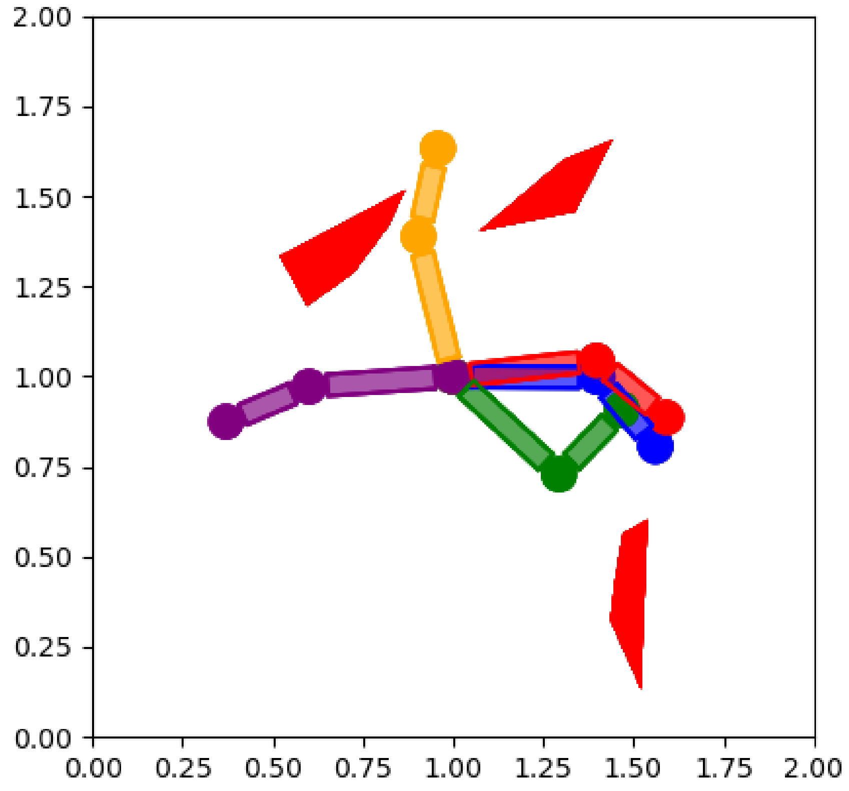
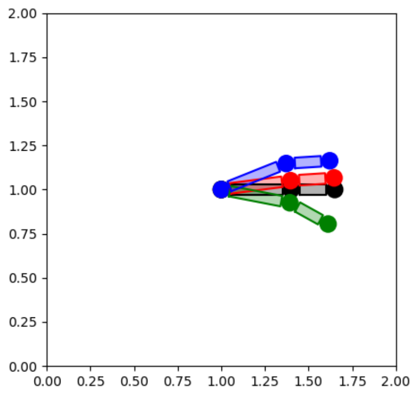
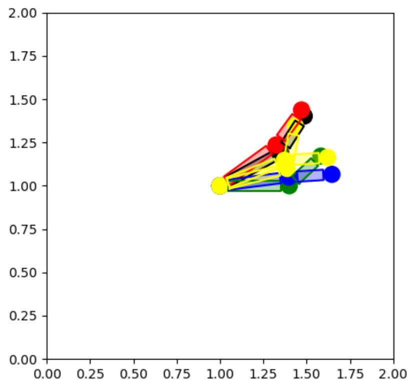
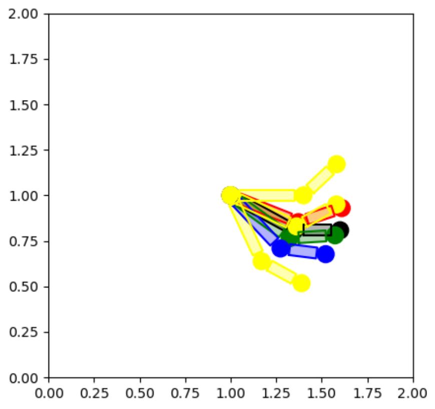
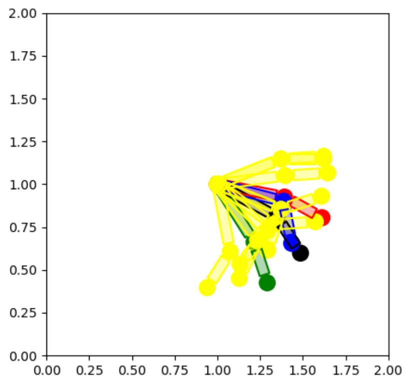
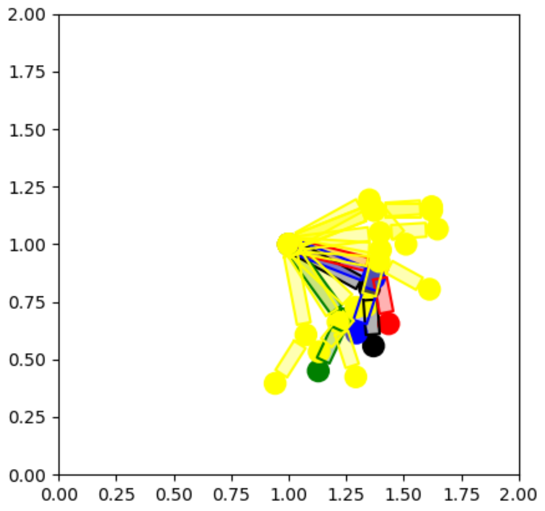
1.1 Sampling random collision free configurations

