Evolving Driving Agent for Remote Control of Scaled Model of a Car

Ivan Tanev
Doshisha University, 1-3 Miyakodani, Tatara, Kyotanabe 610-0321, Japan
ATR Network Informatics Laboratories, 2-2-2 Hikaridai, "Keihanna Science City",Kyoto 619-0288, Japan
itanev@mail.doshisha.ac.jp

Michal Joachimczak
Gdansk Technical University
11/12 Narutowicza Str.
Gdansk 80-952, Poland
mjoach@atr.jp

Hitoshi Hemmi
ATR Network Informatics Laboratories,2-2-2 Hikaridai, "Keihanna Science City", Kyoto 619-0288, Japan
hemmi@atr.jp

Katsunori Shimohara
ATR Network Informatics Laboratories,2-2-2 Hikaridai, "Keihanna Science City", Kyoto 619-0288, Japan
katsu@atr.jp

ABSTRACT

We present an approach for automatic design via genetic programming of the functionality of driving agent, able to remotely operate a scale model of a car running in a fastest possible way. The agent's actions are conveyed to the car via standard radio control transmitter. The agent perceives the environment from a live video feedback of an overhead camera. In order to cope with the inherent video feed latency we propose 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in which the agent considers its current actions based on anticipated intrinsic (rather than currently available, outdated) state of the car and its surrounding. The driving style of the agent is first evolved offline on a software simulator of the car and then adapted online to the real world. Experimental results demonstrate that on long runs the agent's-operated car is only marginally (about 5%) slower than a human-operated one, while the consistence of lap times posted by the evolved driving agent is better than that of a human. Presented work can be viewed as a step towards the development of a framework for automated design of the controllers of remotely operated vehicles capable to find an optimal solution to various tasks in different traffic situations and road conditions.

Categories and Subject Descriptors

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

General Terms

Algorithms, design

Keywords

Anticipatory modeling, genetic programming

1. INTRODUCTION

The success of the computer playing sport games (like chess [3]) 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 [9], F1 racing [12], etc. [1]. 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. For technical circuits, featuring series of tight turns where the way the car enters a particular turn dramatically depends on the way it exits the previous one, the task of obtaining the best line is quite difficult even for experienced drivers without a certain amount of practice (learning) laps. 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 automatic design via genetic programming (GP) 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. Our ultimate long-term objective is to develop a framework for automated design of the control software of remotely operated vehicles capable to find an optimal solution to various tasks in different traffic situations and road 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 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 remaining of the paper is organized as follows. Section 2 introduces the main characteristics of the hardware used in our system configuration. Section 3 elaborates on the anticipatory model employed by the driving agent as an approach to compensate the detrimental effect of the feedback latency on the performance of the agent. Section 4 discusses the key attributes of driving style and the proposed approach of genetic programming employed to automatically evolve them. Section 5 draws an conclusion.

2. SYSTEM HARDWARE

2.1 The Car

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

Figure 1. Scale model of the F1 racing car

This inexpensive, off-the-shelf car features simple remote control (RC) with functionality including "forward", "reverse", and "neutral" throttle control commands and "left", "right" and "straight" steering controls. The car has the following three favorable features: a wide steering angularity, a spring suspension system in frond and rear wheel, and 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 considered model 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 pulse width modulation (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" command is applied for braking of the car, which still runs forward. We anticipate that the absorption of the shocks caused by torque alterations of PWM is relevant for the reliability of the car's transmission. The last mentioned feature - differential rear wheels drive (similar to the real cars') implies that the car turns without a rear wheels spin which results in smooth entrance into the turns and good traction at their exits.

The mechanical characteristics of the car are summarized in Table 1. Characteristics 3)-10) are empirically measured from the car running on the considered surface - a synthetic short-pile carpet. The minimal velocity of *understeer* indicates the velocity at which the front wheel of the turning car start to skid away from the apex of the turn, yielding an increased actual turning radius. This effect however is observed only when the car turns with engaged "forward" and "neutral" throttle control commands. Due to the weight distribution effect observed under braking, the grip levels of the front wheels of the turning car increase, yielding a smooth turning with nominal turning radius without experiencing an understeer. On the surface used in our experiments, the car indicated no signs of *oversteer*.

