International Conference on Engineering, Science and Technology

October 7-10, 2019 www.iconest.net Denver, CO, USA

Genetic Algorithm in Trajectory Optimization for Car Races

Dana Vrajitoru
Indiana University South Bend, USA, dvrajito@iusb.edu

Abstract: In this paper, we present several methods of constructing trajectories for autonomous cars in a car race setting using the TORCS car race simulation system with the goal of improving race completion time. The first method builds the trajectory procedurally using known efficient curves. It starts by mapping the race track using available sensors provided by TORCS. The track is then reconstructed and segmented based on the curvature direction. Last, efficient curve profiles are applied to each segment. The second method applies smoothing and gradient descent optimization algorithms to improve such trajectories built by the first method. Finally, we use a genetic algorithm to find an optimal trajectory by minimizing the overall length of the curve. A coarser segmentation is used in this case and a number of key points are selected for the trajectory definition. We compare the results of these different methods on two chosen race tracks and show that the genetic algorithm is capable of building a trajectory of a lower total length than other methods, but the results in an actual race are less predictable.

Keywords: genetic algorithms, trajectory optimization, autonomous cars, car race

Introduction

In this paper, we present several algorithms constructing a trajectory for an autonomous car in a simulated car race setting and compare their efficiency in terms of trajectory length and car racing time. The car's trajectory is defined as a curve spanning the length of the race track from start to end and confined to its boundaries. The first algorithm segments the track in intervals of constant curvature sign and builds the trajectory using a procedural method based on known efficient curve shapes. The second one starts from either a simple trajectory, or one built by the first method, and applies a combination of a straightening method and a smoothing transformation. The third algorithm is an application of genetic algorithms (GAs) to this problem by also segmenting the track and by aiming to minimize the total length of the trajectory.

Autonomous vehicles have been a growing interest with practical applications in the last years, as they are preparing to be deployed on the road in the near future. The research presented in this paper was performed in the TORCS system that implements simulated car races with multiple vehicles available as well as several tracks of various difficulty (Wymann, 2013). The system also provides a customizable car controller. Our research is related to the work presented in (Vrajitoru, 2018), where a neural network (NN) is trained to compute a target trajectory value based on a procedurally optimized trajectory. The NN takes the local road curvature data as input and produces a target trajectory value as output. The direction unit in the car controller can follow this trajectory value, which is a component we also use in our current research.

Multiple papers can be found showcasing algorithms for computing optimal paths within road constraints. The curves used by our first trajectory computation method can be found in (Velenis, 2005), and are computed both to minimize the time it takes to get through a curve, and the maximum exit velocity. These curves are in line with what driving instructors usually recommend, and are also similar to the trajectory presented in (Liniger, 2015). Our problem settings for the trajectory calculation are similar to (Liniger, 2015); in both cases, the car trajectory is computed for a track situation, aiming to optimize the travel time. The difference is that in their application, they pre-compute safe zones on the road for the car and use them to improve the trajectory computation in real time.

A number of approaches for track prediction are present in the literature, as for example, the track segmentation approach in (Quadflieg, 2010) where the track is divided into fragments classified based on a pre-defined set of polygon types. Onieva et al. (Onieva, 2012) propose another controller based on track segmentation. Their driving controller, named AUTOPIA, is a very successful participant in the TORCS simulated racing car competitions. Such segmentation algorithms are also related to map-matching algorithms such as the ones presented in (Rathour, 2017). Genetic algorithms have recently been used in several studies for trajectory optimization, as for example, in (Schlueter, 2018), (Padhye, 2008). The kind of trajectories built in these studies are for space travel, and the authors report successful results when compared to real-world missions. Air traffic control is another potential GAs application (Marceau, 2013). The paper is organized the following way. Section 2 introduces the procedural trajectory computation algorithm and the optimization methods. Section 3 describes the application of genetic algorithms to this problem. Section 4 presents experimental results and compares the three methods. The paper ends with conclusions.

