Evolutionary Optimization of Autonomous Vehicle Tracks

J. Ignacio Serrano
Insitituto de Automática Industrial, CSIC.
Ctra. Campo Real, km 0.200
Arganda del Rey. 28500 Madrid (Spain)
nachosm@iai.csic.es

M. D. del Castillo
Insitituto de Automática Industrial, CSIC.
Ctra. Campo Real, km 0.200
Arganda del Rey. 28500 Madrid (Spain)
lola@iai.csic.es

Javier Alonso
Insitituto de Automática Industrial, CSIC.
Ctra. Campo Real, km 0.200
Arganda del Rey. 28500 Madrid (Spain)
jalonso@iai.csic.es

J. E. Naranjo
Insitituto de Automática Industrial, CSIC.
Ctra. Campo Real, km 0.200
Arganda del Rey. 28500 Madrid (Spain)
jnaranjo@iai.csic.es

Abstract- This paper presents a method for the optimization of reference tracks which will be used as maps by an autonomous vehicle. Given a track obtained during a manual driving session by a GPS sensor installed in the vehicle, a reduction of the number of points obtained is needed in order to improve the autonomous tracking and control processing times. It is also needed a noise removal in order to avoid the lateral offset error in automatic controlled driving. The optimization is carried out by an Evolutionary Strategy approach. After an empirical fine tuning of the algorithm parameters, the experiments show that the algorithm is able to provide reference tracks that result very closed to the tracks manually optimized by an expert, and with a similar autonomous driving performance.

1 Introduction

Although totally autonomous vehicles are currently an utopia there exist partially autonomous systems that have obtained surprising results in specific scopes. Examples of the latter are the autonomous cars guided by reference pathway maps. The available autonomous system used for this work consists of a car controlled by a fuzzy logic controller resulting from the Autopia Project[1]. One of the big problems of this kind of systems is related to navigation [12] and thus this is the problem confronted here, concretely the optimization of the pathway tracking. Given an initial path the objective is to obtain another path with much less number of reference points in order to improve controller performance, track drawing and time spent in completing the whole circuit. The task of reducing the number of points in a path has been carrying out by an expert in a manual manner until the development of this work. A kind of Evolutionary Computation(EC) algorithms [6] , called Evolutionary Strategies(ES) [2], has been proposed as the method for automatic optimization. There also exist other works that make use of the EC to solve navigation issues, such as control [1], [3], path planning [15], path planning in cooperation [8] and/or obstacle avoidance [10] and/or real-time adaptation on-line and off-line [14]. Since the optimization process presented here can be viewed as a data mining process rather than a typical path planning, consisting of a reduction and noise removal of preexisting data, the process can be classified as an off-line process. One of the special features of the current optimization problem is that the final solution must keep the spacial structure and thus it shall be a key point of the algorithm design. In [13], the authors deal with a similar problem in the optimization of a road network viewed as a graph, by adding structure constraints to the genetic operators. In the algorithm proposed here the structure constraints are controlled only by the initial population generation and the values of the algorithm parameters, without any other design consideration.

In the next section the autonomous car system is outlined, together with its software simulator and the manual track optimization process. A detailed explanation of the ES algorithm dynamics and parameters is presented in Section 3. Section 4 shows the experiments and results concerning to the algorithm parameter setting and the performance of the algorithm solutions compared to the expert solutions, respectively. Finally, conclusions and future work are commented in Section 5.

2 Autonomous Vehicle

The autonomous system used is a computer-controlled car, from the Autopia Project[1]. The basic dynamics of the system is as follows: the car receives as input a pathway map consisting of a sequence of two dimensional points, each one defined by its North and East geodesic coordinates, and two reference speed values, one for the straight segments and the other for curve segments. At each moment, knowing its current position by a GPS sensor [9], a fuzzy logic controller [4] computes the direction and acceleration/decceleration needed to reach the next point in the input sequence at the input reference speed.

2.1 Car Cinematics Simulator

A software simulator of the autonomous car has been implemented. The simulator uses the same fuzzy logic controller as the actual vehicle and takes into account car cinematics to be as realistic as possible. The simulator takes as input a sequence of 2D points plus two reference speed values. The output is formed by the total, maximum and average

[1]http://www.iai.csic.es/autopia/

lateral error in straight and curve segments, the number of points where the car gets out of road and the time spent in completing the track. Since the simulator knows the road width, the lateral error in each point is the orthogonal distance between the current position of the car and the center of the road. All these output values will be used by the evolutionary algorithm to determine the fitness of the tentative solutions, as explained below. Although the simulation performance is not identical to the actual car performance, it is similar enough to extrapolate its results.

