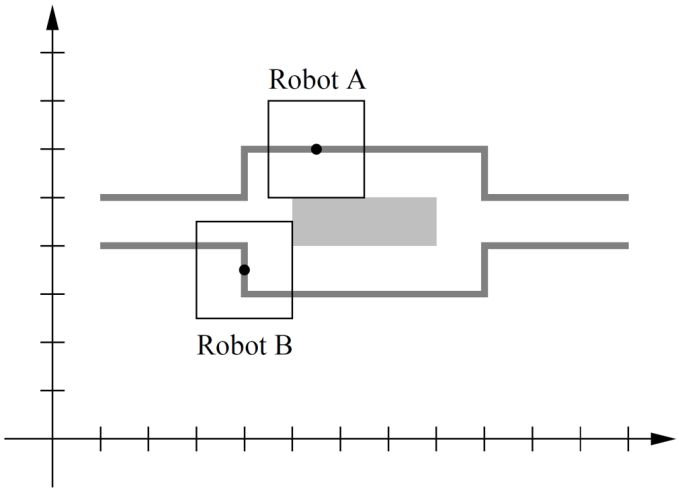
CS 4610/5335: Robotic Science and Systems (Spring 2022) — Lawson L.S. Wong
Northeastern University — Due February 25, 2022

Exercise 2: Motion Planning

Please remember the following policies:

- Exercises are due at **11:59 PM** Boston time (ET).
- **Submissions should be made electronically on Canvas, as a single .pdf file for written parts, and as a single .zip file for code.** Check your submission and make sure you have included all files! You can make as many submissions as you wish, but only the latest one will be considered, and late days will be computed based on the latest submission.
- Each exercise may be handed in up to two days late (24-hour period), penalized by 5% per day. Submissions later than this will not be accepted. There is no limit on the total number of late days used over the course of the semester.
- Written solutions may be handwritten or typeset. For the former, please ensure handwriting is legible. If you write your answers on paper and submit images of them, that is fine, but please put and order them correctly in a single .pdf file. One way to do this is putting them in a Word document and saving as a PDF file.
- Some questions are intended for CS 5335 students only. CS 4610 students may complete these for extra credit.
- You are welcome to discuss these problems with other students in the class, but you must understand and write up the solution yourself, *and* indicate who you discussed with (if any).
- Contact the teaching staff if there are *extenuating* circumstances.