Autonomous Car Pilot and Procedural Trajectory

The research presented in this paper was conducted using the TORCS car race simulation system. TORCS sustains a large community of developers and users, and it is the platform for popular competitions organized every year as a part of various international conferences (Guse, 2010). The program is organized as a server implementing car races combining multiple vehicles on a variety of tracks. A client module can be written by the user to implement the actions of a controlled car. The server provides runtime information about the car's position and orientation on the track, the car's speed, damage, and other measures, and the opponents' relative position in the car's vicinity (Rathour, 2017). The client can control the steering wheel ([-1, +1]), the gas pedal ([0, +1]), the brake pedal ([0, +1]); and the gearbox (-1 through 6) (Vrajitoru, 2018).

The application provides multiple race tracks of various length and difficulty. We have chosen two tracks for this particular research: E-Track5 and E-Road. The first one is an easy track with a few curves in each direction that is ideal for parameter tuning. The second track is longer and more difficult, with fast-switching turns where the procedural algorithm cannot properly optimize the trajectory. It has potential as a good testing ground for the GAs. Figure 1 below shows the outline of the two tracks.

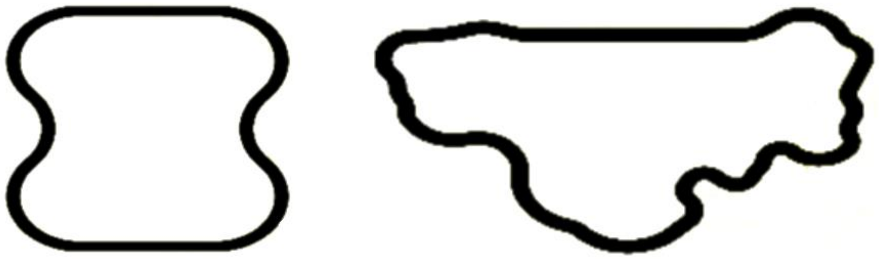
Figure 1. E-Track 5 (left) and E-Road (right)

Since the geometry of the tracks is not provided as such, but is only indirectly available through the car state measures provided by the server at race time, the first step for these trajectory calculations was to map the tracks. We did this by running the car at a low constant speed (50 km/h) and by calculating the curvature of the road using the TORCS sensor information in each frame. These measurements were recorded and then used by the track building algorithms.

Because of artifacts of the curvature measurement algorithm, the reconstructed tracks don't conserve the track shape, even though the main curves are still highlighted. For the reconstructed shape to be somewhat similar to the original, we had to scale the curvature down. The road shape seen in Figure 4 represents the reconstructed E-Track 5 with a curvature scale of 0.18. The shape and especially the amount of turning in the reconstructed track can have an influence on the efficiency of the track building algorithms. In particular for the GA, if the track turns are scaled down too much, the difference between the fitness of a good trajectory and that of a worse one can be too small for the evolution process to perform well.

Figure 2 shows the track E-road reconstructed with two different curvature scales: 0.065 on the left, and 0.1 on the right. The left one is the closest to a simple curve almost closing on itself, while the second one highlights the curves more prominently and might be better suited for trajectory computation. We define the car trajectory as a function r(t) returning a value in the interval [-1, 1] for each point of the track's centerline t. This value represents the target lateral position of the car on a perpendicular line to the road centerline. Figure 3 illustrates this definition. The function r(t) can take values outside the interval [-1, 1] if the car exits the road, but we assume that this is not a desirable trait in such a target function.

