Race car strategy optimisation under simulation

Naveen Chaudhary (Y9359) and Shashank Sharma (Y9544)
Guide: Prof Amitabh Mukerjee
April 17, 2013

Abstract

Optimal racing line selection is an important problem in racing games. Those who select the best racing line are mostly the champions while the others fail to do so. Thus it will be very useful to have a method that can generate optimal racing line. Aim of this project is to find the optimal racing line for a track and design a controller for a car to traverse on it. We have implemented an artificial intelligence method for the same.

Introduction

In racing games the route that the vehicle follows is known as the racing line. For any given track there is an enormous number of racing lines possible. An optimal racing line is the one which reduces the time needed to complete the race.

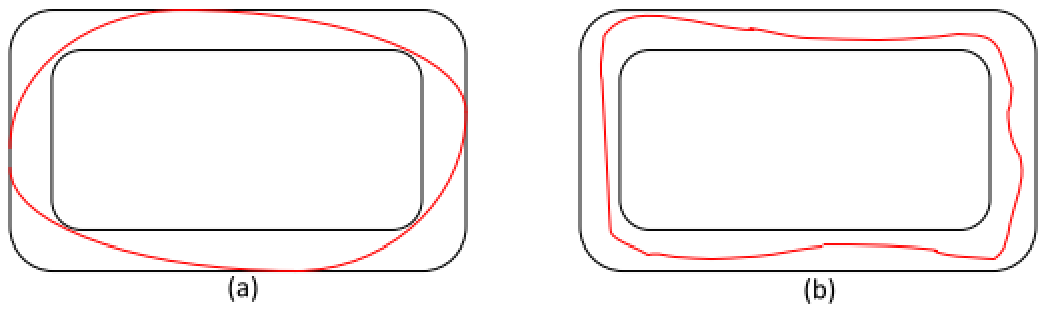

Fig1: Two different racing lines on the same track [6]

An optimal line considers the conditions of the track and makes smart decisions based on the track. Example, (b) is making many random unnecessary turns, while (a) has a smoother path and avoids unnecessary turns. So (a) is a better racing line than (b). However it may not be the best solution.

In racing games, the corners make the most difference in performance. On straight tracks all racers can reach the maximum speed possible and just move in a straight line, but when there is a turn the speed is bounded and there is a trade-off between the speed and the length of the racing line. A smoother racing line is longer and a more curvy racing line may be shorter. And hence to generate the optimal path and follow it is crucial.

Previous Work

In earlier racing games, the waypoints on the track on crucial turns were manually defined, then smoothening curves were fit into these points and points were added where the path led outside the track. The controller of the car was behavioural based and hand coded.

Some work was done trying to implement AI in the games without the computation of racing lines, however it was not so successful on a competitive level.

Efficient computation of racing lines to develop better paths of which the K-1999 algorithm was considered to be the best.

Determination of Optimal Path

The track is represented as a set of connected polygons. Waypoints are defined on the connected edges.

