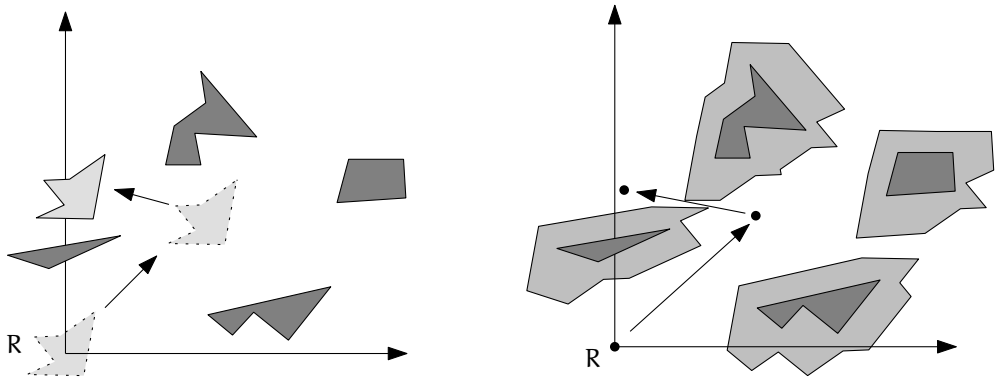
19. Translational Motion Planning

Lecture on Thursday 3rd December, 2009 by Michael Hoffmann <hoffmann@inf.ethz.ch>

In a basic instance of motion planning, a robot—modeled as a simple polygon R—moves by translation amidst a set $\mathcal{P}$ of polygonal obstacles. Throughout its motion the robot must not penetrate any of the obstacles but touching them is allowed. In other words, the interior of R must be disjoint from the obstacles at any time. Formulated in this way, we are looking at the motion planning problem in *working space* (Figure 19.1(a)).

However, often it is useful to look at the problem from a different angle, in so-called *configuration space*. Starting point is the observation that the placement of R is fully determined by a vector $\vec{v} = (x, y) \in \mathbb{R}^2$ specifying the translation of R with respect to the origin. Hence, by fixing a *reference point* in R and considering it the origin, the robot becomes a point in configuration space (Figure 19.1(b)). The advantage of this point of view is that it is much easier to think of a point moving around as compared to a simple polygon moving around.



(a) Working Space. (b) Configuration Space.

Figure 19.1: *Working space and configuration space for a robot* R *and a collection of polygonal obstacles.*

The next question is: How do obstacles look like in configuration space? For an obstacle $P \in \mathcal{P}$ the set $\mathcal{C}(P) = \{\vec{v} \in \mathbb{R}^2 \mid R + \vec{v} \cap P \neq \emptyset\}$ in configuration space corresponds to the obstacle P in the original setting. We write $R + \vec{v}$ for the *Minkowski sum* $\{\vec{r} + \vec{v} \mid \vec{r} \in R\}$. Our interest is focused on the set $\mathcal{F} = \mathbb{R}^2 \setminus \bigcup_{P \in \mathcal{P}} \mathcal{C}(P)$ of *free placements* in which the robot does not intersect any obstacle.

Proposition 19.1 $\mathcal{C}(P) = P - R$.

Proof. $\vec{v} \in P - R \iff \vec{v} = \vec{p} - \vec{r}$, for some $\vec{p} \in P$ and $\vec{r} \in R$. On the other hand, $R + \vec{v} \cap P \neq \emptyset \iff \vec{r} + \vec{v} = \vec{p}$, for some $\vec{p} \in P$ and $\vec{r} \in R$. □

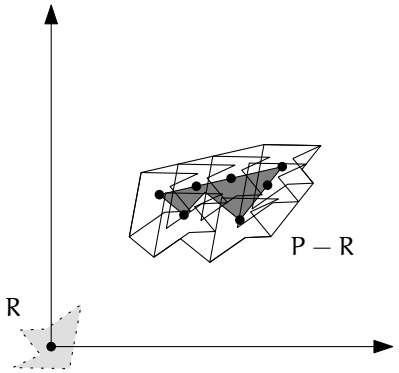


Figure 19.2: *The Minkowski sum of an obstacle with an inverted robot.*

Observe that each edge of a configuration space obstacle $\mathcal{C}(P)$ corresponds to the region of contact of an edge or vertex of P with an edge or vertex of R. It follows that $\mathcal{F}$ is bounded by at most $O(nm)$ line segments, if R has m edges and the objects in $\mathcal{P}$ have n edges in total.

19.1 Constructing a single face

Theorem 19.2 *Given a set S of n line segments and a point $x \in \mathbb{R}^2$, the face of $\mathcal{A}(S)$ that contains x can be constructed in $O(\lambda_3(n) \log n)$ expected time.*

Phrased in terms of translational motion planning this means the following.

Corollary 19.3 *Consider a simple polygon R with k edges (robot) and a polygonal environment $\mathcal{P}$ that consists of n edges in total. The free space of all positions of R that can be reached by translating it without properly intersecting an obstacle from $\mathcal{P}$ has complexity $O(\lambda_3(kn))$ and it can be constructed in $O(\lambda_3(kn) \log(kn))$ expected time.*

Below we sketch[1] a proof of Theorem 19.2 using a randomized incremental construction, by constructing the trapezoidal map induced by the given set S of segments. As before, suppose without loss of generality that no two points (intersection points or endpoints) have the same x-coordinate.