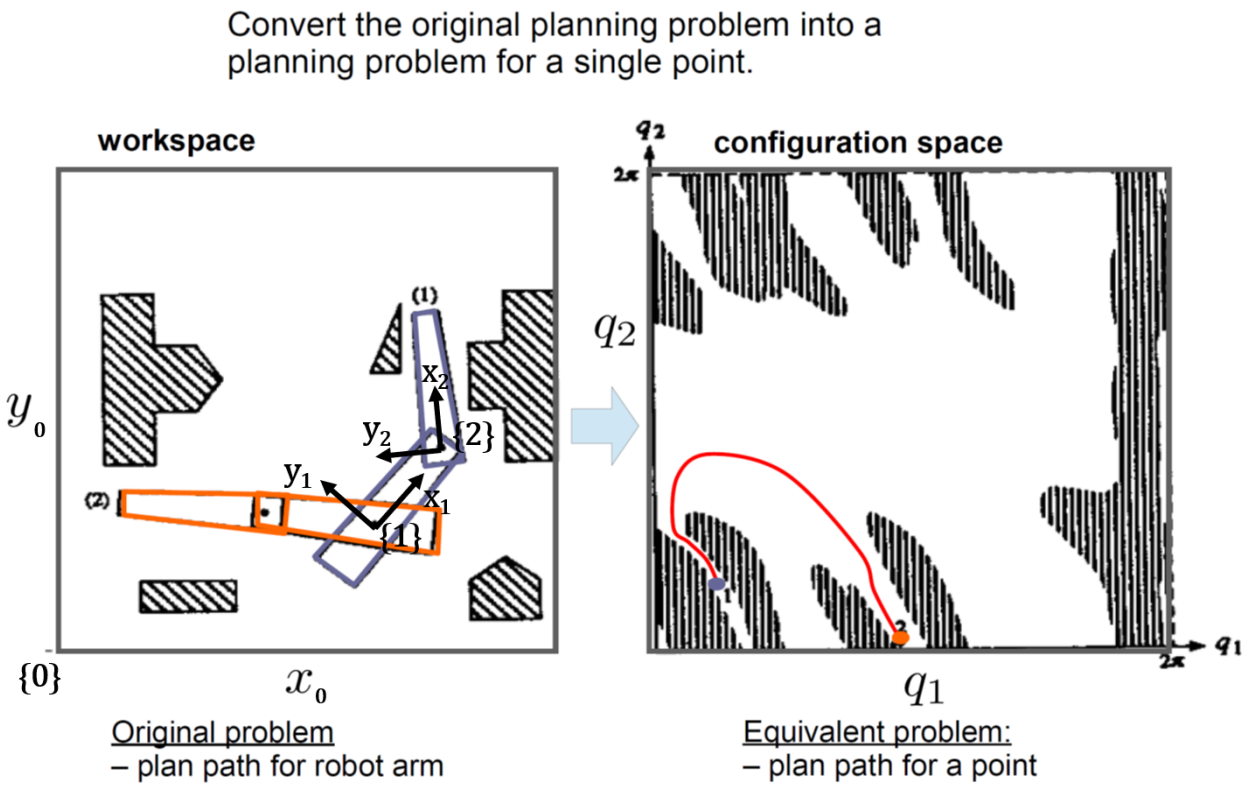
Configuration Space and Motion Planning



C0. **2 points.** Consider the diagram above. Two square robots A and B operate in a 2-D workspace. The two robots do not rotate, and each moves on a fixed track, so that its center remains on the solid gray line shown in the figure. The robots must move so as not to collide with each other. The diagram is to scale, with each tick denoting a distance of one unit.

(a) Even though there are two robots both moving in a 2-D workspace, we can still use a 2-D configuration space to represent the above system. Specify what the axes of our configuration space correspond to in the workspace, what the axis limits are, and what configuration the above diagram depicts.

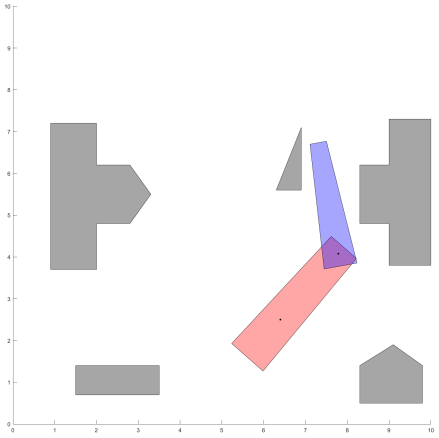
(b) Draw the configuration space. For each configuration-space obstacle, label the coordinates of the vertices, and sketch the workspace configuration corresponding to each. Since the workspace is symmetric, only makes sketches for the left side of the workspace.



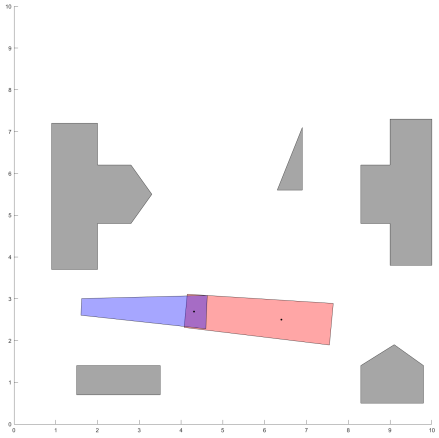
In the remainder of this section, we will revisit the 2-DOF 2-link rotational planar robot shown above that we considered in lecture. The arm is attached to a table surface that we are viewing from a top-down perspective. There are obstacles on the table as shown by the shaded regions. The goal is move the arm from start configuration (1, purple) to the goal configuration (2, orange), without colliding into any obstacles. Assume that the illustrated workspace is 10 units wide and 10 units high; the origin of frame $\{0\}$ is at the bottom-left corner, with x_0 and y_0 axes as shown. The first link of the arm is attached to the table at $(6.4, 2.5)$, the origin of frame $\{1\}$. Link 2 is attached to link 1 at $(2.1, 0)$ with respect to frame $\{1\}$, the origin of frame $\{2\}$. The configuration of the arm is given by $q = (q_1, q_2)$, where q_1 is the angle between x_0 and x_1, and q_2 is the angle between x_1 and x_2. Both joints may rotate between 0 and 2π radians. For example, the start configuration (1, purple) is $q_{\text{start}} = (0.85, 0.9)$, and the goal configuration (2, orange) is $q_{\text{goal}} = (3.05, 0.05)$.

