Evolution of Driving Agent, Remotely Operating a Scale Model of a Car with Obstacle Avoidance Capabilities

Ivan Tanev
Doshisha University, 1-3 Miyakodani, Tatara, Kyotanabe 610-0321, Japan
itanev@mail.doshisha.ac.jp

Michal Joachimczak
Institute of Oceanology, Polish Academy of Sciences, Department of Genetics and Marine, Biotechnology, 5 Sw. Wojciecha Str., Gdynia 81-347, Poland
mjoach@iopan.gda.pl

Katsunori Shimohara
Doshisha University, 1-3 Miyakodani, Tatara, Kyotanabe 610-0321, Japan
kshimoha@mail.doshisha.ac.jp

ABSTRACT

We present an approach for evolutionary design of an agent, remotely operating a scale model of a car running in a fastest possible way. The agent perceives the environment from a video camera and conveys its actions to the car via standard radio control transmitter. In order to cope with the video feed latency we propose an anticipatory modeling in which the agent considers its current actions based on the anticipated intrinsic (rather than currently available, outdated) state of the car and its surrounding. The agent is first evolved on software models of the car and tracks, and then adapted to the real world. During the adaptation, the lap times improve steadily to the values close to the values obtained from the evolution on the models. An evolutionary optimization of the avoidance of a small obstacle results in lap times that are virtually the same as the best lap times achieved on the same track without obstacles. Presented work can be viewed as a step towards developing a racing game in which the human competes against a computer, both operating scale models of racing cars.

Categories and Subject Descriptors

G.1.6–Global Optimization; J.2-Physics

General Terms

Algorithms, design

Keywords

Anticipatory modeling, driving agent, feedback latency, genetic algorithms

1. INTRODUCTION

The success of the computer playing sport games (like chess [4]) has long served as touchstone of the progress in the filed of artificial intelligence (AI). The expanding scope of applicability of AI, when the latter is employed to control the individual characters (agents) which are able to "learn" the environment and to adopt an adaptive optimal (rather than a priori preprogrammed) playing tactics and strategy include soccer [11], F1 racing [15], etc. [2]. Focusing in the domain of car racing, in this work we consider the problem of designing a driving agent, able to remotely control a scale model of a racing car, which runs in a fastest possible way. Our work is motivated by the opportunity to develop an agent, able to address some of the challenges, which a human driver of racing car faces. In order to provide a fastest laps times around the circuit, the driver needs to define the best driving (racing) line, or the way the car enters, crosses the apex, and exits each of the turns in the circuit. Moreover, realizing the once defined optimal driving line, both the artificial driving agent and the human driver are required to make precise judgment about the state (i.e., position, orientation and velocity) of their car and the environment, and to react quickly and precisely.

The *objective* of our work is an automated design via genetic algorithms (GA) of the functionality of driving agent, able to remotely operate a scale model of racing car (hereafter referred to as "car") running in a fastest way around. The agent should be able to control the car in (i) a consistent way and (ii) to avoid small, static obstacles. An agent with such capabilities would open up an opportunity for building a framework of a novel racing games in which the human competes against a computer with both of them remotely operating scale models, rather than simulated cars. The proposed evolutionary approach could be applied for automated design of the control software of remotely operated vehicles capable to find an optimal solution to various tasks in different environmental situations and conditions.

Achieving the objective implies that the following tasks should be addressed:

(i) Developing an approach allowing the agent to adequately control the scale model of the car addressing the challenge of dealing with the control feedback latency,

(ii) Formalizing the notion of driving style with obstacle avoidance capabilities and defining the key parameters which describe it, and

(iii) Developing an algorithm paradigm for automatic definition of fastest driving lines by setting the key attributes of driving lines to their optimal values.

The related work done by Wloch and Bentley [15] demonstrates the feasibility of applying genetic algorithms for automated optimization of the setup of the simulated racing car. However, neither the adaptation of the driving style to the setup of the car (i.e., a co-evolution of the driving style and the setup of the car) nor the use of a physical (scale) model of a car was considered in their work. Conversely, Togelius and Lucas [13] used scale models of cars in their research to demonstrate the ability of the artificial evolution to develop optimal neuro-controllers with various architectures. However, the implication of the inherent latencies in the video feedback on either the precision or the velocity characteristics of the car was beyond the scope of their work. In our previous work [14] we used an evolutionary approach to optimize the controller of a scale model of a racing car. However, neither the driving consistency nor the avoidance of obstacles or "guardrails", which are relevant for real-world applications, has been discussed in our previous work.

The remaining of the article is organized as follows. Section 2 introduces the hardware configuration used in our work. Section 3 elaborates on the anticipatory model employed by the driving agent to alleviate the detrimental effect of the feedback latency on the performance of the agent. Section 4 discusses the key attributes of driving style with obstacle avoidance capabilities and the proposed approach of genetic algorithms employed to automatically evolve them. Section 5 draws a conclusion.

2. SYSTEM CONFIGURATION

2.1 The Car

In our experiments we choose the 1:24 scaled model of an F1 racing car (hereafter referred to as "car"), with bodywork stripped from logos and repainted for more reliable video tracking (Figure 1).

