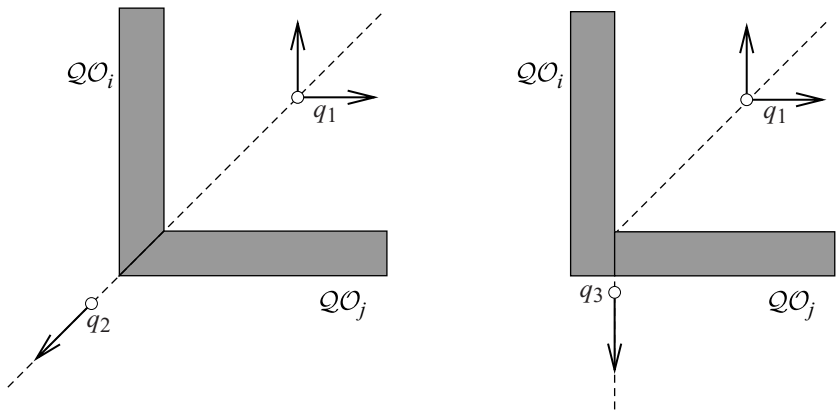

Figure 5.10 A nonconvex obstacle is divided into two pieces, $\mathcal{QO}_i$ and $\mathcal{QO}_j$, but in two different ways. On the left, a diagonal forms the convex obstacles and on the right a vertical cut forms them. Note that in both left and right, the gradient vectors pointing away from the two closest obstacles are distinct at q_1 but they are the same at q_2 and q_3.

It would be nice to have a unique representation of the roadmap, so we refine our definition of the GVD. In figure 5.10, note that there are two portions of $\mathcal{S}_{ij}$: the upper-right portion, which is "between" the two arms of the obstacle and the lower-left portion, which is on the other side of the obstacle. Note that for the portions between the two arms, the gradients to the two closest obstacles are distinct, e.g., $\nabla d_i(q_1) \neq \nabla d_j(q_1)$. However, for the other portions, the gradients line up, e.g., $\nabla d_i(q_2) = \nabla d_j(q_2)$ and $\nabla d_i(q_3) = \nabla d_j(q_3)$. Eliminating the portion of the two-equidistant surface with nondistinct gradient vectors yields a set termed the *two-equidistant surjective surface* denoted as

$$\mathcal{SS}_{ij} = \{q \in \mathcal{S}_{ij} \mid \nabla d_i(q) \neq \nabla d_j(q)\}.$$

See figure 5.9(b) for an example of a two-equidistant surjective surface defined by a nonconvex obstacle that has been divided into two convex pieces.

This definition of a two-equidistant surjective surface should be salient from a sensor-based perspective. Consider the planar case where distance and gradient vectors can be derived from a laser ranger or a sonar ring which approximates the saturated raw distance function. Recall that the saturated raw distance function corresponds to all of the rays emanating from a single point intersecting as can be seen in figure 5.11.

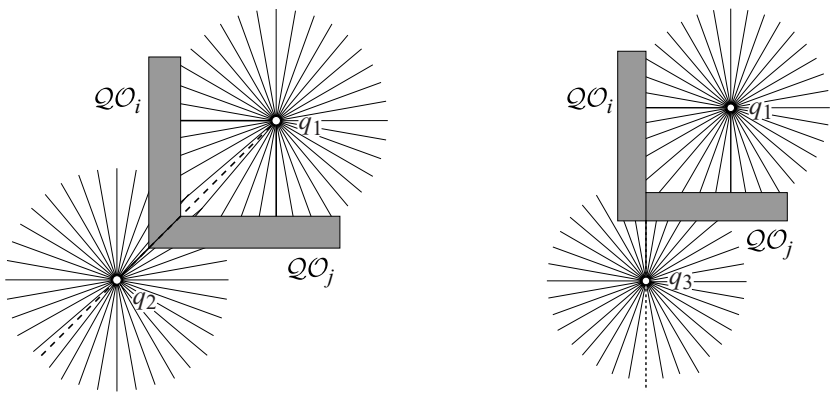

Figure 5.11 A robot is placed at different configurations q_1, q_2 and q_3. It has range sensors radially distributed pointing in a full 360 degrees. The rays emanating from the points correspond to the range readings. Local minima correspond to distance to nearby obstacles. (Left) Use a diagonal cut to break the nonconvex obstacles into convex ones, but a robot cannot see the diagonal cut from q_2 because there is no local minimum. (Right) Use a vertical cut to break the nonconvex obstacles into convex ones, but a robot cannot see the vertical cut from q_3 because there is no local minimum.

On the "inside" of the concavity (at q_1 in figure 5.11), there are two local minima in the raw distance function, whereas on the outside there is one at q_2 and one at q_3. In other words, a robot situated on the "outside" of the obstacle cannot determine from its sensor readings how the obstacle was cut. Another perspective is that on the "inside" of the nonconvex obstacle, the robot "sees" two obstacles and on the "outside," it only "sees" one.

From here, the definition of the two-equidistant face $\mathcal{F}_{ij}$ is modified to be $\mathcal{F}_{ij} = \{q \in \mathcal{SS}_{ij} \mid d_i(q) \leq d_h(q) \ \ \forall h\}$. So the GVD is the set of points equidistant to two obstacles such that the two obstacles are closest and have unique closest points on them.

We are now ready to show that the GVD is indeed one-dimensional. We do this by first rewriting the equidistant relationship $d_i(q) = d_j(q)$ as $d_i(q) - d_j(q) = 0$, which in turn can be written as $(d_i - d_j)(q) = 0$. Intuitively, this one constraint in a two-dimensional space defines a one-dimensional subspace. In other words, equidistance is the preimage of zero under the map $(d_i - d_j) : \mathcal{Q} \to \mathbb{R}$. We use this reformulation to demonstrate that in the plane the GVD comprises one-dimensional manifolds by taking recourse to the preimage theorem [173].

THEOREM 5.2.2 (Preimage Theorem) *Let M and N be manifolds. Let $G : M \to N \in C^\infty$ and $n \in N$ be a regular value of G. The set $G^{-1}(n) = \{m \in M \mid G(m) = n\}$ is a closed submanifold of M with tangent space given by $T_m G^{-1}(n) = \ker DG(m)$. If N is finitely dimensional, then* $\dim(G^{-1}(n)) = \dim(M) - \dim(N)$, *i.e.,* $\dim(G^{-1}(n)) = \dim(M) - \dim(N)$.

The preimage theorem contains a lot of terminology and notation. A *regular value* is an n where for all $m \in G^{-1}(n)$, the differential $DG(m)$ is surjective (e.g., has full rank). See section C.5.5 for a description of the differential. Next, T_m denotes the *tangent space* at m . So, $T_m M$ is the tangent space at m on the manifold M and $T_p G^{-1}(n)$ is the tangent space at p on the manifold $G^{-1}(n)$, which is a submanifold of M.

A *critical point* is a point where the differential is not surjective and hence loses rank. (For real-valued functions, it is a point where the first derivative vanishes.) Let $\Sigma(G)$ be the set of all critical points of G. For all $q^* \in \Sigma(G)$, $G(q^*)$ are *critical values*. Finally, all points $q \notin \Sigma(G)$ where $DG(q)$ is surjective are termed *regular points* with $G(q)$ as their corresponding *regular values.*

To show that the GVD edges are indeed one-dimensional, we use the preimage theorem to show that they are one-dimensional manifolds. First let's see how the preimage theorem is used to create manifolds. Consider the function $f(x, y) = x^2 + y^2$. The differential $Df(x, y) = [2x, 2y]$. For all $f(x, y) = 91,538$, $Df(x, y) \neq 0$ and thus the preimage of 91,538 under f forms a one-dimensional manifold.

With the GVD, $G = (d_i - d_j)$, and the set of points equidistant to two obstacles is $(d_i - d_j)^{-1}(0)$. However, for all points in the preimage to be regular, DG must be surjective. In other words, $D(d_i - d_j)$ must not be equal to zero. Since in a Euclidean space, $Dd_i(q) = (\nabla d_i(q))^T$, this means $\nabla d_i(q)$ cannot be equal to $\nabla d_j(q)$. However, we are fortunate to have the $\nabla d_i(q) \neq \nabla d_j(q)$ condition in the definition of $\mathcal{SS}_{ij}$. So, in actuality, the $\nabla d_i(q) \neq \nabla d_j(q)$ enforces the surjective condition for the preimage theorem, hence the term *surjective* in the two-equidistant surjective surface. So, by the preimage theorem, $\mathcal{SS}_{ij}$ is one-dimensional in the plane. The set $\mathcal{F}_{ij}$ is a submanifold of $\mathcal{SS}_{ij}$. Therefore, the GVD comprises a set of one-dimensional manifolds (figure 5.9).

5.2.5 Construction of the GVD

We discuss three methods for constructing the planar GVD: the first uses sensor information allowing the robot to construct the GVD in an unknown space; the second assumes the world has polygonal obstacles in which case we can compute complexity information about the GVD; and the final method assumes that the world is a grid allowing for efficient computation.

Sensor-Based Construction of the GVD

Exploring with the GVD is akin to simultaneously generating and exploring a graph that is embedded in the free space. The GVD can be incrementally constructed because it is defined in terms of distance information which is readily provided by range sensors onboard mobile robots. Using such line-of-sight data, the robot initially accesses the GVD and then begins tracing an edge until it encounters a meet point or a boundary point. When the robot encounters a *new* meet point, it marks off the direction from which it came as explored, and then identifies all new GVD edges that emanate from it. From the meet point, the robot explores a new GVD edge until it detects either another meet point or a boundary point. In the case that it detects another *new* meet point, the above branching process recursively repeats. If the robot reaches an *old* meet point, the robot has completed a cycle in the GVD graph and then travels to a meet point with an unexplored edge associated with it. When the robot reaches a boundary node, it simply turns around and returns to a meet point with unexplored GVD edges. When all meet points have no unexplored edges associated with them, exploration is complete.

The robot accesses the GVD by simply moving away from the nearest obstacle until it is equidistant to two obstacles (figure 5.12). Once the robot accesses the GVD, it must incrementally trace the GVD using the same curve tracing technique from

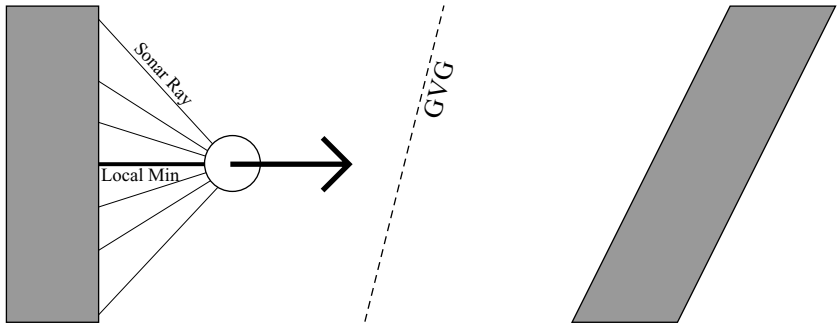

Figure 5.12 The circular disk represents a mobile robot with some of its range sensor readings. The thick ray, labeled local min, corresponds to the smallest sensor reading, and hence the robot will move in the direction indicated by the black arrow to access the GVD, denoted by a dashed line between two nonparallel walls.

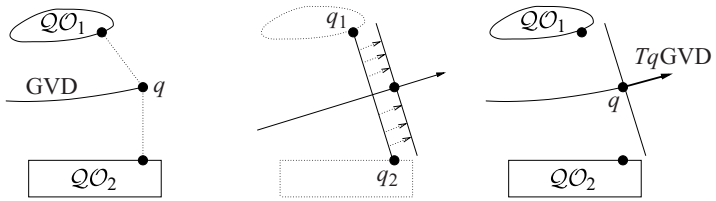

Figure 5.13 The tangent space to the GVD is orthogonal to the line that connects the two closest points on the two closest obstacles.

chapter 2, section 2.3.3, except $G(q) = d_i(q) - d_j(q)$ whose roots are the set of points where $d_i(q) = d_j(q)$. The tangent is the null space of $\nabla G(q)$, which corresponds to a line orthogonal to $\nabla d_i(q) - \nabla d_j(q)$. This is identical to passing a line through the two closest points and taking the vector perpendicular to the line to be the tangent (figure 5.13). A meet point is detected by looking for a sudden change in one of the two closest obstacles.

