Robotics II

Day 6: Trajectory Planning

Chua Tzong Lin

B7TB1703

chuatl@mems.mech.tohoku.ac.jp

1. Making initial character using b-spline.

The chosen initial character is “C”. Initially, 6 points are chosen as inputs for the b-spline program. The chosen points form a course structure of the letter “C” as shown in figure 1.1.

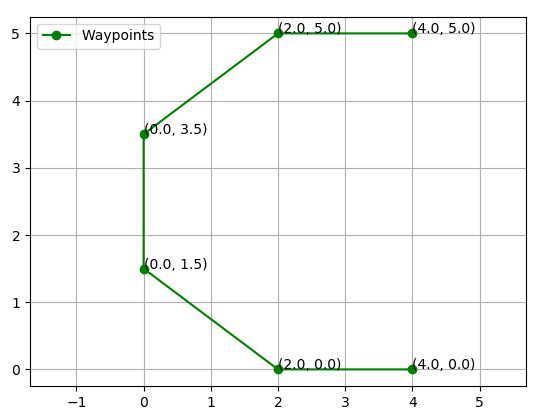


Figure 1.1 Chosen starting points

Using the b-spline interpolation method, the curve is generated which is shown in figure 1.2.

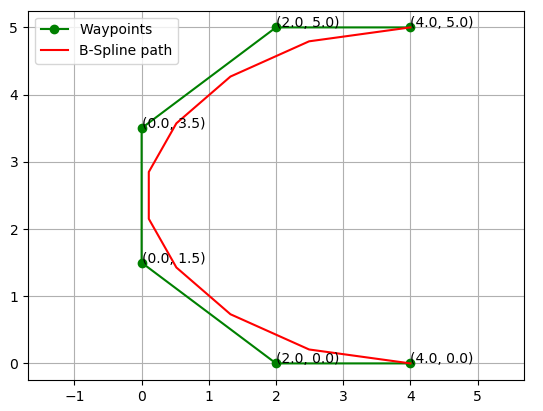


Figure 1.2 B-spline path

From figure 1.2, it can be observed that the generated curve is still discontinuous, to further smoothen the curve, the output points can be further inputted into the B-spline algorithm to generate a smoother curve as shown in figure 1.3.

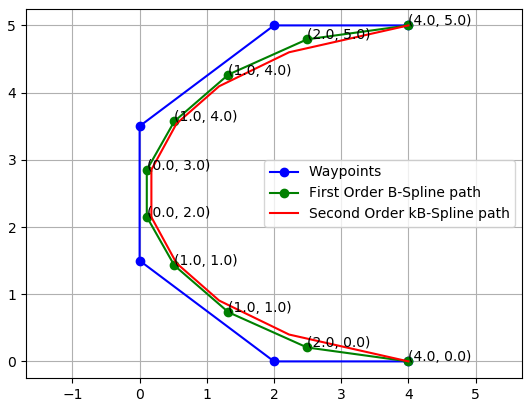


Figure 1.3 2nd Order B-spline path

1. Making initial character using cubic function.

From figures 1.1 to 1.3, it can be observed that the curve generated using b-spline is discontinuous, other than that, it does not always reach the desired points. The curve can be smoothen out by interpolating the newly generated points into the b-spline function as shown in figure 1.3. However, this increases the deviation from our desired points.

Hence, to address this problem, another interpolation method is introduced, which is the cubic function interpolation. The same initial points are used as shown in figure 1.1. The resulting curve is shown in figure 2.1.

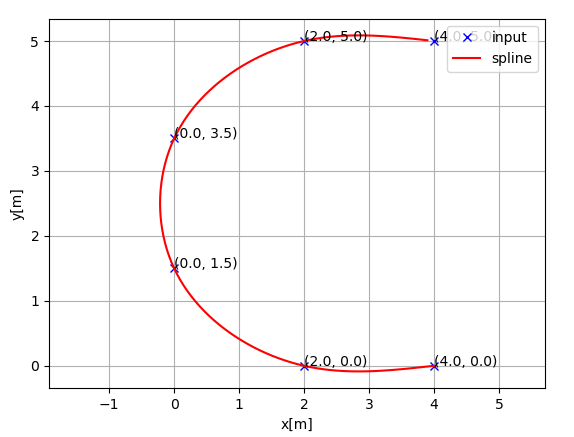


Figure 2.1 Cubic-spline interpolated curve

The amount of turning of each point of the line curve can be compared by calculating the curvature and yaw of each point. The curvature and yaw of each point is shown in figure 2.2 and figure 2.3.