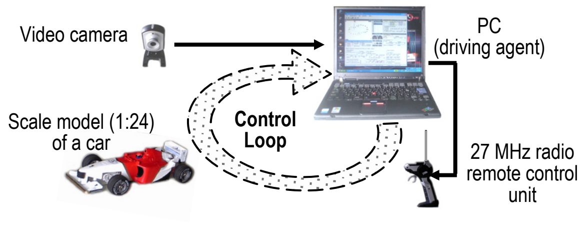


Figure 1. System configuration

This inexpensive (US$20), off-the-shelf car features a simple radio remote control (RC) with functionality including "forward", "reverse", and "neutral" throttle control commands and "left", "right" and "straight" steering controls. The car features neither an onboard speed controller nor analog steering servos. The "analog" control of both the throttle and steering is modeled via pulse-width modulation (PWM) of the controlling signals issued by the driving agent. The car has the following three favorable features: (i) a wide steering angularity, (ii) a spring suspension system in both front and rear wheels, and (iii) a differential drive. The former feature implies a reduced turning angle, and consequently, high maneuverability of the car. The suspension, which is usually designed to cushion the body of the car from the bumps of the track, would be hardly used in its primary destination in our work. Instead, the torsion spring of the rear suspension of the car functions as an elastic buffer, which absorbs the shocks, caused by the sharp and often violent alterations in the torque generated by the car's motor. These torque alterations occur during the PWM of the throttle, by means of which the controlling software regulates the speed of the car within the range from zero to the maximum possible value. In addition, torque alterations occur when the "reverse" throttle command is applied for braking of the car that still runs forward. We suppose that the absorption of the shocks caused by torque alterations is important for the reliability of the car's transmission. The last mentioned feature - differential rear wheels drive (similar to the real cars') implies that that the torque of the motor is split and delivered to the rear wheels in a way that allows them to rotate at different angular speeds if necessary, e.g., under cornering. Therefore, the car turns without a rear wheels spin, which results in a smooth entrance into the turns and a good traction at their exits. The mechanical characteristics of the car are summarized in Table 1. Characteristics #2-#9 are empirically measured from the car running on the considered track surface - a polyvinyl chloride carpet. The minimal velocity of understeer indicates the threshold of velocity at which the front wheels of the turning car start to skid away from the apex of the turn, yielding an increased actual turning radius. In addition to the degradation of the maneuverability of the car, the skidding of the front wheels results in a significant braking momentum. This, in turn, limits the top speed of the turning car (Table 1, parameter #3). Therefore, the understeer reduces the average velocity of the cornering car (due to the lower speed along longer arcs), which has a detrimental effect on the lap time. Moreover, the increased turning radius means that the car might enter the run-off areas or even hit the guardrails on tight corners of the track, which, in turn, might result in a damage of the car or lost of momentum (or both). Consequently, the straightforward strategy of taking corners at full-throttle is unlikely to deliver safe and fast laps.

On the other hand, due to the weight distribution effect, the grip of the rear wheels of the turning car decreases under braking, resulting in an oversteer. Depending on its severity, the car might turn sharper than intended, or even spin out of control. This renders the task of optimizing the driving style of the agent quite challenging, which, in turn, additionally motivated us to consider an automated heuristic approach to address it.

Table 1. Mechanical Characteristics of the Car

Parameter	Value
1) Car: model and scale	Auldey F1, 1:24
2) Max straight line velocity, mm/s (scaled, km/h)	2000 (172)
3) Max turning velocity, mm/s (scaled, km/h)	1400 (121)
4) Min turning radius (without understeer), mm (scaled, m)	300 (7.2)
5) Min velocity of understeer, mm/s (scaled, km/h)	1200 (104)
6) Increase of turning radius due to understeer , mm/(mm/s)	0.5
7) Acceleration on full throttle, mm/s^2	800
8) Deceleration on reverse, mm/s^2	-2000
9) Deceleration on throttle lift-off, mm/s^2	-600

2.2 Perceptions and Actions of the Agent

The agent perceptions are obtained from video camera mounted overhead. The camera features a high definition CCD sensor and lenses with wide field of view (66 degrees), which allows to cover a sufficiently wide area of about 2800mm x 2100mm from an altitude of about 2200mm. In our experiments camera operates at 320x240 pixels mode, with a video sampling interval of about 30ms. The camera is connected to the personal computer (PC) through a PCMCIA-type video capture board.

The agent's actions (a series of steering and throttle commands) are conveyed to the car via standard radio control transmitter operating in 27MHz band. The four mechanical buttons of the transmitter are electronically bypassed by n-p-n-transistor switches (operating in a common emitter configuration) activated by the controlling software. Transistors are mounted on a small board, connected to the parallel (LPT) port of the PC.

2.3 Following Simple Routes

In order to verify the very basic concepts of applying the agency for remote operation of the car, we conducted experiments with the car following saimple routes marked by apexes of the turns. Analogous to the real-world vehicle navigation using GPS waypoints, we assume that the driving agent is a priori aware of the positions of these apexes. They are conveniently set-up via graphical drag-and-drop user interface, which facilitates their augmentation into the scene and uploading of their coordinates into the agent's memory. The three routes are (i) an O-shaped circuit featuring two right, single-apex turns, (ii) 8-shaped circuit with a right and a left, double-apex turns, and (iii) S-shaped circuit with a series of right and left turns.

The agent receives a live video from the camera, tracks the car, computes the current state (i.e., position, orientation and speed) of the car and depending on the values of these parameters issues a series of corresponding throttle- and steering controlling commands. The controlling commands correspond to the very basic, handcrafted functionality needed to follow the route by homing to the apexes of the turns at 20 degrees with a speed of about 800mm/s (scaled speed of about 70km/h). The resulting driving lines, indicated by the traces of the perceived geometrical center of the car on O-, 8-, and S-shaped circuits are shown in Figure 2. As Figure illustrates, the lines (shown in Figures 2a, 2b and 2c) differ significantly from the expected ones (Figures 2d, 2e and 2f). The next section (i) elaborates on the problem causing the discrepancy between the expected driving lines and the really observed ones and (ii) discusses the proposed approach to address it.

