Evolving robust and specialized car racing skills

Julian Togelius
Department of Computer Science
University of Essex, UK
julian@togelius.com

Simon M. Lucas
Department of Computer Science
University of Essex, UK
sml@essex.ac.uk

***Abstract*— Neural network-based controllers are evolved for racing simulated R/C cars around several tracks of varying difficulty. The transferability of driving skills acquired when evolving for a single track is evaluated, and different ways of evolving controllers able to perform well on many different tracks are investigated. It is further shown that such generally proficient controllers can reliably be developed into specialized controllers for individual tracks. Evolution of sensor parameters together with network weights is shown to lead to higher final fitness, but only if turned on after a general controller is developed, otherwise it hinders evolution. It is argued that simulated car racing is a scalable and relevant testbed for evolutionary robotics research, and that the results of this research can be useful for commercial computer games.**

Keywords: Evolutionary robotics, games, car racing, driving, incremental evolution

I. Introduction

Car racing is a remarkably popular preoccupation - both to watch and to participate in - be it in a computer simulation or in the "real world". But it is not only popular, it is also challenging: racing well requires fast and accurate reactions, knowledge of the car's behaviour in different environments, and various forms of real-time planning, such as path planning and deciding when to overtake a competitor. In other words, it requires many of the core components of intelligence being researched within computational intelligence and robotics. The success of the recent DARPA Grand Challenge[1], where completely autonomous real cars raced in a demanding desert environment, may be taken as a measure of the interest in car racing within these research communities.

This paper deals with using evolutionary algorithms to create neural network controllers for simulated car racing. Specifically, we evolve controllers that have robust performance over different tracks, and can be specialized to work better on particular tracks.

Evolutionary robotics (the use of evolutionary algorithms for embodied control problems) and simulated car racing are in many ways ideal companions. The benefit for the development of racing games and simulations is clear: evolutionary robotics offers a way to automatically develop controllers, possibly specialized for specific tracks or types of tracks, driving styles, skill levels, competitors etc. One could envision a racing simulator where the user is allowed to construct his own tracks and cars, and the game automatically develops a set of controllers to drive these tracks. The game could also automatically adapt to the user's driving style, or learn from other drivers (humans or machines) on the Internet.

The benefits for evolutionary robotics might require some explanation. While evolutionary robotics has successfully been used for various interdisciplinary investigations (e.g. of memory mechanisms, neural architectures and evolutionary dynamics), and for parameter tuning of some more complex controllers, its approximately 15 years of development have not seen much scaling up[2]. That is, we have yet to see the evolution of robot controllers (as opposed to just parameters of such) for any really complex problem - problems where artificial evolution becomes a superior alternative to manual design of controllers.

We believe that some of the reason for this lack of progress is the limited environments, sensor data, embodiments, and tasks in most evolutionary robotics experiments. A typical such experiment uses a semi-holonomic robot operating in an impoverished environment (in many ways resembling a "Skinner box", the simplistic boxes pioneered by B. F. Skinner for studying operant conditioning[3]), using simple, low-bandwidth sensor input, doing a task that is hard to incrementally scale up. The car racing task uses a more complex and interesting robot morphology, as a car is more complex to control than a semi-holonomic robot, but at the same time it has more capabilites. While a simple racing track might be as impoverished an environment as ever a Skinner box, it can be scaled up. A controller might be evolved to race a simple track, which can then be progressively complexified (by adding competitors, gears, crossroads, blind alleys, bridges, jumps etc.) up to and above the level of the DARPA Grand Challenge, without ever changing the nature of the fitness function, thus ensuring smooth scaling up. This solution to the problems of the environment and task scalability does come at cost: the car will probably need ever more sophisticated sensors, including high-bandwitdh visual input, to navigate more complex tracks. But such input can be supplied, if we use one of today's graphically sophisticated racing games as experimental environment. This shifts the problem to one of controller encodings that can handle such complex input.

A. Prior research

1) Evolutionary car racing: A few investigations into evolutionary car racing can be found in the recent literature. Togelius and Lucas[4] investigated various controller architectures and sensor input representations for simulated car racing. It was concluded that the only combination out

of those studied that allows evolution to reliably produce good racing controllers uses neural networks informed by egocentric information from range-finder sensors. Best performance was achieved by making ranges and angles of the rangefinders evolvable, and providing the network with a further sensor indicating angle to the next waypoint. The controllers were only tried on one track, but some noise was introduced into the environment and the track was surrounded by impenetrable walls. The best of the evolved controllers outperformed all of a small sample human competitors.

