Optimal Racing Line on a Generated Track

Ross Halpin, Mary Coyle and Nathan Morris

CA4 – Project Milestone 4, Submitted in part towards completion of Strategic Thinking – CCT

Introduction

The purpose of this project was to identify a problem that could be approached using an artificial intelligence model, in the hope of gaining further insight into possible solutions.

The problem we chose to combat was finding the optimal racing line around any given track. In initial brainstorming sessions it became clear than a common interest in sports analytics was prevalent in the group. From this, we began research into which sports would be most suitable for analysis in terms of data collection and availability. This led us to motorsport.

Despite this, after deciding this as our research path, we found difficulty initially in finding suitable track data. A solution to this was proposed by Ross, who suggested creating our own tracks using OpenStreetMap GeoJSON data points. This provided us with seemingly endless opportunity to create suitable tracks.

Objectives

- ❑ Create a suitable track using OpenStreetMap data
- ❑ Design a physics-based simulation of a vehicle traversing the track
- ❑ Apply a genetic algorithm to the simulation to create iterative optimizations of the racing line. Shorter track time is rewarded
- ❑ Map the line taken by the simulation during the lap, eventually achieving a close approximation of the optimal racing line
- ❑ Evaluate findings to determine if results are converging to the optimal path or further optimization required

Track Creation

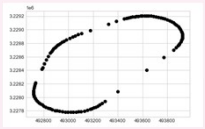
Due to the initial difficulty of accessing appropriate track models, the solution we devised was to create our own tracks.

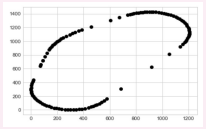
OSM is free-to-use geospatial data and can be used to convert real roads into datapoints. OSM can be queried using their Overpass API to extract the required route in a GeoJSON file format. This will provide us with the central line of our track as the GeoJSON defines the route as multiple nodes with a common relation.

The figures below show the feature querying functionality of OSM, the roadways in the vicinity have been highlighted.

Methods and Tools

Once the roads have been transformed and mapped, each coordinate becomes a datapoint with values of X and Y. Initially, the track exists at a distance away from the origin of 0,0 but can be adjusted using the Python features of map and lambda. The figures below show the before and after of this process





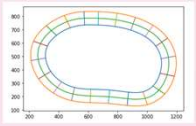
In terms of our project, OSM is not able to provide us with exact widths and shapes of roads, only the central line. This creates a problem in that a racing line can't be defined unless a track exists, meaning it will need to be created with a defined width at any given point.

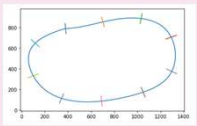
There are multiple ways to approach this. One method would be to utilise satellite imagery of the chosen street and use a computer vision algorithm to estimate the width along it. Another would be to base the width on the road standards of the particular country that road in situated.

As the purpose of the project is more about finding the optimal racing line, the accuracy of the track width isn't necessarily important. Therefore, we decided to implement a set width along the track as the turns and corners would be the more telling factors.

Track Generator

For this project we built a custom piece of software for generating tracks from inputted cartesian coordinates. This is done through interpolation of the inputted points and creates a spline, which is a continuous line representing the sparse inputted points. The track is dividedly evenly by N and at each Nth point we calculate the tangent and normal of the point. We then extrude along the normal until the track width. The end of each normal is then captured and used to interpolate splines for track limits.

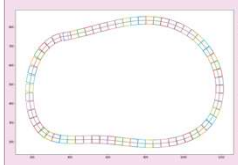


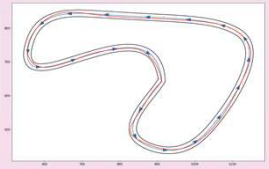


Random racing line generator

To create a sample space from which we will generate random racing lines, we use a method by which we create N sample squares around the track. From these squares we choose a random point within that square.

We then interpolate a spline from these random points to create a random racing line





Genetic algorithm

With a genetic algorithm we are attempting to optimise a function through operations similar to our understanding of natural evolution. We start with a population of random potential solutions and evaluation of their fitness.

We then iterate using the evolutionary operators to optimise the solution further and further based on some desired fitness of the solution. In this project the time taken to traverse the track race track is the fitness.