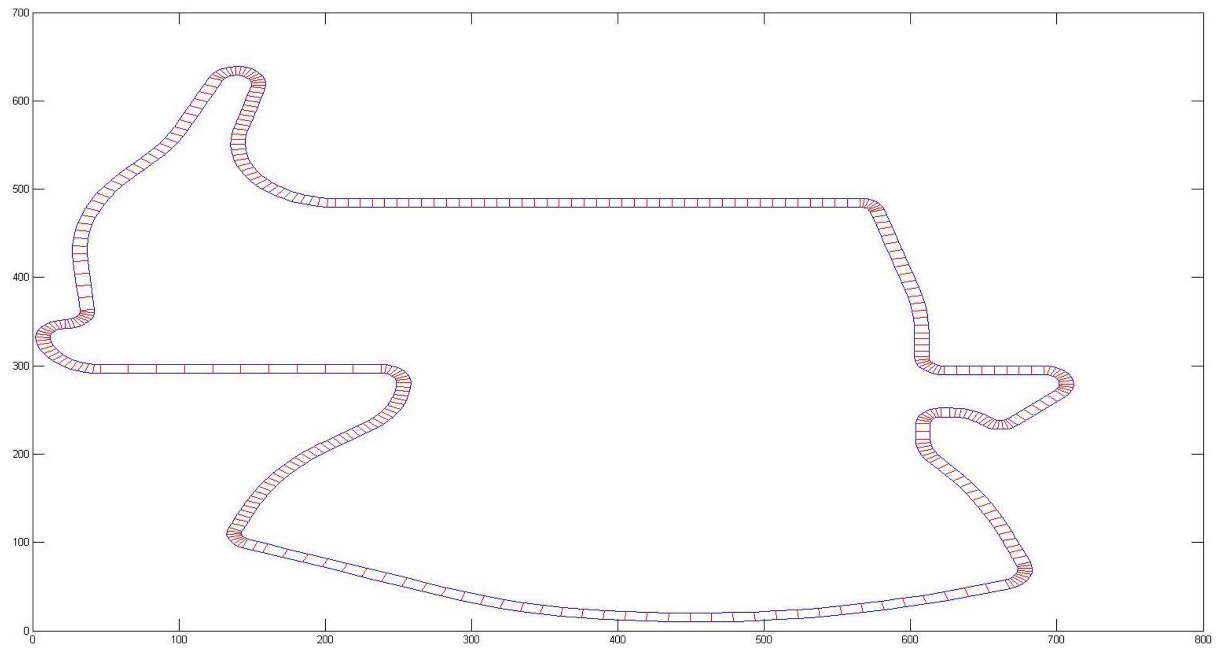

Fig2: Segmented track

The outline of the method follows as:

- Set the starting point as 0.
- For each point I, decide the point i+1 using forward looking algorithm.
- Run many times and select few best Energy racing lines.
- Among these arcing lines we compute the expected finishing time and then chose the best.

  Best energy can be computed using the formula $\int \sqrt{K}\, ds$, where K is the curvature of the path s.

The forward looking algorithm tries to make the selection of the points smarter by looking ahead on the path for turns if any. It computes the difference in curvature at two points ahead on the track. It gives a measure of whether there is a left (+ve), right (-ve) turn, or a straight (0) path on an analog scale from -1 to 1. According to these, it adjusts the position on the track.

The working of the forward looking algorithm is explained below:

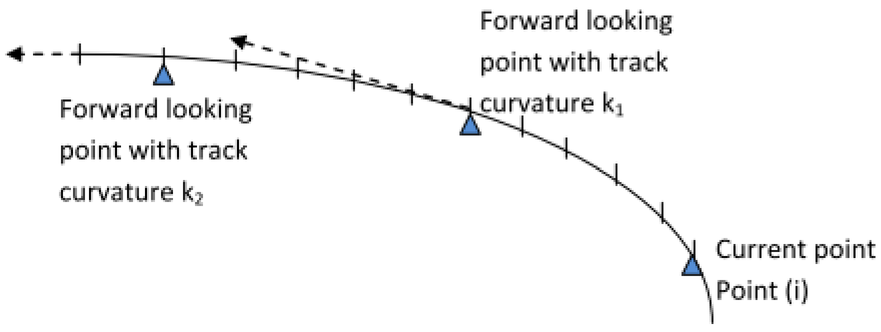
Forward looking point with track curvature k_1
Forward looking point with track curvature k_2
Current point Point (i)

Fig3: Forward looking algorithm [6]

If there is a left turn approaching, then it can be interpreted as composed of a straight path (a), a left turn (b) and again a straight path (c). Consider the two points for the forward looking algorithm be K1, and K2, with K2 being ahead of K1.

- When both K1 and K2 lie on (a), the output of the forward looking algorithm shall be 0, and the racing line will go straight.
- When K2 lie on (b) and K1 on (a), the output of the forward looking algorithm shall be positive, this will cause the racing line to move towards the right edge of the racing track.
- When K1 lie on (b) and K2 on (c), the output of the forward looking algorithm shall be negative, this will cause the racing line to move towards the left edge of the track.
- When both K1 and K2 lie on (c), the output of the forward looking algorithm will again become 0 and the car will move straight until another turn is encountered.

The outputs of the forward looking algorithm, in a nutshell can be visualised as, when the car approaches a left turn, it will try to move towards the outside edge of the track (right), then tries to hit the apex on the inner edge (left), and then exists straight out of the corner. Thus, giving a smooth path.

