16-811 Homework 2 Resubmit

Changsheng Shen (Bobby)
changshs@andrew.cmu.edu

10/16/2019

Q8:

(a): We define two vectors $\vec{ij} = (x^{(j)} - x^{(i)}, y^{(j)} - y^{(i)})$ and $\vec{ik} = (x^{(k)} - x^{(i)}, y^{(k)} - y^{(i)})$, in fact $\vec{ij}$ and $\vec{ik}$ can be visualized as the two edges of the triangle formed by the three points.

Then for any point $p = (x, y)$ falls within the triangle, the vector $\vec{ip}$ can be expressed a linear combination of $\vec{ij}$ and $\vec{ik}$, with constraints on the coefficients:

$$\vec{ip} = (x - x^{(i)}, y - y^{(i)}) = \alpha * \vec{ij} + \beta * \vec{ik}$$

where

$$0 \leq \alpha \leq 1$$
$$0 \leq \beta \leq 1$$
$$\alpha + \beta \leq 1$$

Therefore, we can write the system of linear equations as:

$$\begin{bmatrix} x^{(j)} - x^{(i)} & x^{(k)} - x^{(i)} \\ y^{(j)} - y^{(i)} & y^{(k)} - y^{(i)} \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} x - x^{(i)} \\ y - y^{(i)} \end{bmatrix}$$

with constraints:

$$0 \leq \alpha \leq 1$$
$$0 \leq \beta \leq 1$$
$$\alpha + \beta \leq 1$$

By incorporating the constraints into the equation, and rearranging it:

$$\begin{bmatrix} x^{(i)} & x^{(j)} & x^{(k)} \\ y^{(i)} & y^{(j)} & y^{(k)} \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

where

$$\alpha, \beta, \gamma \geq 0$$

(b):

See q8.py

(c):

First of all, get a subset of all the paths that lie on the same side of the ring of fire. This depends on which side the starting position is at (by checking if $x < y$).

For each triple (i, j, k) with three paths, I checked whether our start point is inside the triangle formed by the start points of the three paths.

If yes, then store this triple combination as a candidate triple. Do this for all possible triples.

Finally, choose one candidate triple whose start points are closest to our starting position, by computing and comparing the sum of three euclidean distances, from each start point of a path, to our starting position.

Once we choose the triple with three paths, we can compute the weights by solving the linear equation in (a). See q8.py for details.

In terms of interpolation, first compute the 50 points on the path as the weighted sum of the chosen three paths, and then do a dense spline interpolation in between every two adjacent points (handled in matplotlib, plt.plot()).

(d).

The three paths being interpolated are marked with red, blue and green. The interpolated path is marked with gray.

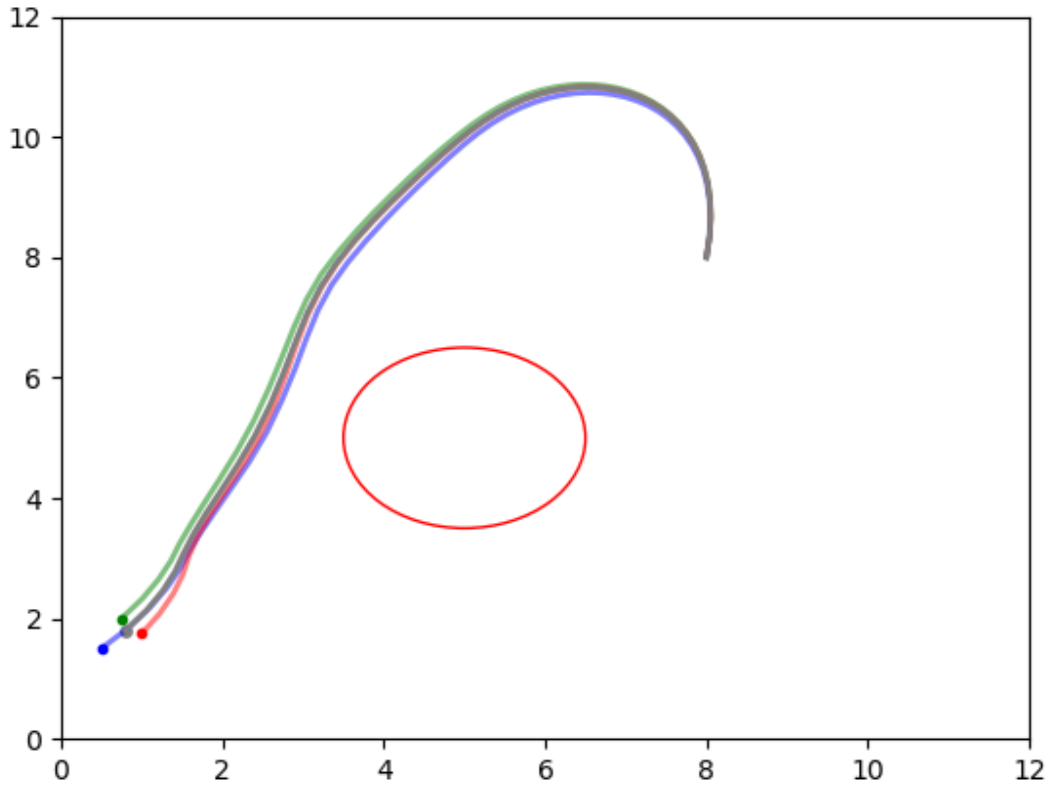

Figure 1: Starting Position: (0.8, 1.8)