Table 1. Characteristics of the Car

Parameter	Value
1) Car Model	Auldey F1
2) Scale	1:24
3) Max straight line velocity, mm/s	2000
4) Max turning velocity, mm/s	1400
5) Min turning radius, mm	300
6) Min velocity of understeer, mm/s	1600
7) Increase of turning radius due to understeer , mm/(mm/s)	0.4
8) Acceleration on full throttle, mm/s^2	1200
9) Deceleration on reverse, mm/s^2	-1200
10) Deceleration on throttle lift-off, mm/s^2	-800
11) Type of the motor	Mabuchi FA-130
12) Normal voltage, V	1.5 – 3.0
13) Normal current, A	0.5
14) Normal torque, g.cm	4
15) Speed at normal torque, rpm	8.600

2.2 Video Feedback

The agent perceptions are obtained from live video feedback of video camera mounted overhead. We use the Creative NX Ultra Web-Camera, which features a high definition CCD sensor and wide field of view (78 degrees) lenses, which allows to cover an area of about 3200mm X 2400mm from an altitude of about 2200mm. Camera operates at 320x240 pixels mode in our experiments. The refresh rate is 40fps. Camera is connected to the USB port of PC.

2.3 Radio Control of the Car via PC

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

2.4 Following Simple Routes – First Experiments

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 sample routes marked by apexes of the turns in three circuits featuring different complexity. These routes are as follows:

(i) 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 driving agent receives a live video feedback from the Web 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/or steering controlling commands. The controlling commands correspond to the very basic, handcrafted functionality of following the route by homing at the apexes of the turns at 20 degrees with speed of about 1500mm/s. The resulting driving lines, indicated by the traces of the detected center of the car on O-, 8-, and S-shaped circuits are shown in Figure 2. As Figure illustrates, the emerged driving lines (shown in Figures 2a, 2b and 2c) differ dramatically from the expected ones (Figures 2d, 2e and 2f). The next section elaborates on the underlying reason for the discrepancy between the expected driving lines of the agent and the really observed ones.

3. ANTICIPATORY MODELING

3.1 Outdated Perceptions

The latencies introduced in the feedback control loop (e.g., by the video feed latency, video feed sampling interval, reaction time of actuators, etc.) imply that the current actions of the driving agents are based on outdated perceptions, and consequently, outdated knowledge about its own state (position, orientation, and speed) and the surrounding environment. For the hardware used in our system, the accumulated latency is about 200ms, which results in a maximum error of perceiving the position of the car of about 400mm when the later runs at its maximum speed of 2000mm/s. This latency is also associated with an error in perceiving the orientation (bearing) and speed of the car. The cumulative effect of these errors renders the straightforward tasks of following simple routes, as shown in Figures 2, virtually insolvable. The driving lines, shown in Figures 2a), 2b) and 2c), conversely to the expected lines shown in 2d), 2e) and 2f) represent the cumulative effects of feedback latency when the car is controlled by driving agent which considers its current actions based on outdated perceptions.

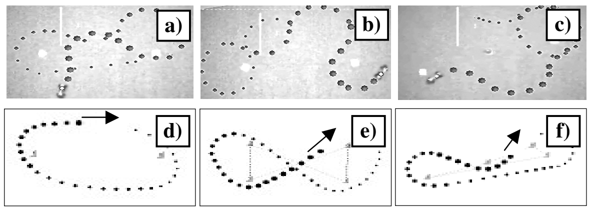


Figure 2. Driving lines of the real scaled model of the car (top) and the expected (bottom) lines of the car, controlled by agent following the apexes of the turns in O-shaped (a and d), 8- shaped (b and e), and S-shaped (c and f) circuits. The real driving lines (a, b, and c) dramatically differ from the expected ones on the same circuits (d, e, and f). The arrows in d), e) and f) indicate the running direction of the car.

