Chapter 6

Path Planning Extensions

6.1 Optimal Paths

6.1.1 Visibility Roadmap

A visibility roadmap is obtained by generating all line segments between pairs of obstacle region vertices. Any line segment that lies entirely in the free space is added to the roadmap. Figure 6.1 shows an example. When a path planning is given, the initial position and goal position are also treated as vertices (in other words, they are connected to other vertices, if possible). This generates a connectivity graph that can be searched for a solution. The naive approach to constructing this graph takes time $O(n^3)$. The sweep-line principle can be applied to yield a more efficient algorithm.

There are two important notes about visibility roadmaps:

- The shortest-path solutions found in the roadmap will actually be the shortest-path solutions for the original problem. In addition, the paths "touch" the obstacle region. This is not acceptable in terms of the original problem, and the resulting paths should be modified.
- Many edges can be removed from the visibility roadmap, and optimal solutions will still be obtained. Any edge can be removed if the following property holds: extend the edge in both direction by a small amount. If either end "pokes" into the obstacle region, then the edge can be removed. Figure 6.1.b shows the edges that remain after this removal of performed.

6.2 Time-Varying Problems

This section addresses a time-varying version of path planning. Although the robot is allowed to move, it has been assumed so far that the obstacle region, $\mathcal{O}$, and the goal configuration, q_{goal} are stationary for all time. It is now assumed that these entities may vary over time, although their motions are predictable. The case in which unpredictable motions occur is covered in Chapter **??**.

Let T denote an interval of time, which is given by $T = [0, t_f]$ for some final time $t_f > 0$, or $T = [0, \infty)$.

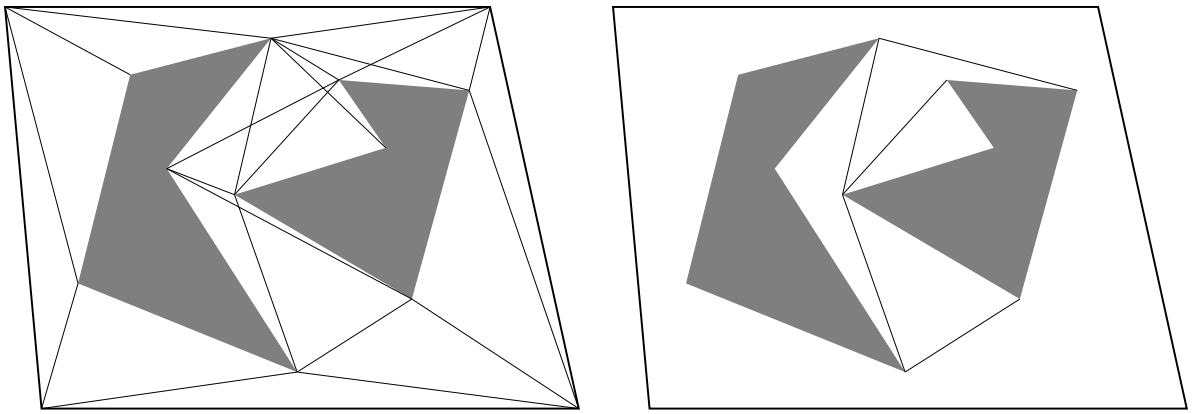
Let $q_{goal}(t)$ denote the goal configuration at time $t \in T$. The initial configuration may be denoted as $q_{init}(0)$.

Let $\mathcal{O}(t)$ denote the obstacle region at time $t \in T$. The model can defined using time-varying primitives. The obstacle region in the configuration space now varies with time, and is therefore written as $\mathcal{C}_{obs}(t)$. This can be defined as

$$\mathcal{C}_{obs}(t) = \{q \in \mathcal{C} \mid \mathcal{A}(q) \cap \mathcal{O}(t) \neq \emptyset\}$$

A time varying obstacle region appears significantly more complicated than a stationary obstacle region; however, it is possible to define a new space that removes the explicit dependency on the parameter, t.

Let a topological space, X be called the *state space*, defined as $X = \mathcal{C} \times T$. A state, $x \in X$, represents both a configuration and a time. Thus, x may be written as (q, t). An obstacle region can be described as a



(a) (b)

Figure 6.1: a) A visibility graph is constructed by joining obstacle region vertices; b) It is generally possible to reduce the number of edges in the visibility roadmap.

subset of X, given by

$$X_{obs} = \{x \in X \mid \mathcal{A}(q) \cap \mathcal{O}(t) \neq \emptyset\}.$$

If the primitives are expressed algebraically in terms of t, then it is possible to construct a semi-algebraic representation of X_{obs}. A given state, $x = (q, t)$, can be tested for inclusion in X_{obs} by running a collision detection algorithm for the configuration q and the obstacle region at time t. The collision-free subset of the state space can be written as $X_{free} = X \setminus X_{obs}$.

