CMSC 754: Spring 2026 Dave Mount

Solutions to Homework 9

Solution 1:

(a) Given a rectangle O_i of height h_i and length ℓ_i, the C-obstacle shape is shown in Fig. 1(a). It can be computed by expressing the robot $\mathcal{R}$ as the union of two rectangles $\mathcal{R}_1 \cup \mathcal{R}_2$, both having the origin as their reference point. Rectangle $\mathcal{R}_1$ is of height h and width w, and rectangle $\mathcal{R}_2$ is of height w and width ℓ. The C-obstacle is the union of the two C-obstacles, $(O_i \oplus (-\mathcal{R}_1)) \cup (O_i \oplus (-\mathcal{R}_2))$.

Letting $q_i = (p_{i,x} + \ell_i, p_{i,y} + h_i)$ denote the upper right corner of the obstacle, both rectangles share this as their upper right corner. The first rectangle has height $h_i + w$ and width $\ell + \ell_i$. The other has height $h + h_i$ and width $\ell_i + w$.

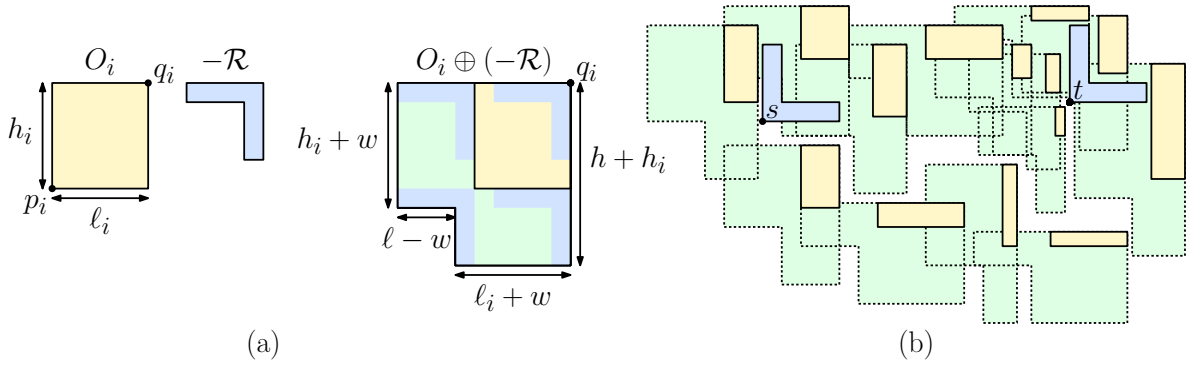


Figure 1: L-shaped robot motion planning.

(b) Here is a high-level sketch. Compute the C-obstacles for each of the rectangles of $\{O_1, \ldots, O_n\}$. Using plane sweep, compute their union in $O(m \log m)$ time, where m is the total combinatorial complexity of the union. (In part (c), we will show that $m = O(n^2)$.) The free space is the compliment of the union of the C-obstacles.

Next, compute a trapezoidal map of the entire configuration, and label cells that lie in the free space versus those that lie within C-obstacles. Construct a dual graph whose nodes consist of just the free-space cells, where two nodes are connected by an edge if their associated trapezoids share a vertical wall. This can be done in $O(m \log m)$ time.

Given the start point s and end point t, first apply point location to determine the cells containing them. (If either cell does not lie in a free-space cell, then report that the initial configuration is invalid.) Let u_s and u_t denote these nodes the dual graph. Determine whether there is path between them (e.g., by BFS) in $O(m)$ time. If so, plan a path in the configuration space along this path (see Fig. 1).

(c) To generate a worst-case example, make the robot's arms extremely long an skinny, and make the obstacles tiny squares. Consider the sort of arrangement shown in Fig. 2. It should be clear that this can be generalized to arbitrarily large numbers of obstacles n, resulting in a boundary complexity of $O(n^2)$.