2.2 Track Building

The construction of the input tracks is carried out in a manual manner. First, the expert drives the car manually with the desired speed and tracking. During this session, the GPS sensor obtains the geodesic position of the car with high frequency. At the end, the final sequence of points from the GPS sensor is too large, with many non useful and noise points, what makes necessary an optimization in order not to overflow the controller cycle times and improve the overall performance. Until now, the expert, knowing the car cinematics and the control functioning, manually selected the minimum number of useful points, setting the reference speed values to the average speed values during the manual driving session for straight and curve segments, respectively. Thus, these optimization targets are the ones pursued by the ES algorithm application but in an automatic process.

3 EA Design

Different aspects will be taken into account at EA design, such as individual representations, recombination and mutation operators, selection of individuals to recombine or mutate, fitness function of individuals (solutions), maximum number of generations, population size, etc. There exist different types of EAs depending on the absence/presence and the kind of the former aspects [6] that basically can be enumerated as Genetic Algorithms (GA), Evolutionary Programming (EP), Evolutionary Strategies (ES), Classifier Systems (CFS) and Genetic Programming (GP).

In our case, the objective is to optimize a track automatically obtained from a GPS sensor during a manual driving session, that is composed of too many points, what makes its autonomous driving very slow, carrying a lot of noise, what introduces lateral offset errors in tracking. So, tentative solutions are track maps, as two-dimensional point sequences plus two reference speed values, which can vary in size and whose fitness is a complex function depending on size, lateral offset error and spent time in computer-controlled driving. Since individuals have not a fixed size nor a coded representation the GA approach does not fit well. In addition, a recombination operator is applied and the degradation of population fitness from one generation to the next one is allowed, so this is not an Evolutionary Programming algorithm. The solutions are not immersed in a varying environment that can make them change for adaptation, what makes the presented algorithm not to be a CFS. Thus, an Evolutionary Strategy [2], [11] algorithm has been designed and implemented, considering all the corresponding aspects described next.

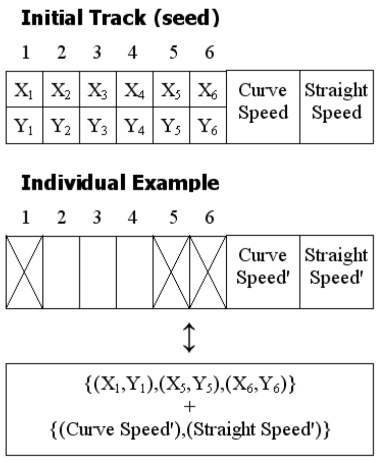

Initial Track (seed)
1 2 3 4 5 6
X_1 X_2 X_3 X_4 X_5 X_6
Y_1 Y_2 Y_3 Y_4 Y_5 Y_6
Curve Speed
Straight Speed
Individual Example
1 2 3 4 5 6
Curve Speed'
Straight Speed'
{(X_1,Y_1),(X_5,Y_5),(X_6,Y_6)}
+
{(Curve Speed'),(Straight Speed')}

Figure 1: Example of EA individual structure and representation.

3.1 Individual Representation

As said before, tentative solutions or individuals are tracks composed of a sorted sequence of 2D points and two reference speed values, for curve and straight segments respectively. Since the objective is the optimization of an existing track, here called seed, it is not logical to include points out of the original track in the individuals. So, solutions are formed by a subset of the initial seed, not necessarily of equal size, plus the speed values. Evolving the reference speeds together with the track points allows to integrate the time spent by the autonomous driving on completing the track into the optimization process. The subset representation ensures the size reduction of the initial track, what means another of the optimization targets. It is important to remark at this moment that the sequence of points is sorted representing a continuous track when driving to one point to the next one. Just this constraint determines, in a high degree, the design of the internal representation of individuals.

Each individual consists of an array of size equal to initial seed size. Each position *i* within the array is a mark that represents whether the individual contains the *i* point of the initial seed or not. Figure 1 shows a naive example of the representation structure. On the other side, if we consider a more intuitive and straight representation, such as a list of points, then the recombination operator, or crossover, may produce bad-formed offsprings that contain transitions from one point to a previous point in the sequence, because the intersection of the appended parts is not empty as consequence mainly derived from size difference. As can be seen below at crossover and mutation sections, the selected representation also benefits the application and design of the evolutionary operators, making it easier and more intuitive.