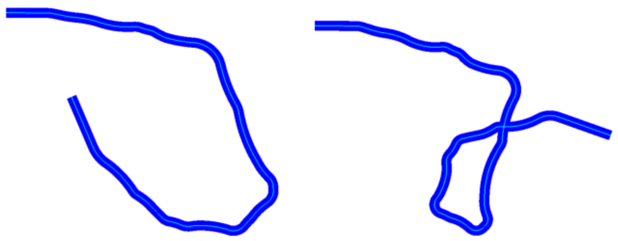
Figure 2. E-Road Reconstructed with Scale 0.065 (left) and 0.1 (right)

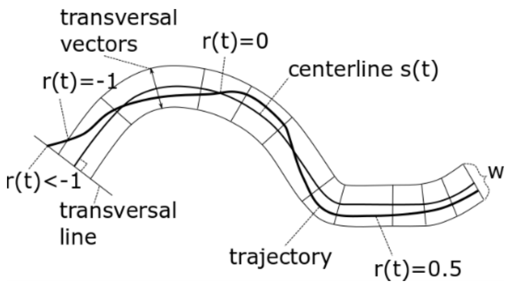

Figure 3. Road and Trajectory Definition

Autonomous Car Pilot

The autonomous car pilot we used for these experiments is called *Meep*, and represents an improvement to the controller called Gazelle (Albelihi, 2015). It is composed of a target direction unit, a speed unit, a recovery unit activated when the car gets outside of the track or crashes against the road shoulder, a special alerts unit that is on the lookout for dangerous road conditions, and an opponent avoidance unit. Since the current experiments focus on the trajectory itself, we have used a constant speed for these experiments, set at 80 km/h. There are no opponents involved in these tests, so that module is not necessary.

Finally, in the direction unit, the pilot uses a target value for the trajectory in the interval [-1, 1], representing an objective value for the lateral position of the car on the road in each frame of the race. This value is provided by one of the pre-computed trajectories in this paper. The steering value is computed using two measurements: the current lateral position of the car on the road, and the current angle between the car's axis and the centerline of the road. Both of these are provided by TORCS.

Thus, the target steering angle is calculated first by reversing the current angle with respect to the road centerline, to align the car's axis with the road. To this angle, we add the difference between the current lateral position on the road and its target value, multiplied by a parameter. To avoid turning too sharply when the road position is far off, we use the square root of the difference if its absolute value is larger than 1. For the same reason, we cap this difference at 2.5. The resulting steering value is computed the following way, then clamped to the interval [-1, 1]:

```
deltaTraj = targetTraj – currentTraj;
clamp(deltaTraj, -2.5, 2.5);
if (|deltaTraj|>1)
        deltaTraj /= sqrt(deltaTraj);
steer = roadAngle + kt*deltaTraj;
steer /= maxAngle;
```

where **targetTraj** is the target trajectory, **currentTraj** is the current lateral position of the car on the road, **roadAngle** is the current angle of the car with the road centerline, and **kt** is an adjustable parameter. The function **clamp** restricts a value to a given interval.

Procedural Trajectory

For the procedural trajectory, the road is first mapped by having a vehicle drive at constant speed along the track and computing the curvature of the road in each point using the free distance information. Then the road is segmented based on the contiguity of the sign of the curvature, as seen in Figure 4 in purple.

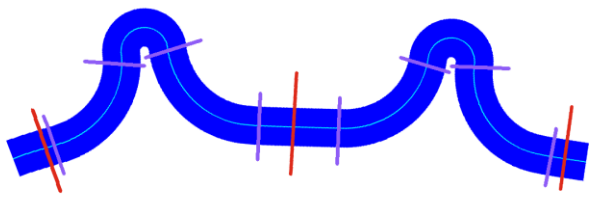
Figure 4. Track Segmentation for Procedural (purple) and GA (red) Optimization

Thus, each segment, marked by purple delimiters, represents either an almost flat region, or one where the road curves only to the right, or only to the left. For each segment, an optimal curve profile, shown in Figure 5 left, is used to calculate a locally optimal trajectory.