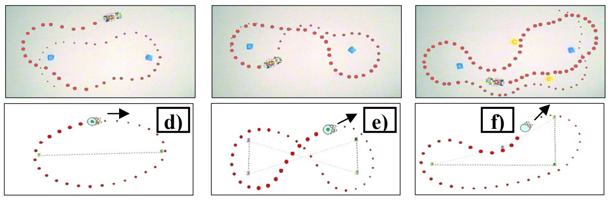


Figure 2. Driving lines of the scaled model of the car (top) and the expected lines of modeled car (bottom), controlled by agent in O-shaped (a and d), 8- shaped (b and e), and S-shaped (c and f) circuits. The real driving lines (a, b, and c) significantly differ from the expected ones on the same circuits (d, e, and f, respectively).

3. ANTICIPATORY MODELING

3.1 Outdated Perceptions

The delays introduced in the feedback control loop (Figure 1) by the latency of the video feed imply that the current actions of the driving agents are based on outdated perceptions, and consequently, outdated knowledge about its own state and the surrounding environment. For the hardware used in our system, the aggregated latency is about 90ms, which results in a maximum error of perceiving the position of the car of about 180mm (scaled error of 4.3m) when the later runs at its maximum speed of 2000mm/s (scaled speed of 172km/h). The latency also causes an error in perceiving the orientation (bearing) and the speed of the car. The cumulative effect of these errors renders the tasks of precisely following simple routes, shown in Figures 2, hardly solvable. The driving lines, shown in Figures 2a), 2b) and 2c), corresponding to the expected lines shown in 2d), 2e) and 2f), respectively, illustrate the detrimental effects of the feedback latency on the precision of control.

3.2 Software Simulator

In order to investigate the detrimental effect of latency on the performance of the driving agent, and to verify the effectiveness of the proposed approach for its alleviation, we developed a software simulation of the car and tracks. The additional rationales behind the development of the software simulation include (i) the possibility to verify the feasibility of certain circuit configurations without the need to be concerned by the risks of possible damage to the environment or the car (or both), and (ii) the opportunity to "compress" the runtime of the fitness evaluation in the eventual implementation of agent's evolution [5][8]. Furthermore, the kernel of the developed simulator is the *internal model* of the car and the environment, which, as elaborated below, is continuously applied by the driving agent in order to anticipate the intrinsic state of the car from currently available (outdated) perceptions. The software simulator takes into consideration the feedback latency of 90ms (3 time steps for the video sampling interval of 30ms), and the Newtonian physics of the car tuned with the concrete values of car's parameters as summarized in Table 1.

3.3 Anticipated Car's State and Environment

The common methods of dealing with latency feedback are based on the idea of either slowing down the flow of control commands in order to allow the feedback to catch-up, or issuing the current control command only after the feedback result of the execution of the previous one have been obtained [7]. However, these methods are not applicable for the considered task of controlling a car due to the relatively high velocities of the latter. In the proposed approach of incorporating an anticipatory modeling [12], the driving agent considers its current actions based on *anticipated* intrinsic (rather than currently available, outdated) state of the car and surrounding environment. The agent *anticipates* the intrinsic state of the car (position, orientation, and speed) from the currently available outdated (by 90ms) state by means of iteratively applying the history of its own most recent actions (i.e., the throttle and steering commands) to the internal model of the car. It further anticipates the perception information related to the surrounding environment, (e.g., the distance and the bearing to the apex of the next turn) from the viewpoint of the anticipated intrinsic position and orientation of the car.

The problem of outdated perceptions in our system is related to the problem of GPS-based vehicle navigation [1]. Each GPS reading provides an absolute position of the vehicle. However, the GPS signal is not always available and in such situation (between tall buildings, in tunnels, under the bridges, etc.) the navigation system has to relay on anticipated car position over some unknown period of time. The position in such case is usually estimated by anticipation (dead-reckoning) utilizing a conventional inertial navigation system with drift sensors. In our system, (i) the absolute and noisy car position is provided from the overhead camera rather than from GPS, and (ii) we are dealing with the history of issued commands over a fixed period of time rather than with the readings of the drift sensors.

3.4 Following Simple Routes with Anticipatory Modeling

The emerged driving lines (superposed over four consecutive laps) of the scaled model of the car are shown in Figure 3. As figure illustrates, the driving lines of the car controlled by an agent employing anticipatory modeling in a system with latency feedback (Figures 3a, 3b, and 3c) are much similar to the expected driving lines of the modeled car with non-latency feedback (Figures 2d, 2e and 2f, respectively). These results experimentally verify the compensatory effect of the anticipation on the performance of the driving agent.