3.2 Software Simulator

In order to investigate the effect of latency on the performance of the driving agent, and to verify the effectiveness of the eventually proposed approaches for its compensation, we developed a software simulation of the car and circuits. The additional rationales behind the development of the software simulation can be summarized as (i) possibility to verify the feasibility of certain circuit configurations without the need to be concerned by the risks of possible damage to the environment and/or the car, and (ii) an opportunity to "compress" the runtime of fitness evaluation routine (the most time-consuming part) in the eventual implementation of agent's evolution [4][7]. Furthermore, as elaborated below, the developed simulator comprises the kernel of the internal model of the car and the environment, which the driving agent continuously applies in order to anticipate the intrinsic current state of the car based on currently available outdated perceptions. The software simulator takes into consideration the feedback latency of 200ms (8 time steps for sampling interval of 25ms), the physics of the car and the concrete values of mechanical parameters of the car (on the surface used in our experiment) as summarized in Table1.

The driving lines of simulated car controlled by driving agent without- and with simulated feedback latency of 200ms are shown in Figure 3.

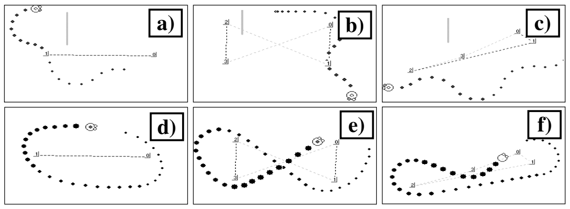
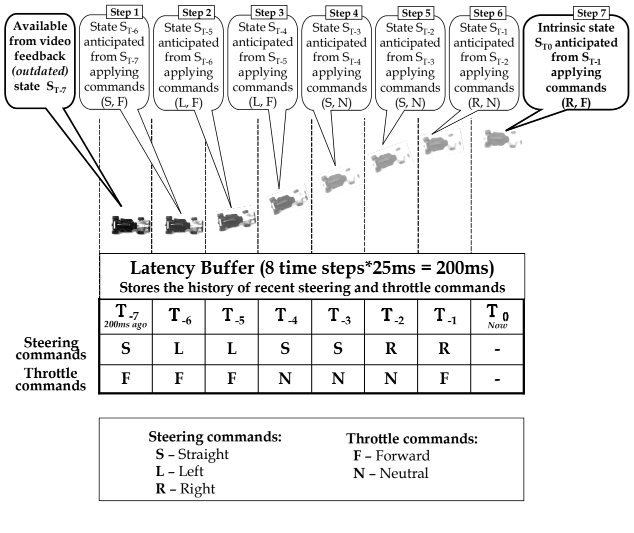


Figure 3. Driving lines of the simulated car controlled by agent following the apexes of the turns with- (top) and without (bottom) feedback latency of 200ms in O- shaped (a and d), 8- shaped (b and e), and S-shaped (c and f) circuits. The behavior of the simulated car with feedback latency (a, b, and c) is much similar to the behavior of the real scaled model of the car on the same circuits, shown in Figures 2a, 2b, and 2c.

As Figure indicates, the characteristic behavior of the simulated car with feedback latency of 200ms in all three sample circuits (Figures 3a, 3b, and 3c) is quantitatively much similar to the behavior of the real scaled model of the car on the same circuits (Figures 2a, 2b, and 2c), which experimentally supports the conclusion about the significant detrimental effects of the feedback latency on the performance of the driving agent. In both the simulated and the real car the agent is unable to control the car adequately, and the car cannot follow the routes marked by apexes of the turns. Moreover, in all 6 cases considered the agent crashes the car into the fences of the circuits.

3.3 Anticipated Intrinsic State of the Car and Anticipated 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 [6]. However, these methods are not applicable for the considered task of controlling a scaled model of racing car due to the relatively high velocities of the controlled object. In the proposed approach of incorporating an anticipatory modeling [10], the driving agent considers its current actions based on *anticipated* intrinsic (rather than currently available, outdated) state of the car and surrounding environment. As illustrated in Figure 4, the agent *anticipates* the intrinsic state of the car (position, orientation, and speed) from the currently available outdated (by 200ms) state by means of iteratively applying the recorded history of its own most recent (during the last 200ms) actions (i.e., the throttle and steering commands) to the internal software model of the car.