The NLinkArm class models the robot arm, utilizing transformation matrices to update joint positions based on given joint angles. Collision checking is facilitated by creating rectangles representing the arm links and verifying their intersection with polygons in the provided environment map. The collision_free_config function randomly generates joint angles, repeating the process until a collision-free configuration is achieved. The script visualizes the robot arm and environment using matplotlib. The collision avoidance strategy involves adjusting the rectangles representing the links and checking for collisions with the polygons.



1.2 Nearest neighbors with linear search approach

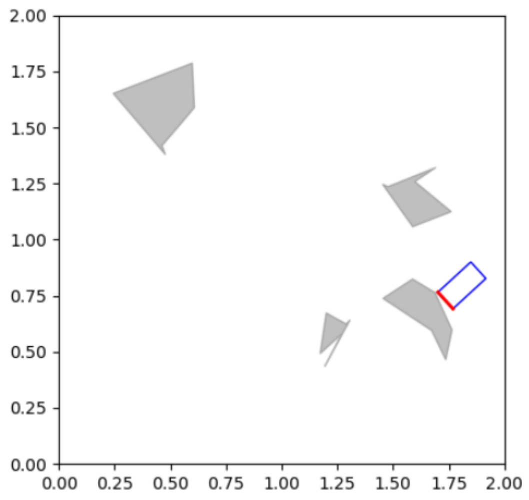
We used np.linalg.norm(config1 - config2) to calculate the Euclidean norm, which is the square root of the sum of squared differences between corresponding elements of the two configurations. This distance metric effectively captures the geometric intuition of the separation between two points in the joint space.

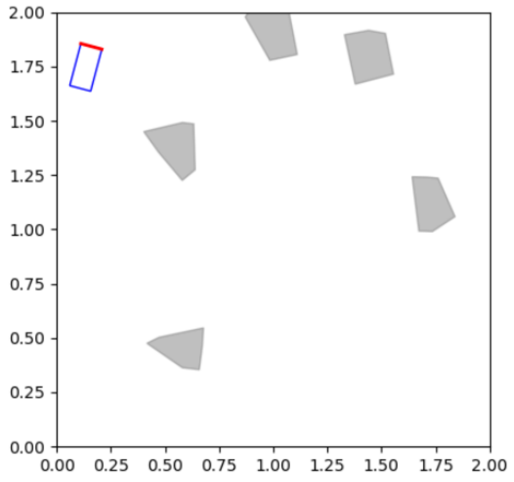

2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00

2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00

2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00

2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00

2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00

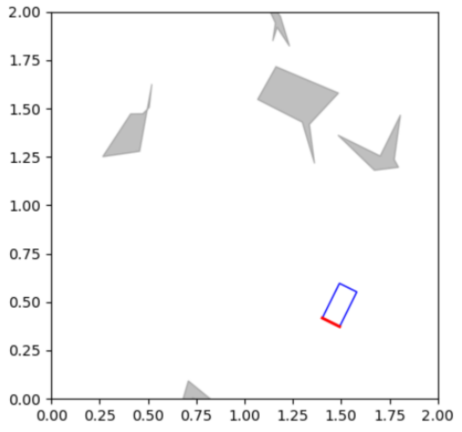
1.3 Interpolation along the straight line in the C-space