Download the starter code (`ex2.zip`). In this file, you will find:

- `ex2_cspace.m`: Main function. Call `ex2_cspace(<questionNum>)` with an appropriate question number (1–7) to run the code for that question. Do not modify this file! (Feel free to actually make changes, but check that your code runs with an unmodified copy of `ex2_cspace.m`.)
- `C1.m` – `C7.m`: Code stubs that you will have to fill in for the respective questions.
- `q2poly.m`: Code stub that you will have to fill in, helper function for various questions.
- `plot_obstacles.m`: Helper function to draw workspace obstacles defined in `ex2_cspace.m`.



C1: Start configuration.



C1: Goal configuration.

C1. **2 points.** Plot the robot in the workspace, as shown above. The demonstration code in `C1.m` shows how to plot the robot at the zero configuration ($q = (0, 0)$). You will need to make appropriate transformations to both links' polygons and their pivot points (frame origins). Consider filling in `q2poly.m` first, which is a useful helper function for C1 and future questions. If you provide q_{start} and q_{goal} as input, you should get the figures above.
Useful functions: Read the documentation for `polyshape`, a MATLAB class for defining 2-D polygons.

C2. **2 points.** Convert the problem into configuration space (right diagram on slide shown above) by discretizing the configuration space, and checking for collisions at each discrete grid point. Using the specified grid for each axis given in `q_grid`, compute whether the configuration at each point is in collision or not, by intersecting the links' 2-D polygon with the obstacles' 2-D polygons. Assume that the robot is never in collision with itself. The resulting matrix should look similar to the configuration space diagram shown on the slide.
Useful functions: `intersect`

Hint: Future questions rely on the output of `C2.m`, which may take a while to compute. To avoid re-computing it in future questions, we have provided functionality to save it in your MATLAB workspace, and pass it in on future calls:

```
cspace = ex2_cspace(2);
ex2_cspace(3, cspace);
```

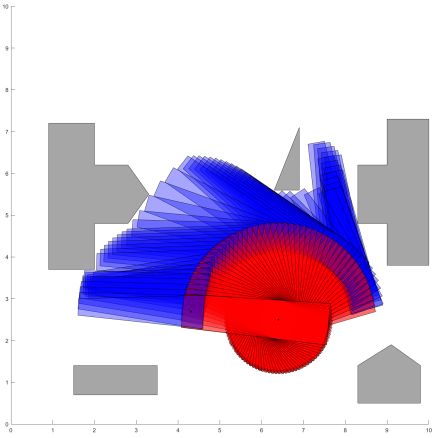
C3: Distance transform from goal configuration.

C4: Path from start to goal.

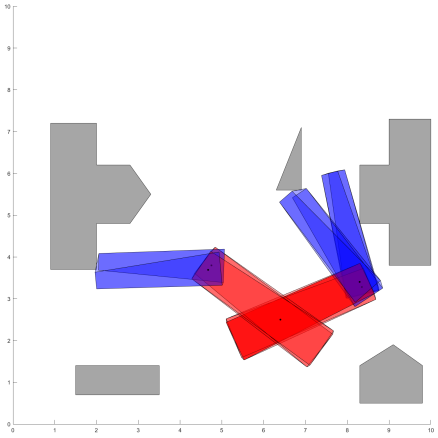
C3. **2 points.** Given a specified goal configuration and the configuration-space grid from C2, compute the distance transform from the grid point nearest to the goal.

C4. **2 points.** Using the distance transform from C3, find a path from the specified start configuration's closest grid point to the goal's grid point. Descend the distance transform in a greedy fashion, breaking ties with any strategy you wish. Diagonal neighbors are allowed.

C5. **1 point.** Convert the path in grid point indices, found in C4, into a path in configurations. Remember to include the actual start and goal configurations. This should trigger a visualization similar to the one shown on the next page.



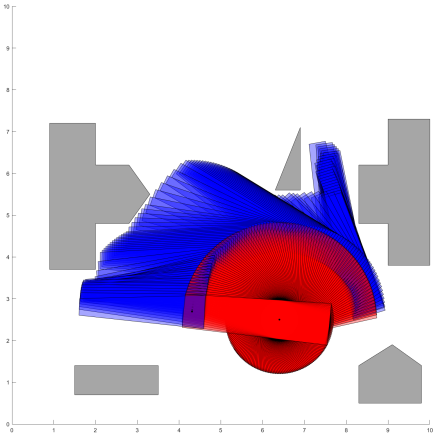
C5: Trajectory from start to goal.



C6: Swept-volume collisions along the path.