Available from video feedback (*outdated*) state S_{T-7}
Step 1
State S_{T-6} anticipated from S_{T-7} applying commands (S, F)
Step 2
State S_{T-5} anticipated from S_{T-6} applying commands (L, F)
Step 3
State S_{T-4} anticipated from S_{T-5} applying commands (L, F)
Step 4
State S_{T-3} anticipated from S_{T-4} applying commands (S, N)
Step 5
State S_{T-2} anticipated from S_{T-3} applying commands (S, N)
Step 6
State S_{T-1} anticipated from S_{T-2} applying commands (R, N)
Step 7
Intrinsic state S_{T0} anticipated from S_{T-1} applying commands (R, F)

Latency Buffer (8 time steps*25ms = 200ms)
Stores the history of recent steering and throttle commands

	T_{-7} *200ms ago*	T_{-6}	T_{-5}	T_{-4}	T_{-3}	T_{-2}	T_{-1}	T_0 *Now*
Steering commands	S	L	L	S	S	R	R	-
Throttle commands	F	F	F	N	N	N	F	-

Steering commands:
S – Straight
L – Left
R – Right

Throttle commands:
F – Forward
N – Neutral

Figure 4. Anticipatory modeling: driving agent anticipates the intrinsic state of the car (position, orientation, and speed) from the currently available outdated state by means of iteratively applying the history (recorded in the latency buffer) of its own most recent actions (i.e., the steering and throttle commands) to the internal software 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.

3.4 Following Simple Routes with Anticipatory Modeling

The emerged driving lines of the simulated car and the real scaled model of the car are shown in Figure 5. As figure illustrates, the behavior of the simulated car with feedback latency and anticipatory modeling (Figures 5a, 5b, and 5c) is much similar to the behavior of the real scaled model of the car on the same circuits (Figures 5d, 5e, and 5f). Moreover, it is virtually identical to the behavior of the simulated car without feedback latency (Figures 3d, 3e and 3f), which experimentally verifies the compensatory effect of applying an anticipatory model on the performance of the driving agent with latency feedback.

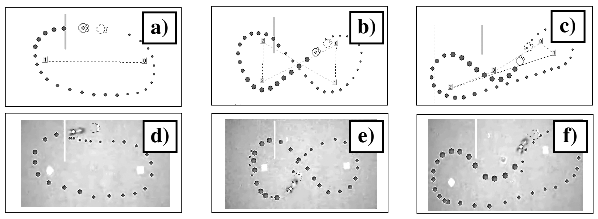


Figure 5. Driving lines of the simulated (top) and the real (bottom) car controlled by agent in a system featuring a feedback latency of 200ms and anticipatory modeling in O-shaped (a and d), 8- shaped (b and e), and S-shaped (c and f) circuits respectively.

4. EXPERIMENTAL RESULTS

4.1. Attributes of the Driving Style

Before starting discussing the proposed approach of employing genetic programming to automatically evolve the driving styles, we should define the key parameters, which describe them. In our work we considering the notion of 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 negotiates turns, a view which is shared among the racing drivers from various F1 racing teams. Indeed, the driver is trying to pass the turn at maximum possible speed, in which the centrifugal forces do not excess the grip provided by the interaction of the tires and the surface. The centrifugal forces depend inversely proportional to the turning radius. Depending on the characteristics of the circuit however, higher turning radius, which result in lower centrifugal forces and higher turning speed may increase the actual distance the car travels around the circuit, which in turn may result in longer lap times. Moreover, the way drivers negotiates particular turn depends on the characteristics of the following section of the circuit, as a higher turning radius, and thus higher exit speed might be important in cases when the turn is followed by long straight where the car can reach its maximum speed. However, higher turning radius and consequently, running wide at the exit of the turn might negatively affect the trajectory of approaching the following turn incases when both turns tightly follow each other in a form of chicane. 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. When the velocity is below this threshold, the car slows down by turning to neutral (or lifting-off the throttle),
(v) Approach angle – the bearing of the apex of the turn. Higher values of this parameter yield wider driving lines featuring higher turning angles.