3.2 Initial Population

The initial population is an important issue in design process of an EA because, as the starting point of the optimum solution search, conditions the quality of the explored pathways and so the final result, and the convergence time. The generation of the initial individuals is not an entirely random process like in other EA approaches. For each new individual a random number of points is selected. However, this number must be comprised within an interval defined by two values. Both, bottom and upper interval boundaries, are manually settled depending on the initial track. So, the individuals will have a maximum and a minimum size.

Afterwards, the concrete points taken from the initial seed that compounds the new individuals are not randomly selected. A random selection could produce a concentrated distribution of the points. For example, and individual may have many of the beginning and ending points and no of the middle ones, what would fall into a track far different from the track to be optimized. To this extent, the points are selected following an uniform distribution by picking a point from every k, being k the integer division between the individual size and the initial track size. In order not to produce identical individuals when they have the same size, this uniform selection is not carried out in an exact manner. Given a selected position as explained before, this can be moved randomly backward or forward with a random offset, never greater than $k/2$ to avoid breaking the order between two consecutive points. With respect to the two reference speed values, the initial individuals will suffer a slight variation from the original values. The variation is small (less than 5%) because the initial values are extracted from real-driving sessions and they are already very tuned. Higher variations may situate the car out of road. It is also interesting to vary the speed values because a reason of adaptation to the optimized sequence of points.

This way, the initial population is generated. One can think that the initial individuals are good enough solutions, making the EA not necessary, because they have a much smaller number of points than the original and keep the drawing. However, the optimal distribution is not necessarily uniform because usually the straights segments need a little number of points to be well-traced and curves need much more points. So, the application of the ES algorithm to the initial individuals is appropriate. The number of individuals that compose the population is an algorithm parameter experimentally determined, as explained later.

3.3 Recombination Operator

Recombination operator, usually called crossover, carries out the mixing of individuals with the expectation of obtaining offsprings better than its parents. The proposed operator in this algorithm is a typical and quite simple one. Two different designs have been proposed, called simple and two-point respectively. Given two parent individuals, the simple crossover selects a random position or split point and produces two new offsprings, composed of the part before the split point of one of the parents and the part after the split point of the other father, and viceversa. On the other hand, the two-point crossover select two split positions so the interchanged parts are the one between the split points and the one outside them, producing two offsprings again. The performance of both kinds of crossover is explained in Experiments section. As can be derived, what the application of this operator actually means is the interchanging of track segments between individuals.

3.4 Individual Selection

Individual selection is responsible of the speed convergence and final reached solution of the algorithm. There exist many ways to do this selection but fitness based is the most intuitive because one can hope that the recombination of two good individuals will result in two good offsprings. Here, three different selection methods have been considered, such as random, direct and roulette. The first selects the individuals randomly, as named. At this point, it is necessary to introduce a new algorithm parameter, the operator application space, i.e., the percentage of population over which the crossover will be applied to in each generation. So, the random selection picks pairs of points until the application space is covered. The other two methods are fitness based. In direct selection, the crossover is applied over each two consecutive individuals when population is decreasingly sorted by fitness values, until completing the application percentage. So, the search is completely guided by the fitness function. However, in roulette selection, although the fitness value plays an important role, a random component takes part in the process. Each individual has a probability of being a crossover target in proportion to its fitness value respect to the total population fitness sum. This way, high fitness individuals have more possibilities of being selected than low fitness. Finally, pairs of individuals are selected until totally covering the application space.

In addition, there exist two selection strategies in ES approaches: plus strategy and comma strategy. In plus strategy the parents participate in the new generation. However, in comma strategy the parents are replaced by their offsprings in the new generation. In this case, the latter is the used mode because plus strategy implies a population size growing through the evolution process and this is not a desired consequence when a fast convergence is wanted.