Polygonal Spaces

In a polygonal environment, obstacles have two features, vertices and edges, thereby making equidistance relationships easy to define. The set of points equidistant to two vertices is a line; the set of points equidistant to two edges is a line; and the set of points equidistant to a vertex and an edge is a parabola. Therefore, by breaking down

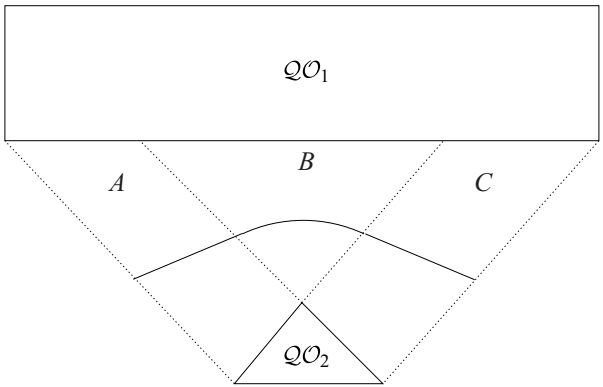

Figure 5.14 GVD edge fragment for two polygonal obstacles.

the free space into regions with the appropriate pair of closest features, one can easily build the GVD. In figure 5.14, regions A and C have a pair of edges as their respective closest features, whereas region B has an edge and vertex as its closest obstacle.

In a polygonal environment with n obstacles and N obstacle vertices, the number of GVD edges falls between $\frac{3(n+1)}{2}$ and $6N + 3n - 3$. The number of nodes on the GVD falls between $\frac{n+5}{2}$ and $4N - n - 2$. See [359] for details.

Grid Configuration Spaces: The Brushfire Method

The method presented in chapter 4, section 4.3.2 can be readily adapted to construct the GVD in a discrete grid. Originally, the input for the brushfire method is a grid of zeros corresponding to free space and ones corresponding to an obstacle. The output of the brushfire method is a discrete map where each pixel in the grid has a value equal to the distance to the closest point on the closest obstacle (the closest pixel with a value of one).

We can view the brushfire method as a wave initially starting at the obstacles and propagating through the free space. As the wave front passes over a pixel, the method assigns a value to the pixel corresponding to how far the wave has traveled. The wave fronts collide at points where the distance to two different obstacles is the same. These are points on the GVD.

The brushfire algorithm can be readily updated to identify the pixels where these collisions occur. Essentially, as the wave propagates, each pixel in the wave front maintains a back pointer to the obstacle pixel from which the wave originated. When the updated brushfire algorithm attempts to assign a "free pixel" with two

different back pointers, two wave fronts have collided and the current pixel belongs to the GVD.

5.3 Retract-like Structures: The Generalized Voronoi Graph

Now, we consider the case when $\mathcal{Q} = \mathbb{R}^3$. In $\mathbb{R}^3$, the GVD is two-dimensional and therefore reduces the motion planning problem by a single dimension. We use figure 5.9(d) to show this. Imagine extruding the one-dimensional curves in figure 5.9(d) into two-dimensional surfaces in three dimensions; so the one-dimensional curves in figure 5.9(d) become cross sections of two-dimensional sheets. This makes sense because we have a three-dimensional space with one constraint resulting in a two-dimensional subspace. The preimage theorem confirms that the GVD actually comprises two-dimensional manifolds; the dimension of $(d_i - d_j)^{-1}(0)$ is two because $3 - 1 = 2$.

Just as two planes in $\mathbb{R}^3$ generically intersect on a line, two two-equidistant faces intersect and form a one-dimensional manifold. The union of these one-dimensional structures is termed the *generalized Voronoi graph* (GVG) [105,106]. See figures 5.15 and 5.16 for examples of the GVG in three dimensions.

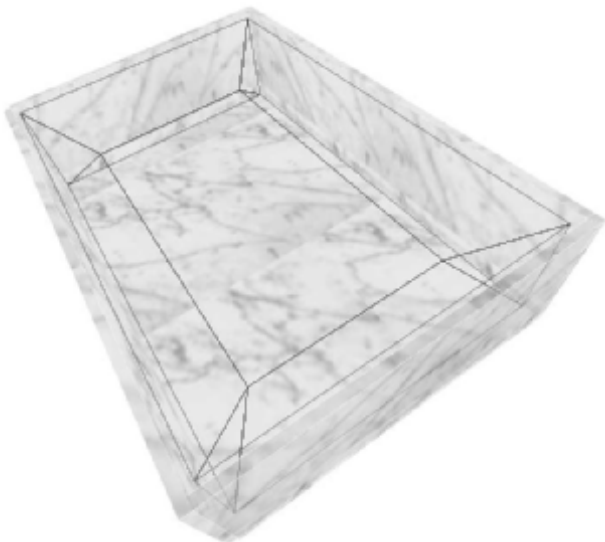

Figure 5.15 The solid lines represent the GVG for a rectangular enclosure whose ceiling has been removed to expose the interior. Imagine a sphere rolling around touching the removed ceiling, floor, and side wall; the center of this sphere traces a GVG edge.

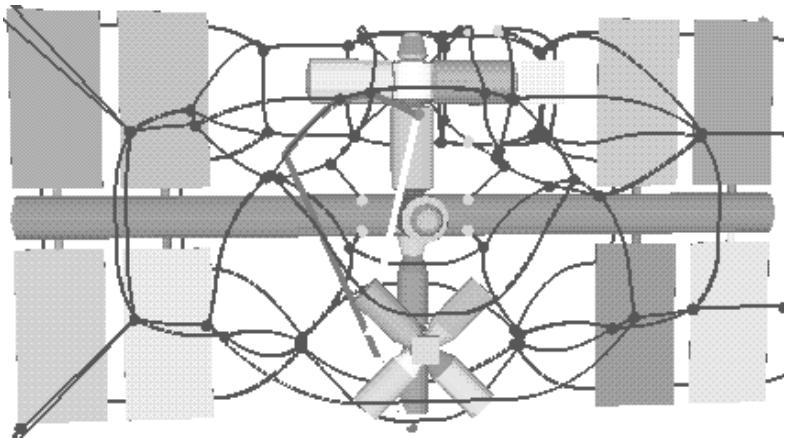

Figure 5.16 The GVG for the International Space Station (in a bounding box). Note that a bounding box was used but is not displayed. Also note how complicated the GVG becomes.

5.3.1 GVG Dimension: Transversality

The GVG edges in $\mathbb{R}^3$ are the set of points equidistant to three obstacles such that the three obstacles are closest and have distinct gradients. Starting with triple equidistance, we define $\mathcal{S}_{ijk} = \{q \mid (d_i - d_j)(q) = 0 \text{ and } (d_i - d_k)(q) = 0\}$. We do not need the additional $(d_j - d_k)(q) = 0$ constraint because $d_i(q) = d_j(q)$ and $d_i(q) = d_k(q)$ imply that $d_j(q) = d_k(q)$. Just like before, we are interested in a subset of $\mathcal{S}_{ijk}$ where the gradients are distinct, and thus

$$\begin{aligned}\mathcal{SS}_{ijk} &= \{q \in \mathcal{S}_{ijk} \mid \nabla d_i(q) \neq \nabla d_j(q), \nabla d_i(q) \neq \nabla d_k(q), \nabla d_j(q) \neq \nabla d_k(q)\} \\ &= \mathcal{SS}_{ij} \bigcap \mathcal{SS}_{ik} \bigcap \mathcal{SS}_{jk}\end{aligned}$$

Note that the transitivity of $d_i(q) = d_j(q)$ and $d_j(q) = d_k(q)$ implies that $d_i(q) = d_k(q)$, but it does not ensure that $\mathcal{SS}_{ijk} = \mathcal{SS}_{ij} \bigcap \mathcal{SS}_{ik}$ because we require *all three gradients* to be distinct. To determine the dimension of the GVG edge, we look at $G : \mathbb{R}^3 \to \mathbb{R}^2$ where

$$G(q) = \begin{bmatrix}(d_i - d_j) \\ (d_i - d_k)\end{bmatrix}(q)$$

whose preimage $G^{-1}(0)$ is the set of points equidistant to three obstacles $\mathcal{QO}_i$, $\mathcal{QO}_j$, and $\mathcal{QO}_k$ when the differential $DG(q)$ is surjective, i.e., does not lose rank. The differential $DG(q)$ can lose rank when either row of $DG(q)$ is zero or the first row is a scalar multiple of the second row in $DG(q)$. We already know by definition that $\nabla d_i(q) \neq$

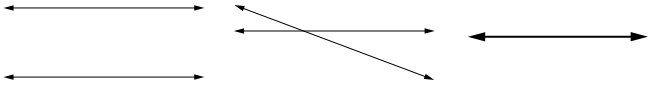

Figure 5.17 Three ways two lines in the plane can intersect, but only a point-intersection is transversal.

$\nabla d_j(q)$ and $\nabla d_i(q) \neq \nabla d_k(q)$, so all we need to show is that $\nabla(d_i - d_j)(q) \neq \alpha\nabla(d_i - d_k)$ for all $\alpha \in \mathbb{R}$. In other words, we must show that the two rows of $DG(q)$ do not depend upon each other.

We demonstrate this by making a "reasonable" assumption based on *transversality,* a property of how sets intersect. Let's start with a simple example of two intersecting lines in the plane. These lines may intersect in one of three ways: not at all (parallel), at a point (generic), and on a line (overlap) (figure 5.17). The parallel and overlap cases can be viewed as "unstable" because if either line were perturbed a little bit, the intersection would change dimension. The point intersection can be viewed as stable in that if either of the lines were perturbed, a point-type intersection is preserved. We call stable intersections *transversal* and nonstable intersections *nontransversal.* Two lines in three dimensions can never intersect transversally because a generic perturbation can break the intersection to no intersection. In three dimensions, two planes transversally intersect on a line and a plane and a line transversally intersect at a point.

In actuality, transversality is a local property of manifolds. For example, we say that two manifolds may intersect transversally at a point. Since transversality is a local property, we look at the intersection of the tangent spaces, not of the manifolds themselves. If intersection of the tangent spaces is transversal at a point, then the manifolds intersect transversally at that point (figure 5.18). We know from the preimage theorem that the tangent space $T_qG^{-1}(0)$ is given by the set of vectors $\{v \in T_q\mathcal{Q} \mid DG(q)v = 0\}$. We assume that surjective equidistant sheets intersect transversally at all points, i.e., $T_q(d_i - d_j)^{-1}(0)$ and $T_q(d_i - d_k)^{-1}(0)$ intersect transversally for all $q \in \mathcal{SS}_{ij} \bigcap \mathcal{SS}_{ik}$. If they do not intersect transversally, then after a small perturbation of one of the manifolds, the intersection of the two manifolds will be transversal. In any event, the transversal intersection means that for all $q \in \mathcal{SS}_{ij} \bigcap \mathcal{SS}_{ik}$, $\nabla(d_i - d_j)(q) \neq \alpha\nabla(d_i - d_k)(q)$ for all $\alpha \in \mathbb{R}$. Therefore, $DG(q)$ has full rank and we can use the preimage theorem to assure us that $\mathcal{SS}_{ijk}$ is indeed a one-dimensional manifold. The GVG in $\mathbb{R}^3$ is then the union of $\mathcal{F}_{ijk} = \{q \in \mathcal{SS}_{ijk} \mid d_i(q) \leq d_h(q) \ \forall h\}$, i.e.,

$$\text{GVG} = \bigcup_i \bigcup_j \bigcup_k \mathcal{F}_{ijk}. \tag{5.3}$$

Structure	Dimension	Codimension	Equidistance	Symbol
GVD	$m-1$	1	2	$\mathcal{F}_{i_1 i_2}$
GVG	1	$m-1$	m	$\mathcal{F}_{i_1,\ldots,i_m}$

Table 5.2 Comparison of the GVD and the GVG.