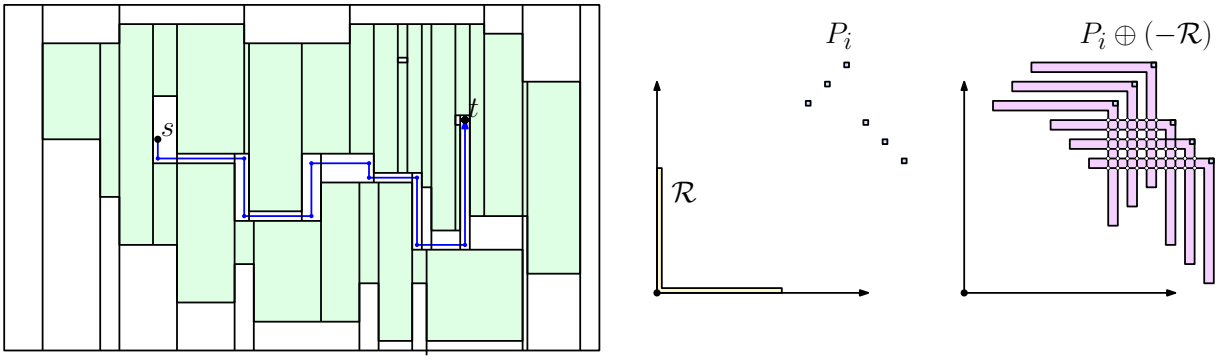


Figure 2: Union of C-obstacles.

Solution 2:

(a) Let $n_\circ$ denote the numbers of vertices of the original polygon that lie within the union interior. Define an *intersection vertex* to be a vertex of the union boundary that lies at the intersection of the interiors of two edges of the original polygons. Let $n_\times$ denote the number of intersection vertices, and let N denote the total number of vertices on the union boundary. We have:

$$n \;=\; m + n_\circ \qquad \text{and} \qquad N \;=\; m + n_\times.$$

We showed in the counting argument from class that each intersection vertex could be charged to an interior vertex, so that each interior vertex receives at most two charges. Therefore, $n_\times \leq 2n_\circ$. This implies that

$$N \;=\; m + n_\times \;\leq\; m + 2n_\circ \;=\; 2(m + n_\circ) - m \;=\; 2n - m,$$

as desired.

(b) Such an example is shown in Fig. 3. It consists of two large triangles and $k \geq 1$ small triangles. The total number of original vertices is $n = 3(k + 2) = 3k + 6$, which implies that $k = (n - 6)/3$.

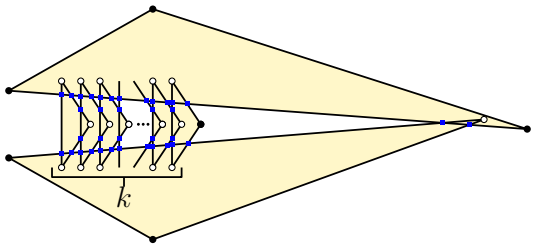


Figure 3: Lower bound on the complexity of the union of a collection of pseudodisks.

The only original polygon vertices appearing in the union boundary are the five vertices of the two large triangles and the central vertex of the rightmost small triangle, for a total of

six. All the other original vertices lie in the interior of the union. The number edge-edge vertices on the union boundary is two for the two big triangles (on the right side of the figure), and each of the k small triangles contributes six vertices except the rightmost one, which contributes only four. Therefore, the total number of edge-edge vertices of the union is $2+6(k-1)+4 = 6k = 2n-12$. Summing up the number of original vertices and the number of edge-edge vertices on the union boundary, we have $6 + (2n-12) = 2n-6$, as desired.

Solution 3:

(a) In this problem we make use of the easily verified fact that Minkowski addition is both *commutative*, that is, $A \oplus B = B \oplus A$ and *associative*, that is, $A \oplus (B \oplus C) = (A \oplus B) \oplus C$. This follows immediately from the definition of Minkowski addition and the facts that vector addition is both commutative and associative.

Consider convex polygons R, R_1, and R_2, such that $R_1 \oplus R_2 = R$ and suppose that R is a common summand of P and Q. This means that there exist convex polygons P' and Q' such that $P = P' \oplus R$ and $Q = Q' \oplus R$. Define $P'' = P' \oplus R_2$ and $Q'' = Q' \oplus R_2$. By the commutativity and associativity of Minkowski addition, we have

$$P \;=\; P' \oplus R \;=\; P' \oplus (R_1 \oplus R_2) \;=\; P' \oplus (R_2 \oplus R_1) \;=\; (P' \oplus R_2) \oplus R_1 \;=\; P'' \oplus R_1.$$

Applying a symmetrical argument to Q, we have $Q = Q'' \oplus R_1$. Therefore, R_1 is a common summand of P and Q. By applying the same argument with R_1 and R_2 swapped, it follows that R_2 is also a common summand of P and Q.