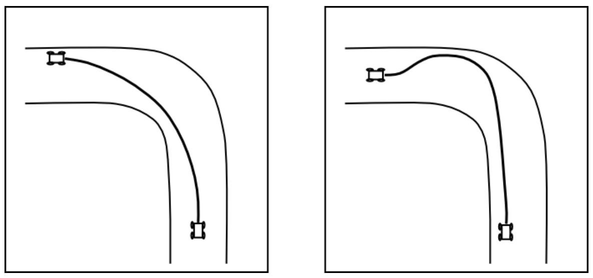
Figure 5. Curvature Profile Optimizing the Time (left) and the Exit Velocity (right)

Theoretically, the trajectory is a continuous function. Practically, it is discretized on the set of points of the centerline where the car is found in each frame of the application. One constraint that arises from the particular setting of this problem is that the trajectory should be fast to compute. Thus, it must allow the car's pilot to communicate with the race server quickly enough for the car not to exit the road or crash against a hard shoulder.

The notion of curvature of a curve is employed widely through the paper and so we shall discuss its definition. For a continuous curve s(t) defined in a multi-dimensional space, the first derivative s'(t) is the tangent to the curve, and the second derivative s''(t), called the curvature, is normal to the curve. Together with their cross-product, they form the Fermat frame. If the curve describes the motion of an object, the first derivative represents the velocity, and the second one is the acceleration. For the current project, we are not interested in the curvature's direction axis, but only in its norm and orientation sign.

In our model, the centerline of the road can be used to represent the curve s(t), where t is the distance from the start of the race track. Since this is part of a simulated environment, the curve is not continuous, but rather composed of a sequence of line segments. We have used the angle between adjacent line segments as a measure of the curvature of the road. As the segments are not directly available, we use the distance measures available from the TORCS sensors to calculate the angle between observed adjacent road segments. A more detailed description of this process can be found in (Vrajitoru, 2018).

The trajectory is computed in three phases also described in (Vrajitoru, 2018). Phase one traverses the entire track and assigns values of 0 (middle of the road) to all the sections where the track is almost straight for long enough, and 0.8 and -0.8 respectively to all the long sections where the road turns in the same direction. The value 0.8 was chosen based on the width of the car, such that car resides completely inside the track at all times. These values are assigned to leave a buffer zone of a few meters at the beginning and end of each continuous section.

Phase 2 traverses the entire trajectory again and connects the pieces of track created during Phase 1. This part uses a linear interpolation of the trajectory values between the end of one section and the beginning of the next one. Phase 3 smooths the resulting trajectory even more by using anti-aliasing techniques. Multiple experiments were done with the trajectory calculating algorithm and variations of parameters such as the value of the curvature for which the road can be considered almost straight or the length of the buffer zones between the different sections of constant trajectory. Figure 6 shows the trajectory computed this way for the E-Track 5 road.

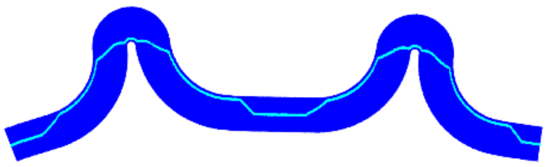
Figure 6. Procedural Trajectory for E-Track 5

Trajectory Straightening and Smoothing

A smoothing algorithm can be used to optimize such a trajectory further. This algorithm comprises of a combination of two steps: a curvature descent, and an anti-aliasing method. The curvature descent method starts by computing the real coordinates of the trajectory points. Then it moves each point on a direction normal to the tangent to the road's centerline in that point, in a direction that goes against the curvature sign, and by an amount proportionate to the curvature value. The movement is capped to the road border. The moved points are converted back to displacement values with respect to the road centerline, which in turn give us the new trajectory values. This is similar to methods used by image processing or vector drawing programs to smooth Bezier curves.

The anti-aliasing method is similar to the blending or smoothing transformation used in image processing to reduce the contrast in a given region. It is in fact a 1-dimensional version of the 2D image transformation. In this transformation, each point of the trajectory is replaced by a weighted average of the point's value with the values of the neighboring points in a given radius. The weights of the neighbors' values decrease with the distance to the transformed point.

These methods are not applied in separate steps, but rather combined through several iterations. For example, the first method can be applied 3 times, followed by the second one applied 10 times, a process which is repeated until no significant changes happen anymore and the trajectory converges towards a given shape. Figure 7 shows the profile of the curve obtained by these two algorithms applied to the procedural trajectory from Figure 6.