For a time-varying path planning problem the following components are defined:

1. A 2D or 3D world
2. A robot $\mathcal{A}$
3. A configuration space $\mathcal{C}$ (the set of all transformations for $\mathcal{A}$)
4. A time interval, T
5. The obstacle region $\mathcal{O}(t)$ for all $t \in T$
6. The initial state x_{init}
7. The goal region X_{goal}

It is assumed that x_{init} is of the form $x_{init} = (q_{init}, 0)$. A stationary goal configuration, q_{goal}, can be specified as

$$X_{goal} = \{(q, t) \in X \mid q = q_{goal}\}.$$

The task is to compute a continuous path, $\tau : [0, t] \rightarrow X_{free}$ such that $\tau(0) = x_{init}$ and $\tau(t) \in X_{goal}$.

Since X_{obs} and $\mathcal{C}_{obs}$ are conceptually similar, it is natural to ask whether standard path planning algorithms can be adapted easily to the time-varying case. There is one important distinction: a path in X must be monotonically increasing with respect to time.

Randomized path planning methods can generally be adapted by defining an appropriate metric. Suppose that $(\mathcal{C}, \rho)$ is a metric space. A suitable metric, ρ_x, can be define over X as

$$\rho_x(x, x') = \begin{cases} 0 & \text{If } q = q' \\ \infty & \text{If } q \neq q' \text{ and } t' \leq t \\ \rho(q, q') & \text{Otherwise} \end{cases}$$

As an example, to extend the randomized roadmap approach, the metric above would be used to select nearest neighbors to attempt possible connection. The resulting roadmap would be a directed graph in which edge progress through the state space by increasing time.

There has been no consideration so far of the speed at which the robot must move to avoid obstacles. It is obviously impractical in many applications if the solution requires the robot to move arbitrarily fast. One simple approach is to enforce a bound on the speed of the robot. For example, a constraint such as $\|dq/dt\| \leq c$ for some constant c can be enforced.

6.3 Multiple-Robot Coordination

Suppose that M robots, $\mathcal{A}^1$, $\mathcal{A}^2$, ..., $\mathcal{A}^M$, operate in a common world, $\mathcal{W}$. Each robot, $\mathcal{A}^i$, has its associated configuration space, $\mathcal{C}^i$, and its initial and goal configurations, q^i_{init} and q^i_{goal}.

A state space can be defined that considers the configurations of all of the robots simultaneously,

$$X = \mathcal{C}^1 \times \mathcal{C}^2 \times \cdots \times \mathcal{C}^M.$$

A state $x \in X$ specifies all robot configurations, and may be expressed as $x = (q^1, q^2, \ldots, q^M)$.

There are two kinds of obstacle regions in the state space: 1) robot-obstacle collisions, and 2) robot-robot collisions. For each i from 1 to M, the subset of X that corresponds to robot A^i in collision with the obstacle region is defined as

$$X^i_{obs} = \{x \in X \mid \mathcal{A}^i(q^i) \cap \mathcal{O} \neq \emptyset\}.$$

For each pair, $\mathcal{A}^i$ and $\mathcal{A}^j$, of robots, the subset of X that corresponds to $\mathcal{A}^i$ in collision with $\mathcal{A}^j$ is given by

$$X^{ij}_{obs} = \{x \in X \mid \mathcal{A}^i(q^i) \cap \mathcal{A}^j(q^j) \neq \emptyset\}.$$

Finally, the obstacle region in X is given by

$$X_{obs} = \left(\bigcup_{i=1}^{M} X^i_{obs} \right) \bigcup \left(\bigcup_{ij,\ i \neq j} X^{ij}_{obs} \right).$$

The task is to compute a continuous path, $\tau : [0,1] \rightarrow X_{free}$ such that $\tau(0) = x_{init}$ and $\tau(1) \in x_{goal}$, in which $x_{init} = (q^1_{init}, q^2_{init}, \ldots, q^M_{init})$ and $x_{goal} = (q^1_{goal}, q^2_{goal}, \ldots, q^M_{init})$.

It is straightforward to adapt general path planning methods from the standard configuration space, $\mathcal{C}$, to the state space X. One drawback of this approach is that the dimension of X can be high. For example, of there are 7 robots which each have 6 degrees of freedom, then the dimension of X is 42. A semi-algebraic representation of X_{obs} can be constructed, and complete path planning techniques can be applied; however, they are unlikely to be efficient. Randomized path planning methods offer a reasonable alternative is one is willing to sacrifice completeness.

Another option is to take a *decoupled* approach to the coordination of multiple robots. Rather then considering all interactions simultaneously, a decoupled approach plans for some robots independently, and then considers interactions.

Prioritized planning One decoupled approach, called *prioritized planning*, proceeds as follows:

- Compute a path and motion for $\mathcal{A}^1$.
- Treat $\mathcal{A}^1$ as a moving obstacle, and compute a path for $\mathcal{A}^2$.
- Treat $\mathcal{A}^1$ and $\mathcal{A}^2$ as moving obstacles, and compute a path for $\mathcal{A}^3$.
- *vdots*
- Treat $\mathcal{A}^1, \ldots, \mathcal{A}^{M-1}$ as moving obstacles, and compute a path for $\mathcal{A}^M$.