3.5 Mutation Operator

Mutation slightly vary some individuals. An operator application space is also introduced here, defining the percentage of non-recombined individuals that can be mutated in each generation. The application space is expressed as a probability value so each of the non-recombined individuals has such probability for being mutated. For the current purposes, four different types of mutation have been proposed, three of them related to the points sequence and one related to the speed values. With respect to the points sequence, the proposed variation modes are, adding, moving or removing points. But, how many points will suffer the modifications? This value is determined by a parameter here called mutation variation rate, and it expresses the maximum percentage of points that will be added, moved or deleted in each individual during the mutation process. This process has a random nature so target points are randomly selected. Adding or deleting is a trivial process. However, moving implies the offset of the selected point positions in the global array. The offset sign and value will be determined randomly but, as in individual generation, it must not be greater than half the uniform interval (number of individual points/number of initial seed points) in order not to produce significant changes in the spacial distribution of points. The speed changes consist of the increasing or decreasing of one randomly selected speed value. Since the objective of the former change is to fit the speeds to the sequence of points the variation is small. As said before, the initial speed values are very fitted and high variations may cause the car get much out of road.

Thus, the outline of the mutation operator is as follows: A random number is generated for each of the candidate individuals. If the former number is lower than a certain probability (application space) then the individual is selected. One of the four described types of mutation is randomly selected for being applied. Afterwards, a percentage of individual points is randomly selected, always lower than the mutation variation rate. Finally, mutation is carried out for each of the selected points and the mutated individual replaces the original one within the population.

3.6 Copy Operator

This is a trivial operator. The copy operator simply keeps intact an individual to the next generation. Since not every individual of each generation are targets for the operators and the population size must keep constant, a copy operator is applied to those individuals that are not recombined nor mutated. This way, the recombination leads the search to local or global maximums, the mutation opens new paths that can improve the search or, at least, save it of local maximums, and the copy operator keeps a base that affords a chance to backtracking.

3.7 Fitness Function

The fitness function is a key point in the EA design, because it guides the search to its optimum value. The fitness must reflect the individual goodness, that is, how well a solution performs. In this case, there are three factors to be considered for a good solution: number of points, total lateral error with respect to the road boundaries and spent time for completing the track. These values are obtained as output from a simulation of the evaluated solution, given the sequence of points and the speed values as input to the simulator outlined in Section 2. The lesser the number of points, the lateral error and the time the better the solution. To combine these three factors in a function, a weighted sum is used as seen in Equation 1.

$$Fitness' = w_{np} \cdot \frac{1}{NumberOfPoints} + w_{le} \cdot \frac{1}{LateralError} + w_{st} \cdot \frac{1}{SpentTime} \quad (1)$$

The values are normalized from zero up to one. Since the optimization consists of the fitness function maximization its value is inversely proportional to the normalized factors, that is, the lower the number of points, lateral error and time, the higher the fitness value. The weights are manually adjusted based on an expert criterium and relevance, resulting in 0.45 for points (w_{np}), 0.45 for lateral error (w_{le}) and 0.1 for time (w_{st}). There exists a direct correlation between the number of points and the lateral error so they are same weighted. Since the formers are the main optimization targets the time is the less considered.

$$Fitness = Fitness' - (Fitness' \cdot \frac{NumberOfWaysout}{NumberOfPoints}) \quad (2)$$

There is a component in Equation 2 to go, that has not been described yet. It means a penalization factor depending on the outs of road. This way, the fitness is decremented proportionately to the percentage of points that cause outs of road. This factor is introduced because, despite of a low number of points and lateral error, it is obviously not desirable at all a tracking with way out points.

3.8 Other Parameters

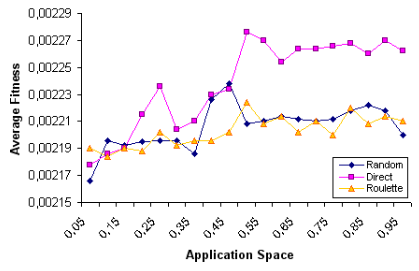
There exists other parameters to be considered in the algorithm design. These are mainly population size, number of generations and elitism. All these parameters together with the operator application spaces and the mutation variation rate are set based on experiments, as explained in the next section.

4 Experiments

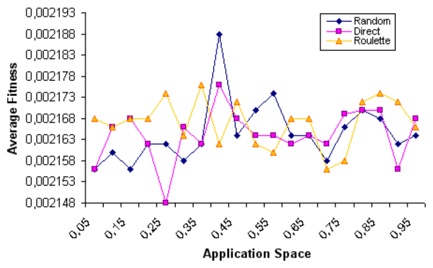
Two classes of experiments have been carried out depending on the target nature. The first is intended to set the EA parameters, whether to select the best option, such as type of individual recombination and selection, or to discover the optimum value, such as application spaces, mutation variation rate, number of generations and population size. The second is intended to prove the validity and performance of the ES solutions against the manual optimization.

