Shortest Path for Marathon Runners

Jennifer Liu

Cost Function Implementation and Justification:

Time = Final Cost

The algorithm envisions the speed of a traveller as a time function. From physics, we know that time = distance / speed. The time is my final cost value. (The G function)
The following sections will describe how distance and speed is computed.

The most optimal path is the path that takes the least time to get there.

Distance

Since elevation is included in addition to the 2D pixels, this map can be visualized as a 3D space. We know from mathematics that to calculate the distance between 2 points on a 3D space, we can use the Euclidean distance. The equation used to calculate the distance in 3D space is shown in Eq 1.

Eq.1

$$\text{2D Distance} = \sqrt{(\Delta x * 10.29)^2 + (\Delta y * 10.29)^2}$$

$$\text{3D Distance} = \sqrt{(\text{2D Distance})^2 + (\text{Elevation Difference})^2}$$

Scaling from pixel to real distance

We need to consider the real distance of the traversing. As described in the write up, traversing 1 pixel in the longitude direction = 10.29m. Traveling 1 pixel in the latitude direction = 7.55m. To consider the scaling, I multiplied the (x, y) location by its respective constants. Hence x_i is multiplied by 10.29, and y_i is multiplied by 7.55

Speed

The speed is determined based on the terrain type. For terrains that are easy to navigate in (i.e Open land, Easy movement forest etc...), I gave it a higher speed value. That is because it can reach its destination quicker if the terrain is easier to move in.
For instance, I gave open land a speed constant of 12, and out of bound a constant of 0.1.

Using just the speed of the terrain would be inaccurate as it does not take the elevation into account. To consider the elevation factor, we multiply the speed by the elevation factor. We normalize the elevation so that it doesn't dominate the speed function too much.

The Equation for Speed:

Uphill Elevation: (If the elevation is uphill, the elevation should reduce the speed)
speed = speed * (1 - elev_diff/100)

Downhill Elevation: (If the elevation is downhill, the elevation should increase the speed)
speed = speed * (1 + elev_diff/100)

Heuristic Function Implementation and Justification:

The way heuristic is calculated is the same as cost function. The difference is that rather than calculating the distance between its current point and its neighboring points, it is calculating the distance between the current point and the final destination point. I take the elevation difference between final destination and current location. The terrain I consider is the terrain of the current point.

The heuristic function is designed to underestimate the real distance. As it is only considering a straight line distance between current location and final location. The elevation is the elevation different between the final point and current point and the terrain of the current point.

Seasonal Algorithms:

Summer

For summer, there is nothing to be added.

Fall

We identify pixels that are the easy movement color, and we identify edges of easy movement. Anytime a traveller crosses these pixels, their speed decreases.

Winter

We identify pixels that are water edges. We look at every water edges, and using BFS traverse 7 pixels out from the water edge in all possible direction and color the pixel to any icy blue. Anytime a traveller cross these pixels, the speed is decreased.

Spring

Implementation is similar to Winter. We identify all the water edge pixels. We look at every water edge pixel, and using BFS traverse 15 pixels out from the water edge in all possible direction. In addition to coloring the pixel, we also check whether the elevation from current pixel and the original water pixel that this traversal started from is greater than 1m. If the elevation is greater than 1m, it is not a mud, else it is. Walking on mud increases the speed.

Sample Output:

Season: summer
The graphical path is saved to: Results/Brown/brownSummer.png
The total distance of this route is: 5006.83 m

Output Explanation:

Line 1: Tells reader what season this route is generated on
Line 2: Tels reader where the output with path outlined is saved to
Line 3: The total distance of this route