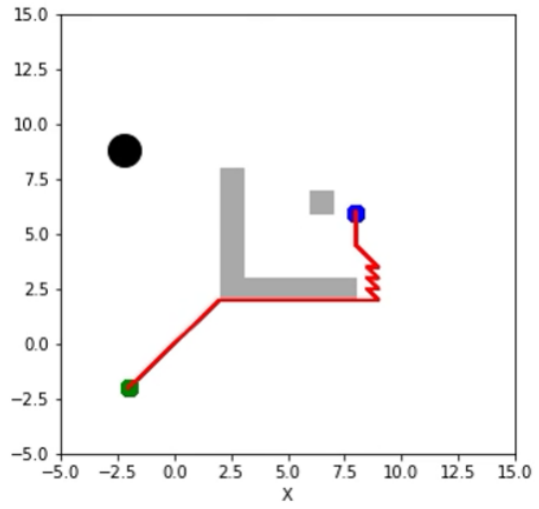
Path Planning Algorithm Implementation

Srijan Pal

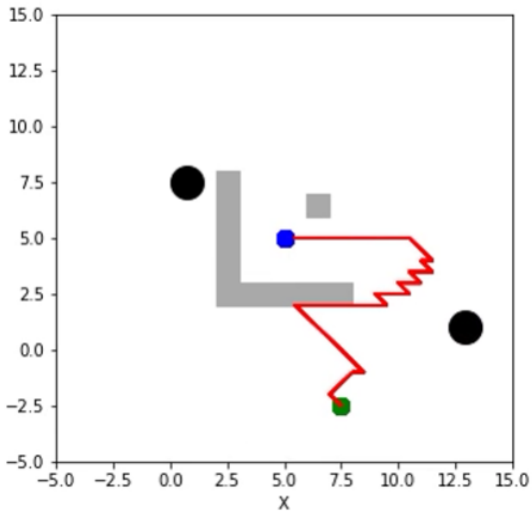
The path planning algorithm implemented here is mainly concerned with finding the neighboring points of the tip of the red line in the animation and directing the red line in a direction that will not collide with any obstacles (static or dynamic).
These paths generated guide the tip of the red line in almost the shortest direction during each particular frame considering the position of both dynamic and static obstacles, even though these are certainly not optimal for the dynamic environment.

Performance :

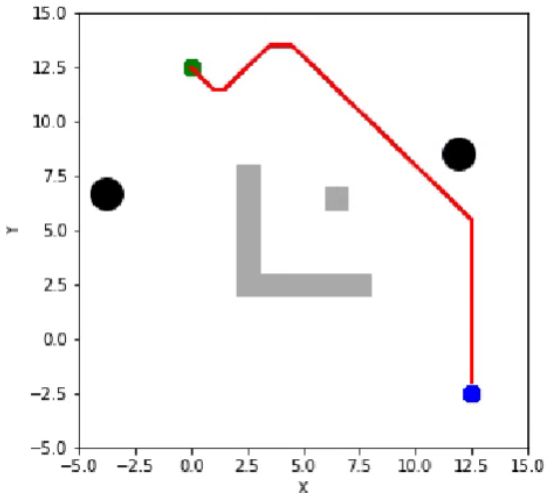
Start: (-2,-2) —Goal: (8,6)



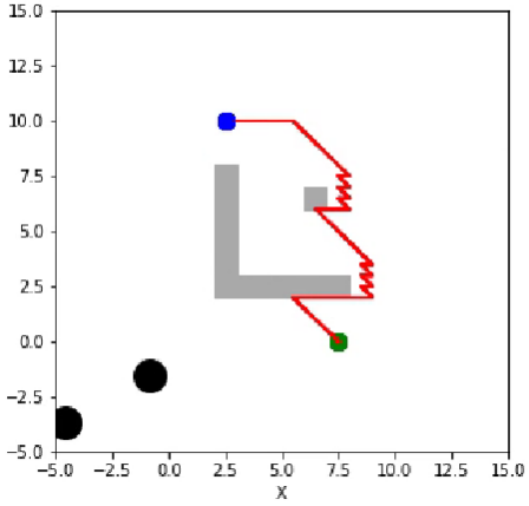
Start: (7.5,-2.5) —Goal: (5,5)