4.1 ES parameters setting

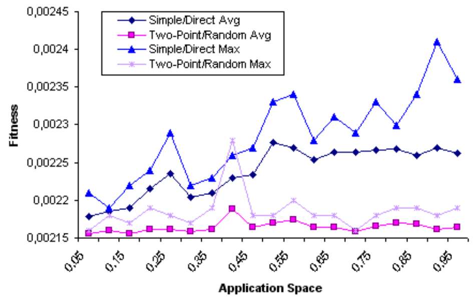
Basically, these experiments consists of running the EA varying the target parameter each time. For each configuration ten executions will be run, and the average and maximum fitness of the best individual in the last generation are the reference values. This way, the first experiments are concerned to the recombination parameters. Considering the simple crossover, the application space has been modified from 0.05 up to 0.95 with an increment of 0.05. This variation has been applied using each of the three proposed selection methods. Results can be seen in Figure 2(a). As can be observed, the best fitness for all the selection methods is reached between 0.4 and 0.6 values of the application space parameter, resulting that the direct selection with a value of 0.5 for the application space is the best performance. Analogously, the same experiments have been done



(a)



(b)



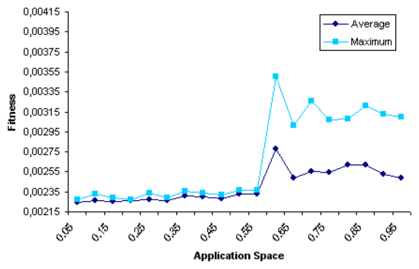
(c)

Figure 2: (a) Average fitness values over ten executions for different selection methods and application space values using simple crossover. (b) Same as (a) but using two-point crossover. (c) Average and maximum fitness values over ten executions between the best configurations using simple and two-point crossover.

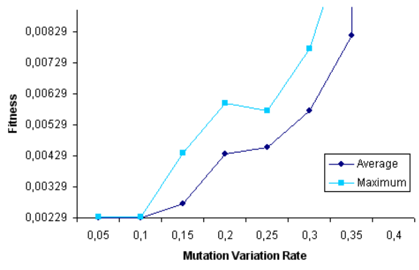
for the two-point crossover, as presented in Figure 2(b). In this case, it does not seem to have a regular behavior, presenting many oscillations for all the selection methods. This means that the two-point crossover has a random nature since it can produce better or worse offsprings with the same probability and, consequently, it is not useful to guide the search. In fact, a comparison on average and maximum results over ten executions between the best configurations using two-point crossover (random selection) and simple crossover (direct selection) is presented in Figure 2(c), resulting in a better and most regular performance of the latter.

The rest of parameters used for these experiments are: a population size of 50, a maximum number of generation of 50, a mutation application space of 0.05 and a mutation variation rate of 0.05. Obviously, the formers parameters are also targets of optimization, but they are taken as an initial reasonable baseline.

Next, analogous tests have been carried out for mutation parameters. Fixing the variation rate to 0.05, the application space has been varied from 0.05 up to 0.95 within intervals of 0.05. Then, using the optimized recombination parame-



(a)



(b)

Figure 3: (a) Average and maximum fitness values over ten executions for different application space values of the mutation operator. (b) Average and maximum fitness values over ten executions for different variation rates values of the mutation operator.

ters and the base values for the rest, the algorithm has been run ten times for each different application space value. Average and maximum results, over ten executions for each value, are presented in Figure 3(a). As can be observed, results seem to improve surprisingly from an application space of 0.6. The analysis of the best individuals with these high fitness reveals that they are bad-formed. The high mutation in the evolution produces individuals that represent almost straight segments, because they concentrate the majority of the points at the beginning and at the end of the original track, so they have a few points and a very low lateral error and, consequently, a high fitness value. However, they are not valid solutions because they are far different from the original track, the target of the optimization. Thus, it seems not to exist, from 0.05 to 0.5 values of application space, any irregularity influencing the quality and the performance of the solutions. Accordingly with the EA nature and the literature, a small value of 0.1 is selected for the application space parameter.

Next, the same test is carried out in order to determine the optimum value of the mutation variation rate, using the parameters values previously optimized. The average and maximum results are showed in Figure 3(b). Analogous to the application space, the spectacular improvement from 0.15 value is due again to the high fitness of bad-formed individuals. It seems clear that as the number of changed points increases the individuals differs more and more from the initial structure, falling into non-valid solutions. So, the variation rate value must be small too, taken as 0.1, for the same reason as the application space.