(b) Let $\widehat{R} = \alpha R \oplus (1-\alpha)R$, for some α, where $0 < \alpha < 1$. We will show that $\widehat{R} \subseteq R$ and $R \subseteq \widehat{R}$, from which it will follow that $\widehat{R} = R$. Recall that $A \oplus B = \{a+b \,:\, a \in A, b \in B\}$. Replacing A and B with αR and $(1-\alpha)R$, we have

$$\widehat{R} \;=\; \{a+b \,:\, a \in \alpha R,\, b \in (1-\alpha)R\}.$$

For any $r \in R$, setting $a = \alpha r$ and $b = (1-\alpha)r$, we have $r = a+b$, and hence it follows that $r \in \alpha R \oplus (1-\alpha)R = \widehat{R}$, which implies that $R \subseteq R$.

Conversely, for any $r \in \widehat{R}$, we have $r = a+b$, where $a = \alpha r'$ for some $r' \in R$ and $b = (1-\alpha)r''$ for some $r'' \in R$. Therefore, $r = \alpha r' + (1-\alpha)r''$ for $0 < \alpha < 1$, implying that r lies on the line segment between r' and r''. By convexity, all such points lie within R, which implies that $\widehat{R} \subseteq R$, which completes the proof.

(c) If two convex polygons P and R are in general position (implying that they have no parallel edges), then the edge set of $P \oplus R$ is just the union of the edge sets of each of the two polygons, merged tail to head in cyclic order. It follows that if R is a common summand of P and Q, and these polygons are in general position, then the edges of R are just the common edges of P and Q (see Fig. 4(a)). (If P and Q are not in general position, then the edges of R will be drawn from parallel edges from P and Q, but the lengths of the edges of R is not as obvious. See the solution to the Challenge Problem.)

Let us direct the edges of P and Q in counterclockwise order, and think of each edge as a vector. The algorithm merges these vectors in angular order. Whenever a vector from P and

a vector from Q have the same orientations, we check that their lengths are equal. If not, we report that P and Q have no common summand. Otherwise, we take the sum of all the common vectors. If they do not sum to the zero vector, then we again report that P and Q have no common summand. Otherwise, we take R to be the convex polygon formed by concatenating these vectors in a tail to head manner (see Fig. 4(b)). The algorithm clearly takes $O(n+m)$ time, where n and m are the number of sides of the two polygons.

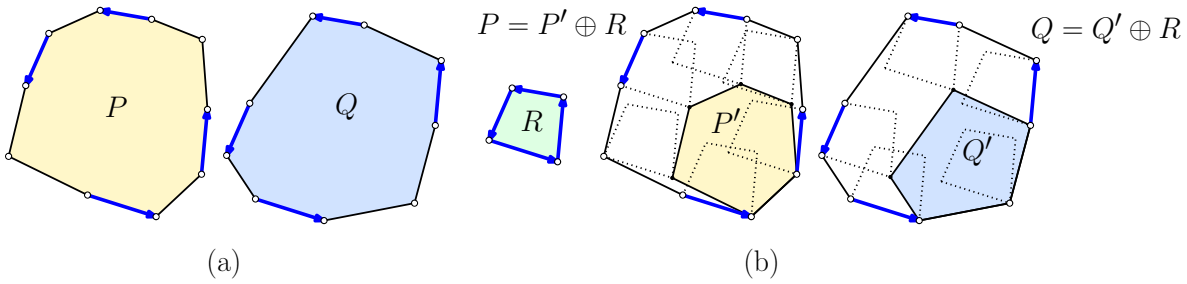


Figure 4: Maximal common summands in the general-position case.

Solution 4:

(a) The C-obstacle $\mathcal{C}_{\mathcal{R}}(P_i)$ in the horizontal case is the Minkowski sum of a horizontal segment of length 2 centered at the origin and the rectangle $[x_i^-, x_i^+] \times [y_i^-, y_i^+]$. This just involves widening the rectangle by one unit on each side, that is, $\mathcal{C}_{\mathcal{R}}(P_i) = [x_i^- - 1, x_i^+ + 1] \times [y_i^-, y_i^+]$ (see Fig. 5(a)).