In contrast to the algorithm you know, here we want to construct a single cell only, the cell that contains x. Whenever a segment closes a face, splitting it into two, we discard one of the two resulting faces and keep only the one that contains x. To detect whether a face is closed, use a disjoint-set (union-find) data structure on S. Initially, all segments are in separate components. The runtime needed for the disjoint-set data structure is $O(n\alpha(n))$, which is not a dominating factor in the bound we are heading for.

[1] For more details refer to the book of Agarwal and Sharir [1].

Insert the segments of S in order $s_1, \ldots, s_n$, chosen uniformly at random. Maintain (as a doubly connected edge list) the trapezoidal decomposition of the face f_i, $1 \leq i \leq n$, of the arrangement $\mathcal{A}_i$ of $\{s_1, \ldots, s_i\}$ that contains x.

As a third data structure, maintain a *history dag* (directed acyclic graph) on all trapezoids that appeared at some point during the construction. For each trapezoid, store the (at most four) segments that define it. The root of this dag corresponds to the entire plane and has no segments associated to it.

Those trapezoids that are part of the current face f_i appear as *active* leaves in the history dag. There are two more categories of vertices: Either the trapezoid was destroyed at some step by a segment crossing it; in this case, it is an interior vertex of the history dag and stores links to the (at most four) new trapezoids that replaced it. Or the trapezoid was cut off at some step by a segment that did not cross it but excluded it from the face containing x; these vertices are called *inactive* leaves and they will remain so for the rest of the construction.

Insertion of a segment s_{r+1} comprises the following steps.

1. Find the cells of the trapezoidal map f_r^* of f_r that s_{r+1} intersects by propagating s_{r+1} down the history dag.

2. Trace s_{r+1} through the active cells found in Step 1. For each split, store the new trapezoids with the old one that is replaced.

   Wherever in a split s_{r+1} connects two segments s_j and s_k, join the components of s_j and s_k in the union find data structure. If they were in the same component already, then f_r is split into two faces. Determine which trapezoids are cut off from f_{r+1} at this point by alternately exploring both components using the DCEL structure. (Start two depth-first searches one each from the two local trapezoids incident to s_{r+1}. Proceed in both searches alternately until one is finished. Mark all trapezoids as discarded that are in the component that does not contain x.) In this way, the time spent for the exploration is proportional to the number of trapezoids discarded and every trapezoid can be discarded at most once.

3. Update the history dag using the information stored during splits. This is done only after all splits have been processed in order to avoid updating trapezoids that are discarded in this step.

The analysis is completely analogous to the case where the whole arrangement is constructed, except for the expected number of trapezoids created during the algorithm. Recall that any potential trapezoid τ is defined by at most four segments from S. Denote by t_r the expected number of trapezoids created by the algorithm after insertion of $s_1, \ldots, s_r$. Then in order for τ to be created at a certain step of the algorithm, one of these defining segments has to be inserted last. Therefore,

$$\Pr[\tau \text{ is created by inserting } s_r] \leq \frac{4}{r} \Pr[\tau \text{ appears in } f_r^*]$$

and

$$\begin{aligned}
t_r &= \sum_{\tau} \Pr[\tau \text{ is created in one of the first } r \text{ steps}] \\
&\leq \sum_{\tau} \sum_{i=1}^{r} \frac{4}{i} \Pr[\tau \text{ appears in } f_i^*] \\
&= \sum_{i=1}^{r} \frac{4}{i} \sum_{\tau} \Pr[\tau \text{ appears in } f_i^*] \\
&\overset{\text{Theorem 18.6}}{\leq} \sum_{i=1}^{r} \frac{4}{i} O(\lambda_3(i)) \\
&\leq \sum_{i=1}^{r} \frac{4}{i} ci\alpha(i) = 4c \sum_{i=1}^{r} \alpha(i) \leq 4cr\alpha(r) = O(\lambda_3(r)) \; .
\end{aligned}$$

Using the notation of the configuration space framework, the expected number of conflicts is bounded from above by

$$\begin{aligned}
\sum_{r=1}^{n-1} (k_1 - k_2 + k_3) &\leq 16(n-1) + 12n \sum_{r=1}^{n-1} \frac{\lambda_3(r+1)}{r(r+1)} \\
&\leq 16(n-1) + 12 \sum_{r=1}^{n-1} \frac{n}{r+1} \lambda_3(r+1) \frac{1}{r} \\
&\leq 16(n-1) + 12 \sum_{r=1}^{n-1} \frac{\lambda_3(n)}{r} \\
&= 16(n-1) + 12\lambda_3(n) H_{n-1} \\
&= O(\lambda_3(n) \log n) \; .
\end{aligned}$$

Questions

50. *What is the configuration space model for (translational) motion planning, and what does it have to do with arrangements (of line segments)?* Explain the working space/configuration space duality and how to model obstacles in configuration space.

51. *Can one construct a single face of an arrangement (of line segments) more efficiently compared to constructing the whole arrangement?* Explain the statement of Theorem 19.2 and give a rough sketch of the proof.

References

[1] Pankaj K. Agarwal and Micha Sharir, *Davenport-Schinzel sequences and their geometric applications*, Cambridge University Press, New York, NY, 1995.