Stanley *et al.*[5] used a similar setup - neural networks informed by range-finders - in an experiment aimed at evolving both controllers and crash-warning systems for subsequently impaired controllers. The experiment was conducted on a single track in simulation (using the RARS simulator[6]), and the track was not surrounded by walls, so the car was allowed (at a fitness penalty) to venture outside the track.

In another interesting experiment, Floreano *et al.* evolved neural networks for simulated car racing using first-person visual input from the driving simulator Carworld[7][8]. However, only 5 x 5 pixels of the visual field was used as inputs for the network; the position of these pixels was dynamically selected by the network, in a process known as active vision.

A different approach to evolutionary car racing was taken by Tanev *et al.*, who evolved parameters for a hand-coded racing car controller, using anticipatory modeling of the car's position[9]. While the amount of human input into the controller design process is arguably higher in this case, this approach allowed evolution of controllers for real radio-controlled cars without an intermediary simulation step.

Also related is the work of Wloch and Bentley, who used a human-designed controller built into a high quality racing simulator, but used artificial evolution to optimize all physical and mechanical parameters of the car[10]. Evolution here managed to come up with car configurations that performed better than any of the stock cars in the simulator.

2) Supervised learning and real-world applications: Machine learning techniques have also been used in real-world car driving and racing applications, though these techniques have been forms of supervised learning rather than evolutionary learning. Perhaps most well-known of these is Pomerleau's use of backpropagation to train neural networks to associate pre-processed video input with a human driver's actions, leading to a controller able to keep a real car on the road and take curves appropriately[11]. More recently, the winning team in the DARPA Grand Challenge made extensive use of machine learning in developing their car controller.

Going from physical reality to virtual reality, the Microsoft's Xbox video game Forza Motorsport is worthy of mention, as all the opponent car controllers have been trained by supervised learning of human player data, instead of the usual racing game technique of blindly following precalculated racing lines[12]. The player can even train his own "drivatars" to race tracks in his place, after they have acquired his or her individual driving style.

Supervised learning, however, ultimately suffers from requiring good training data. Sometimes such training data is simply not available, at other times it is prohibitively expensive to obtain, and at yet other times imitating human drivers is simply not what we want.

B. Motivations for this paper

While the research referred to above has shown the usefulness of evolutionary robotics techniques for car racing, the controllers have in all those cases only been tested on a single track, and sometimes with severe simplifying assumptions, such as being able to drive through walls. Thus, the first objective of the research reported in this paper is to evolve neural network controllers each capable of competitively and reliably navigating a variety of different tracks, including tracks they have not been trained on. Based on the range-finding and aimpoint sensors proposed in[4], we investigate which sensor setup and evolutionary sequence allows us to create such controllers.

A second objective is to investigate whether evolution of a specialized controller, i.e. one performing very well on a particular track, can be sped up by starting from an already evolved "general" controller. Such a process could be useful for example in a racing game, where users are allowed to design tracks and a controller providing good performance on such tracks needs to be created on the fly.

The concrete questions we pose and try to answer are the following: How robust is the evolutionary algorithm, that is, how certain can we be that a given evolutionary run will produce a proficient controller for a given track? Is the layout of the racing track directly influencing the fitness landscape so that some tracks are much harder than others to evolve, while not being impossible to drive? What is the transferability of knowledge gained in evolving for one track in terms of performance on other tracks? Can we evolve controllers that can proficiently race all tracks in our training set? How? Can such generally proficient controllers be used to reliably create specialized controllers that perform well, but only on particular tracks? Finally, can this be done even for tracks for which it is not possible to evolve a good controller from scratch?

While this investigation primarily addresses the scalability of the problem domain (and to some extent of the sensor/network combination), it may also be of use for practical applications such as racing games to find out the most reliable ways to evolve proficient controllers.