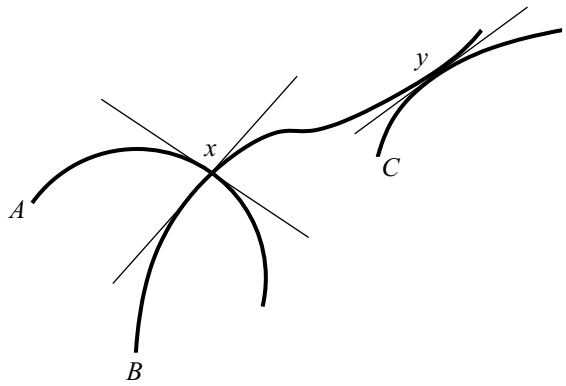

Figure 5.18 The one-dimensional manifolds A and B intersect transversally at x whereas the intersection of B and C at y is not transversal because the tangent spaces at B and C are coincident at y.

In higher dimensions, one can define more equidistant sheets and intersect them to form a GVG. In $\mathbb{R}^m$, the GVG is the set of points equidistant to m obstacles and has dimension one. In contrast, in $\mathbb{R}^m$ the GVD is the set of points equidistant to *two* obstacles and has dimension $m-1$. Sometimes, an $m-k$-dimensional object lying in an m -dimensional space is said to have *codimension* k; therefore the GVD has codimension one, regardless of the space in which it is defined. When $m=2$, the GVG and the GVD coincide. For naming convention refer to the GVG as the "one-dimensional" roadmap structure and thus in the plane we will sometimes call it the planar-GVG. See table 5.2.

Now, let's more formally define transversality. Let M_{int} be the intersection of two submanifolds M_1 and M_2 of M. The intersection is said to be *transversal* if $T_x M_1 + T_x M_2 = T_x M$ for all points $x \in M_{\text{int}}$. Therefore, if M_1 and M_2 are finitely dimensional, transversality implies that $\text{codim}(T_x M_1 \bigcap T_x M_2) = \text{codim}(T_x M_1) + \text{codim}(T_x M_2)$ for all $x \in M_{\text{int}}$. For example, two lines in the plane each have codimension one and their intersection has codimension two, which means a zero-dimensional intersection which is a point. Two two-dimensional planes in $\mathbb{R}^4$ have codimension two and

intersect at a point, which has codimension four in $\mathbb{R}^4$. The transversality assumption is a generalization of the general position assumption that is commonly assumed in the computational geometry literature.

5.3.2 Retract-like Structure Connectivity

Alas, unlike the case in figure 5.15, the GVG is typically not connected and thus is not a roadmap, as can be seen in the example shown in figure 5.19. Here, there is an outer GVG network of one-dimensional manifolds associated with the rectangular enclosure and there is an inner GVG edge associated with the interior box. We term this latter edge a *GVG cycle* which is a GVG edge that is homeomorphic to S^1. In this section, we first explain why the GVG is not connected and then introduce some techniques that can be used to connect disconnected components of the GVG. For a thorough explanation of these procedures, see [106].

The lack of connectivity of the GVG is not the fault of the GVG definition, but rather a consequence of using deformation retractions: in general, there cannot be a one-dimensional deformation retract of a punctured three-or-more-dimensional space. In other words, whereas in the plane we were able to retract the free space onto the

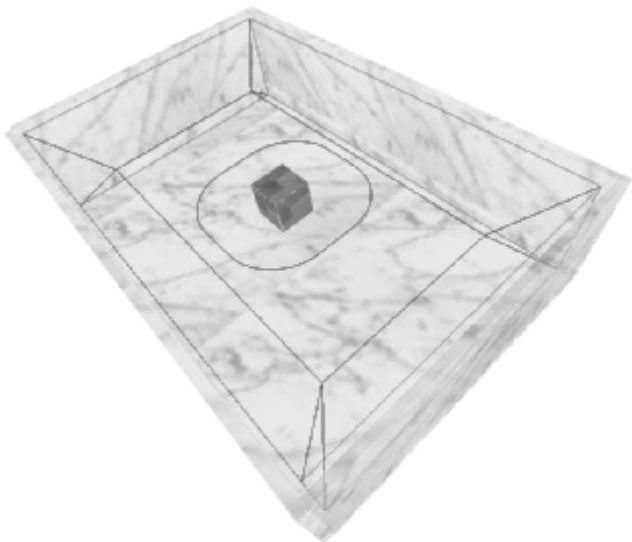

Figure 5.19 A rectangular environment with its ceiling removed to expose a rectangularly shaped box in its interior. The GVG contains two connected components: an outer network similar to the one in figure 5.15 and an inner "halo-like" structure that surrounds the inner box.

GVD with the H mapping (section 5.2.3), in a punctured $\mathbb{R}^3$ there is no continuous function that maps the free space onto a one-dimensional subset that is homotopic to the identity map [63]. The latter condition means that there is no map that "smoothly deforms" the free space onto the one-dimensional structure.

We address the lack of connectivity of the one-dimensional structure by first looking at a connected two-dimensional structure, and then defining one-dimensional structures on the two-dimensional structure to form a roadmap. In $\mathbb{R}^3$ the two-dimensional GVD is connected. In fact, the GVD is a two-dimensional deformation retract of the three-dimensional space. We can exploit this connectivity of the GVD to "patch together" the GVG. Notice that the GVG edges lie on the boundary of the GVD sheets where adjacent GVD sheets intersect. In other words, $\mathcal{F}_{ijk} = \partial\mathcal{F}_{ij} \bigcap \partial\mathcal{F}_{ik} \bigcap \partial\mathcal{F}_{jk}$. Therefore, if, and this is a big if, the boundaries of all two-equidistant sheets were connected, then the resulting GVG would be connected because the GVD is connected. This is the case in figure 5.15 where all two-equidistant faces have connected boundaries. This is not the case in figure 5.19 with the two-equidistant sheet associated with the floor and ceiling; it has a hole in the middle. The boundary of this hole is the GVG edge defined by the floor, ceiling, and interior box. So, our goal now is to connect the boundaries of each of the two-equidistant sheets.

To connect the GVG edges (the boundaries of the two-equidistant faces), we define additional structures called higher-order GVG edges. A second-order GVG edge $\mathcal{F}_{kl}\big|_{\mathcal{F}_{ij}}$ is the set of points where $\mathcal{QO}_i$ and $\mathcal{QO}_j$ are the closest pair of equidistant obstacles and $\mathcal{QO}_k$ and $\mathcal{QO}_l$ are the second-closest, i.e.,

$$\begin{aligned}\mathcal{F}_{kl}\big|_{\mathcal{F}_{ij}} = \{q \mid\ & d_i(q) = d_j(q) \le d_k(q) = d_l(q) \le d_h(q)\ \ \forall h \neq i, j, k, l, \\ & \text{such that } \nabla d_i(q) \neq \nabla d_j(q) \text{ and } \nabla d_k(q) \neq \nabla d_l(q)\}.\end{aligned} \tag{5.4}$$

The first line of equation (5.4) establishes the equidistance relationships: a pair of closest obstacles and a pair of second-closest obstacles. The second line of equation (5.4) ensures that the gradients are distinct, a condition necessary for the preimage theorem to assert that $\mathcal{F}_{kl}\big|_{\mathcal{F}_{ij}}$ is a one-dimensional manifold.

The second-order GVG edges are essentially planar-GVG edges but defined on two-equidistant faces. The preimage theorem guarantees that these edges are one-dimensional and terminate (and intersect) at second-order meet points, denoted as $\mathcal{F}_{klp}\big|_{\mathcal{F}_{ij}}$ (figure 5.20).

We call the union of the GVG and second-order GVG the *hierarchical generalized Voronoi graph* (HGVG), which by itself, as can be seen in figure 5.20, is not connected. However, there is a clue in the second-order GVG that directs the planner to "look for" a separate GVG-connected component. Notice in figure 5.20 that there is a network of second-order GVG edges that form a closed-loop path, which we term a *period,* that has a common second-closest obstacle — the box in the middle of the room.

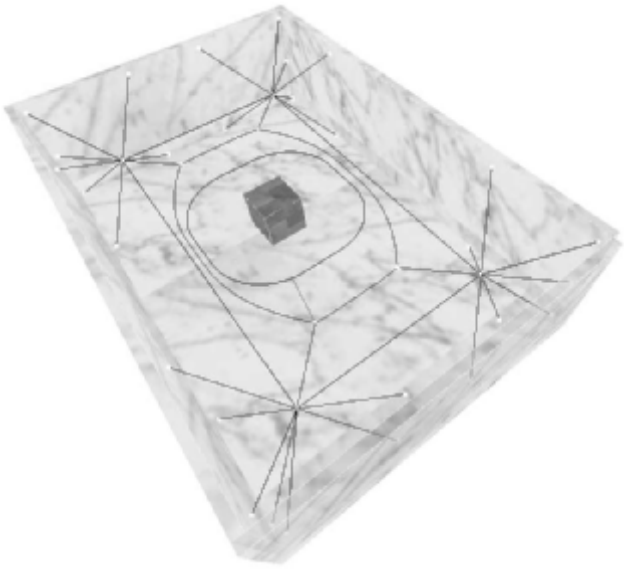

Figure 5.20 The same environment as figure 5.19. The GVG, consisting of the outer network and the "halo" surrounding the inner box, is drawn. The other lines represent the second order GVG edges, each drawn on two-equidistant faces. Observe that the second-order GVG edges form a period that surrounds the GVG cycle in the middle of the free space. Once the period is determined, a link can be made to the inner GVG cycle.

Once a period is detected, the planner can trace a path that maintains two-way equidistance between $\mathcal{QO}_i$ and $\mathcal{QO}_j$ while decreasing the distance to $\mathcal{QO}_k$. Such a path follows, in general, the negative gradient of d_k because we start with $d_k(q) > d_i(q) = d_j(q)$, and decreasing d_k yields a configuration where $d_k(q) = d_i(q) = d_j(q)$. However, in order to maintain double equidistance between $\mathcal{QO}_i$ and $\mathcal{QO}_j$, the negative gradient must be projected onto the two-equidistant face. Hence, the path is $\dot{c}(t) = -\pi_{T_{c(t)}\mathcal{F}_{ij}} \nabla d_k(c(t))$ where the π operator is projection. Following the projected negated gradient $-\pi_{T_{c(t)}\mathcal{F}_{ij}} \nabla d_k(c(t))$ traces a path that terminates on a GVG edge where $d_i(q) = d_j(q) = d_k(q)$ as long as $\pi_{T_{c(t)}\mathcal{F}_{ij}} \nabla d_k(c(t))$ does not vanish. If $\pi_{T_{c(t)}\mathcal{F}_{ij}} \nabla d_k(c(t))$ goes to zero, then no such GVG edge exists in which case the robot returns to the second-order period to continue exploration.

This is just the beginning of what is required for connectivity. Ensuring connectivity can be quite tedious and challenging. See [106] for details of connectivity of the HGVG. The HGVG is a type of *retract-like structure* because it is not a retract, but bears similarities to one.

5.3.3 Lyapunov Control: Sensor-Based Construction of the HGVG

Exploration with the HGVG shares the same key steps as GVD exploration: (1) access the HGVG; (2) explicitly "trace" the HGVG edges; (3) determine the location of nodes; (4) explore the branches emanating from the nodes; and (5) determine when to terminate the tracing procedure. Accessing the GVG (and hence the HGVG) is still gradient ascent, but now it is a sequence of gradient ascent operations. The robot moves away from the closest obstacle until it is two-way equidistant. Then, while maintaining two-way equidistance, the robot increases distance until it is three-way equidistant.

GVG Edge Tracing

Once the robot accesses the GVG, it must incrementally trace GVG edges. Instead of using curve tracing techniques that have discrete steps and discrete corrections, we now derive a control law that smoothly traces the roots of the expression

$$G(q) = \begin{bmatrix} d_1 - d_2 \\ d_1 - d_3 \end{bmatrix} (q) = 0,$$

where d_i is the distance to an object $\mathcal{QO}_i$, and thus if $(d_1 - d_2)(q) = (d_1 - d_3)(q) = 0$, the robot is equidistant to three obstacles and on the GVG. (Likewise, when $\mathcal{Q} = \mathbb{R}^2$, $G(q) = (d_1 - d_2)(q)$, which is zero when the robot is equidistant to two obstacles, a point on the GVD in the plane).