Viewing these attributes as key attributes of the driving style, the functionality of agent exhibiting a given driving style can be algorithmically formalized in a way as shown in Figure 6.

```
1. At each time step do begin
2. //--- Perceptions:
3. Obtain the agent's perceptions of its own state:
4.   position (P), orientation (O) and velocity (V);
5. Obtain the agent's perceptions of the environment:
6.   approach angle ($A_A$), distance ($A_D$) to the current apex,
7.   and whether the car is inside or outside the preferred
8.   throttle lift off zone;
9. //--- Reaction of the agent to the current perceptions
10. //--- Steering control:
11. if ($A_A$> Preferred $A_A$)
12.     and (abs($A_A$ – Preferred $A_A$)>Preferred Threshold $A_A$)
13.  then SetSteering(Left)
14.  else  if ($A_A$< Preferred $A_A$)
15.      and (abs($A_A$ – Preferred $A_A$)> Preferred Threshold $A_A$)
16.   then SetSteering(Right)
17.   else SetSteering(Straight);
18.//--- Throttle control:
19. if Car is outside Preferred Throttle Lift-off Zone
20.   then ShiftGear (Preferred Straight Line Gear)
21.   else begin
22.          if V > Preferred_Braking Velocity
23.            then SetThrottle(Reverse)
24.            else ShiftGear( Preferred Turning Gear);
25.        end;
26. end
```

Figure 6. Functionality of driving agent.

The usage of the hypothetical optimal (i.e. yielding a fastest lap times) values of the key driving style attributes are underlined in the figure and indicated as "Preferred". As Figure illustrates, both the orientation and the velocity of the car are continuously adjusted to match the preferred values of the corresponding attributes. The open-loop adjustment of the car's velocity (Figure 6, lines 20 and 24) is implemented by macro-commands `ShiftGear(Gear)`, implemented via PWM of the sequence of "forward" and "neutral" throttle commands with duty cycle of 150ms. The possible values of the input parameter `Gear` are 1, 2, 3 or 4, which correspond to the duty ratios of the underlying PWM of 0.5, 0.66, 0.73, and 1 respectively.

The way the driving agent perceives its own state and the environment (Figure 6, lines 2-8) is illustrated in Figure 7.

4.2 Evolving Driving Styles

Assuming that the key attributes of optimal driving style around a particular turn of particular 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 on the proposed approach of employing genetic programming for automatic discovery of these optimal values on software simulator of the car and on the method used to adapt the evolved solution to the concrete characteristics of the real scaled model of the car on the real track.

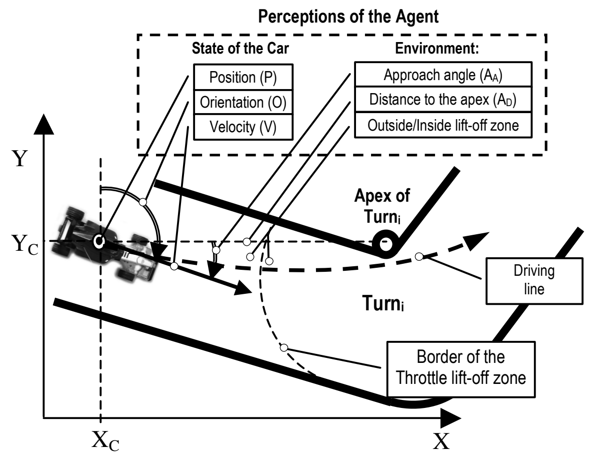


Figure 7. Perceptions of the driving agent.

4.1.1 GP

GP [5] is a domain-independent problem-solving approach in which a population of computer programs (individuals' genotypes) is evolved to solve problems. The simulated evolution in GP is based on the Darwinian principle of reproduction and survival of the fittest. The fitness of each individual is based on the quality with which the phenotype of the simulated individual is performing in a given environment.

4.1.2 Genetic Representation

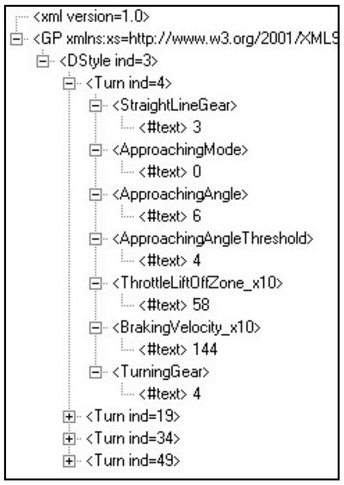
The genotype in GP encodes for the evolving optimal values of the key parameters (as elaborated earlier in 4.1) of the driving style for each of the turns of given racing circuit. In principal, the genotype could be represented as a linear chromosome featuring numerical values of the corresponding parameters, and genetic algorithms [2] (rather than GP) could be applied for automatic discovery of these values. However, trying to prototype such linear representation of the genotype we observed the following difficulties which required solutions not foreseen in the canonical implementations of genetic algorithms:

(i) Variable length representation problem: because different racing circuits feature different number of turns, implying *variable length* of genetic representation,
(ii) Different driving style parameters feature *different ranges* of their possible respective values which need to be individually considered at the stage of creating the initial population, and during the mutation operation too,
(iii) 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,

(iv) The crossover operation should allow for the values of the *same attributes* of *different turns* to be swapped.