In a genetic algorithm we have 3 operations: Selection, Crossover, Mutation. Selection is the process through which better solutions are selected over worse solutions for crossover. Crossover would be the mixture of parameters between two fit individuals to create a new individual. Mutation is a random low chance that parameters of a new individual will change to a new value.

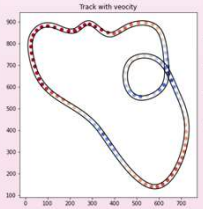
```
def selection(pop, k=4):
    sel_rand = randint(0, len(pop))
    for sel_i in randint(0, len(pop), k):
        if pop[sel_i][2] < pop[sel_rand][2]:
            sel_rand = sel_i
    return pop[sel_rand]

def crossover(p1, p2, r_cross):
    c1 = np.array(p1[:].copy())
    c2 = np.array(p2[:].copy())
    if rand() < r_cross:
        crossover_point = randint(1, control_points)
        c1 = np.concatenate((c1[:crossover_point],c2[crossover_point:]))
        c2 = np.concatenate((c2[:crossover_point],c1[crossover_point:]))
    return [c1, c2]

def mutation(child, r_mut):
    for i in range(len(squares)):
        if rand() < r_mut:
            rp = random_point_within(squares[i])
            child[3][i] = rp
    return child
```

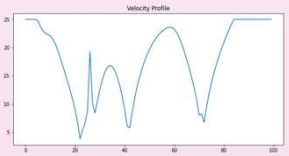
Physics based fitness

To evaluate the fitness of a racing line solution we must determine the time a given vehicle might take to traverse the track. We are doing this in 2D with simplified physics. In this case we an equation for velocity which takes two constants, a force F and a mass m. These can be simplified to a single constant c. For Formula 1 cars this value is generally around 1.3 m/s and represents the maximum turning velocity.

$$v_{max} = \sqrt{\frac{Fr}{m}} = c\sqrt{r}$$

The value for r is the radius of curvature at each point which is calculated non-trivially using each point as a centre point along with the immediate previous and next points as points on the circle. We Can see the output of this velocity calculation visualised for sparse points on the track along with a velocity profile. Time is then calculated by setting a total distance for the track and dividing it by the average velocity.





Results

In determining how well the genetic algorithm performed we chose to both randomly generate racing lines and to generate racing lines with our genetic algorithm.

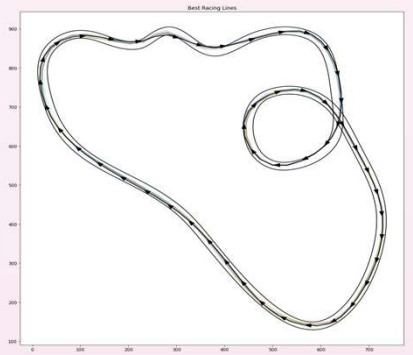
How well would random perform if we generate the same number of random racing line as our Genetic Algorithm?

Random generated racing lines with 50 iterations and population of 500
25000 racing lines randomly generated
Best time: **311.2580106466346**
Average Best Time: **314.28778991582976**
Average Worst Time: **326.6378874060011**
Average Time: **321.096051052866**

How well did our Genetic Algorithm Perform?

Racing line optimisation with 50 generations and population of 500
Best time: **309.00084374047975**
Average Best Time **309.97085054308224**
Average Worst Time **321.9977028304318**
Average Time **314.9527337410784**

Here we can see the best racing line plotted alongside some of the other best racing lines generated by the GA



Conclusions

	Random	Genetic Algorithm	GA Improvement
Best Time	311.2580	309.0008	-2.2572
Average Best Time	314.2877	309.9708	-4.3169
Average Worst Time	326.6378	321.9977	-4.6401
Average Time	321.0960	314.9527	-6.1433
		Average Improvement:	-4.3393

As can be seen in the table above the Genetic Algorithm was very successful in only a short amount of 50 generations. We managed to beat random generation of racing lines by 4 seconds on average. In the racing world pole positions are won by hundredths of a second so these are great results..
During each phase we ensured to meet the deliverables that we had previously outlined in the gantt chart and updating when needed.
Our team each brought a unique set of skill to this project which complemented each other.
Although our team was somewhat restricted by Covid-19 and conflicting Work schedules we are all proud of this project.. It is of an overall opinion that we each learned skills that we will all be able to carry forward.
Overall each team member is happy with the results of this project and how we came to those results.