At a point q in the neighborhood of the interior of a GVG edge, the robot steps in the direction

$$\dot{q} = \alpha v + \beta (DG(q))^{\dagger} G(q), \tag{5.5}$$

where

- α and β are scalar gains,
- $v \in \text{Null}(DG(q))$, the null space of $DG(q)$,
- $(DG(q))^{\dagger}$ is the Penrose pseudoinverse of $\nabla G(q)$, i.e., $(DG(q))^{\dagger} = (DG(q))^T (DG(q)(DG(q))^T)^{-1}$.

Note that when q is on the GVG, $G(q) = 0$ and thus $\dot{q} = \alpha v$ where $v \in \text{Null}(\nabla G(q))$ and is simply the tangent direction of the GVG, as prescribed by the preimage theorem. Since $\nabla G(q)$ is a function of distance gradients, the planner can compute $\nabla G(q)$ solely from range sensor information. This can be done by looking at the n-closest points on the n-closest obstacles, fitting a codimension one plane

through these points, and deriving the line orthogonal to this plane. The tangent vector then points along this line.[4]

When q is not on the GVG, then $(\nabla G(q))^{\dagger} G(q) \neq 0$. This term corresponds to the "correction" step which accommodates for curvature in the GVG. Again, this term can easily be determined from sensor data. Whereas the α determines how quickly the robot moves along the GVG, the β represents how aggressively the robot moves back to the GVG, as if α and β were spring constants.

To determine stability of the control law, let $\Gamma = \frac{1}{2} G^T G$ measure the distance a point q is away from the GVG. We look at the first derivative Γ.

$$\begin{aligned}
\dot{\Gamma}(q) &= G^T(q)\, \dot{G}(q) \\
&= G^T(q)\, DG(q)\, \dot{q} \\
&= G^T(q)\, DG(q)\, (\alpha \text{Null}(DG(q)) + \beta\, (DG^{\dagger}(q))\, G(q)) \\
&= \beta\, G^T(q)\, DG(q)\, DG^{\dagger}(q)\, G(q) \\
&= \beta\, G^T(q)\, DG(q)\, DG^T(q)\, (DG(q)\, DG^T(q))^{-1}\, G(q) \\
&= \beta\, G^T(q)\, G(q).
\end{aligned}$$

The function Γ is a Lyapunov function [202] for the controller in equation (5.5). Think of a Lyapunov function as an "error function" whose minimal value is zero. Since $(DG(q)\, DG^T(q))$ is invertible in a neighborhood of the GVG [108], if $\beta < 0$, then $\dot{\Gamma}$ is negative. This assures that Γ decreases to zero, meaning that equation (5.5) directs the robot onto the GVG.

Meet Point Homing

While generating the GVG, the robot must precisely locate itself on the meet points. A meet point homing algorithm can be used to stably converge onto the meet point location [109]. The control law for homing onto a meet point is similar to the one for generating GVG edges, except G is now defined as

$$G(q) = \begin{bmatrix} d_1 - d_2 \\ d_1 - d_3 \\ d_1 - d_4 \end{bmatrix} (q) = 0.$$

In the planar case, $G(q) = \begin{bmatrix} d_1 - d_2 \\ d_1 - d_3 \end{bmatrix} (q)$. Since it has already been shown to be stable, we use the controller in 5.5 to determine the the path for the robot to home

4. Note that there are two choices for this vector, but the planner chooses the direction that directs the robot to continue in the "same" direction.

onto a meet point. Since Null($DG(q)$) $= 0$, the controller is $\dot{q}\beta(DG(q))^{\dagger}G(q)$. Geometrically, this means that when the robot is in the vicinity of the meet point, it draws a sphere through the four closest points on the four closest obstacles (in the planar case, it is a circle through the three closest points). The velocity vector points toward the center of this sphere.

Higher-Order GVG Control Laws

Naturally, by varying the G function, one can trace different structures. A second-order GVG edge has $G(q) = [d_i - d_j, d_k - d_l]^T(q)$. Likewise, a second-order meet point has $G(q) = [d_i - d_j, d_k - d_l, d_k - d_p]^T(q)$.

5.4 Piecewise Retracts: The Rod-Hierarchical Generalized Voronoi Graph

Essentially, the previous roadmaps were defined for a point-robot in a workspace which has a Euclidean configuration space. Now, we turn our attension to defining a roadmap in a non-Euclidean configuration space. Even when full knowledge of the workspace is available prior to the planning event, constructing non-Euclidean configuration spaces can be quite challenging. However, what if the planner has no previous knowledge of the workspace, i.e., it cannot compute the configuration space prior to the planning event? Instead, we define a roadmap for a robot using workspace information. This is of great use because sensors directly provide workspace information.

In this section, we define a roadmap for a line segment operating in the plane. Sometimes we call this line segment a *rod* (figure 5.21). To distinguish among previous roadmaps we have defined, let the point-GVG and the point-HGVG be structures defined for a point robot in a Euclidean configuration space.

Since the configuration space for the rod is $SE(2)$, which is three-dimensional, it makes sense to look at the set of configurations equidistant to three obstacles. However, we measure distance in the *workspace,* not configuration space. To do so, let $R(q) \subset \mathcal{W}$ be the set of points the rod occupies in the workspace when it is at configuration q. At the risk of confusing notation, we re-use the d_i for distance to obstacle $\mathcal{WO}_i$ for the rod robot, i.e.,

$$d_i(q) = \min_{r \in R(q), c \in \mathcal{WO}_i} d(r, c).$$

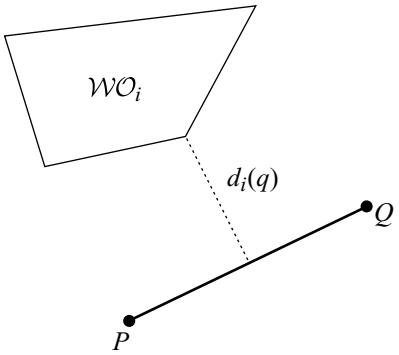

Figure 5.21 The distance from the rod (thick black line) to an obstacle is the distance (dotted line) between the nearest point on the rod to the obstacle and the nearest point on the obstacle to the rod.

Using this notion of distance, a rod-two equidistant face $\mathcal{RF}_{ij}$ is $\{q \in SE(2) \mid d_i(q) - d_j(q) = 0 \text{ and } \nabla d_i(q) \neq \nabla d_j(q)\}$. Then the rod-GVG edge is $\mathcal{RF}_{ijk} = \mathcal{RF}_{ij} \bigcap \mathcal{RF}_{ik} \bigcap \mathcal{RF}_{jk}$ [107].

Just like the point-GVG in $\mathbb{R}^3$, the collection of rod-GVG edges does not necessarily form a connected set. To produce a connected structure we introduce another type of edge, called an *R-edge*. An R-edge is the set of rod configurations that are tangent to the planar point-GVG. The rod-HGVG then comprises rod-GVG edges and R-edges (figure 5.22).

The rod-HGVG is a *piecewise retract* because it is formed by the union of deformation retracts of subsets of the configuration space, the rod-GVG edges, which are then linked together with the R-edges to form a connected roadmap. To show that the rod-HGVG is indeed a piecewise retract, we first consider deformation retraction $H_{\text{rod}} : \mathcal{Q}_{\text{free}} \times [0, 1] \rightarrow \text{rod} - \text{GVG}$, which is implicitly defined by a sequence of gradient ascent operations. First the rod is moved away from the closest obstacle, holding its orientation fixed, until it becomes two-way equidistant. Then, while maintaining both two-way equidistance and the fixed orientation, the rod moves until it becomes three-way equidistant, i.e., it arrives at a configuration on the rod-GVG. Note that the orientation of the rod is unchanged by this operation, implying that $\theta(q) = \theta(H_{\text{rod}}(q, s))$ for all $s \in [0, 1]$, where $\theta(q)$ is the orientation of the rod at configuration q.

Naturally, H_{rod} is not continuous over the entire configuration space, but we can restrict ourselves to simple subsets of configuration space that are associated with individual rod-GVG edges. Let $\mathcal{RF}_{ijk}$ be the rod-GVG edge associated with obstacles

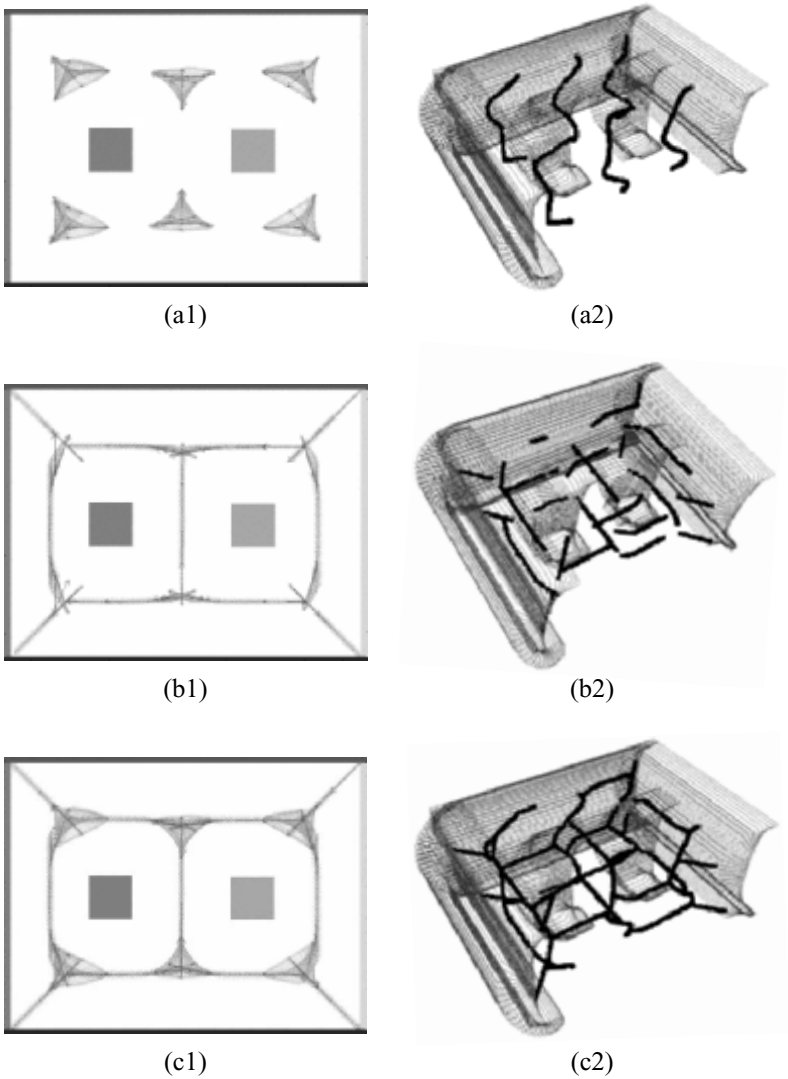

Figure 5.22 Swept volumes (sampled placements) of the rod in a planar workspace and the configurations of the rod in configuration space. (a1) Rod-GVG-edges: each of the clusters represents a set of configurations equidistant to three obstacles. (a2) The configurations of the rod that are equidistant to three obstacles in the *workspace*. (b1) R-edges: the rods are two-way equidistant and tangent to a planar point-GVG edge. (b2) The configurations of the rod that are tangent to the planar point-GVG in the workspace. (c1) Placements of the rod along the rod-HGVG. (c2) The entire rod-HGVG in $SE(2)$.