C. Overview of the paper

The paper is laid out as follows: first, we describe the characteristics of the car racing simulation we will be using, including sensor models, tracks, and how this models differs from the problem of racing real radio-controlled cars. The next section details the neural networks and evolutionary algorithm we employ. We then proceed to describe experiments on evolving controllers optimized for the individual tracks from scratch, followed by a section where we investigate

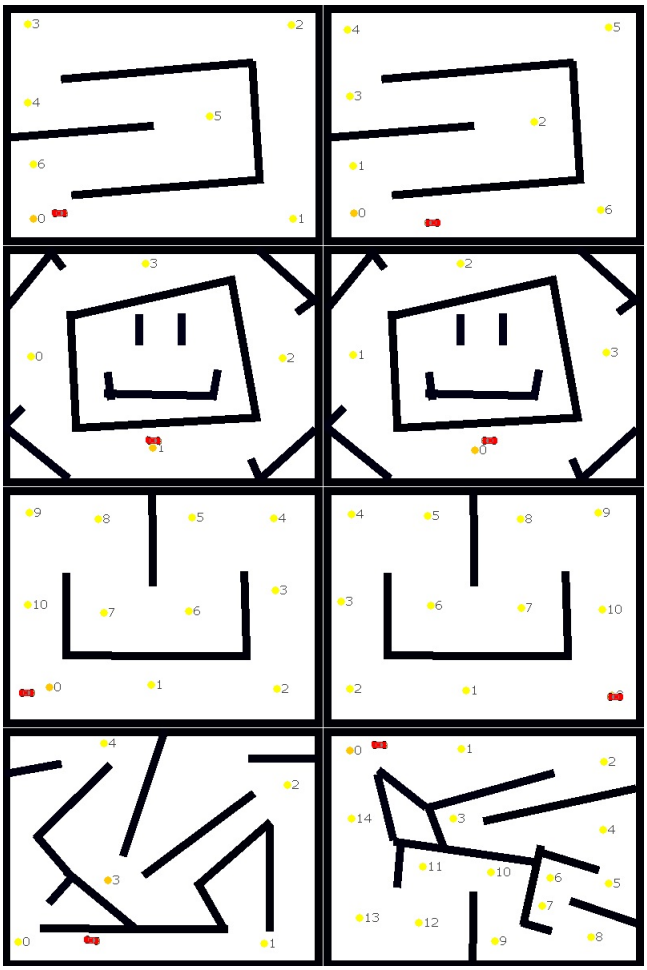

Fig. 1. The eight tracks. Notice how tracks 1 and 2 (at the top), 3 and 4, 5 and 6 differ in the clockwise/anti-clockwise layout of waypoints and associated starting points. Tracks 7 and 8 have no relation to each other apart from both being difficult.

how to evolve controllers that provide robust performance over several tracks. These controllers are then validated on tracks for which they have not been evolved. Finally, these controllers are further evolved to provide better fitness on specific tracks, conclusions are drawn, and further research is suggested.

II. The car racing model

The experiments in this article were performed in a 2-dimensional simulator, intended to qualitatively if not quantitatively, model a standard radio-controlled (R/C) toy car (approximately 17 centimeters long) in an arena with dimensions approximately 3*2 meters, where the track is delimited by solid walls. The simulation has the dimensions 400*300 pixels, and the car measures 20*10 pixels.

R/C toy car racing differs from racing full-sized cars in several ways. One is the simplified controls; many R/C cars have only three possible drive modes (forward, backward, and neutral) and three possible steering modes (left, right and center). Other differences are that many toy cars have bad grip on many surfaces, leading to easy skidding, and that damaging such cars in collisions is harder due to their low weight.

The dynamics of the car are based on a reasonably detailed mechanical model, taking into account the small size of the car and bad grip on the surface, but is not based on any actual measurement [13][14]. The model is similar to that used in [4], and differs mainly in its improved collision handling; after more experience with the physical R/C cars the collision response system was reimplemented to make collisions more realistic (and, as an effect, more undesirable). Now, a collison may cause the car to get stuck if the wall is struck at an unfortunate angle, something often seen in experiments with physical cars.

A track consists of a set of walls, a chain of waypoints, and a set of starting positions and directions. When a car is added to a track in one of the starting positions, with corresponding starting direction, both the position and angle being subject to random alterations. The waypoints are used for fitness calculations.

For the experiments we have designed eight different tracks, presented in figure 1. The tracks are designed to vary in difficulty, from easy to hard. Three of the tracks are versions of three other tracks with all the waypoints in reverse order, and the directions of the starting positions reversed.