Similarly, in the vertical case, we expand vertically, yielding $\mathcal{C}_{\mathcal{R}}(P_i) = [x_i^-, x_i^+] \times [y_i^- - 1, y_i^+ + 1]$ (see Fig. 5(b)).

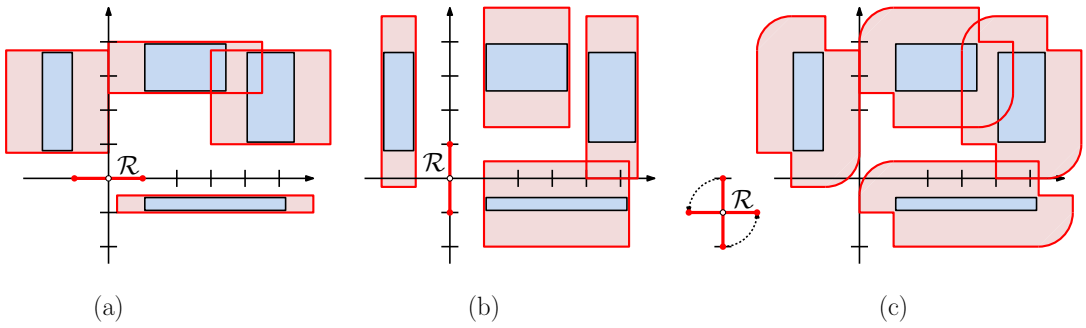
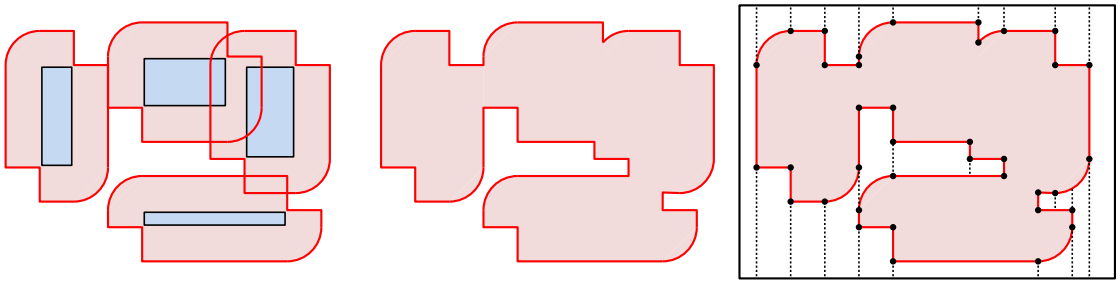


Figure 5: C-Obstacles.

(b) The C-obstacle for the counterclockwise rotation is a nonconvex shape (see Fig. 5(c)). The NW and SW corners are rounded and the NE and SW corners are "notched". The C-obstacle for the clockwise rotation is symmetrical.

(c) We test reachability as follows. There are four different motion types: H/V: horizontal/vertical translation and CW/CCW: clockwise/counterclockwise rotations. First, for each of the motion type, we construct the C-obstacles for the set of rectangles (see Fig. 6(a)). For each

motion type, we then construct the union of the C-obstacles (e.g., by applying plane sweep, see Fig. 6(b)). Note that this is done independently for each motion type, so we have four separate unions. Finally, for each motion type, we compute a decomposition of the compliment of the union (e.g., using a trapezoidal map, see Fig. 6(c)). Let's call this a *motion map* for the associated motion type.



(a) (b) (c)

Figure 6: Building a motion map.

We now have four distinct motion maps, one for each motion type. Let's imagine that these maps reside on four separate planes. We then construct a graph connecting the trapezoids of these maps as follows. First, if two trapezoids are neighbors in the same map, sharing a common vertical wall, we connect them. Next, if two trapezoids from the following different pairs of motion types overlap spatially, we connect them:

$$\text{H} \longleftrightarrow \text{CW}, \quad \text{V} \longleftrightarrow \text{CW}, \quad \text{H} \longleftrightarrow \text{CCW}, \quad \text{V} \longleftrightarrow \text{CCW}.$$

Intuitively, this is because the robot can transition from horizontal/vertical translational motion to rotational motion (but it cannot transition directly between horizontal and vertical translational motion). These connections can be computed in polynomial time by overlaying pairs of maps and sweeping over them simultaneously. (Let's not worry about the messy details.)