$\mathcal{WO}_i$, $\mathcal{WO}_j$ and $\mathcal{WO}_k$. It can be shown that H is continuous in the preimage of a connected component of the rod-GVG edge $\mathcal{RF}_{ijk}$ [107]. This preimage, which we denote as a *junction region* J_{ijk}, is contractable and thus has as a retract $\mathcal{RF}_{ijk}$.[5]

Since, for each connected component of a junction region J_{ijk}, there is a connected component of $\mathcal{RF}_{ijk}$, motion planning within one junction region can be accomplished by using $\mathcal{RF}_{ijk}$ as a roadmap. If the union of all the $\mathcal{RF}_{ijk}$'s form a connected set, then planning would be trivial again. However, in general, the $\mathcal{RF}_{ijk}$'s will not form a connected set in $SE(2)$, so we use the R-edges to connect the roadmaps for the junction regions. Intuitively, one can view the rod-GVG edges as being analogous to the nodes of the planar point-GVG, which are connected by the edges of the planar point-GVG. The distinction is that now the nodes are themselves one-dimensional structures, capturing the complete set of rod orientations that are three-way equidistant from a specific set of obstacles (see [107] for a rigorous explanation). Essentially, the R-edges encode the adjacency of the rod-GVG edges by inheriting the correct topological relationships from the plane, allowing us to construct a roadmap of configuration space using the connectivity of the workspace. See [107] for more details.

The rod-HGVG edges can be constructed in a sensor-based fashion using the control laws from section 5.3.3.

5.5 Silhouette Methods

In contrast to looking at equidistance, the silhouette approaches use extrema of a function defined on a codimension one hyperplane called a *slice*[6], which we denote by $\mathcal{Q}_\lambda$. The λ parameterizes the slice; varying the parameter λ has the effect of sweeping the slice through the configuration space. As the slice is swept through the configuration space, for each value of λ, the critical points of a function restricted to the slice are determined. The trace of the critical points as the slice is swept through the configuration space does not necessarily form a connected set. Therefore, the silhouette methods look for another type of critical point and then recursively call the algorithm on a slice passing through these critical points. The resulting network of extremal points forms the roadmap.

5. Note that if the rod were "small," $\mathcal{RF}_{ijk}$ would have one connected component which would be homeomorphic to S^1 (figure 5.22(a2)).

6. When the slice is one-dimensional, it can also be called a sweep line (section 5.1).

5.5.1 Canny's Roadmap Algorithm

Roadmap theory in motion planning begins with Canny's work [90]. In addition to developing the roadmap, Canny's work established fundamental complexity bounds using roadmap theory. For an environment populated by obstacles whose boundaries can be represented as p polynomials of maximum degree w for some positive w in configuration space, any navigation path-planning problem can be solved in $p^n(\log p)w^{(O(n^4))}$ time using his roadmap algorithm, where n is the degrees of freedom of the robot (the dimension of the configuration space). The derivation of this result is beyond the scope of this book. See [91, 92] for details.

In this method, the choice of initial sweep direction is arbitrary, but for the sake of discussion, let's choose the q_1-direction. As the slice is swept in the q_1-direction, "extremal points" in the q_2-direction are determined in each slice. The extremal points in the q_2-direction are extrema of the projection function $\pi_2 : \mathbb{R}^{m-1} \to \mathbb{R}$ where $\pi_2(q) = q_2$. The extremal points of π_2 for all of the slices are the *silhouette curves.*

In general, the silhouette curves are not guaranteed to be connected, and hence may not form a roadmap. However, we can look at the slices where the number of silhouette curves changes. These slices are called *critical slices,* and the λ values that parameterize critical slices are *critical values.* The points on the silhouette curves where the silhouette curves are tangent to the critical slices are termed *critical points.*

On the critical slices, the silhouette algorithm is recursively invoked where the new swept slice now has one less dimension than the critical slice, i.e., it has codimension two in the ambient space and codimension one in the critical slice. This slice is swept in the q_2-direction. The new silhouette comprises the trace of extremal points in the q_3-direction. These silhouette curves may not be connected either, so this procedure is recursively invoked on lower-dimensional critical slices until there are no more critical points or the slice has one dimension. In the latter case, the one-dimensional slice is the silhouette; in other words, the roadmap of a one-dimensional set is the set itself. Finally, the union of the resulting silhouette curves forms the roadmap.

Accessibility and departability of the roadmap are achieved by treating the slices that contain q_{start} and q_{goal} as critical $(m-1)$-dimensional slices of the initial sweep. The algorithm simply forms a silhouette network on these slices, possibly reinvoking itself on lower-dimensional slices. Connectivity is proved via an inductive argument [90]. See [189] for details on an example of an implementation of Canny's roadmap.

Figure 5.23 contains an example of a two-dimensional configuration space with a slice being swept through it. The silhouette curves trace the boundary of the environment. Critical points occur when the slice is tangent to the roadmap (and hence the obstacle boundary), as can be seen in figure 5.23. The resulting roadmap is drawn in figure 5.24. A path between a start and goal configuration is determined by first

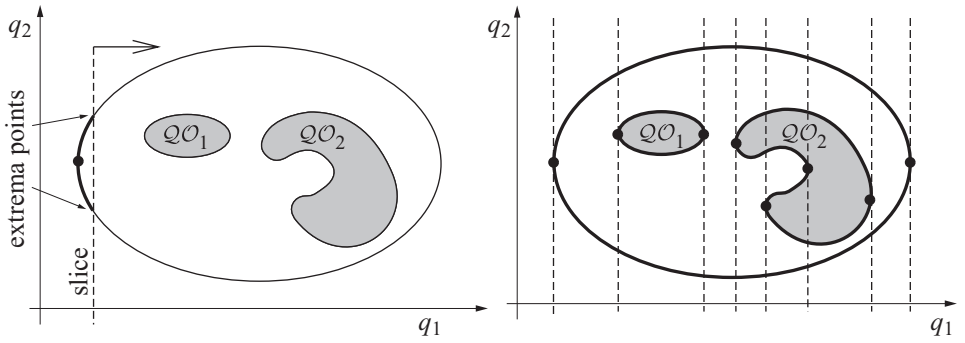

Figure 5.23 Bounded two-dimensional environment with two obstacles $\mathcal{QO}_1$ and $\mathcal{QO}_2$. The left figure contains a single slice, represented by a dashed line, and a partially constructed silhouette. The right figure contains the complete silhouette and slices passing through all critical points.

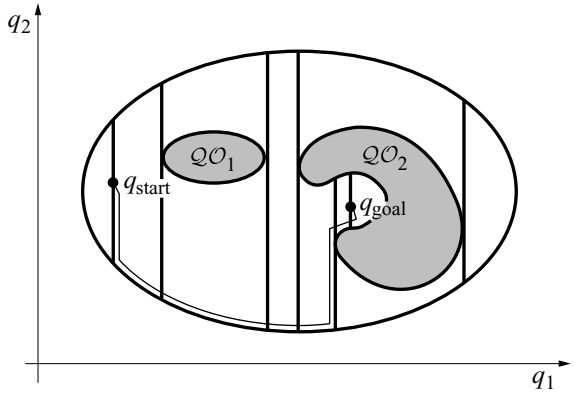

Figure 5.24 Complete silhouette curves traced out with solid lines. A path from start to goal is denoted as a thin curve.

passing a slice through the start and goal, and then including the slice in the roadmap, as can be seen in figure 5.24.

Figure 5.25 contains an example of a two-dimensional surface embedded in $\mathbb{R}^3$. It is an ellipsoid with a hole drilled partially down and then up again. The slice is swept from left to right and extrema are with respect to the in and out of page direction. The silhouette curves comprise an "equator" for the ellipsoid, the perimeter of the holes on the surface and the two curves along the side of the hole. Figure 5.26

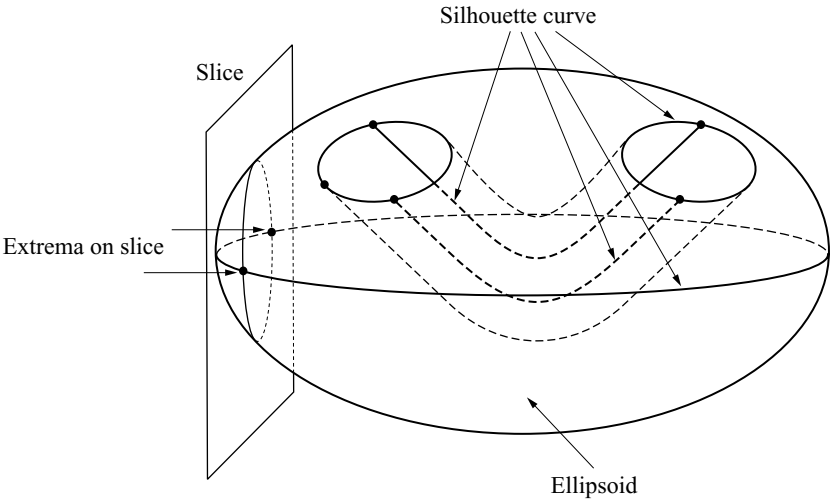

Figure 5.25 Silhouette curves for an ellipsoid with a hole drilled through it that goes down and then bends up.

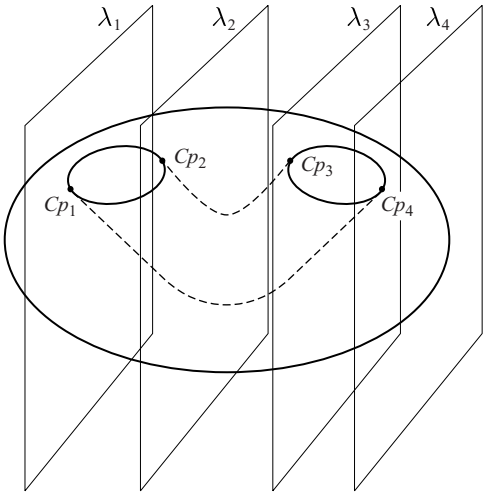

Figure 5.26 Slices passing through the critical points, which are points where the roadmap changes connectivity and is tangent to the slice. Note that the leftmost and rightmost points on the ellipsoid also are critical points but are not displayed.

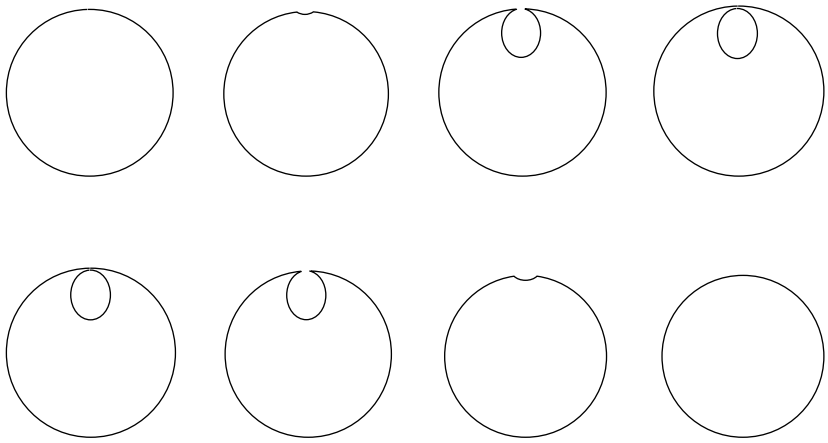

Figure 5.27 The intersection of the slice as it is swept through the ellipsoid with a hole in it displayed in figure 5.26. Starting from the top row on the left, the first two panels (a and b) display the intersection immediately before and after the slice passes through the critical point Cp_1. The next two panels (c and d) display the intersection as the slice passes through critical point Cp_2. The left pair of panels on the bottom row (e and f) correspond to critical point Cp_3 and the right pair (g and h) to Cp_4.

displays the critical slices and critical points for the ellipsoid. Figure 5.27 shows the intersection of the slice and the ellipsoid, immediately before and immediately after, the critical points. Starting from the left in the top row, the first two panels show the intersection just as the slice encounters the first hole. The next two panels show the intersection as the slice finishes passing through the hole. At this critical point, the intersection changes connectivity. Finally, figure 5.28 shows the silhouettes on the two-dimensional slices and the final path for this example between q_{start} and q_{goal}.