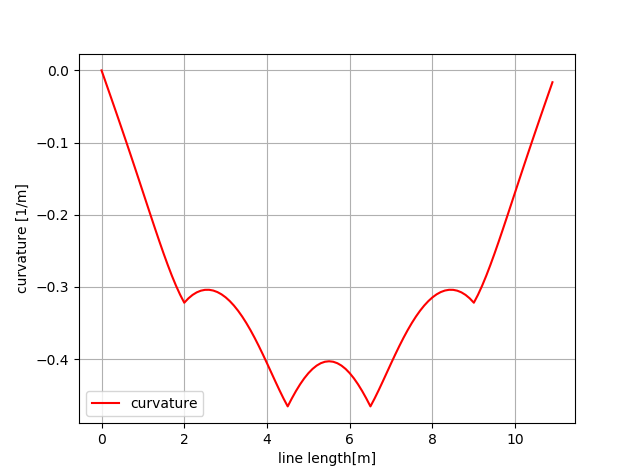


Figure 2.2 Curvature of each point in the line curve

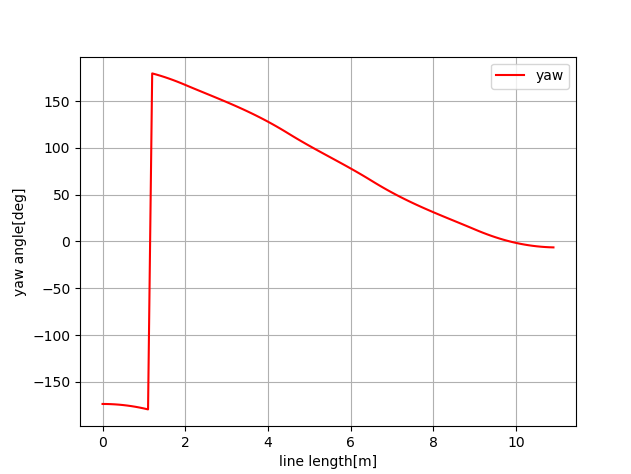


Figure 2.3 Yaw of each point in the line curve

1. A cost map is generated using the points in the curve in figure 1.3 for navigation. The points are considered as obstacles in our simulation. However, in most cases when the obstacles are too close to the initial starting point or to the goal, the program will fail to navigate its way through the maze. Examples are shown in figure 3.1.

When selecting the obstacles, it is important to place objects away from the places stated above. The successful cases are shown in figure 3.2 and figure 3.3. The interesting case is shown in figure 3.2, where the navigation point will take a long detour if it fails to reach the desired goal in the first try.

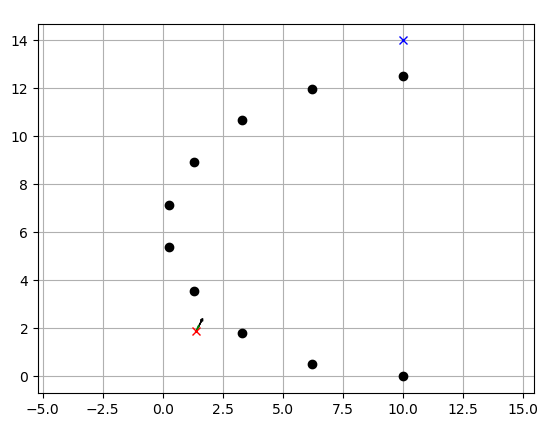


Figure 3.1 Example of failed case

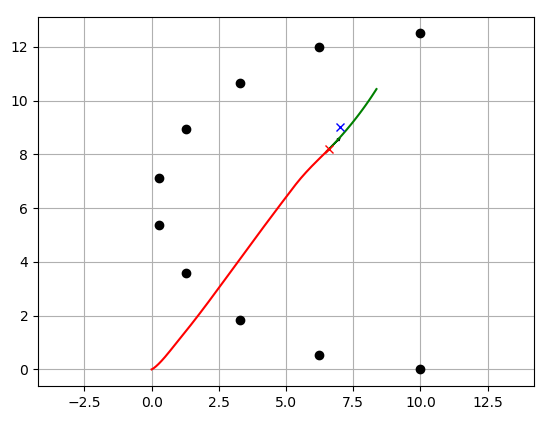


Figure 3.2 Example of successful case

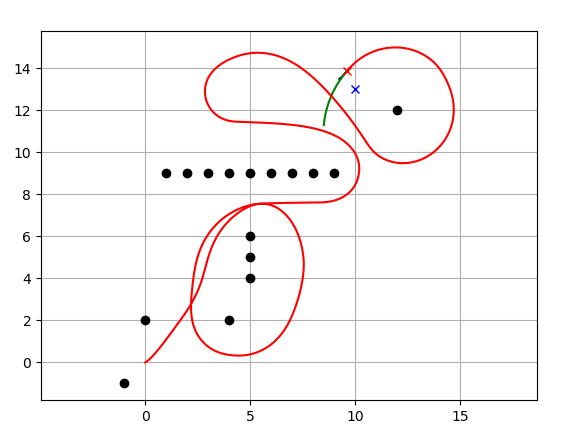


Figure 3.3 Example of successful case