Even if the path planning method used for each robot is complete, the prioritized planning method is not complete because the motions generated in the earlier steps might prevent later robots from reaching their goals. One can try to improve the likelihood of success by trying different orderings, but there is still no guarantee of success.

Fixed-path coordination Another decoupled approach, called *fixed path coordination*, plans all paths independently, and then attempts to schedule the motions of the robots along their paths to avoid collisions.

Suppose $M = 2$. A collision-free path, $\tau_i : [0,1] \rightarrow \mathcal{C}^i_{free}$ is computed for each robot, $\mathcal{A}^i$ for $i = 1, 2$. Define a state space $X = [0,1] \times [0,1]$. Each state $x \in X$ is of the form (s_1, s_2) in which s_i represents a configuration $\tau_i(s_i)$ of $\mathcal{A}^i$. In other words, each s_i places $\mathcal{A}^i$ at a configuration given by its path τ_i. The task is to find a collision-free path in X from $(0,0)$ to $(1,1)$. A state $x \in X$ is in collision if $\mathcal{A}^1(\tau_1(s_1)) \cap A^2(\tau_2(s_2)) \neq \emptyset$.

6.4 Manipulation Planning

In all of the problems considered thus far, one primary objective has been to avoid collisions. In the case of manipulation planning, contacts between bodies are allowed. A manipulation task usually involves using the robot to move an object or part.

Let $\mathcal{P}$ denote a *part*, which is modeled in terms of geometric primitives, as described in Section **??**. It is assumed that $\mathcal{P}$ is allowed to undergo transformations, and will therefore have its own configuration space $\mathcal{C}^p$.

The robot, $\mathcal{A}$, will be referred to as a *manipulator* because it is allowed to interact with the part. Let $\mathcal{C}^m$ denote the configuration space of the manipulator. It will be convenient for manipulation planning to define a state space as $X = \mathcal{C}^m \times \mathcal{C}^p$, in which each state $x \in X$ is of the form $x = (q^m, q^p)$. As usual, states in which the manipulator is in collision with the obstacle region should be avoided, leading to

$$X^m_{obs} = \{(q^m, q^p) \in X \mid \mathcal{A}(q^m) \cap \mathcal{O}\}.$$

States in which the part is in collision with the obstacle region should also be avoided; however, it is usually necessary to allow contact between the part and obstacle region. Imagine, for example, placing a vase on a table; the vase and table are required to touch. These "touching" configurations can be defined as $\mathcal{C}^p_{contact} = \partial\mathcal{C}^p_{free}$ (the boundary of $\mathcal{C}_{free}$). The complete set of permitted configurations for the part is then given by $\mathcal{C}^p_{valid} = \mathcal{C}^p_{free} \cup \mathcal{C}^p_{contact}$. The set of forbidden configurations for the part is $\mathcal{C}^p_{obs} = \mathcal{C}^p \setminus \mathcal{C}^p_{valid}$. This results in another obstacle region in the state space, given by

$$X^p_{obs} = \{(q^m, q^p) \in X \mid q^p \in \mathcal{C}^p_{obs}\}$$

The sets X^m_{obs} and X^p_{obs} consider collisions with the obstacle region, and the remaining task is to consider interactions between the manipulator and the part. The manipulator might be allowed to push or carry the part. In general, the interactions could become quite complicated and even unpredictable. The obstacle region, X_{obs} can be formed in terms of X^m_{obs}, X^p_{obs}, and the set of states in which the manipulator and part are in an illegal configuration.

Since the general manipulation planning problem is quite complicated, a restricted version will be defined. Assume that the manipulator carries the part only if the part is grasped in a specified way. When it is being carried, the part can be considered as part of the manipulator. When the part is lying in its initial configuration, q^p_{init}, it can be considered as part of the obstacle region in the world.

Assume that the manipulation problem can be solved by picking up the part once and delivering it to its destination, q^p_{goal}. In this case, any solution path can be considered as the concatenation of two paths:

- **Transit Path:** The manipulator moves from its initial configuration, q^m_{init}, to any configuration that enables the part to be grasped.

- **Transfer Path:** The manipulator carries the part to its desired goal configuration.

A state space, $X = \mathcal{C}^m \times G$, can be defined for the restricted problem in which $G = \{0, 1\}$. States in which $g = 0$ for $g \in G$ indicate the *transit mode*, in which the part is in its initial configuration, and is treated as an obstacle. States in which $g = 1$ indicate the *transfer mode*, in which the part is being carried, and is treated as part of the manipulator. The manipulation planning task can now be formulated as computing a path, $\tau : [0,1] \rightarrow X_{free}$, in which

- $\tau(0)$ is the initial state x_{init}

- $\tau(1)$ is a state of the form (q^m, q^p_{goal}).
- X_{free} is the set of allowable states.

A variety of path planning algorithms can be adapted to solve this problem. For example, the randomized roadmap or rapidly-exploring random trees can be easily adapted.

Bibliography