Moreover, the superposition of the driving lines obtained over four consecutive laps demonstrates the consistence of trajectory of the car. Indeed, as Figure 3 illustrates, the maximal lateral deviation of the center of the car (shown as dark trailing circles) from the imaginable average is less than 60% of the width of the car, i.e. less than 55mm. In the most challenging, S-circuit, the small variations in both the actual length of the lap and the average lap speed result in a relative standard deviation of the lap time of about 2.2% (120ms of the average 5300ms). We view this consistence as an important prerequisite for an efficient evolutionary optimization of the driving agent. Such an optimization implies that the lap time (being the fitness value of the evolved driving agent) is the only feedback obtained from the interaction of the agent and the environment. An eventual inaccurate evaluation of the fitness due to inconsistent lap times would result in a noisy fitness, which in turn has a detrimental effect on the *computational effort* (convergence) of the evolutionary algorithms [9]. Because the noisy fitness value can be viewed as a sum of the real fitness plus a random, unbiased (mean of zero) noise component, increasing the number of the laps during the fitness evaluation of the evolving agent would cancel, to some extent, the random noise component and consequently, reduce the noisiness of the fitness. Such an approach, however, would increase the runtime of the fitness evaluation, which, in turn would reduce the *computational performance* of the simulated evolution.

The above-discussed inconsistency of both the driving lines and lap times is caused by the combination of the following inaccuracies:

(i) Inaccuracy of the computer vision: imprecise, noisy determination of the position of the car due to color irregularities (i.e. shadows) in the perceived image of the car and variable foreshortening. This problem is alleviated by Kalman filter [6], which exploits the Newtonian mechanics of the car, as defined in its internal model, and

(ii) Sampling error, caused by the discrete perceptions-actions control loop of the agent with a finite sampling interval.

The small differences between the modeled driving lines (Figure 2d, 2e, and 2f) and the real ones with anticipatory modeling (Figure 3a, 3b and 3c, respectively) is caused by an inaccuracy of the internal model of the car. The perspective distortion of the camera yields a variable scale of the scene (mm per pixel), and consequently, a variation in the perceived mechanical characteristics of the car (acceleration, maximal speed, turning radius, etc.) depending on the actual distance between the camera and the car. Neither the vision subsystem nor the model of the car attempts to correct this distortion.

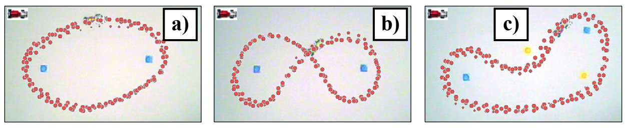


Figure 3. Driving lines superposed over four consecutive laps in O- (a), 8- (b), and S-shaped (c) circuits respectively. The agent employs an anticipatory modeling to compensate the feedback latency. The car, shown in the top left corner of (a), (b) and (c) quantitatively illustrates the scale of the snapshots.

4. EXPERIMENTAL RESULTS

4.1. Attributes of the Driving Style

Before starting discussing the proposed approach of employing an evolutionary approach to automatically evolve the driving styles, we should define the key parameters, which describe them. In our work we consider the driving style as the driving line, which the car follows *around the turns* in the circuits combined with the speed, at which the car travels along this line. Our choice of driving styles' parameters is based on the view of the importance of the way the driver approaches, negotiates, and exits turns, a view that is shared among the racing drivers from various racing teams. We introduce the following key attributes of the driving style of the agent:

(i) Straight-line velocity – the velocity at which the car approaches the turn,

(ii) Turning velocity,

(iii) Throttle lift-off zone – the distance from the apex at which the car begins slowing down from the straight line velocity to turning velocity,

(iv) Braking velocity - the threshold, above which the car applies brakes for slowing down,

(v) Approach angle – the bearing of the apex of the turn. Higher values of this parameter yield wider driving lines featuring higher turning radii.

Viewing the desired values of these attributes as values that the agent tries to maintain, the functionality of the agent can be algorithmically formalized in a way as shown in Figure 4. The usage of the values of the key driving style attributes are underlined in the figure and indicated as "desired". As Figure 4 illustrates, both the orientation (Figure 4, lines 8-11) and the speed (Figure 4, lines 13-16) of the car are continuously adjusted to match the desired values of the corresponding attributes. The open-loop adjustment of the car's velocity (Figure 4, lines 14 and 16) is implemented by macro-commands `ShiftGear(Gear)`, implemented via PWM of the sequence of "forward" and "neutral" throttle commands with duty cycle of 120ms (4 sampling intervals). The possible values of the input parameter `Gear` are 1, 2, 3 or 4, which correspond to the duty ratios of PWM of 0.25, 0.5, 0.75, and 1 respectively. The way the driving agent perceives its own state and the environment (Figure 4, lines 3-5) is illustrated in Figure 5.

```
1. At each time step do begin
2. //--- Perceptions:
3. Obtain the agent's perceptions of car's state: position (P), orientation and speed (V);
4. Obtain the agent's perceptions of the environment:
5.    approach angle (A_A), and distance (A_D) to the current apex
6. //--- Reaction of the agent to the current perceptions
7. //--- Steering control:
8. if (A_A> Desired A_A) and (abs(A_A - Desired A_A)> Desired Threshold A_A)
9.    then SetSteering(Left)
10.   else if (A_A< Desired A_A) and (abs(A_A - Desired A_A)> Desired Threshold A_A)
11.            then SetSteering(Right)   else SetSteering(Straight);
12.//--- Throttle control:
13. if A_D > Desired Throttle Lift-off Zone
14.    then ShiftGear (Desired Straight Line Gear)
15.    else if (V > Desired  Braking Velocity)
16.            then SetThrottle(Reverse)   else ShiftGear( Preferred Turning Gear);
17. end
```

Figure 4. Functionality of driving agent.

