Obstacle-avoidant Function Fitting

Artur Roos

October 2022

Contents

1 Introduction

1.1 Rationale

Not longer than two days ago I was challenged to a mathematical standoff by a good friend of mine, the weapons would be polynomial functions and their graphs. Winning this competition was a matter of honor, so I used all of my skill and wits to prepare. The fight would be held in "Graph Wars", a small competitive 1v1 game published on the 23rd of February 2022.

The goal of the game is to design a function that can be plotted from one point to another without intersecting any circles in shortest time. You are given 60 seconds to input as many functions as you want, then the turn is passed onto your opponent. This is wrapped into a war-like setting where your function is a projectile, directed by your soldiers against your enemy's.

It is also usually played without assistance of any external programs like plotters or calculators, that's for good reason. Most importantly, game is considered solved: for any initial state of the field there can be deduced a function to guarantee a "hit".

1.2 Aim

My ultimate goal set out to be simply designing an algorithm, which when given a list of all circle obstacles, their radii, and the edge points would express the path to go from to another without colliding with any of the obstacles.

2 Investigation

The task consists of the small independents subtasks: finding the points and fitting the function. Task-specific restrictions are mentioned in the corresponding section.

2.1 Finding The Points

This problem can be boiled down to pathfinding. However, usual pathfinding techniques operate on discrete spaces like graphs and grids which our plane is not, we need to convert it to one. Cells of the grid can be in two states: occupied or empty. The occupied cells are the ones covered by an obstacle. Since obstacle circles only occur at integer coordinates and the smallest radius of a circle is 1, we can safely assume that the smallest size of a grid cell we might need is a half. After that we can apply the A* pathfinding algorithm to yield us a set of points A, those are the points that the function should go through.

TODO

2.2 Fitting The Function

Let A be a set of length n where each point is like $\langle x, y \rangle$. For simplicity we can impose a strict total ordering on the set, and reindex it in such way that $\forall a_1, a_2 \in A(a_1 < a_2 \Leftrightarrow a_{1_x} < a_{2_x})$. In this way the L_n is guaranteed to be the rightmost point, which will make the mathematics a lot easier. Let's also move the coordinate system so that the point at bottom left $\langle -50, -20 \rangle$ is at the origin. That will make all the coordinates positive and they are easier to reason about. Simple transation $g(p) = \langle p_x + 50, p_y + 20 \rangle, p \in A$ does the trick. To reverse this transformation later we might just use g^{-1}. We

may also ignore all the x values in ranges $(-\infty; a_{1_x})$ and $(a_{n_x}; \infty)$, since the function will not be evaluated there at any point.

There is an infinite number of ways to make a function that goes through a set of points, the simplest one is the Lagrange Interpolation which does fit this usecase perfectly.

Sadly, the game does not accept piecewise functions so we have to design a way to express it as a, for example, polynomial. We may follow a specific example and try to generalise it later. Let $A' = \{\langle 0, 1\rangle, \langle 6, 9\rangle, \langle 17, 4\rangle, \langle 19, 22\rangle\}$.

First we may try to make a function $\phi_i(x)$ such that it equates to 0 at every point, but x_i. For simplicity we may make it equate to 1, this will allow for better composability in the future. For the second point the process might look something like this:

First make it be zero at all the points, but x_i.

$$\hat{\phi}_2(x) = (x - 0)(x - 17)(x - 19) \tag{1}$$

Now the value at x_2 is non-zero, we can just devide it by $\hat{\phi}_2(x_2)$.

$$\hat{\phi}_2(x_2) = (6 - 0)(6 - 17)(6 - 19) \tag{2}$$

$$\phi_2(x) = \frac{(x - 0)(x - 17)(x - 19)}{\hat{\phi}_2(x_2)} \tag{3}$$

$$\phi_2(x) = \frac{(x - 0)(x - 17)(x - 19)}{(6 - 0)(6 - 17)(6 - 19)} \tag{4}$$

$$\phi_2(x_1) = 0, \phi_2(x_2) = 1, \phi_2(x_3) = 0, \phi_2(x_4) = 0$$

To make ϕ_i completely representative of the point at x_i we need to scale it by y_i. Let's define a new helper function $\psi_i(x)$.

$$\psi_2(x) = \frac{(x - x_1)(x - x_3)(x - x_4)}{(x_2 - x_1)(x_2 - x_3)(x_2 - x_4)} * y_2 \tag{5}$$

$$\psi_2(x) = y_2\phi_2(x) \tag{6}$$

$$\psi_2(x_1) = 0, \psi_2(x_2) = 9, \psi_2(x_3) = 0, \psi_2(x_4) = 0$$

Looking at the table of all the values of ψ_i we can deducde that their linear combination will satisfy our condition of $\psi_i(x_i) = y_i$ and $l_i(x_j) = 0, j \neq i$. So for set A' the fit function is

$$f(x) = \psi_1(x) + \psi_2(x) + \psi_3(x) + \psi_4(x)$$

If we generalise this function to for any set A it would look something like the following.

$$f(x) = \sum_{i=1}^{|A|} \psi_i(x)$$

$$\psi_i(x) = y_i \phi_i(x)$$

$$\phi_i = \prod_{j=0, j \neq i}^{|A|} \frac{x - x_j}{x_i - x_j} \tag{7}$$

$$f(x) = \sum_{i=1}^{|A|} \prod_{j=0, j \neq i}^{|A|} y_i * \frac{x - x_j}{x_i - x_j} \tag{8}$$

3 Limitations

This method covers a lot of scenarios, but it doesn't yield correct results when the next intended point on the path a_2 which goes after some a_1 satisfied $a_{2_x} < a_{1_x}$. Because of the reindexing this changes the order of points. Beyond that, this arises only in situations when the f is not a well-defined function, since that requires it to have to two different values at same x. Luckily, those scenarios never occured in a real game. Perhaps the game accounts for that and intentionally avoids it.

4 Reflection

TODO

5 Conclusion

TODO

6 Bibliography

TODO