C6. **[CS 5335 only.] 2 points.** Unfortunately, since collisions have only been checked at discrete grid points, we cannot guarantee that the segments between those grid points are collision-free. In fact, the trajectory we found in our implementation of C5 contains three collisions, shown in the right above. These collisions can be detected by considering the *swept volume* between two configurations. The swept volume can be approximated by appropriate convex hulls of the robot links' 2-D polygons. Check if any segments of the trajectory you found in C5 are in collision, plot the violating swept volumes (similar to right diagram above), and return the number of collisions. Depending on how you found your trajectory, it may not actually have any such swept-volume collisions!
Useful functions: `convhull`

C7. **[CS 5335 only.] 1 point.** Most of the collisions above were caused by planning a path that was too close to obstacles. One simple conservative way to avoid such collisions is to pad the obstacles by a small buffer zone. Pad the obstacles in configuration space by one grid cell (including diagonal neighbors), and verify that the resulting trajectory does not contain any swept-volume collisions.



C7: More conservative trajectory from start to goal, with no swept-volume collisions.

Sampling-based Motion Planning

Download the starter code (`ex2.zip`). In this file, you will find:

- `ex2_motion.m`: Main function. Call `ex2_motion(<questionNum>)` with an appropriate question number (0–5) to run the code for that question. Do not modify this file! (Feel free to actually make changes, but check that your code runs with an unmodified copy of `ex2_motion.m`.)
- `M0.m` – `M5.m`: Code stubs that you will have to fill in for the respective questions.
- `check_collision.m, check_edge.m`: Helper functions to perform collision checking.

In this part of the assignment, you will implement two sampling-based motion planning algorithms, the probabilistic roadmap (PRM) and the rapidly exploring random tree (RRT). A 4-DOF 2-link arm has been defined, together with a spherical obstacle. The cylinders depicting the arm's links should not collide with the obstacle, but the cylinders located at the joints are just markers visualizing the axes of rotation and do not contribute to potential collisions.

M0. **[CS 5335 only.] 2 points.** Familiarize yourself with the provided code in `ex2_motion.m` and the robot that has been defined. Use the code in `M0.m` to visualize the robot in various configurations. The code in `ex2_motion.m` that calls `M0` also checks whether the configuration is within joint limits, and whether the configuration is in collision.

Look at the code in `check_collision.m` and `check_edge.m`. Explain what each function is doing to check for collisions, for individual configurations and for configuration-space line segments respectively. Identify and describe two potential issues in the collision-checking algorithm (but do not try to fix them).

M1. **1 points.** Despite the issues you may have identified in M0, we will proceed with the provided collision checking functions. Given the provided joint limits, generate uniformly distributed samples from configuration space. The configurations should be within the joint limits, but they can be in collision.

M2. **2 points.** Implement the PRM algorithm to construct a roadmap for this environment. Your graph should contain `num_samples` collision-free configurations within joint limits. For each vertex, consider the nearest `num_neighbors` neighbors, and add an edge if the segment between them is collision-free. Edges are specified using the output `adjacency` matrix, where `adjacency(i,j)` is the distance between vertices v_i and v_j if an edge exists between them, and is 0 otherwise. Note that the adjacency matrix should be symmetric (edge between v_i and v_j implies edge between v_j and v_i), and that each vertex may have $\leq$ `num_neighbors` neighbors (if some edges are in collision) or $\geq$ `num_neighbors` neighbors (v_j may be closest to v_i, but there may be other vertices closer to v_j).

Hint: The next question relies on the output of `M2.m`, which may take a while to compute. To avoid re-computing it in future questions, we have provided functionality to save it in your MATLAB workspace, and pass it in on future calls:

```
[samples, adjacency] = ex2_motion(2);
ex2_motion(3, samples, adjacency);
```

M3. **2 points.** Use the roadmap to find a collision-free path from the start configuration to the goal configuration. You will need to appropriate points to get "on" and "off" the roadmap from the start and goal respectively. *Useful functions*: `shortestpath`

M4. **2 points.** Implement the RRT algorithm to find a collision-free path from the start configuration to the goal configuration. Choose appropriate hyperparameter values for the step size and frequency of sampling q_{goal}. Remember to traverse the constructed tree to recover the actual path.

M5. **2 points.** The path found by RRT likely has many unnecessary waypoints and motion segments in it. Smooth the path returned by the RRT by removing unnecessary waypoints. One way to accomplish this is to check non-consecutive waypoints in the path to see if they can be connected by a collision-free edge.

M6. **0 points.** If M4 and M5 have been implemented correctly, this question should require no further implementation. Watch your algorithm work on a more challenging motion planning problem. Can you make it even harder?