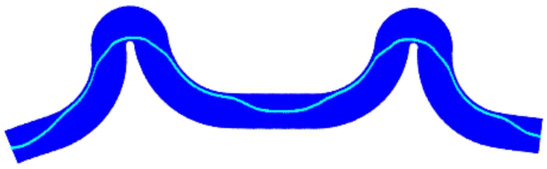
Figure 7. Procedural Trajectory Optimized

Preliminary experiments have shown that the ideal number of steps varies from a road to another, and depend on the starting trajectory. It is also apparent that starting from the procedural trajectory yields better results than starting from a constant trajectory that follows the road's centerline.

Trajectory by Genetic Algorithms

Genetic algorithm have been used for similar problems before, as well as for numerous other applications. Their advantage for such problems is that they can avoid a painstaking process of parameter and algorithm tuning, and they can also take advantage of high performance computing resources. To apply the GAs to this problem, first we notice that the scale of the problem is so big that it might take a very long time to run even one trial. Thus, E-Track is defined by about 3000 curvature points, while E-road by about 20000. To improve the use of computational resources, first we segmented the road in intervals where the trajectory can be computed independently. For this, we selected stretches of road that are almost flat and long enough (at least 10 points for E-Track 5 and 15 for E-Road) and used their middle points as segment delimiters. An example of this segmentation can be seen in Figure 4 marked in red. Note that by this method, the segments end up being longer than for the procedural method. The trajectory can be computed by the GA on each of the segments, which can also lead to an easily parallel implementation. This segmentation aims to optimize the trajectory even between the smaller segments used by the procedural algorithm and take advantage of possible shortcuts that can be taken.

For each segment, we represent the potential trajectory in the chromosome as a set of control points or key frame points equally spaced on the road. Each of them is represented by several genes that compose a real value in the interval [-1, 1], representing the trajectory value. As a reminder, -1 marks the left border of the road, +1 the right border, and in general, the values represent a transversal proportional displacement with respect to the road centreline. Each of these points is represented by the same number of binary genes. Then the actual trajectory is computed by linear interpolation between the control points. We have tried to vary the spacing of these key frame points, as is shown in Section 4. Surprisingly, it seems that a larger spacing between them leads to better performance.

In terms of fitness, the potential candidates are the total length of the trajectory within the segment measured on the reconstructed road and the curvature, either total over the segment or its maximum value. Preliminary tests using these functions have shown the total distance to be the best measure for evolutionary purposes. The goal is to minimize the length or the resulting curve, such that lower values are better. We used a population of size 100, a number of generations limited to 5000, the uniform crossover (Syswerda, 1989) with a probability of swap of 0.3, and a mutation rate of 0.01. Each of the results presented in Section 4 represents an optimum value over 10 different trials with a different seed for the random number generator. The optimal trajectory is selected independently for each segment.

From some of the early experiments on this, we noticed that sometimes undesired side-effects appear from the segmentation process. Thus, the GA may evolve a good trajectory for each segment, but a big skip can happen between segments that can degrade the performance. This is especially true for the start and end of the trajectory. To address this issue, we tried a variation of this method as a boundary value problem, where for each segment the trajectory value is set to 0 (middle of the road) in the delimiting points marking the segments. We call this a bound GA in the next section. As we will see, this has led to performance improvement.

Experimental Results

Figure 8 shows a plot of the total trajectory length for E-Track 5 as a function of the key frame step value for the GA. The orange line represents simple GA runs while the dark blue one represents a GA run with fixed boundary values. The lines for the constant (green), procedural (red) and optimized by straightening and smoothing (gray) trajectories are constant because they do not depend on the key frame points. Since we're trying to minimize the total trajectory length, smaller values in this chart are better.