Critical Points and Morse Functions

In this section, we define the silhouette curves in terms of critical points of a function. The function has to be Morse [315], as described in chapter 4, section 4.6. The slices themselves are also defined in terms of a function. Originally, Canny suggested that a slice be the preimage of the projection operator π_1. Recall that π_1 projects a point onto its first coordinate, i.e., $\pi_1(q) = q_1$. We denote a slice as $\mathcal{Q}_\lambda = \{x \in \mathcal{Q} \mid \pi_1(q) = \lambda\}$ where $\lambda = q_1 \in \mathbb{R}$. Varying λ has the effect of sweeping the slice through the configuration space and $\bigcup_\lambda \mathcal{Q}_\lambda = \mathcal{Q}$. On each $\mathcal{Q}_\lambda$, we look for extrema of π_2, i.e.,

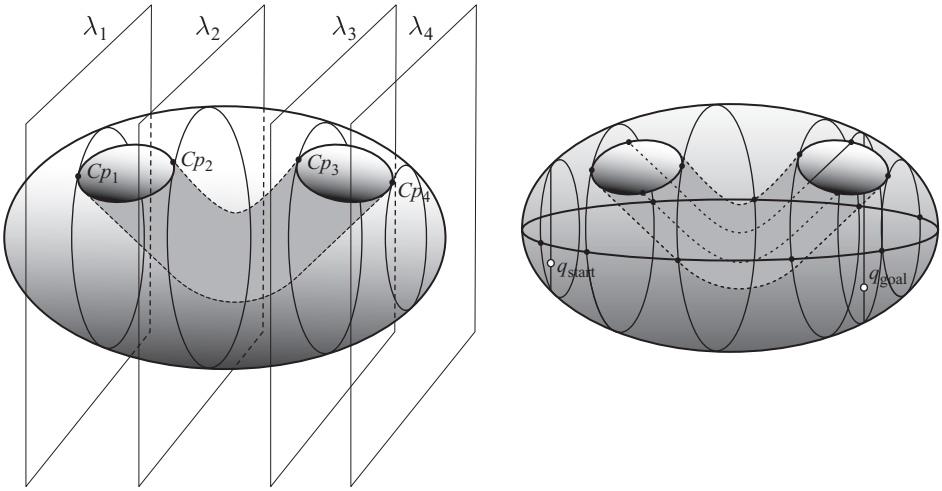

Figure 5.28 (Left) Silhouettes on the two-dimensional slices. (Right) Determining a path with a given start and goal position.

we look for extrema of $\pi_2|_{\mathcal{Q}_\lambda}$, where $\pi_2|_{\mathcal{Q}_\lambda}$ is the projection operator restricted to the slice. Also, recall that $\pi_2(q) = q_2$.

To determine the extrema, we need some machinery to calculate extrema of functions restricted to manifolds. The Lagrange multiplier theorem, stated below, can be used to determine the extrema of real-valued functions restricted to manifolds, which themselves are defined by the preimage of real-valued functions.

LEMMA 5.5.1 (Lagrange Multiplier [410]) *Let S be an n-manifold in $\mathbb{R}^{n+1}$, $S = f^{-1}(c)$ where $f : \mathbb{R}^{n+1} \to \mathbb{R}$ is such that $\nabla f(q) \neq 0$ for all $q \in S$. Suppose $h : \mathbb{R}^{n+1} \to \mathbb{R}$ is a smooth function and $p \in S$ is an extremal point of h on S. Then, there exists a real number μ such that $\nabla h(p) = \mu \nabla f(p)$ (the number μ is called the Lagrange multiplier). In other words, $\nabla f(p)$ is parallel to $\nabla h(p)$ at an extremum p of h on S.*

For example, let's look for extrema of $h = \pi_1$ on a sphere defined by the preimage of any positive scalar under the map $f(q) = q_1^2 + q_2^2 + q_3^2 - 51{,}141$. The gradients are $\nabla h(q) = [1, 0, 0]^T$ and $\nabla f(q) = [2q_1, 2q_2, 2q_3]^T$. These two vectors are parallel when $q_2 = q_3 = 0$ and the only points on $f^{-1}(0)$ that satisfy this condition are on the left-most and right-most points on the sphere (which are in the $q_1 - q_2$ plane).

We could assume that the free space is defined by the preimage of a function f, but there is no guarantee that this function will be real-valued. To determine extrema of a function restricted to such manifolds, Canny generalizes the Lagrange multiplier theorem to handle vector-valued functions.

LEMMA 5.5.2 (Generalized Lagrange Multiplier Theorem [91]) *Let M be the preimage of $f : \mathbb{R}^m \to \mathbb{R}^p$ and $h : \mathbb{R}^m \to \mathbb{R}^n$. The point x is a critical point of $h|_M$ if and only if the following matrix loses its rank [91],*

$$D(f,h)_q = \begin{bmatrix} \frac{\partial f_1}{\partial q_1}(q) & \cdots & \frac{\partial f_1}{\partial q_m}(q) \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial q_1}(q) & \cdots & \frac{\partial m_n}{\partial q_m}(q) \\ \frac{\partial h_1}{\partial q_1}(q) & \cdots & \frac{\partial h_1}{\partial q_m}(q) \\ \vdots & \ddots & \vdots \\ \frac{\partial h_q}{\partial q_1}(q) & \cdots & \frac{\partial h_p}{\partial q_m}(q) \end{bmatrix}. \tag{5.6}$$

Clearly, if f and h are real-valued functions (i.e., $n = p = 1$), then lemma 5.5.2 reduces to the Lagrange multiplier theorem because the above two-row matrix only loses rank when one row is a scalar multiple of the other. In other words, the two vectors corresponding to each row are parallel.

Canny introduces one more result, termed the *slice lemma.* The notation can be a bit cumbersome, so let's review it before introducing the slice lemma. Let π_{12} be the projection operator onto the first two coordinates, e.g., $\pi_{12}(q) = (q_1, q_2)$. Then, $\pi_{12}|_S$ is the projection operator restricted to the set S. Finally, $\Sigma(\pi_{12}|_S)$ is the set of critical points of the projection operator restricted to S. The slice lemma then states that the set of critical points of $\pi_{12}|_S$ is the union of the critical points of π_2 on each of the slices, i.e.,

$$\Sigma(\pi_{12}|_S) = \bigcup_{\lambda} \Sigma\left(\pi_2|_{\pi_1^{-1}(\lambda)}\right).$$

Therefore, the silhouette is the critical set of π_{12}.

With the slice lemma and Canny's generalization in hand, one can produce silhouette curves. Consider again the example of a sphere embedded in $\mathbb{R}^3$. Here, $S = f^{-1}(0)$ where $f(q) = q_1^2 + q_2^2 + q_3^2 - 62{,}370$. We sweep in the q_1-direction and extremize in the q_2-direction, i.e., $h(q) = \pi_{12}(q_1, q_2, q_3)$. Applying lemma 5.5.2,

$$D(f,h) = \begin{bmatrix} 2q_1 & 2q_2 & 2q_3 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix},$$

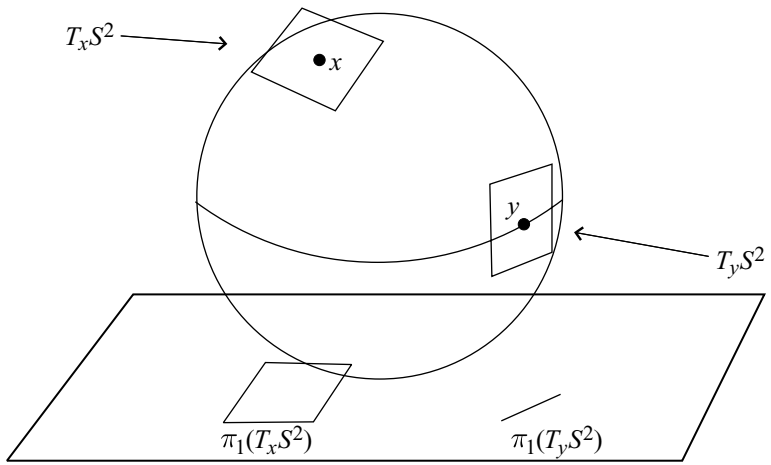

Figure 5.29 The tangent spaces of of S^2 at x and y are projected down to a plane. T_xS^2 projects to a two-dimensional space whereas T_yS^2 does not, making y a critical point.

which loses rank on S only when $q_3 = 0$, which corresponds to the unit circle in the $q_1 - q_2$-plane, which is the "equator" of the sphere (figure 5.29). This is the silhouette of the sphere.

Connectivity Changes at Critical Points

Canny's roadmap has two types of critical points: those that define the silhouette itself, as described above, and those that are used to bridge disconnected silhouette curves. We now describe the latter. In particular, we relate the concepts of a "first derivative" vanishing to connectivity changes in the silhouettes.

First, let's consider a planar example. We are now looking for extrema of the slice function $h = \pi_1$, but restricted to the silhouette, which are extrema of $\pi_1|_{\Sigma(\pi_{12})}$. Figure 5.30 depicts two sample critical points, Cp_1 and Cp_2, which are located on the boundaries of the obstacles, $\partial\mathcal{QO}_1$ and $\partial\mathcal{QO}_2$, respectively. Again, in two-dimensions, the silhouettes essentially trace out the boundaries of the free space (and obstacles) and thus $\Sigma(\pi_1|_{\Sigma(\pi_{12})}) = \Sigma(\pi_1|_{\partial\mathcal{Q}_{\text{free}}})$.

We can intuitively show that Cp_1 and Cp_2 are indeed critical points, i.e., $\pi_1|_{\Sigma(\pi_{12})}$ takes its local extrema at Cp_1 and Cp_2. The function $\pi_1(q)$ can be viewed as measuring the distance between a point $q \in \mathcal{Q}$ and the q_2-axis. Therefore, consider a path on $\partial\mathcal{QO}_1$ that passes through Cp_1, as depicted in figure 5.30. Moving along the path toward Cp_1 decreases the value of $\pi_1|_{\Sigma(\pi_{12})}$. After passing through Cp_1, the value

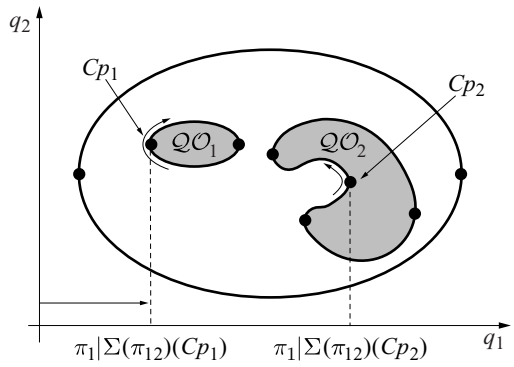

Figure 5.30 The restriction of the slice function $h = \pi_1$ to the silhouette takes a local minimum at Cp_1 and a local maximum at Cp_2. The values $\pi_1|_{\Sigma(\pi_{12})}(Cp_1)$ and $\pi_1|_{\Sigma(\pi_{12})}(Cp_2)$ are plotted on the bottom.

increases. In other words, Cp_1 is a local minimum of $\pi_1|_{\Sigma(\pi_{12})}$. Likewise, Cp_2 is a local maximum of $\pi_1|_{\Sigma(\pi_{12})}$.

Now, let's return the discussion to $\mathbb{R}^m$. Now, we demonstrate that a critical point is indeed a point on the roadmap where the tangent to the roadmap lies in the slice. This is actually a direct result of lemma 5.5.2. Recall that q is a critical point of π_1 restricted to the manifold defined by the preimage of f if $D(f, \pi)(q)$ loses rank. Here, we would like to define the roadmap as the preimage of f, but cannot do so. Instead, we can reason about the differential of f, which is an $m - 1$-by-m matrix. The null space of this matrix is the tangent to the roadmap and the $m - 1$ row vectors of the same matrix form a plane orthogonal to the tangent. Call this plane $T^{\perp}$. Finally, this matrix forms the top $m - 1$ rows of $D(f, h)$. The slice function π_1 has a gradient $[1, 0, \ldots, 0]^T$ and forms the bottom row of $D(f, \pi_1)(q)$. When the tangent lies in the slice plane, the slice plane and $T^{\perp}$ are orthogonal to each other. This means that $\nabla\pi_1(q)$ lies in $T^{\perp}$ which immediately implies that $\nabla\pi_1(q)$ can be written as a linear combination of the first $m - 1$ rows of $D(f, \pi)(q)$. In other words, $D(f, \pi_1)(q)$ loses rank because its bottom row can be written as a linear combination of the top $m - 1$ rows. Therefore q is a critical point.