What is the size of the graph? Well, it is certainly polynomial in the number of obstacles n. We cannot directly apply results in class about pseudodisks, because the rotational C-obstacles are not convex. Given n rectangular obstacles, the total boundary complexity of the C-obstacles is $O(n)$, and hence the complexity of their union is $O(n^2)$. The complexity of the overlay of any pair is at most $O(n^4)$. (I conjecture that a more refined analysis will reveal that the total complexity is just $O(n)$.)

Once this graph has been constructed, we compute its connected components. Now, given this graph, reachability queries can be answered very efficiently. Given a starting configuration $s = (x_s, y_s, o_s)$ and a target $t = (x_t, y_t, o_t)$, we locate the associated nodes in the graph based on the location (x, y) and the orientation (horizontal translation if the orientation is horizontal and vertical translation if the orientation is vertical). This can be done efficiently through point location in $O(\log n)$ time. We then simply check whether s and t are in the same connected component, which can be done in constant time.

Solution to the Challenge Problem: Given P and Q, direct the edges of both in counterclockwise order, merge them and identify those that are parallel. Suppose that there are k parallel edges. Let $\{u_1, \ldots, u_k\}$ denote the unit vectors in the directions of these edges. Let a_i and b_i denote the lengths of the edges associated with u_i for P and Q, respectively, and let $c_i = \min(a_i, b_i)$. This information can be extracted in $O(n+m)$ time, where n and m are the number of sides of the two polygons.

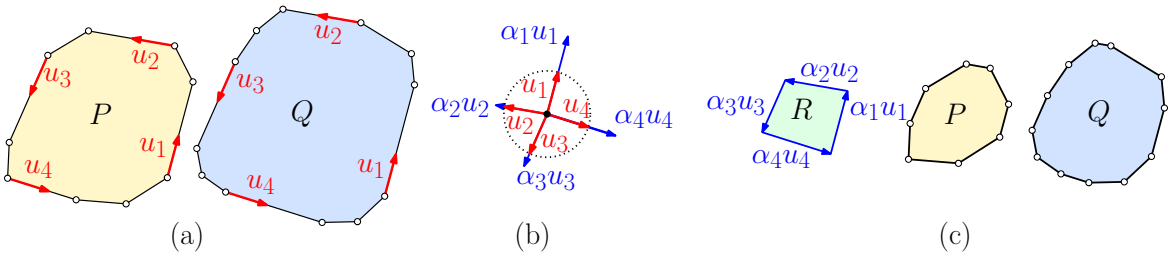


Figure 7: Maximal common summands in the general case.

Any common summand consists of directed edges oriented parallel to the u_i's and having lengths ranging from 0 to c_i. Letting α_i denote the length of the ith directed edge, it follows that $0 \leq \alpha_i \leq c_i$, and (in order to form a polygon) $\sum_{i=1}^{k} \alpha_i u_i = 0$. Together, these constraints define a linear programming problem in dimension k involving $2k + 2 = O(k)$ constraints. In order to obtain a maximal solution, we set the objective function of maximizing $\sum_{i=1}^{k} \alpha_i$.

We can then apply the randomized LP algorithm from class. (Given the 2-dimensional nature of the problem, I suspect that there are more efficient ways to solve this LP.) Note that there is always a feasible, bounded solution. It is feasible since setting $\alpha_i = 0$ satisfies all the constraints. It is bounded, due to the constraints $\alpha_i \leq c_i$.

We assert that the LP optimum solution yields a maximal common summand. This is because any augmentation to the solution would result in longer edges (a consequence of Problem 3(a)). Increasing the edge lengths would allow us to increase the value of the objective function, violating the hypothesis that this is a maximal solution to the LP. By the analysis given in class, the expected time to solve the LP is $O(k!k) = O(k^{k+1})$ time. Thus, if k is a constant, the overall running time is dominated by the $O(n+m)$ time needed to set up the LP. (As mentioned above, I suspect that there is a much more efficient solution.)