While this forward looking algorithm provides a near optimal solution, but better paths that are close to it are possible. For each point along the line, the probability distribution of possible points along the width of the track is a Gaussian distribution that is skewed according to the previous steps taken and future road conditions.

This Gaussian distribution is not computationally efficient, thus it is approximated to a linear probability distribution model.

Where the slope of the distribution, m is defined as:

$$m = \min\left(\max\left(\frac{k2 - k1}{maxK}, -1\right), 1\right)$$

K1, k2 are the curvatures at the forward looked points K1, K2, and maxK is the maximum curvature along the complete track.

Thus the probability distribution function is set as

$$f(x) = \frac{1}{2} + \frac{1}{2}\, m.x$$

According to the current direction of the car, to ensure a smooth path, a scope is defined in the form of a cone with angle as 15^0. The next point shall lie within this scope. According to the probability distribution function an appropriate point is selected within this scope.

As said earlier, this may not be the perfect point, thus a random element is added to this point chosen so as to get a large variety of tracks.

This algorithm is repeated 10,000 times and for every path, energy is computed, lower energy implies a less curvy path but not necessarily the quickest. However, it has been shown in previous works, a lower energy path is close to the quickest path. Thus, 10 best energy paths are chosen.

For these 10 paths, the following integral is computed:

$\int \frac{1}{v} ds,$ where v is the maximum possible speed at the point on the path

The path which gives the minimum value of the integral is chosen as the racing line for the given track.

Making ANN for path

When the car is running, at every instant a target point is assigned to the car based on the waypoint on the next segment of the optimal racing line calculated.

This target point, the current position of the car, the speed of the car, the yaw of the car with respect to the track, and the current gear of the car form the inputs of the ANN.

The outputs that are required are the steering angle (-1 to 1), acceleration value (-1 to 1), and the gear change (-1 or +1).

The hidden layer for the ANN was decided to be 40 x 1. This hidden layer has to be chosen via experiments, but due to lack of time, this size was chosen randomly.

Due to lack of training data, the car was driven along the track manually trying to follow the racing line computed as closely as possible. As this line was not graphically available, the training data generated was not perfect.

A simple back-propagation algorithm of gradient descent algorithm was implemented to train the network.

Results

The controller was tested on 3 different tracks available on TORCS, viz. Aalborg, Forza and Alpine2.

Track	Our time	Best time	Energy of racing line
Aalborg	1:22:81	1:18:23	57.86
Forza	1:42:86	1:40:97	22.7944
Alpine2	1:52:40	1:49:67	49.4576

The racing lines found for different tracks are shown below:

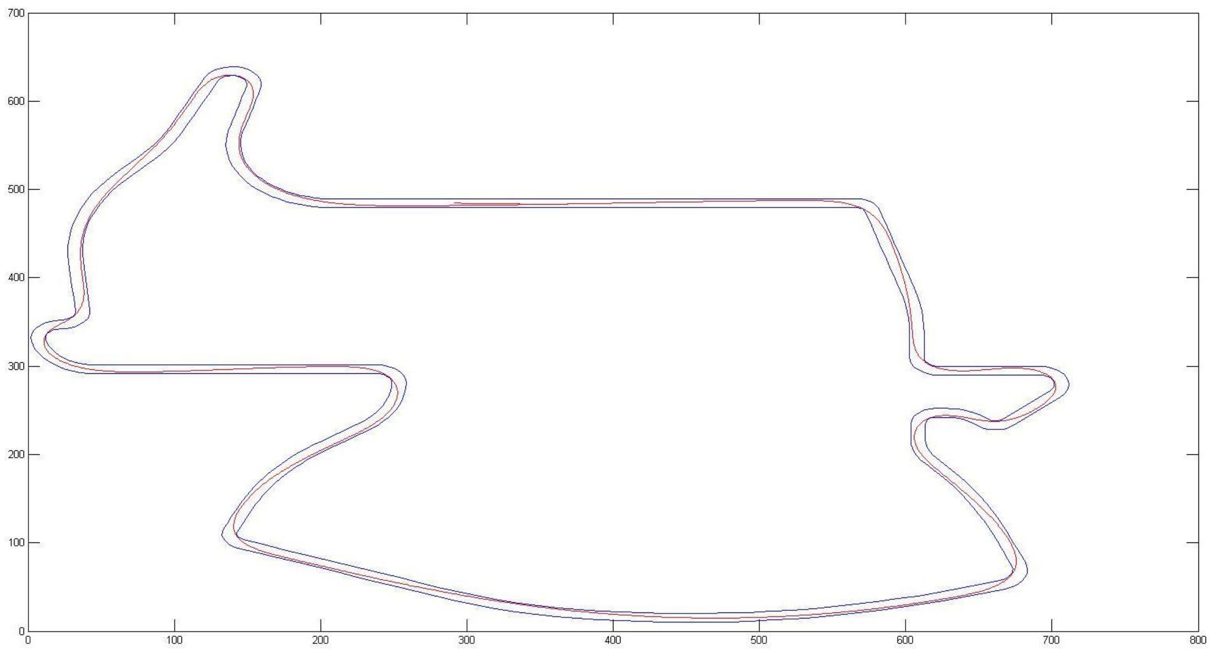

Fig4: Racing line for Aalborg track

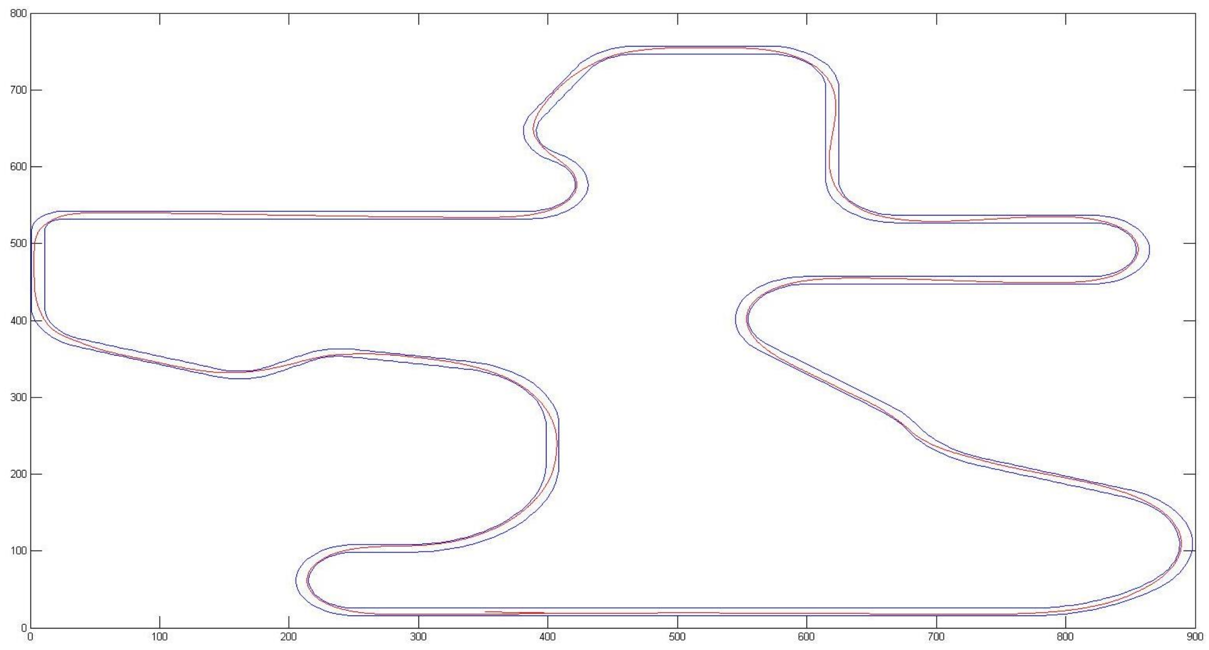

Fig5: Racing line for Alpine2 track

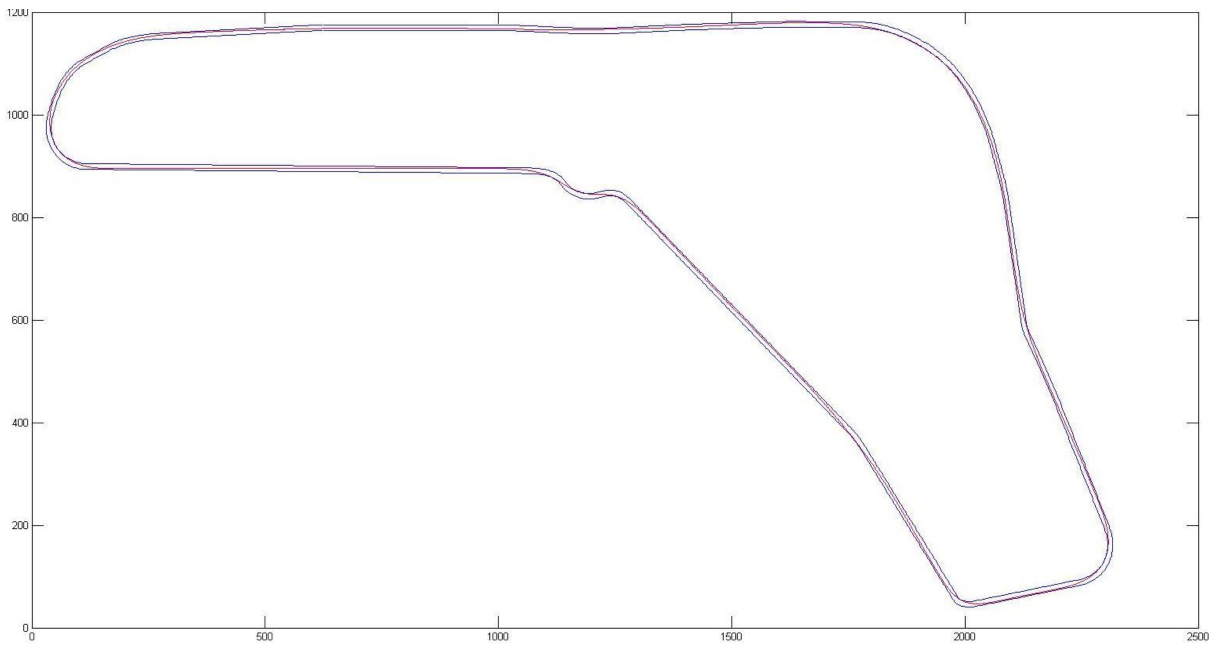

Fig6: Racing line for Forza track

Conclusion

The times obtained by the developed controller were only a few seconds far from the overall best times obtained on the selected tracks.

This difference may be explained through the lack of training data available. Another explanation can be the improper choice of the size of ANN. ANN has to be chosen with experiments with different number of nodes, thus manoeuvrability of the car on the racing line is compromised.

Future Work

As a part of future work, the controller can be modified for the introduction of opponents. This will require the addition of tactics like, drafting, overtaking, collision avoidance, etc.

If the car gets stuck outside the track, the drive mechanism has to be changed completely to get it back on the track.

Introduction of real racing scenarios like, pits-tops to deal with damages and fuel problems.