The issues (i) and (iii) might be addressed in generic way by a variable length hierarchical representation of the genotype such as a parsing tree, usually employed in GP. The second and fourth issues suggest the need of method for maintaining data types in the genetic representation, and performing the mutation and crossover operations in a strongly typed way, which is consistent with the notion of strongly typed GP (STGP) [8]. These concerns motivated us to employ a three-based structure for the genotypic representation of the evolving driving styles of the agent. And inspired by its flexibility, and the recently emerged widespread adoption of document object model (DOM) and extensible markup language (XML), we represent the evolved genotypes as DOM-parse trees featuring equivalent flat XML-text. XML-tags offer a generic support for the data types in STGP, while the standard API of the off-the-shelf DOM-parsers contributes to the efficiency of manipulating the genetic representation [11]. A DOM-tree of sample genotype is shown in Figure 8.



```
<xml version=1.0>
<GP xmlns:xs=http://www.w3.org/2001/XMLS
  <DStyle ind=3>
    <Turn ind=4>
      <StraightLineGear>
        <#text> 3
      <ApproachingMode>
        <#text> 0
      <ApproachingAngle>
        <#text> 6
      <ApproachingAngleThreshold>
        <#text> 4
      <ThrottleLiftOffZone_x10>
        <#text> 58
      <BrakingVelocity_x10>
        <#text> 144
      <TurningGear>
        <#text> 4
    <Turn ind=19>
    <Turn ind=34>
    <Turn ind=49>
```

Figure 8. DOM-tree of sample genotype. The genotype represents the values of attributes of driving style of the agent for a sample circuit featuring four turns. The sub-tree associated with the values of attributes of the First Turn of the circuit is shown expanded.

The main parameters of STGP are shown in Table 2. 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 9. This circuit is a challenge for human operator, featuring a combination of a low-sped (4), medium-speed (2) and two high-speed turns (1 and 3) 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 4900mm.

Table 2. Main parameters of GP

Category	Value
Population size	200 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 four flying laps for the real scale model of the car
Fitness	Average lap time in milliseconds
Termination criteria	Number of generations = 40

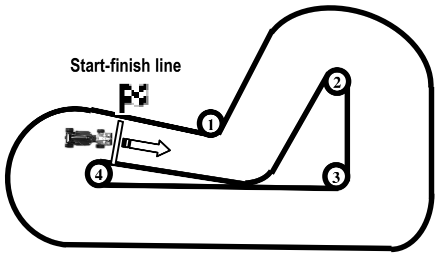


Figure 9. A sample circuit used for evolution of driving style of agent. The length of the track (lap distance) measured between the apexes of the turns is about 4900mm. The circuit is used for experiments with both the (i) software model and (ii) the real scaled model of the racing car.

4.1.3 Offline Evolution of Driving Styles on a Software Model of The Car.

In this experiment we conducted an offline evolution employing the proposed STGP framework on the software anticipatory model of the car. The fitness (i.e. the lap time of a single flying lap) convergence results, aggregated over 50 independent runs of STGP are shown in Figure 10. As figure illustrates, the best lap time average over all runs of STGP improved from 3800ms to about 3240ms (i.e., 15%) in 25 generations, which for a single run of STGP consumes only about 15 minutes of runtime on PC with 3GHz CPU, 512MB RAM and Windows XP OS.

4.1.4 Porting the Evolved Solution to the Real Car

Adaptation in Nature is viewed as an ability of species to discover the best phenotypic (i.e. pertaining to biochemistry, morphology, physiology, and behavior) traits for survival in continuously changing fitness landscape. For the considered task of evolving driving styles, the process of porting the solution, evolved offline on the model to the real car can be viewed as process of changing the fitness landscape (because of changing both the morphology of the car and the surrounding environment from modeled to the real ones) of the task.