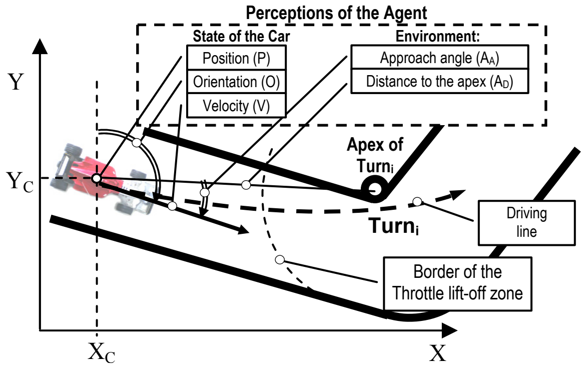


Figure 5. Perceptions of the driving agent.

4.2 Evolving Driving Styles

Assuming that the key attributes of optimal driving style around different turns of a circuit will feature different values, our objective of automatic design of optimal driving styles can be rephrased as an automatic discovery of the optimal values of these parameters for each of the turns in the circuit. This section elaborates (i) on the proposed evolutionary approach for automatic discovery of these optimal values on the software simulator of the car and (ii) on the method used to adapt the evolved solution to the concrete physical characteristics of the real scaled model of the car on the real track.

4.2.1 GA

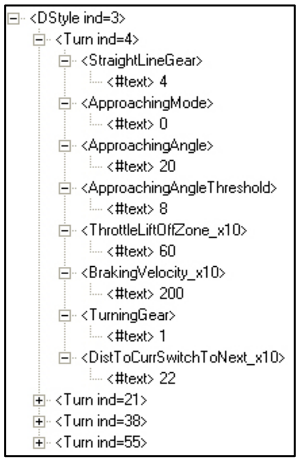
GA [3] is a domain-independent problem-solving approach in which a population of individuals representing the parameters of the candidate solution to the problem (individuals' genotypes) is evolved applying the Darwinian principle of reproduction and survival of the fittest. The fitness of each individual is based on the quality with which the candidate solution is performing in a given environment

4.2.2 Genetic Representation

The genotype in the proposed GA encodes for the evolving optimal values of the key parameters (as elaborated earlier in 4.1) of the driving style for each of the turns on given circuit. In principle the genotype could be represented as a linear chromosome featuring numerical values of the corresponding parameters. However, trying to prototype such linear representation we observed the following difficulties, which required solutions not foreseen in the linear chromosomes in the canonical GA:

(i) Different parameters of the driving style feature *different ranges* of their possible respective values, which need to be individually considered both at the stage of creating the initial population, and during the mutation operation,
(ii) The crossover operation should allow for the *complete set* of driving style parameters associated with particular turn to be swapped with the *complete set* of parameters of another turn in order to protect higher granularity building blocks from the potentially destructive effects of crossover,
(iii) The crossover operation should only allow for the values of the *same attributes* of (either the same or different) turns of the parent's driving styles to be swapped.

The first and third issues suggest the need of a method for maintaining data types in the genetic representation, and performing the mutation and crossover operations in a strongly typed way. Addressing these issues we proposed a strongly-typed GA (STGA), which defines a strongly-typed semantic of the genetic operations in a way, consistent with that of the strongly-typed genetic programming [10]. In order to address the second issue in a generic way we propose a hierarchical, tree-based representation of the genotype as a parsing tree, usually employed in genetic programming. A sample genotype is shown in Figure 6. The main parameters of STGA are shown in Table 2.



```
⊟ <DStyle ind=3>
   ⊟ <Turn ind=4>
      ⊟ <StraightLineGear>
           <#text> 4
      ⊟ <ApproachingMode>
           <#text> 0
      ⊟ <ApproachingAngle>
           <#text> 20
      ⊟ <ApproachingAngleThreshold>
           <#text> 8
      ⊟ <ThrottleLiftOffZone_x10>
           <#text> 60
      ⊟ <BrakingVelocity_x10>
           <#text> 200
      ⊟ <TurningGear>
           <#text> 1
      ⊟ <DistToCurrSwitchToNext_x10>
           <#text> 22
   ⊞ <Turn ind=21>
   ⊞ <Turn ind=38>
   ⊞ <Turn ind=55>
```

Figure 6. Sample genotype represented as XML DOM tree. The sub-tree associated with the values of attributes of the first turn of the circuit is shown expanded.

The sample circuit considered in our experiments of evolving driving styles of the agent operating both the software model and the real scale model of the car is shown in Figure 7. The circuit, which is a real challenge for a human operator, features a combination of one high-speed (3), one medium-speed (1) and two low-speed hairpin turns (2 and 4). Turns are represented in the figure with their respective apexes. The series of turns 4-1-2 form a technical S-shaped sector of right, left, and right turn. The length of the track, measured between the apexes of the turns is about 3800mm. The walls ("guardrails") are virtually introduced in a sense that they are not physically constructed on the track. Consequently, in both cases (simulated and real car), "hitting" the walls in no way effects the dynamics of the car. However, each "crash" is penalized with 0.4s (about 10% of the expected lap time), added to the actual lap time. This reflects our intention to evolve driving styles that avoid the potentially dangerous crashes into the eventual real walls rather than trying to exploit the occasional benefits of bouncing from them.