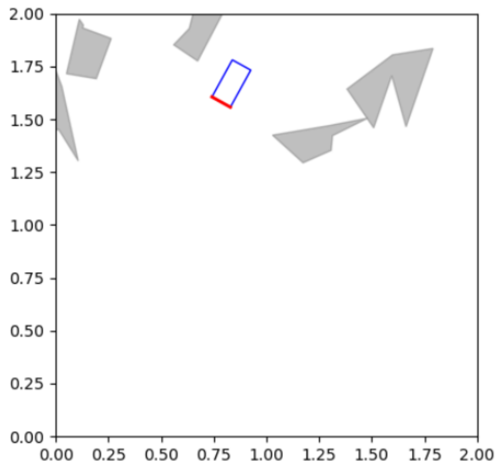
This employs a transformation matrix approach to update joint positions based on angles, creating a visual representation of the arm's configuration through rectangles and circles. The animation smoothly interpolates between these configurations using the FuncAnimation class, generating a clear and visually appealing representation of the arm's motion

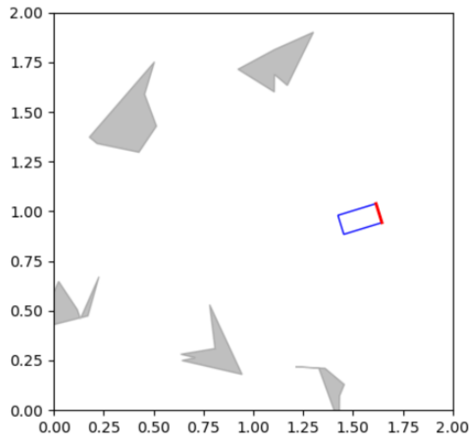
2.1 Sampling random collision-free configurations







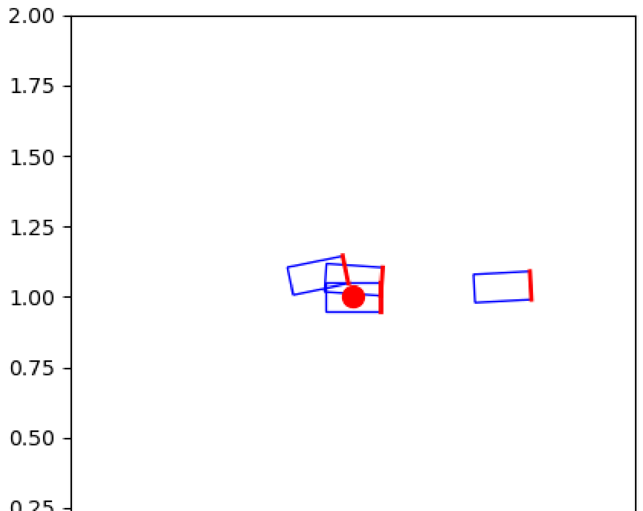




Using the same rigid body class from Assignment 1, we checked collisions between the line edges of the robot and polygons for both the initial location and also as the robot moves in order to determine if its path will collide with a polygon.

2.2 Nearest neighbors with linear search approach

Using the distance metric for translation and rotation distance, we combine the two distances with the given weighted average (α = 0.7) which gives more bias to the translation component and finds and then plots the k nearest neighbors to the target.



2.3 Interpolation along the straight line in the C-space

(Videos in zip file) The interpolate function makes a series of configurations between the start and goal coordinates. It then interpolates the x, y, and theta (orientation) between the start and goal. For our code it reaches the goal not when the lines cross the coordinate but when the geometric center of the rigid body is on the exact target coordinate to ensure accuracy.

2.4 Implement RRT

(Videos in zip file)

2.5 Implement PRM

(Videos in zip file)