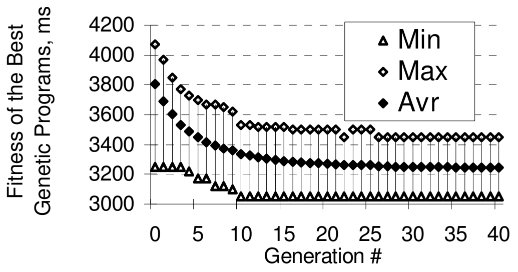


Figure 10. Aggregated over 50 independent runs of STGP fitness convergence characteristics of offline evolution of driving styles on software anticipatory model of the car.

In our approach we employ the same STGP framework (as used for offline evolution of the driving styles) for implementing a phylogenetic adaptation of the obtained solution to changes in the fitness landscape caused by switching from the simulated world into the reality. At the beginning of the adaptation process the STGP is initialized with a population comprising 20 best-of-run genetic programs (i.e. driving styles) obtained from experiments with 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 propose an approach of employing out-lap, several flying timed laps, and in-lap during each time trial in a way much similar to the current qualifying format in many car racing formulas. After crossing the start-finish line (shown immediately after the Turn 4 in Figure 9) completing the final timed lap governed by the current genetic program (i.e., 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 complete rest at a point somewhere between Turn 1 and Turn 2. At this point, which can be seen as an improvised team pit, the next genetic program (driving style) to be evaluated is loaded into the agent's controller, and the car starts an out-lap. Controlled by the new agent's controller, 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 quite sufficient to cancel any effect of the quality of the previous time trial on the performance of the current genetic program. Actually, already on the back straight between turns 3 and 4 the car the driving line and the car speed during the out lap virtually match these parameters of the timed flying lap. In order to compensate for the eventual effect of perceptional noise, inherent for the real systems, a total amount of 4 timed laps are conducted during the time trial of each genetic program, and the average lap time is considered as a corresponding fitness value.

The online 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 generations. A single run has been completed, and within 8 generations an improvement of fitness value of the best solution from the initial value of 3460s to 3220s has been observed.

4.3 Comparative Analysis of Human and Computer Driven Cars

In order to comparatively evaluate the quality of driving style obtained by evolution via STGP we conducted an comparative analysis of the performance results obtained on a simulated race distance of 80 flying laps (20 stints of 4 timed flying laps per each stint) around the same sample circuit as shown in Figure 9 for two cases: a computer- and a human controlled car. In the former case the anticipatory agent drives the car employing the best driving style evolved as elaborated before in 4.2.3 and 4.2.4. In the latter case the car was operated via the standard RC unit by the second author of this work, who proved to be the fastest among the 6 candidates considered. The results are summarized in Table 3.

Table 3. Comparative results of the performance of computer and a human operator

Parameter	Computer	Human
Total amount of covered laps	80	80
Total distance, mm	441381	411833
Total time, s	296	280
Average speed, mm/s	1491	1470
Max speed, mm/s	2058	2150
Average lap time, s	3.74	3.54
Best lap time, s	3.29	3.11
Standard deviation (inconsistence)	0.17	0.24

As shown in the table, the human is faster than computer with a relatively narrow gap of 16s (280s vs. 296s, of about 5%) in the total time, and 0.2s in the average lap time (3.54s vs.3.74s). However, the computer laps around the circuit more consistently with a standard deviation of a lap time about 0.17 vs. 0.24 for the human. While for the human the standard deviation can be attributed to the effects of the slight misjudgments and/or delays in responses (due to tiredness), for computer it is primarily related to the effect of noise on the precision with which the agent perceives its own state (position, orientation, velocity) and the environment (distance and the bearing to the apex of the current turn). This noise stems from the noise in determining the perfect geometrical center of car by the video-tracking subsystem. The latter employs a simple color-spot detection algorithm and it is not intended to compensate for the deformations of the shape of the spot caused by the shadows of irregular relief of the bodywork of the car. The consistency of lapping around the sample circuit is illustrated by the histograms of lap times shown in Figure 11.