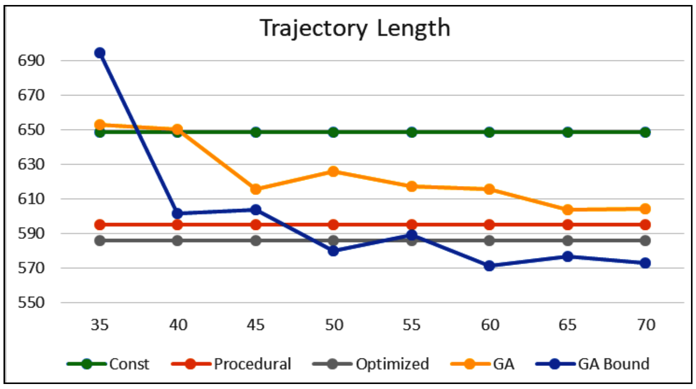

Figure 8. Total Trajectory Length as a Function of the Key Frame Step for E-Track 5

We can see from this figure that the bound GA performs better than the simple one, and that when the key frame step is 60 or more, it outperforms all the other models. For this reason, for the E-road track we only run the bound GA model. Figure 9 shows the total curvature along the track for the same models.

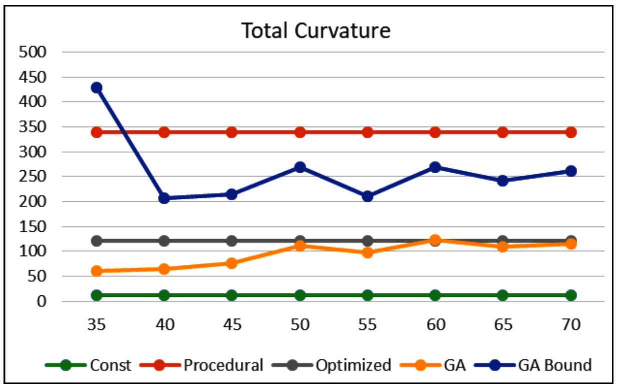

Figure 9. Total Trajectory Curvature as a Function of the Key Frame Step for E-Track 5

Table 1 shows a summary of the best results by GA simple and bound for E-track 5. For each of these runs, we applied the anti-aliasing (AA) or smoothing step at the end, and the rows marked GA+AA and GA Bound+AA reflect the results obtained that way. The number of AA iterations is between 10 and 40. As before, the smoothing transformation is repeated until there is no more visible change. From this table, it appears that the bound GA produces the best result in terms of trajectory length and applying a number of anti-aliasing steps can improve it even further.

Table 1. Best Trajectory Length and Curvature by GA for E-Track 5

Method		Distance	Curvature
Const		648.55	12.21
Procedural		595.09	338.32
Optimized		585.85	120.72
	Step		
GA	65	603.88	122.95
GA+AA	65	571.52	116.26
GA Bound	60	571.35	269.61
GA Bound+AA	60	569.47	122.10

To measure the effectiveness of the trajectories in a race setting, we ran the Meep pilot in TORCS using each of these trajectories. Table 2 shows the results of each of them in terms of total time (seconds) it takes to complete the race and total amount of turning done by the car during the race, measured in radians. We ran the car at a constant speed of 80 km/h. From this table, we can see that all the models performed better than the constant trajectory, but the procedural trajectory still yields the best results and it is matched by the bound GA producing the best trajectory length. This also shows that a shorter trajectory, such as the optimized one, doesn't always allow the car to finish the race faster.

Table 2. TORCS Race Results for E-Track 5 at 80km/h

Method		Time (s)	Total turn
Const		76.79	231.17
Procedural		73.42	199.73
Optimized		73.51	201.38
	Step		
GA+AA best length	65	74.28	224.23
GA+AA best curvature	40	76.09	244.02
GA Bound best length	60	73.42	216.98
GA Bound best curvature	40	73.44	215.30

Figures 10 and 11 show a plot of the total trajectory length and curvature respectively as a function of the key frame step for the track E-Road with a road scale of 0.065. We can see that the GA produces a track of a shorter length than the other methods for a step larger than 70 and that the anti-aliasing addition at the end improves this trajectory even further, both in terms of length and of curvature. In the case of this track, the length and curvature seem to be more correlated than for E-Track 5.