The main differences between our simulation and the real R/C car racing problem have to do with sensing. As reported in Tanev et al. as well as [4], there is a small but not unimportant lag in the communication between camera, computer and car, leading to the controller acting on outdated perceptions. Apart from that, there is often some error in estimations of the car's position and velocity from an overhead camera. In contrast, the simulation allows instant and accurate information to be fed to the controller.

III. Evolvable intelligence

A. Sensors

The car experiences its environment through two types of sensors: the waypoint sensor, and the wall sensors. The waypoint sensor gives the difference between the car's current orientation and the angle to the next waypoint (but not the distance to the waypoint). When pointing straight to a waypoint, this sensor thus outputs 0, when the waypoint is to the left of the car it outputs a positive value, and vice versa. As for the wall sensors, each sensor has an angle (relative to the orientation of the car) and a range, between 0 and 200 pixels. The output of the wall sensor is zero if no wall is encountered along a line with the specified angle and range from the centre of the car, otherwise it is a fraction of one, depending on how close to the car the sensed wall is. A small amount of noise is applied to all sensor readings, as it is to starting positions and orientations.

In some of the experiments the sensor parameters are mutated by the evolutionary algorithm, but in all experiments they start from the following setup: one sensor points straight forward (0 radians) in the direction of the car and has

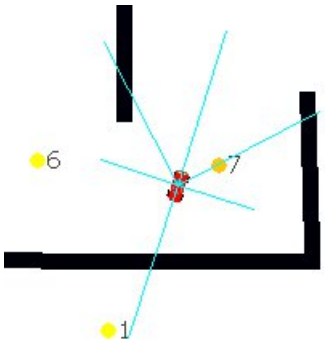

Fig. 2. The initial sensor setup, which is kept throughout the evolutionary run for those runs where sensor parameters are not evolvable. Here, the car is seen in close-up moving upward-leftward. At this particular position, the front-right sensor returns a positive number very close to 0, as it detects a wall near the limit of its range; the front-left sensor returns a number close to 0.5, and the back sensor a slightly larger number. The front, left and right sensors do not detect any walls at all and thus return 0.

range 200 pixels, as has three sensors pointing forward-left, forward-right and backward respectively. The two other sensors, which point left and right, have reach 100; this is illustrated in figure 2.

B. Neural networks

The controllers in the experiments below are based on neural networks. More precisely, we are using multilayer perceptrons with three neuronal layers (two adaptive layers) and tanh activation functions. A network has at least three inputs: one fixed input with the value 1, one speed input in the approximate range [0..3], and one input from the waypoint sensor, in the range [-Π..Π]. In addition to this, it might have any number of inputs from wall sensors, in the range [0..1]. All networks have two outputs, which are interpreted as driving commands for the car.

C. Evolutionary algorithm

The genome is an array of floating point numbers, of variable or fixed length depending on the experimental setup. Apart from information on the number of wall sensors and hidden neurons, it encodes the orientation and range of the wall sensors, and weights of the connections in the neural network.

The evolutionary algorithm used is a kind of evolutionary strategy, with $\mu = 50$ and $\delta = 50$. In other words, 50 genomes (the elite) are created at the start of evolution. At each generation, one copy is made of each genome in the elite, and all copies are mutated. After that, fitness value is calculated for each genome, and the 50 best individuals of all 100 form the new elite.

There are two mutation operators: Gaussian mutation of all weight values, and Gaussian mutation of all sensor parameters (angles and lengths), which might be turned on or off. In both cases, the standard deviation of the Gaussian distribution was set to 0.3.

Last but not least: the fitness function. The fitness of a controller is calculated as the number of waypoints it has passed, divided by the number of waypoints in the track, plus an intermediate term representing how far it is on its way to the next waypoint, calculated from the relative distances between the car and the previous and next waypoint. A fitness of 1.0 thus means having completed one full track within the alloted time. Waypoints can only be passed in the correct order, and a waypoint is counted as passed when the centre of the car is within 30 pixels from the waypoint. In the evolutionary experiments reported below, each car was allowed 700 timesteps (enough to do two to three laps on most tracks in the test set) and fitness was averaged over three trials.