Table 2. Main parameters of STGA

Category	Value
Population size	100 individuals
Selection	Binary tournament, selection ratio 0.1, reproduction ratio 0.9
Elitism	Best 4 individuals
Mutation	Random sub-tree mutation, ratio 0.01
Trial interval	Single flying lap for the software model and two flying laps for the real car
Fitness	Average lap time in milliseconds
Termination criteria	Number of generations = 40

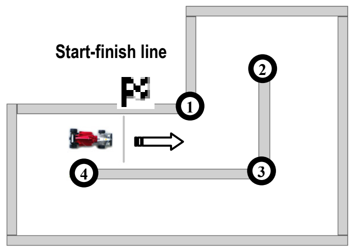


Figure 7. Sample circuit used for evolution of driving style of agent. The circuit is used for experiments with both the (i) software model and (ii) the real scaled model of the car.

4.2.3 Offline Evolution of Driving Styles on a Software Model of the Car

In this experiment we conducted an offline (i.e. on the software model of the car and tracks) evolution employing the proposed evolutionary framework. The results of the convergence of fitness (i.e. the lap time of a single flying lap), aggregated over 50 independent runs of STGA are shown in Figure 8. As figure illustrates, the best lap time average over all runs of STGA improved from 4770ms to about 4200ms (i.e., about 14%) within 40 generations, which for a single run of STGA consumes about 24 minutes of runtime on PC with 3GHz CPU, 512MB RAM and Windows XP OS. For the measured average speed of about 1100mm/s the achieved average reduction of lap time by 570ms corresponds to an advantage of about 63cm (more than 3 lengths of the car) per lap.

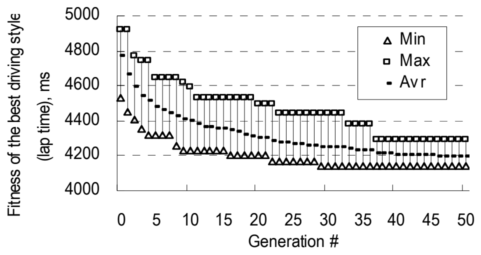


Figure 8. Fitness convergence characteristics of offline evolution of driving styles on a software model of the car.

4.2.4 Porting the Evolved Solution to the Real Car

For the considered task of evolving driving styles, the process of porting the solution, evolved offline on the model to the car can be viewed as a process of adaptation to the changed fitness landscape of the task. In our approach we employ the same STGA framework, as used for offline evolution, for a phylogenetic adaptation of the already obtained set of good solutions to the changes in the fitness landscape caused by switching from the simulated world into the reality. The parameters of the STGA are as shown in Table 2 with the only difference that the population size is set to 20 individuals. At the beginning of the adaptation the STGA is initialized with the 20 best-of-run driving styles obtained from experiments with an offline evolution as elaborated in 4.2.3. In order to address the challenges of (i) guaranteeing an equal initial conditions for the time trials of all candidate solutions and (ii) automatic positioning of the real car before each time trial, we employ a time trial comprising an out-lap followed by a series of flying timed laps, and finally, an in-lap in a way similar to the current qualifying format in the car racing formulas. After crossing the start-finish line (shown immediately after the turn 4 in Figure 7) completing the final timed lap governed by the current driving style, the car enters the in-lap slowing down by implementing the "Neutral" throttle command. Depending on the speed at the start-finish line, the car comes to a rest at a point somewhere between turn 1 and turn 2 (Figure 7). At this point, which can be seen as an improvised team pit, the next driving style which has to be evaluated is loaded into the agent's controller, and the car starts its out-lap. Controlled by the new driving style, the car negotiates turns 2, 3 and 4. During the out lap the car covers a distance from the improvised pit stop to the start-finish line, which is sufficient to cancel any effect of the previously evaluated driving style on the performance of the current driving style. In order to compensate for the eventual inconsistence of the lap time (as discussed in 3.4), a total amount of 2 timed laps are conducted during the time trial of each driving style, and the average lap time is considered as a corresponding fitness value.

The online (i.e., on the scale model of the car) evolution of the initial population of 20 best-of-run solutions obtained offline was allowed to run until no improvement in fitness value of the best driving style have been registered for 4 consecutive generations. A single run has been completed, and improvement of the aggregated (with the time penalty for hitting the "walls") fitness value of the best solution from the initial value 4930ms (due to penalty) to 4140ms (average speed of 1120mm/s, scaled to about 97km/h) has been observed within 10 generations. The emergent features of the evolved best driving style of the anticipatory agent are shown in Figure 9. Figure 9a illustrates the trajectory of the car as it starts its flying lap entering the turn

4 relatively wide and exiting it as close as possible to the apex. As Figure 9b depicts, this allows the car to negotiate the left turn 1 in a way that allows reaching the following right hairpin turn 2 using the shortest possible driving line. The car exits the turn 2 (Figure 9c) in a way that allows for a favorable orientation at the entrance of the following turn 3 (Figure 9d) and an early acceleration well before its apex, contributing to the achievement of the faster speed down the back straight between turns 3 and 4. The car uses the full width of the track and enters the hairpin turn 4 wide (Figure 9a) and exits it close to its apex preparing for the first turn of the next flying lap.

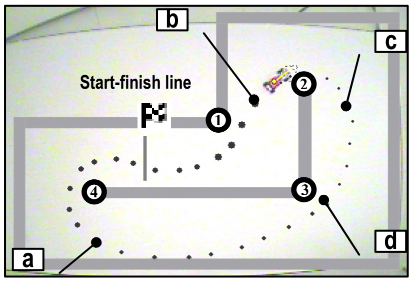
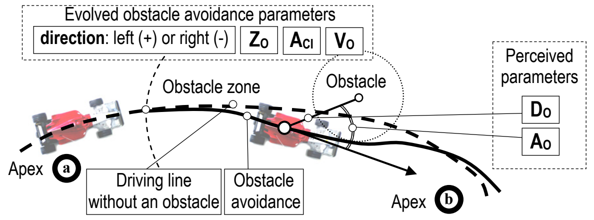


Figure 9. Emergent features of a sample best evolved driving style of the anticipatory agent. The dark trailing circles depict the trajectory of the center of the car. Timestamp interval between each of these circles is 120ms (4 sampling intervals).