To conclude with the EA settings, experiments related to population size and number of generations have been

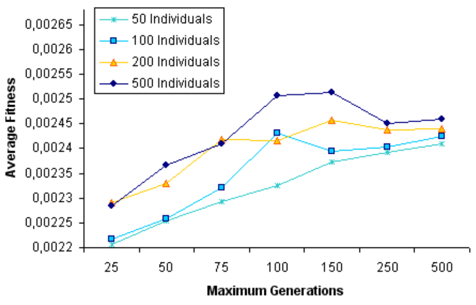


Figure 4: Average fitness values over ten executions for different numbers of generations and population sizes.

done. For different number of generations, such as 25, 50, 75, 100, 150, 250 and 500, the algorithm has run ten times using 50, 100, 200 and 500 individuals as population size. Figure 4 presents the average results. As can be observed, the performance is mostly increasing until 150 generations for all population sizes. From this point the performance decreases or suffers a lower increment, with a tendency to convergence. Thus, the best configuration seems to be 150 generations with a population size of 500. A smaller population size does not afford enough population variety to improve the performance. Therefore, the evolution of the individuals during more than 150 generations does not produce significant improvement even falling into degradation in some cases, such as size values of 200 and 500.

This way, the EA configuration is fine tuned in order to produce the best feasible solutions.

4.2 Track Optimization

The objective of these experiments is to prove the validity of the ES solutions for real usage and the comparison against manually optimized tracks. Ten executions of the algorithm have been run for each tested track, and the solution with best fitness is selected for comparison. The CPU time consumed during the ten runs using the previously optimized configuration values was over five minutes, requiring 5 MB of memory space approximately.

The first example is not a whole circuit but a fragment. It is peculiar because one of the curves it contains. This is an special kind of curve, called clotoid, very typical in many of the real roads. The initial track is composed of 1133 points, with both speed values of 5 km/h. The time spent in its simulation is 147 ms. The expert solution has 38 points and the same speed values, spending 80 ms in simulation. The ES solution results in 39 points with 5.30 km/h and 4.85 km/h for reference speeds, spending 81 ms in simulation. As can be observed in Figure 5, expert track and ES output track in (a) and its corresponding simulations in (b), the ES solution is similar in structure and performance to the expert solution even in the presence of non standard (90 degrees) curves.

For the second example track, the original seed have 2274 points, with a straight speed of 12 km/h and a curve speed of 10 km/h. The time spent by the simulator in completing the circuit was 290 ms. The expert turned this track

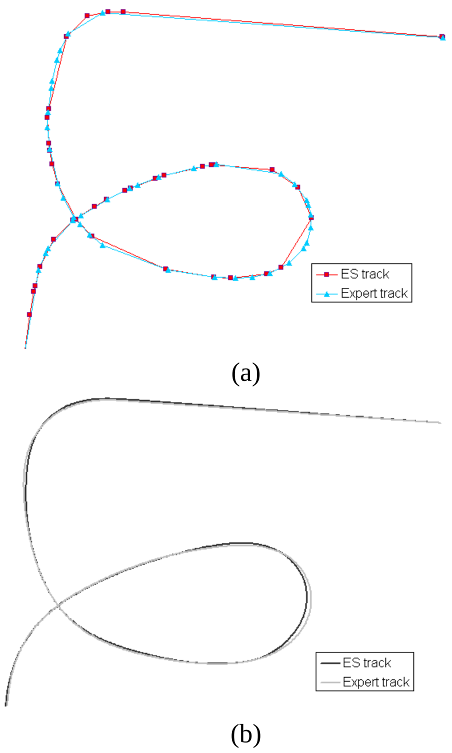


Figure 5: (a) Expert and ES output tracks. (b) Simulation drawing of (a).

into another with 26 points keeping the same speed values, spending 217 ms in its simulation. The best individual over the ten executions of the algorithm resulted in a track with 31 points, straight and curve speeds of 12 km/h and 11 km/h respectively, spending 221 ms in simulation. Figure 6 shows (a) the manually optimized track and (b) its simulation drawing, (c) the best track over ten algorithm executions and (d) its simulation drawing. As can be seen, the algorithm affords an automatic way of optimization with similar performance to an expert solution.