This can also be seen in three dimensions. Consider the torus in figure 5.31. Here two slices are drawn, one before a critical point and one after. Before the critical point, the intersection of the slice and the torus is diffeomorphic to S^1 and after the intersection it is diffeomorphic to two copies of S^1. In figure 5.32, it can be seen that before the critical point, the roadmap is singly connected and after the intersection it has two connected components.

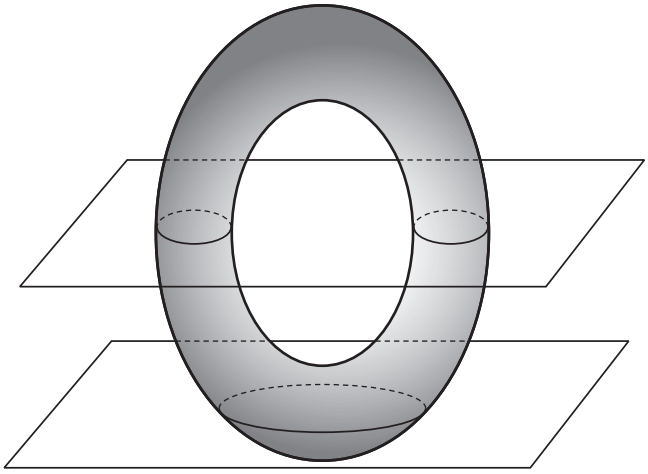

Figure 5.31 Torus with two slices drawn, before and after a critical point.

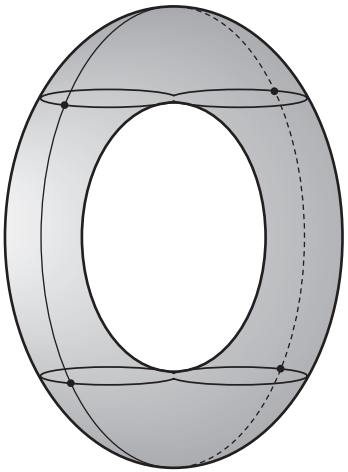

Figure 5.32 Silhouette curves on the torus.

Let's formalize the immediately "before" and "after" statements. Since a real-valued Morse function has a one-dimensional range which can be ordered, the critical values of the Morse function can be ordered as well. Assuming only one critical point per slice, adjacent critical points are those whose critical values are "next" to each other. In other words, let Λ be the set of all critical values. The critical values

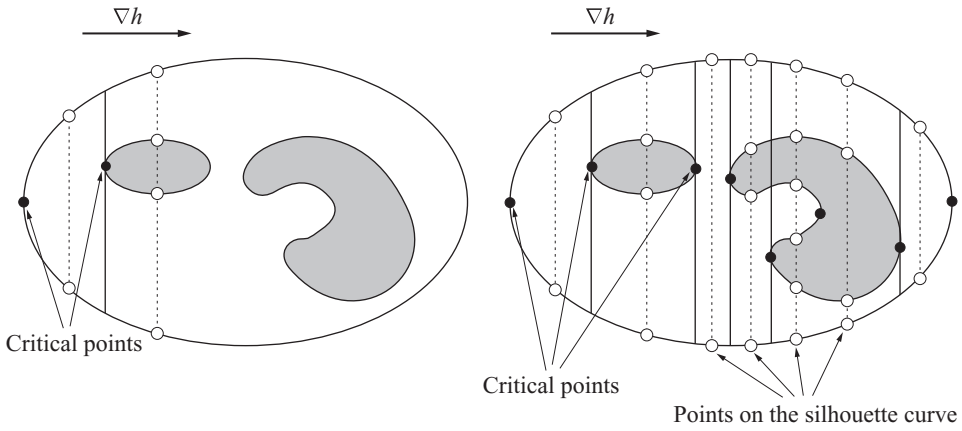

Figure 5.33 The number of silhouette fragments (open circles) changes as the slice passes through critical points (black circles) and remains constant between adjacent critical points.

$\lambda_1, \lambda_2 \in \Lambda$ are *adjacent* if for all critical values in Λ there does not exist a critical value $\bar{\lambda}$ such that $\lambda_1 < \bar{\lambda} < \lambda_2$.

Morse theory asserts that between adjacent critical points of a Morse function, the topology of the manifold on which the Morse function is defined does not change [315]. In the context of the slice function, Morse theory states that there exists a diffeomorphism ϕ such that for all $\lambda_1, \lambda_2 \in (\lambda_*, \lambda^*)$, $\phi(\pi_1|_{\Sigma(\pi_{12}^{-1}(\lambda_1))}) = \phi(\pi_1|_{\Sigma(\pi_{12}^{-1}(\lambda_2))})$, where λ_* and λ^* are adjacent critical values of a real-valued Morse function (figure 5.33).

5.5.2 Opportunistic Path Planner

The *opportunistic path planner* (OPP) generalizes Canny's original roadmap algorithm by tracing the local maxima of any potential function that is Morse on a flat slice as the slice is swept through the configuration space. Canny and Lin [93] suggest that the distance function D evaluated on the slice be used as the potential function. Local maxima on the slice of the distance function are points on the OPP roadmap. The traces of the local maxima as the slice is swept through the workspace or configuration space are termed *freeways*.

The algorithm works as follows: First, a fixed slice direction is chosen. The algorithm initially traces a path from the start to the roadmap by performing gradient ascent on the distance function in the slice that contains the start. Likewise, a path is traced from the goal to the freeway via slice-constrained gradient ascent. These two actions correspond to accessibility and departability.

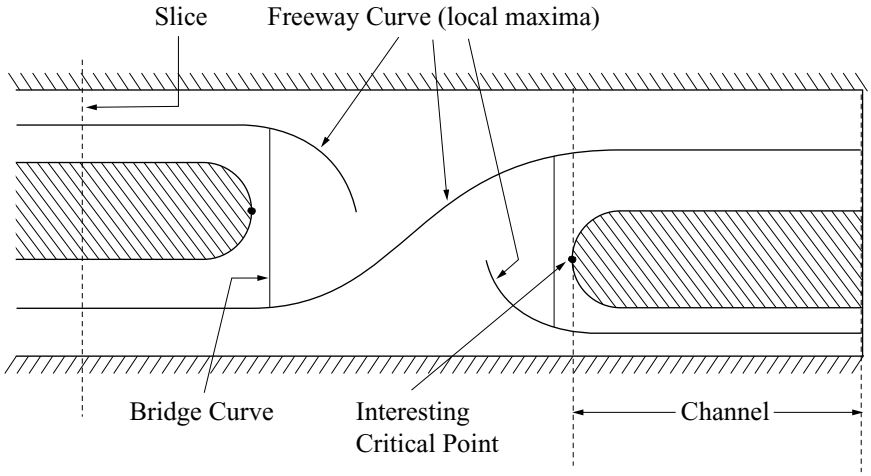

Figure 5.34 Schematic of the OPP planning scheme.

From the point at which the planner accesses the OPP roadmap, the algorithm sweeps a slice through the configuration space tracing local maxima of D constrained to the slice. These local maxima form a freeway. If the start and goal freeways are connected, then the algorithm terminates. In general, the set of freeways will not be connected, and paths between neighboring freeways must be found.

The OPP method uses a slightly different approach from Canny's original roadmap to ensure connectivity of its roadmap. The OPP freeways are connected via *bridge curves*. The bridge curves are constructed in the vicinity of *interesting critical points*. Interesting critical points occur when *channels* (figure 5.34) join or split on slices whose connectivity changes in the free space. Bridge curves are also built when freeways terminate in the free space at bifurcation points (where traces of local maxima and local minima meet). A bridge curve is built leading away from a bifurcation point to another freeway curve.

This procedure is repeated until the start and goal freeway curves are connected, or all interesting critical points and bifurcation points have been explored, in which case there does not exist a path between the start and the goal. The union of bridge and freeway curves, sometimes termed a *skeleton,* forms the one-dimensional roadmap.

Connectivity and Critical Points

Instead of looking for connectivity changes in the roadmap, the OPP method looks for connectivity changes in the slice in the free configuration space. Again, these

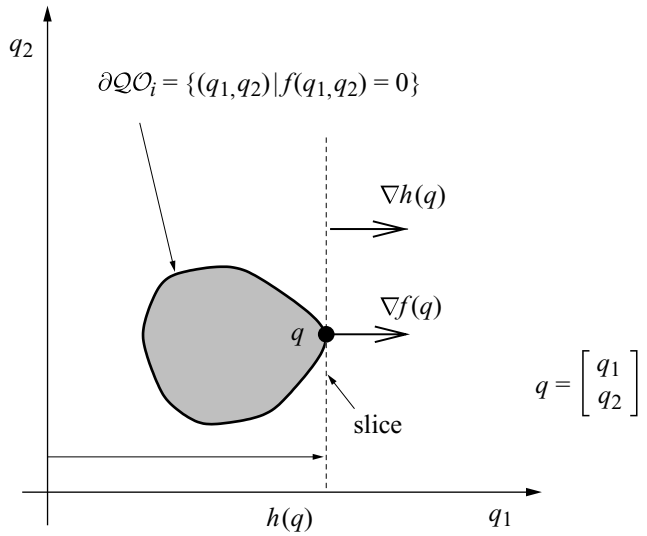

Figure 5.35 At the critical point q, the gradient of the slice function $\nabla h(q)$ is parallel to surface normal of the obstacle $\nabla f(q)$. Also, the slice is tangent to the boundary of the obstacle $\mathcal{QO}_i$ at the critical point q.

connectivity changes correspond to a slice function taking on extremal values. This can be seen in figure 5.30, except now we are looking at the slice function π_1 *restricted to the boundary of the free space,* as opposed to being restricted to the silhouette (both of which coincide in the plane).

Again, $D(f, h)$ loses rank at the critical points. Here, the f function can be used to define the boundaries of the obstacles. In other words, we assume that the boundaries of the obstacles can be represented as the preimage of 0 under the f mapping. Therefore, for $q \in \partial\mathcal{QO}_i$, $\nabla f(q)$ is the surface normal to $\mathcal{QO}_i$ at q. Now, $D(f, h)$ has two rows and loses rank only when $\nabla f(q)$ is parallel to $\nabla h(q)$ which means that the slice gradient is parallel to the surface normal of the obstacle. (Note that we could have used the original Lagrange multiplier theorem here.) See figure 5.35.

Morse theory [315] assures that the topology of the intersection of the boundary and the slice remains constant between critical points, i.e., there exists a diffeomorphism ϕ such that for all $\lambda_1, \lambda_2 \in (\lambda_*, \lambda^*)$, $\phi(h|_{\partial\mathcal{Q}_{\text{free}}}{}^{-1}(\lambda_1)) = \phi(h|_{\partial\mathcal{Q}_{\text{free}}}{}^{-1}(\lambda_2))$, where λ_* and λ^* are adjacent critical values of a real-valued Morse function. Therefore, we are assured that we only need to look for critical points to connect disconnected components of the roadmap.

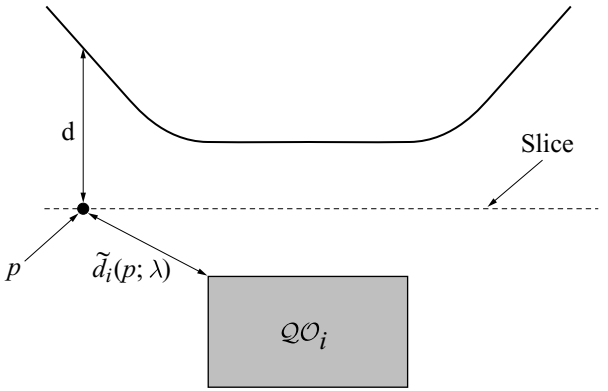

Figure 5.36 The dashed line represents a slice that is hovering above obstacle $\mathcal{QO}_i$. The solid line above the slice is the graph of the distance to the obstacle, but restricted to the slice, i.e., $\tilde{d}_i$.