References

[1] Jung-Ying Wang and Yong-Bin Lin, “Game AI: Simulating Car Racing Game by Applying Pathfinding Algorithms”, International Journal of Machine Learning and Computing, Vol 2, No.1, Feb 12.

[2] J. Togelius and Simon M. Lucas, “Evolving Controllers for Simulated Car Racing”, Proceedings of the congress on evolutionary computation, 2005, pp. 1906-1913.

[3] C. H. Tan , J. H. Ang , K. C. Tan and A. Tay "Online adaptive controller for simulated car racing", Proc. IEEE Congr. Evol. Comput., pp.2239 -2245 2008.

[4] L. Cardamone , D. Loiacono and P. L. Lanzi "On-line neuroevolution applied to the open racing car simulator", Proc. IEEE Congr. Evol. Comput., pp.2622 -2629 2009.

[5] L. Cardamone, D. Loiacono, P.L. Lanzi, and A.P. Bardelli, “Searching for the optimal racing line using genetic algorithms”, In Computational Intelligence and Games (CIG), 2010 IEEE Symposium on, pages 388-394, aug. 2010.

[6] Y. Xiong, “Race Line Optimization”, thesis submitted to MIT, September 2010.

[7] TORCS – The Open Racing Car Simulator, “http://www.torcs.org”.