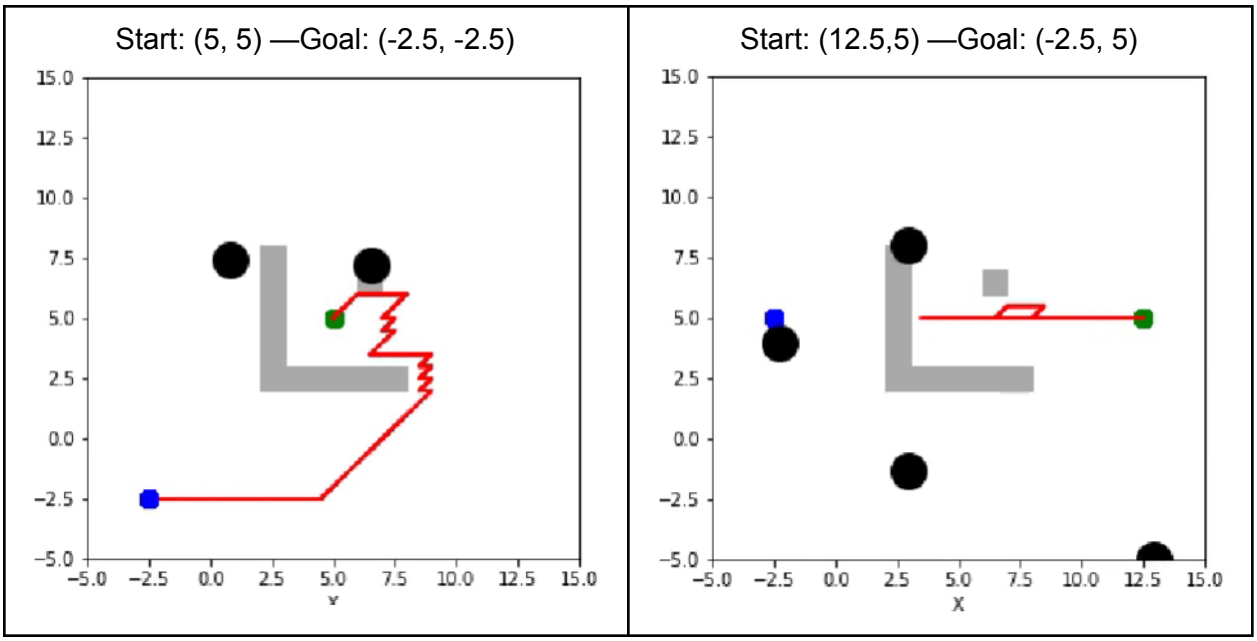


Start: (0,12.5) —Goal: (12.5, -2.5)



Start: (7.5, 0) —Goal: (2.5, 10)

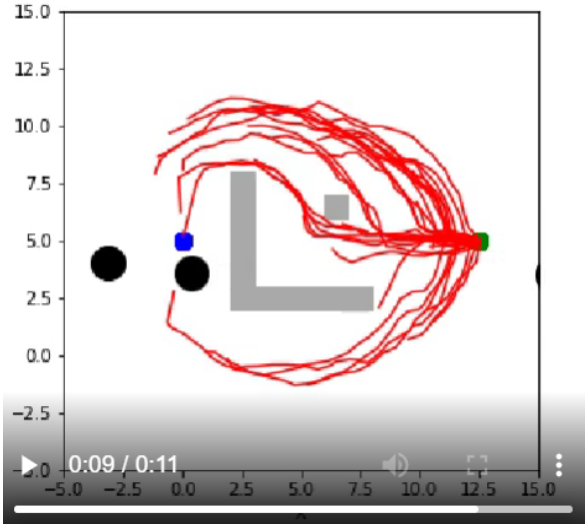




For all of the above cases, the algorithm took less than 50 frames or timesteps to reach the goal from the starting point except the last case.

In the last case, the red line could not reach the goal and is stuck in a loop in the region. This was a drawback in the algorithm implemented. After analyzing it was seen that if the red line approaches a static obstacle from the right (i.e. the goal being toward the left side of the plane) if somehow the static obstacle is directly at a right angle to the shortest path then it gets stuck in a loop.

For this reason, I tried a different approach which is almost similar to RRT* path planning for a segment of the path generated at a particular timeframe. Due to randomization in the RRT* there are multiple numbers of paths generated:



Implementation of the above-mentioned algorithm can be found here.