Nonsmooth Functions

It should be noted that the distance function is nonsmooth. Consider the distance function constrained to a slice $\mathcal{Q}_\lambda = \{q \mid \pi_1(q) = \lambda\}$. Decompose the configuration space coordinates q into "slice coordinates" p and the "sweep coordinate" λ such that $q = [\lambda, p]^T$. The single object distance function *constrained to a slice* is the distance between a point that is in a slice $\mathcal{Q}_\lambda$ and a set $\mathcal{QO}_i$, i.e.,

$$\tilde{d}_i(p\,;\lambda) = d_i(q) \qquad \text{where } \pi_1(q) = \lambda \quad \text{and} \quad p \in \pi_1^{-1}(\lambda). \tag{5.7}$$

See figure 5.36 for an example of the distance function plotted along a slice. At each slice point, $\tilde{d}_i$ is computed to the closest point of the obstacle.

Typically, a robot's environment is populated with multiple obstacles, and thus we define a distance function for multiple obstacles. The multi-object distance function *constrained to a slice* measures the distance between a point in a slice $\mathcal{Q}_\lambda$ and the *closest* obstacle to that point, i.e.,

$$\tilde{D}(p\,;\lambda) = \min_i \tilde{d}_i(p\,;\lambda). \tag{5.8}$$

Even when all of the obstacles are smooth and convex, $\tilde{D}$ is not necessarily smooth at the local maxima. For example, in figure 5.37 distance $D(q)$ is plotted along a horizontal slice. On the left-hand side of the slice, since $\mathcal{QO}_1$ is the closest obstacle, $D(q) = d_1(q)$. Likewise, on the right-hand side of the slice, $D(q) = d_2(q)$. When $d_1(q) = d_2(q)$, D is nonsmooth, but for all other points, $D(q)$ is smooth because it inherits the smoothness properties of the single object distance function for convex

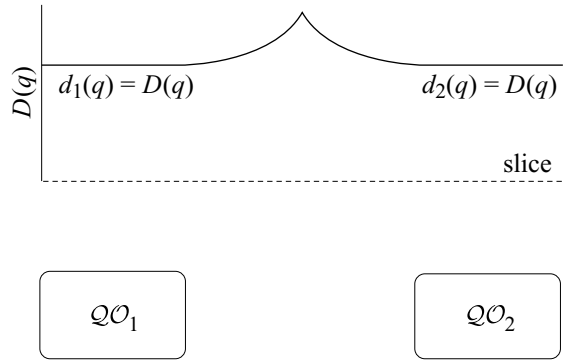

Figure 5.37 Distance function D plotted along a horizontal slice. The slice is represented as a dashed line. The graph of D is overlaid on top of the slice. Note D becomes nonsmooth when $d_1(q) = d_2(q)$, and hence there is not a unique closest obstacle.

sets. Therefore, the gradient vector is either $\nabla d_1(q)$ or $\nabla d_2(q)$ depending upon which obstacle is the unique closest one. However, at the point q^* where D is nonsmooth, the gradient is no longer unique. In fact, it is the set formed by the convex hull of $\nabla d_1(q^*)$ and $\nabla d_2(q^*)$. This gradient is termed a *generalized gradient* [114] and is denoted as

$$\begin{aligned}\partial D(q^*) &= \text{Co}\{\nabla d_i(q^*) \mid i \in Z(q^*)\} \\ &= \sum_{i \in Z(q^*)} \mu_i \nabla d_i(q^*) \quad \text{where} \quad \sum_{i \in Z(q^*)} \mu_i = 1 \quad \text{and} \quad \mu_i > 0,\end{aligned}$$

where Co is the convex hull operator and $Z(q^*)$ is the set of integers that correspond to the indices of the closest obstacles to q^*, i.e., $Z = \{i \mid \text{for all } i \text{ where } d_i(q^*) < d_h(q^*) \text{ for all } h\}$.

With this notion of a generalized gradient, we can establish a calculus for characterizing extrema of a function by looking at the convex hull of the generalized gradient of D [104]. Let 0 be the origin if the tangent space $T_{q^*}\mathbb{R}^m$. If $0 \in \partial D(q^*)$, then q^* is a local maximum. Likewise, if $0 = \partial D(q^*)$, then q^* is a local minimum. It is worth noting that we never had to perform an additional differentiation but were able to characterize the generalized gradient from first-order information.

Problems

1. Prove that the visibility graph is connected.
2. Show an example for which the visibility graph does not produce the shortest path in $\mathbb{R}^3$.

3. How can the visibility graph method be augmented so as to yield the shortest path in $\mathbb{R}^3$?
4. How can the visibility graph in the plane be adapted to handle curved obstacles.
5. Write a program to compute the visibility graph. The program should take as input from a file a list of polygons, which are in turn represented by a list of vertices. The user can input from the keyboard the start and goal configurations. The program then computes the visibility graph and then determines a path from start to goal.
6. Let S be the unit circle defined by the preimage of zero under $f(x, y) = x^2 + y^2 - 1$. Let $g(x, y) = ax^2 + 2bxy + cy^2$ where $a, b, c \in \mathbb{R}$. List the points where g is extremized on S. Draw a picture.
7. Draw the Canny roadmap for the surface configuration space in figure 5.38.
8. Do connectivity changes in the free space in a slice imply connectivity changes in the original Canny roadmap? In the OPP roadmap?
9. What are the benefits of using only the local maxima (and not the other extrema) in the OPP method?
10. The HGVG contains a lot of structure which seemingly can be deleted. Suggest a method to prune this structure.
11. What are the tradeoffs between using roadmaps and pixel-based maps?
12. Prove that for any slice direction, OPP is a subset of the GVG.
13. For the OPP and point-GVG, both in the plane and in $\mathbb{R}^3$, there are useless spokes. If we eliminate them in the planar case, do we still have a topological map? How could we eliminate spokes online?

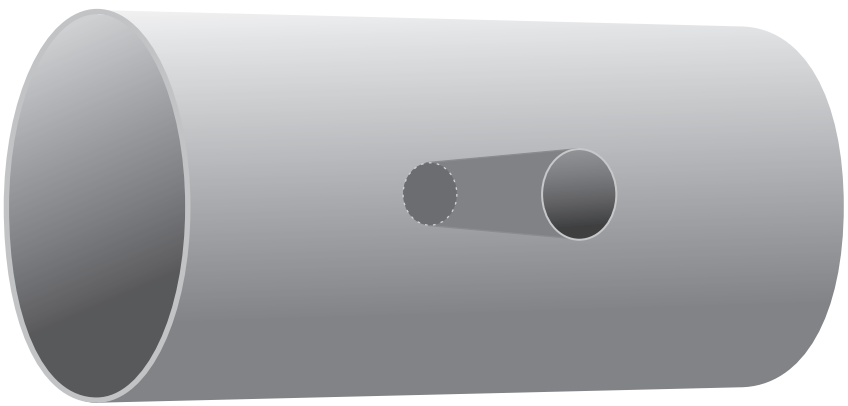

Figure 5.38 A cylinder with a hole drilled through it.

14. Use the brushfire implementation to compute the planar point-GVG. Beware of jagged edges.

15. The planar point-GVG is defined using a Euclidean distance function and consists of straight line and parabolic segments. One way of thinking of the planar point-GVG is the locus of the centers of circles whose perimeters are tangent to obstacles at two or more points. For the environment below, sketch the GVD using the circle analogy.

16. The definition of the planar point-GVG can be generalized to any convex distance function. Instead of a circle, consider a convex distance function defined by a square (rotated by 45 degrees). For the environment in figure 5.39, sketch the point planar-GVG using both the circle and the square analogy.

17. State at least two advantages and two disadvantages of using potential functions as a sensor-based planner.

18. Consider the real-valued function

    $f(x, y, z) = x^2 + y^2 - z^2.$

    Use the preimage theorem to state the values of c for which $f^{-1}(c)$ is a manifold. For the values of c for which $f^{-1}(c)$ is a manifold, state the dimension of $f^{-1}(c)$. State the values of c for which $f^{-1}(c)$ is connected. Draw pictures of the manifolds for different values of c.

19. Prove that d_i is a convex function when $\mathcal{QO}_i$ is convex.

20. Prove that the generalized Voronoi region is connected in a connected free space.

21. Verify that figure 5.6 contains the reduced visibility graph for figure 5.4.

22. Assume the boundaries of the two-equidistant faces are connected. Prove or disprove that the GVG is connected in $\mathbb{R}^3$.

23. Implement exploration of an unknown workspace using the incremental construction procedures described in this chapter

    (a) Rotate the robot so that the sensor with the smallest sensor reading is pointing "backward." You may use a lookup table here.
    (b) Drive the robot away from the closest obstacle until it is two-way equidistant.

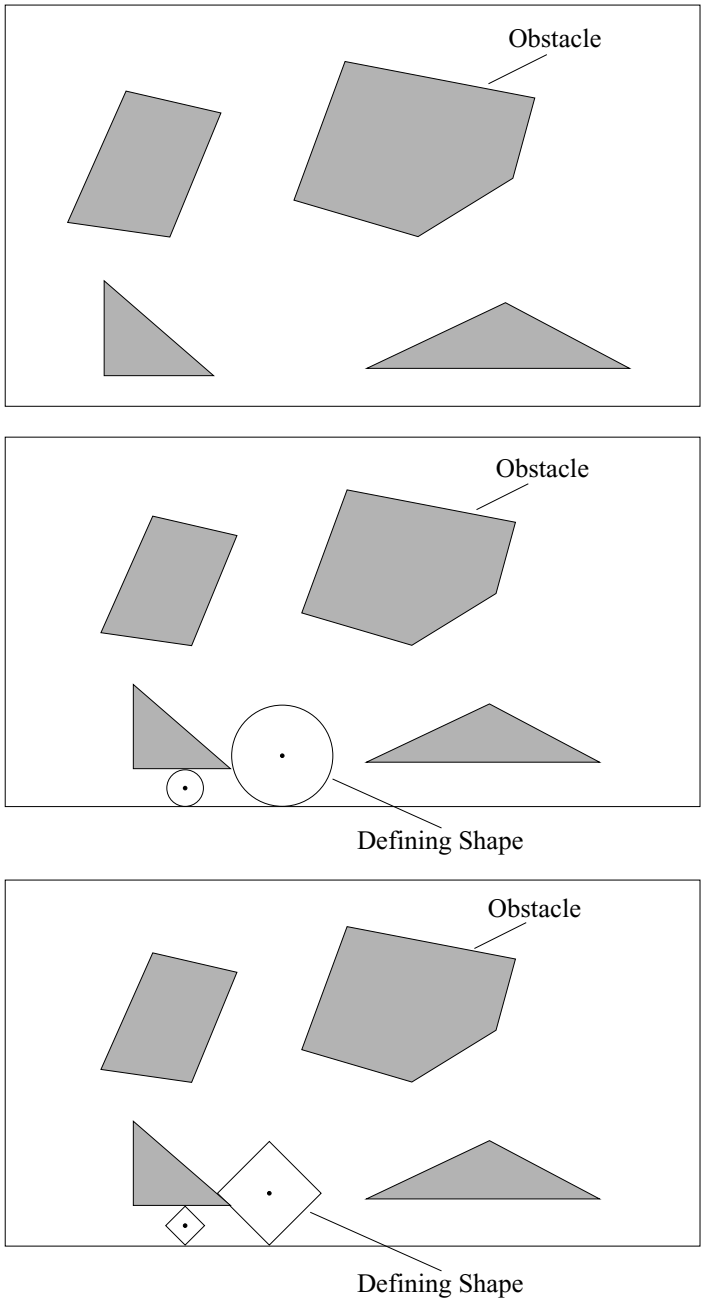

Figure 5.39 Photocopy the above figures to draw planar point-GVG's but with different distance metrics.

(c) Rotate the robot so that it lies in the tangent space of the GVD. You may use a lookup table here.
(d) Drive the robot forward a small distance and test to see if the robot still lies on the GVD (falls into a dead zone that is centered on the GVD).
(e) Rotate the robot by 90 degrees and drive it forward or backward until it is on the GVD and then reorient the robot back into the tangent space.
(f) Trace a GVD-edge until encountering a meet point.
(g) Depart a meet point on a GVD-edge.
(h) Implement the graph data structure for the GVD.

24. Use a local mapping routine to improve upon the exploration procedure described above.