The initial track of the third circuit has 3846 points and 15 km/h and 7 km/h for straight and curve speed values, respectively, spending 402 ms in simulation. The manual solution had 52 points with the same speed values and a simulation time of 286 ms. In this case, the ES solution has 59 points, standing for a straight speed of 15.75 km/h and a curve speed of 7.28 km/h, spending 287 ms in simulation. Figure 7 presents in (a) and (b) the manual track and its simulation respectively and analogously the ES solution and its simulation in (c) and (d). Again, the ES algorithm produces a track with a good performance.

These are three representative examples of the algorithm performance. The ES approach has been also tested over several different tracks in the circuit of Figure 7 with success in all cases.

5 Conclusions and Future Work

An Evolutionary Strategy design an implementation has been presented in this paper in order to optimize the path tracks that an autonomous vehicle takes as input for au-

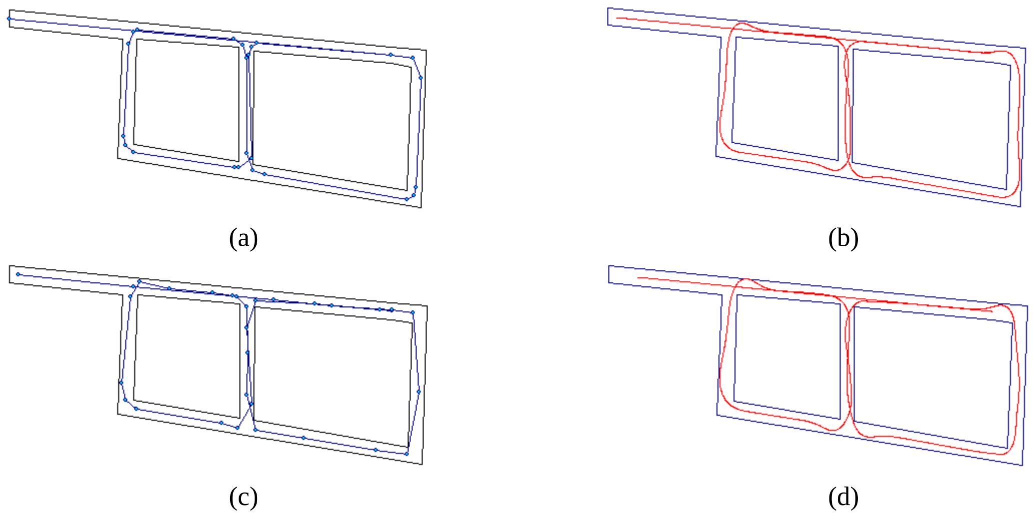


Figure 6: (a) Manually optimized solution. (b) Simulation drawing of (a). (c) Best solution from ten executions of ES algorithm. (d) Simulation drawing of (c).

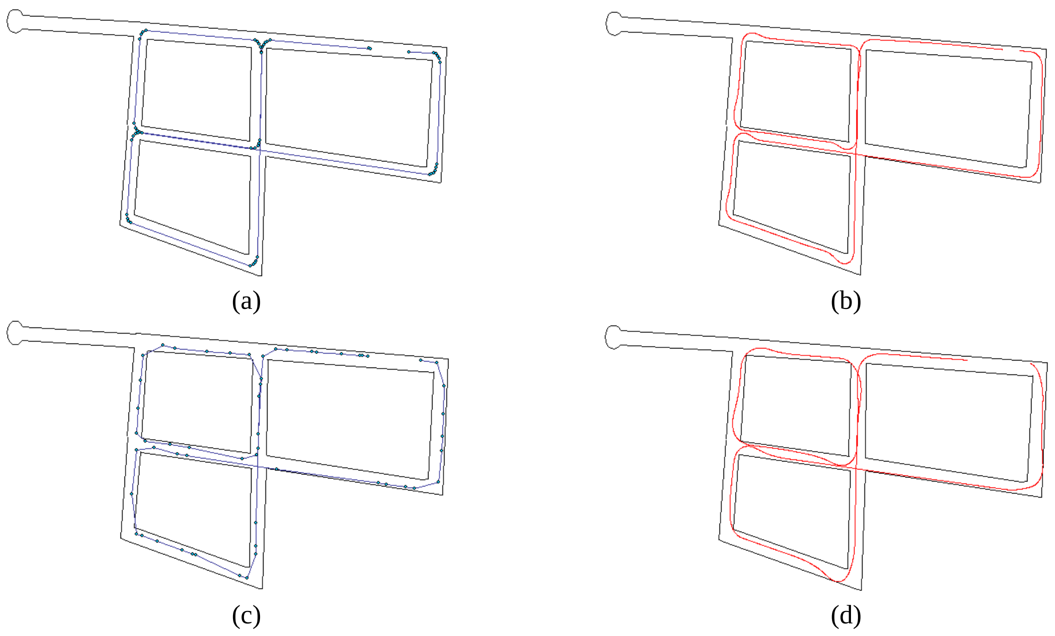


Figure 7: (a) Manually optimized solution. (b) Simulation drawing of (a). (c) Best solution from ten executions of ES algorithm. (d) Simulation drawing of (c).