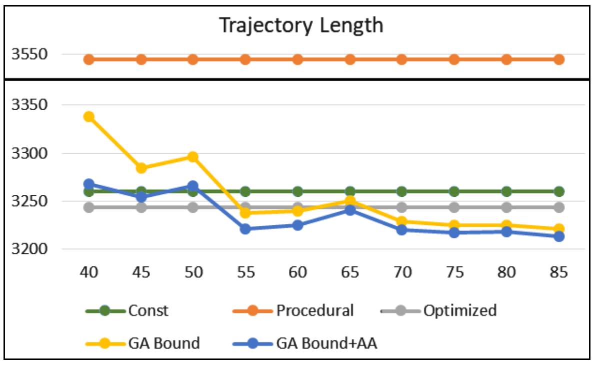

Figure 10. Total Trajectory Length as a Function of the Key Frame Step for E-Road

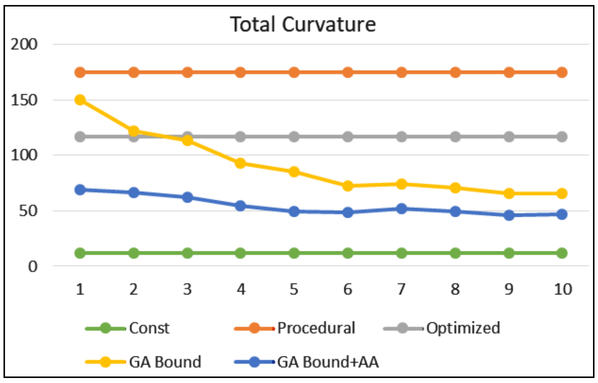

Figure 11. Total Trajectory Curvature as a Function of the Key Frame Step for E-Road

Table 3 shows a summary of the best results by bound GA for E-Road. We also applied the anti-aliasing (AA) or smoothing step at the end for these GA runs. The number of AA iterations was between 50 and 100, based on convergence. From this table, we can see that the bound GA produces the best result again in terms of trajectory length and that the additional anti-aliasing steps at the end have a positive impact. It seems that a larger key frame step is also a better choice for this track.

Table 3. Best Trajectory Length and Curvature by GA for E-Road

Method		Distance	Curvature
Const		3260.52	12.06
Procedural		3544.76	175.10
Optimized		3243.74	116.88
	Step		
GA Bound	85	3221.23	66.05
GA Bound + AA	85	3213.53	46.89

Finally, we examined the influence of the rendered road scale on the quality of the evolved trajectory. Table 4 shows the trajectory length measured on E-Road rendered with a scale of 0.1, where the road scale used by the GA varies. The key frame step here is of 85. From this table we can see that using a scale of 0.1 gives a better result than 0.65.

Table 4. Best trajectory length and curvature by GA for E-Road, measured on a road scale of 0.1

Method		Distance	Curvature
Const		3260.52	18.49
Procedural		3452.98	182.06
Optimized		3184.12	122.56
	R. Scale		
GA Bound	0.065	3181.35	75.37
GA Bound +AA	0.065	3175.25	53.76
GA Bound	0.1	3151.22	88.78
GA Bound +AA	0.1	3142.13	63.94
GA Bound	0.2	3182.43	121.44
GA Bound +AA	0.2	3159.60	87.09

To test these trajectories in race settings, we have run the Meep pilot in TORCS using the best trajectories obtained with the various settings. Table 5 shows these results in terms of total time (seconds) that the car takes to finish the track and in terms of total amount of turning (radians) that the car does during the race. From this table we can see that the procedural trajectory performs the best for this track too, although the bound GA with a road scale of 0.2 is only 2 seconds behind it. Even though the road scale of 0.1 produced a shorter trajectory, the road scale of 0.2 lets the car finish the race faster. This suggest that the car's performance in the race could be further improved by using even higher values for the road scale.