$Track$	10	50	100	200	$Pr.$
1	0.32 (0.07)	0.54 (0.2)	0.7 (0.38)	0.81 (0.5)	2
2	0.38 (0.24)	0.49 (0.38)	0.56 (0.36)	0.71 (0.5)	2
3	0.32 (0.09)	0.97 (0.5)	1.47 (0.63)	1.98 (0.66)	7
4	0.53 (0.17)	1.3 (0.48)	1.5 (0.54)	2.33 (0.59)	9
5	0.45 (0.08)	0.95 (0.6)	0.95 (0.58)	1.65 (0.45)	8
6	0.4 (0.08)	0.68 (0.27)	1.02 (0.74)	1.29 (0.76)	5
7	0.3 (0.07)	0.35 (0.05)	0.39 (0.09)	0.46 (0.13)	0
8	0.16 (0.02)	0.19 (0.03)	0.2 (0.01)	0.2 (0.01)	0

TABLE I

THE FITNESS OF THE BEST CONTROLLER OF VARIOUS GENERATIONS ON THE DIFFERENT TRACKS, AND NUMBER OF RUNS PRODUCING PROFICIENT CONTROLLERS. FITNESS AVERAGED OVER 10 SEPARATE EVOLUTIONARY RUNS; STANDARD DEVIATION BETWEEN PARENTHESES.

IV. EVOLVING TRACK-SPECIFIC CONTROLLERS

The first experiments consisted in evolving controllers for the eight tracks separately, in order to the test the software in general and to rank the difficulty of the tracks.

For each of the tracks, the evolutionary algorithm was run 10 times, each time starting from a population of "clean" controllers, with all connection weights set to zero and sensor parameters as explained above. Only weight mutation was allowed. The evolutionary runs were for 200 generations each.

A. Fixed sensor parameters

1) Evolving from scratch: The results are listed in table I, which is read as follows: each row represents the results for one particular track. The first column gives the mean of the fitnesses of the best controller of each of the evolutionary runs at generation 10, and the standard deviation of the fitnesses of the same controllers. The next three columns present the results of the same calculations at generations 50, 100 and 200, respectively. The "Pr" column gives the number of proficient best controllers for each track. An evolutionary run is deemed to have produced a proficient controller if its best controller at generation 200 has a fitness (averaged, as always, over three trials) of at least 1.5, meaning that it completes at least one and a half lap within the allowed time.

For the first two tracks, proficient controllers were produced by the evolutionary process within 200 generations, but only in two out of ten runs. This means that while it is possible to evolve neural networks that can be relied on to

$Track$	10	50	100	200	$Pr.$
1	0.3 (0.05)	0.58 (0.17)	0.65 (0.18)	0.89 (0.4)	1
2	0.32 (0.09)	0.72 (0.4)	0.81 (0.49)	0.91 (0.6)	3
3	0.53 (0.22)	1.39 (0.51)	2.77 (0.66)	1.99 (0.7)	7
4	1.37 (0.89)	2.25 (0.34)	2.42 (0.37)	2.41 (0.36)	10
5	0.4 (0.07)	0.64 (0.35)	0.95 (0.55)	1.31 (0.66)	4
6	0.48 (0.12)	0.7 (0.29)	0.83 (0.39)	0.99 (0.65)	2
7	0.33 (0.11)	0.43 (0.08)	0.44 (0.08)	0.5 (0.15)	0
8	0.16 (0.02)	0.21 (0)	0.21 (0)	0.21 (0)	0

TABLE III

EVOLVING CONTROLLERS FOR INDIVIDUAL TRACKS FROM SCRATCH WITH SENSOR MUTATION TURNED ON; FORMAT AS IN TABLE I.

race around one of these track without getting stuck or taking excessively long time, the evolutionary process in itself is not reliable. In fact, most of the evolutionary runs are false starts. For tracks 3, 4, 5 and 6, the situation is different as at least half of all evolutionary runs produce proficient controllers. The best evolved controllers for these tracks get around the track fairly fast without colliding with walls. For tracks 7 and 8, however, we have not been able to evolve proficient controllers from scratch at all. The "best" (least bad) controllers evolved for track 7 might get halfway around the track before getting stuck on a wall, or losing orientation and starting to move back along the track.

2) Generality of evolved controllers: Next, we examined the generality of these controllers by testing their performance of the best controller for each track on each of the ten tracks. The results are presented in figure II, and clearly show that the generality is very low. No controller performed very well on any track it had not been evolved on, with the interesting exception of the controller evolved for track 1, that actually performed better on track 3 than on the track for which it had been evolved, and on which it had a rather mediocre performance. It should be noted that both track 1 and track 3 (like all odd-numbered tracks) run counterclockwise, and there indeed seems to be a slight bias for the other controllers to get higher fitness on tracks running in the same direction as the track for which they were evolved. We have not analysed this further.