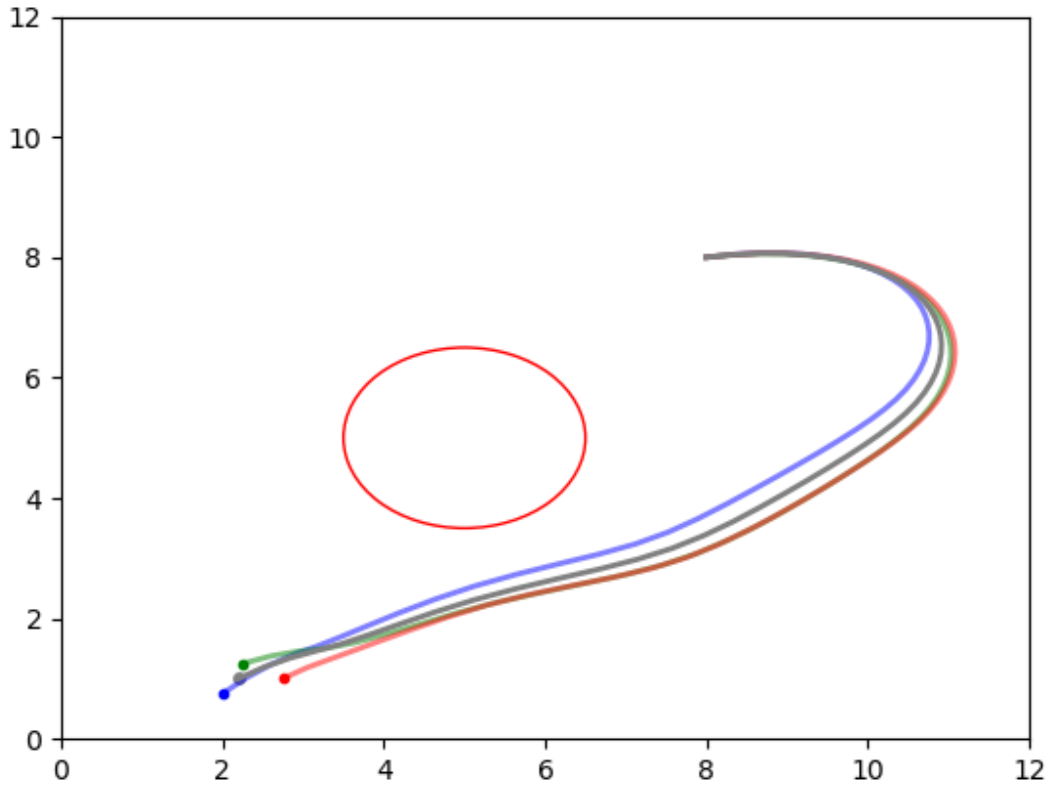

Figure 2: Starting Position: (2.2, 1.0)

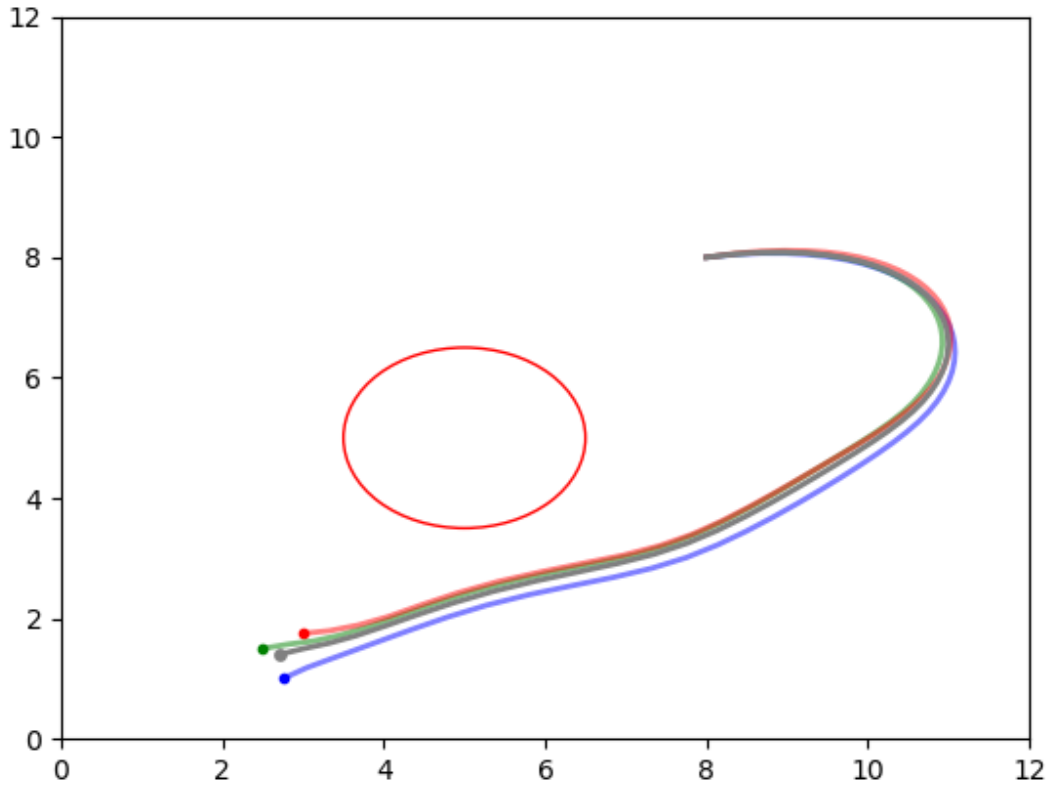

Figure 3: Starting Position: (2.7, 1.4)

(e).

If more obstacles were to be introduced, I will do a more complicated initial filtering to get a subset of paths that lies "together". Namely, any pair of two paths in the subset should not be on different sides of any of the obstacles.