Table 5. TORCS Race Results for E-Road at 80km/h, step 85

Method		Time (s)	Total turn
Const		151.97	605.13
Procedural		146.27	521.01
Optimized		147.79	545.13
	R. Scale		
GA Bound	0.065	150.45	586.86
GA Bound	0.1	149.83	565.93
GA Bound	0.2	148.57	530.21

Overall, it seems that applying the optimization or the anti-aliasing transformation improves the length of the trajectories, but not their performance when tested in TORCS.

Conclusion

In this paper, we presented several methods to build trajectories for autonomous cars in a race setting, including an application of genetic algorithms. We used a road segmentation approach to build a procedural trajectory based on known optimal curves, then a straightening and anti-aliasing optimizing method to improve this trajectory. Finally, using a coarser segmentation, we also applied genetic algorithms to build the trajectory, using trajectory length as the fitness function.

Our experiments showed that the procedural trajectory is the most efficient one in the simulated race settings, but the GA can produce trajectories of a better length. They also showed that using a smaller number of control points is sufficient and can improve the overall performance. An additional factor that can lead to better results is the road scale in the reproduced road used for the trajectory computation, and this has the potential to yield even better results with further exploration.

Acknowledgements

Part of the experiments presented here were performed on the Karst server from the Indiana University Intelligent Infrastructure system.

References

Albelihi, K. and Vrajitoru, D. (2015). An application of neural networks to an autonomous car driver, *Proceedings of the 17th International Conference on Artificial Intelligence*, Las Vegas, NV, 716–722.

Guse, C. and Vrajitoru, D. (2010). The Epic adaptive car pilot. *Proceedings of the Midwest Artificial Intelligence and Cognitive Science Conference*, South Bend, IN, 30–35.

Liniger A. and Lygeros, J. (2015). A viability approach for fast recursive feasible finite horizon path planning of autonomous RC car. *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control, (HSCC '15)*, New York, NY, USA, 1–10.

Marceau, G., Savéant, P., and Schoenauer, M. (2013). Multiobjective optimization for reducing delays and congestion in air traffic management. *Proceedings of the Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'13)*, Amsterdam, The Netherlands, 187-188.

Onieva E. and Pelta, D. A. (2012). An evolutionary tuned driving system for virtual racing car games: The AUTOPIA driver. *International Journal of Intelligent Systems,* 27(3), 217–241.

Padhye, N. (2008). Interplanetary trajectory optimization with swing-bys using evolutionary multi-objective optimization. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'08)*, Atlanta, GA, USA, 1835-1838.

Quadflieg, J. and Preuss, M. (2010). Learning the track and planning ahead in a racing car controller. *Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG10)*, Copenhagen, Denmark, 395–402.

Rathour, S. S., Boyali, A., Zheming, L., Mita, S., and John, V. (2017, August). A Map-based Lateral and Longitudinal DGPS/DR Bias Estimation Method for Autonomous Driving. *International Journal of Machine Learning and Computing*, Vol. 7, No. 4, 67-71.

Schlueter, M. and Munetomo, M. (2018). Massively parallelized co-evaluation for many-objective space trajectory optimization. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'18)*, Kyoto, Japan, 306-307.

Syswerda, G. (1989). Uniform crossover in genetic algorithms. *Proceedings of the International Conference on Genetic Algorithms*, San Mateo (CA): Morgan Kaufmann Publishers.

Velenis E., and Tsiotras, P. (2005). Minimum time vs maximum exit velocity path optimization during cornering. *Proceedings of 2005 IEEE International Symposium on Industrial Electronics*, Dubrovnik, Croatia, 355–360.

Vrajitoru, D. (2018). Global to Local for Path Decision using Neural Networks. *Proceeding of the Pattern Recognition and Artificial Intelligence Conference (PRAI'18)*, ACM International Conference Proceedings Series, Union, NJ, 117-123.

Wymann, B., Dimitrakakis, C., Sumner, A., Espié, E., Guionneau, C., and Coulom, R. (2013). *TORCS, The Open Racing Car Simulator*, v1.3. http://www.torcs.org.