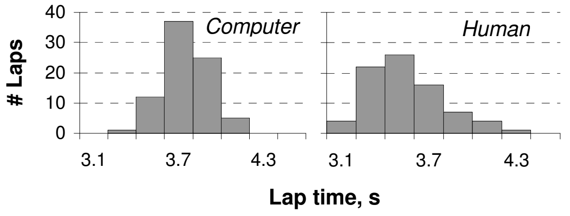


Figure 11. Histogram of lap times registered by computer-operated (left) and human-operated (right) car.

The emergent features of the evolved best driving style of the anticipatory agent, which are virtually identical to the corresponding features of the driving style of the human operator, are illustrated in Figure 12.

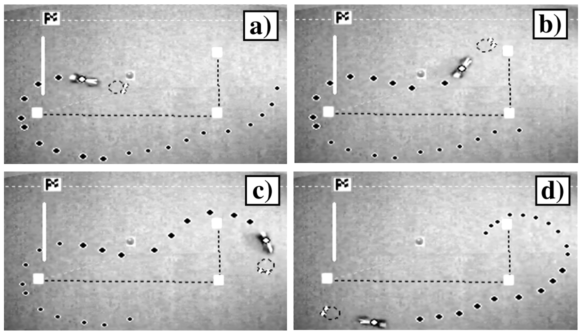


Figure 12. Emergent features of the best evolved driving style of the anticipatory agent.

The circle running just in front of the car indicates the anticipated intrinsic position and orientation of the car. As Figure 12a illustrates the starts the flying lap entering the Turn 4 relatively wide and exiting it as close as possible to the apex of the turn. As Figure 12b depicts, this allows the car to negotiate the left Turn 1 and the following right Turn 2 using the shortest possible driving line. As Figure 12c illustrates, the car exits the Turn 2 quite wide, interpreting the combination of right single-apex turns 2 and 3 as an improvised single double-apex turn. As Figure 12d demonstrates, such driving line contributes to the favorable orientation of the car at the entrance of the turn 3, which additionally extends the length of the back straight contributing to the achievement of the faster speed along the straight. The car enters the Turn 4 wide to allow for the exit to be as close as possible to the apex preparing for Turn 1 of the next flying lap.

5. CONCLUSIONS

The objective of this work is an automatic design via genetic programming of the functionality of driving agent, able to remotely operate a scale model of racing car in a fastest possible way. The agent's action are conveyed to the car via simple remote control unit featuring "forward", "reverse", and "neutral" throttle control commands and "left", "right" and "straight" steering controls. The agent perceives the environment from live video feed of a camera mounted overhead. In order to cope with the inherent video feed latency, which renders even the tasks of following simple routes virtually unsolvable, we proposed and implemented an approach of anticipatory modeling. In the proposed approach the agent considers its current actions based on anticipated intrinsic (rather than currently available, outdated) state of the car and surrounding environment. The driving style (considered in our approach as a driving line, which the car follows around the turns in the circuits together with the speed at which the car travels along this line) is both (i) evolved offline on a software simulator of the scaled model of the car and (ii) adapted online to real world applying strongly typed genetic programming. Comparative analysis demonstrates that on long runs the agent's operated car is only marginally (about 5%) slower than a human-operated one. However, the lap times of evolved agent are more consistent than that of human because of the occasional misjudgments and/or delays in the responses of the latter. This work can be viewed as a step towards the development of a framework for automated design of the control software of remotely operated vehicles capable to find an optimal solution to various tasks in different traffic situations and road conditions.

Acknowledgments

We thank Simon Lucas for his suggestion regarding hardware implementation of the presented system. The research was supported in part by the National Institute of Information and Communications Technology of Japan.

Bibliography

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

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

[3] IBM Corporation (1997), "Deep Blue",

URL: http://www.research.ibm.com/deepblue/

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

[5] Koza, J. R. (1992) "Genetic Programming: On the Programming of Computers by Means of Natural Selection", Cambridge, MA, MIT Press

[6] 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

[7] 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

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

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

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

[11] Tanev, I (2004) "DOM/XML-Based Portable Genetic Representation of Morphology, Behavior and Communication Abilities of Evolvable Agents", Artificial Life and Robotics, Vol.8, Number 1, pp.52-56, Springer-Verlag

[12] 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