tonomous driving over a circuit. The ideal track is obtained by a periodical capture of data from a GPS sensor during a manual driving session, but this track has too much points what makes the driving control processing too slow. The process of removing points keeping the minimum number not to vary the track drawing was currently carried out manually by an expert. After an experimental fine tuning of the algorithm parameters, that found out the advantages and disadvantages of the different proposed configurations, the solutions obtained resulted very similar to the expert ones in performance, what makes them suitable for real usage. It is planed to test the algorithm in longer and more sophisticated circuits than the current ones, such as competition circuits, by applying it fragment by fragment, because it seems very difficult to keep the original structure of whole long tracks during the evolution process, and because the algorithm performs better in short fragments when some special kinds of curves are present. It will be also interesting to design and try other types of recombination and mutation operators as well as fitness functions which take into account other interesting factors.

Bibliography

[1] J. Alonso, J. Ignacio Serrano, M. T. De Pedro, C. Gonzalez, and R. Garcia. Optimization of an autonomous car fuzzy control system via genetic algorithms. In *Proc. of the Int. Workshop on Genetic Fuzzy Systems (GFS05)*, 2005.

[2] T. Back, F. Hoffmeister, and H. Schewefel. A survey of evolution strategies. Technical report, Dpt. of Computer Science XI, University of Dortmund, D-4600 , Dortmund 50, Germany, 1991.

[3] M. D. Castillo, J. Gasos, and C. Garcia-Alegre. Genetic processing of the sensorial information. *Sensors and Actuators*, 37-38(A):255–259, 1993.

[4] R. Garcia, T. de Pedro, J. E. Naranjo, J. Reviejo, C. Gonzalez, and J. Revuelto. Fuzzy logic based lateral control for gps map tracking. In *Proc. of the IEEE Intelligent Vehicles Symposium*, pages 397–400. IEEE Catalog, 2004.

[5] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.

[6] J. Heitktter and D. Beasley. *The Hitch-Hiker's Guide to Evolutionary Computation*. Number Issue 8.1. 2000.

[7] J. H. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, 2nd edition, 1992.

[8] D. Kumar and W. Bibel. Path planning for cooperating robots using a ga-fuzzy approach. In *Proc. of the Int. Seminar on Advances in Plan-Based Control of Robotic Agents, Lecture Notes in Computer Science 2466*, pages 193–210. Springer-Verlag, 2001.

[9] K. Monahan and D. Douglass. *GPS Instant Navigation: A practical Guide from Basics to Asvanced Techniques*. Fine Edge Productions, 2000.

[10] A. C. Nearchou and N. A. Aspragathos. Obstacle avoidance control of redundant robots using genetic algorithms. In *Proc. of the 3rd IEEE Mediterranean Symposium on New Directions in Control and Automation*. IEEE, 1995.

[11] I. Rechenberg. *Evolutionsstrategie*. Frommann-Holzboog, 2nd edition, 1994.

[12] A. Saffiotti. *Handbook of Fuzzy Computation*, chapter Autonomous Robot Navigation. Oxford University Press and IOP Press, 1998.

[13] F. Schweitzer, H. Rose, W. Ebeling, and O. Weiss. Optimization of road networks using evolutionary strategies. *Evolutionary Computation*, 5(4):419–438, 1997.

[14] K. Sugihara. Ga-based on-line path planning for SAUVIM. In *Proc. of the Int. Conf. ont Industrial and Engineering Appications of Artificial Intelligence and Expert Systems (IEA/AIE)*, volume 2, pages 329–338, 1998.

[15] K. Sugihara and J. Smith. Genetic algorithms for adaptive motionplanning of an autonomous mobile robot. In *Proc. of the IEEE Int. Symposium on Computational Intelligence Robotics and Automation*, pages 138–146. IEEE, 1997.