4.2.5 Evolution of Optimal Obstacle Avoidance

Obstacle avoidance is a key feature of any mobile robot. However, depending on *what* the characteristics of the obstacle are (large or small, static or moving), *whether* the artifact is a priori aware of it or not, and *when* it is introduced to the scene (before the trial or at runtime), the implementation of obstacle avoidance requires an addressing of numerous algorithmic or technological (or both) challenges. In this very preliminary work we consider the simplest case of a *static* obstacle with *known* properties (position and size), introduced to the scene *before* the time trial. For the considered car-racing domain, the problem of optimal obstacle avoidance can be viewed as discovering the driving line of circumnavigating an obstacle and the speed along this line that result in a minimal lap time around a predefined circuit.

Adopting the repulsive potential field approach of obstacle avoidance, we view the steering the car away from the obstacle as a mechanism of correcting the desired angle of approach (Desired_A_A) of the apex of the following turn. The parameterization of the maneuver is shown in Figure 10. As figure illustrates, the steering correction is initiated when the car enters the obstacle zone. Consonant with the repulsive potential field approach, the degree of this correction depends on the angular distance between the car and the obstacle (Figure 10, parameter A_O) as follows: the correction A_C of Desired_A_A is set to its maximal (initial) value (Figure 10, parameter A_{CI}) when the bearing of the obstacle is minimal (i.e., A_O=0, when the car travels head on into the obstacle). Then the correction A_C decreases inversely proportionally to zero with the increase of the bearing A_O to its maximal value (i.e., A_O=90 degrees when the car is lined-up with the obstacle). In addition to the steering correction, a throttle control is also applied to maintain the desired velocity V_O while negotiating the obstacle. The evolutionary optimization of the obstacle avoiding implies, in addition to the values of general parameters of the driving style (as elaborated in Section 4.1), an automated discovery of the values of the key parameters of the obstacle avoidance maneuver that result in a fastest lap around the circuit. The obstacle avoidance parameters are the direction of avoidance (left or right), radius of the obstacle zone (Z_O), initial correction of the apex approach angle (A_{CI}), and speed inside the obstacle zone (V_O). Notice that due to the interdependence of (i) the general parameters of the driving style and (ii) the obstacle avoidance parameters, in an ultimate case the driving line defined by the former might provide avoidance without the need for an explicit avoidance maneuver.


Evolved obstacle avoidance parameters
direction: left (+) or right (-)
Z_O
A_CI
V_O
Perceived parameters
D_O
A_O
Obstacle zone
Obstacle
Apex a
Driving line without an obstacle
Obstacle avoidance
Apex b

```
Algorithm of steering- and throttle correction for obstacle avoidance:
1. Obtain the evolved apex approaching parameters Desired_$A_A$ and Desired_V;
3. Obtain the perceived $D_O$ and the evolved $Z_O$;
3. if ($D_O$<$Z_O$) then begin                                 // Inside the obstacle zone
3.     Obtain the perceived $A_O$;                              // Bearing of the obstacle
2.     Obtain the evolved direction, $A_{CI}$, and $V_O$;          // Obstacle avoidance parameters
4.     Desired_$A_A$ = Desired_$A_A$ + sign(direction) $A_C$($A_{CI}$, $A_O$); // Steering correction
5.     Desired_V = $V_O$;                                       // Throttle correction
6. end
7. Perform the normal functionality of driving agent;           // Shown in Figure 5
```

Figure 10. Parameterization (above) and the algorithm (below) of obstacle avoidance. Desired_V, Desired_A_A, Z_O, A_{CI} and V_O are the *evolved* (encoded in the genotype) desired (optimal) values of the velocity, apex approach angle, radius of the obstacle zone, initial (at the entrance of obstacle zone) correction to the Desired_A_A and the speed inside the obstacle zone respectively; D_O and A_O are the *perceived* distance to- and bearing of the obstacle; and A_C is the *computed* correction to the Desired_A_A.

The ability of the agent to adapt to a changeable fitness landscape caused by the obstacle introduced in the same circuit as used in Section 4.2.3 (Figure 9) by gradually improving the lap time by simulated evolution is shown in Figure 11. Demonstrated fitness convergence results are averaged over 20 independent runs, where GA is initialized with a population comprising 90 randomly created individuals, plus 10 best-of-run driving styles (with randomly initialized part of chromosome that encodes for the obstacle avoidance parameters) obtained from the experiments as elaborated in Section 4.2.3. A round obstacle with a diameter of 180mm (two car widths) is placed 400mm before the apex of the turn 4 (Figure 9) on the driving lines of these 10 best-of-run styles. Analogous to the collisions with the walls, "hitting" the obstacle has no effect on the dynamics of the car. Rather, the agent is penalized with 0.4s added to the actual lap time. As Figure 11 illustrates, the best lap time average over all runs of STGA improved from 4920ms to about 4200ms (i.e., about 17%), i.e., to the same lap time as in a circuit without an obstacle.

The online evolution of the initial population of 20 best-of-run solutions obtained offline was performed in a way as discussed in 4.2.4. An improvement of lap time of the best solution from the initial value of 5250ms (due to penalty) to 4170ms was observed within 16 generations. The snapshot of the best-evolved driving style is shown in Figure 12.

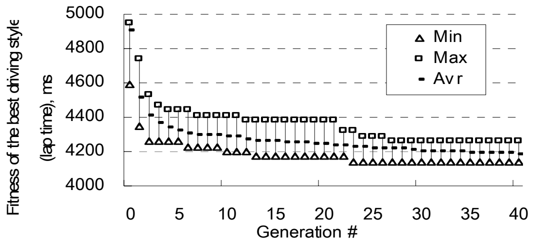


Figure 11. Fitness convergence characteristics of offline evolution of obstacle avoiding driving styles.

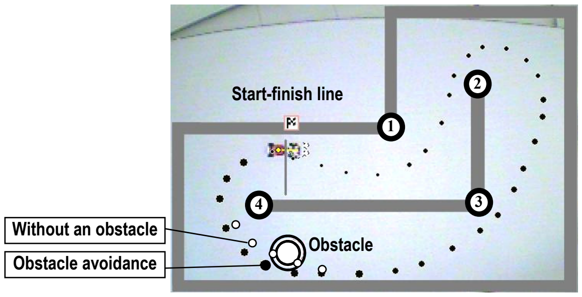


Figure 12. Sample best evolved obstacle avoiding driving style. The dark trailing circles depict the trajectory of the center of the car. Timestamp interval between each of these circles is 120ms (4 sampling intervals).

5. CONCLUSIONS

The objective of this work is an automatic design of the agent, able to remotely operate a fast scale model of a car. The agent's actions are conveyed to the car via RC unit featuring "forward", "reverse", and "neutral" throttle commands and "left", "right" and "straight" steering controls. The agent perceives the environment from an overhead video camera. In order to cope with the inherent video feed latency we proposed and implemented an approach of anticipatory modeling in which the agent considers its current actions based on anticipated intrinsic (rather than currently available, outdated) state of the car and surrounding environment. We formalized the notion of driving style (i.e., driving line with the speed along this line) with obstacle avoidance capabilities and defined the key parameters, which describe it. We demonstrated the feasibility of applying genetic algorithms to evolve the optimal values (i.e., yielding fastest lap times) of these parameters first offline on a software simulator of the car and then online in the real world. During the online evolution, the lap times improved to the values very close to the values obtained from the evolution of the software model of the car. An online evolutionary optimization of the avoidance of a small static obstacle with a priori known properties results in lap times that are virtually the same as the best lap times achieved on the same track without obstacles. Presented work can be viewed as a step towards developing a framework of a racing game in which the human competes against a computer, both of them remotely operating scale models of racing cars.

Bibliography

[1] Abbott, E. and Powell D. (1999) "Land-vehicle Navigation Using GPS", The Proceedings of the IEEE January 1999, vol 87, no 1, pp 145-162

[2] Funge, J. D. (2004) "Artificial Intelligence for Computer Games", Peters Corp.

[3] Goldberg, D. (1989) "Genetic Algorithms in Search, Optimization, and Machine Learning", Addision-Wesley, 1989

[4] IBM Corporation (1997), "Deep Blue", URL: http://www.research.ibm.com/deepblue/

[5] Jacobi, N. (1998) "Minimal Simulations for Evolutionary Robotics", Ph.D. thesis, School of Cognitive and Computing Sciences, Sussex University

[6] Kalman, R. E. (1960) "A New Approach to Linear Filtering and Prediction Problems", Transactions of the ASME-Journal of Basic Engineering, Vol.82, Number Series D, pp. 35-45

[7] Lane, J. C., Carignan, C. and Akin, D (2001) "Time Delay and Communication Bandwidth Limitation on Telerobotic Control", Proc. SPIE Mobile Robots XV and Telemanipulator and Telepresence Technologies VII, SPIE, pp.405-419

[8] Meeden, L. and Kumar, D. (1998) "Trends in Evolutionary Robotics", Soft Computing for Intelligent Robotic Systems, edited by L.C. Jain and T. Fukuda, Physica-Verlag, New York, NY, 1998, pp.215-233

[9] Miller, B. L., & Goldberg, D. E. (1995). Genetic algorithms, tournament selection, and the effects of noise. Complex System, 9(3), 193-212.

[10] Montana, D. (1995) "Strongly Typed Genetic Programming", Evolutionary Computation, Vol.3, No.2, pp.199-230

[11] Robocup (2005) URL: http://www.robocup.org/02.html

[12] Rosen, R. (1985) "Anticipatory Systems", Pergamon Press

[13] Togelius J., and Lucas S. M. (2005) "Evolving Controllers for Simulated Car Racing". Proceedings of IEEE Congress on Evolutionary Computations (CEC-2005), Edinburgh, UK, September 2-5, 2005, pp.1906-1913

[14] Tanev, I., Joachimczak, M., Hemmi H. and Shimohara, K. (2005) "Evolution of the Driving Styles of Anticipatory Agent Remotely Operating a Scaled Model of Racing Car", Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC-2005), Edinburgh, UK, September 2-5, 2005, pp.1891-1898

[15] Wloch, K. and Bentley, P. (2004) "Optimizing the Performance of a Formula One Car Using a Genetic Algorithm", Proceedings of the 8th International Conference on Parallel Problem Solving from Nature, Birmingham, UK, September 18-22, pp.702-711