B. Evolved sensor parameters

1) Evolving from scratch: Evolving controllers from scratch with sensor parameter mutations turned on resulted in somewhat lower average fitnesses and numbers of proficient controllers, as can be seen in table III. The controllers that reached proficiency seemed to be roughly equally fit as those evolved with fixed sensors, but more evolutionary runs got stuck in some local optimum and never produced proficient controllers when sensor parameters were evolvable. It is not known whether this is simply because of the increase in search space dimensionality caused by the addition of sensor parameters, or if they complicate the evolutionary process in some other way.

2) Generality of evolved controllers: Controllers evolved with evolvable sensor parameters turn out to generalize really badly, almost as badly as the controllers evolved with fixed sensors. However, there are some interesting differences, and the controllers evolved for track 1, 2 and 6 (but not the others) actually perform better on tracks for which they were not evolved. It is quite hard to see any kind of logic in which controllers will do well on which tracks, except those they were evolved for, and more data would definitely be needed to resolve this.

V. EVOLVING ROBUST DRIVING SKILLS

The next suite of experiments were on evolving robust controllers, i.e. controllers that can drive proficiently on a large set of tracks.

A. Simultaneous evolution

Our first attempt consisted in evolving controllers on all tracks at once. For this purpose, we ran several evolutionary runs where each controller was tested on all the first six tracks, each for three trials, and the fitness was averaged over all these trials. We ran several evolutionary runs with this setup, and with both evolvable and fixed sensor parameters, for long periods of time, but found very little progress - no controller reached an average fitness above 1.

B. Incremental evolution

Abandoning this method, we tried incremental evolution. The idea here was to evolve a controller on one track, and when it reached proficiency (mean fitness above 1.5) add another track to the training set - so that controllers are now evaluated on both tracks and fitness averaged - and continue evolving. This procedure is then repeated, with a new track added to the fitness function each time the best controller of the population has an average fitness of 1.5 or over, until we have a controller that races all of the first six tracks proficiently. The order of the tracks was 5, 6, 3, 4, 1 and finally 2, the rationale being that the balance between clockwise and counterclockwise should be as equal as possible in order to prevent lopsided controllers, and that easier tracks should be added to the mix before harder ones.

This approach turned out to work much better than simultaneous evolution. Several runs were performed, and while some of them failed to produce generally proficient controllers, some others fared better. A successful run usually takes a long time, on the order of several hundred generations, but it seems that once a run has come up with a controller that is proficient on the first three or four tracks, it almost always proceeds to produce a generally proficient controller. One of the successful runs is depicted in figure 3, and the mean fitness of the best controller of that run when tested on all eight tracks separately is shown in IV. As can be seen from this table, the controller does a good job on the six tracks for which it was evolved, bar that it occasionally gets stuck on a wall in track 2. It never makes its way around track 7 or 8.

The successful runs were all made with sensor mutation turned off. Some runs of incremental evolution were made with sensor mutation allowed; however, they failed to produce any proficient controllers. We speculate that this is

Evo/Test	1	2	3	4	5	6	7	8
1	1.02 (0.14)	0.87 (0.1)	**1.45 (0.18)**	0.52 (0)	1.26 (0.17)	0.03 (0)	0.2 (0.18)	0.13 (0)
2	0.28 (0.06)	**1.13 (0.35)**	0.18 (0.1)	0.75 (0.26)	0.5 (0.13)	0.66 (0.19)	0.18 (0.15)	0.14 (0.02)
3	0.58 (0.16)	0.6 (0.22)	**2.1 (0.48)**	1.45 (0.66)	0.62 (0.13)	0.04 (0.1)	0.03 (0.09)	0.14 (0.02)
4	0.15 (0.01)	0.32 (0.02)	0.06 (0.05)	**1.77 (0.52)**	0.22 (0.1)	0.13 (0.13)	0.07 (0.09)	0.13 (0.02)
5	0.07 (0.02)	-0.02 (0)	0.05 (0)	0.2 (0.11)	**2.37 (0.28)**	0.1 (0.04)	0.03 (0.05)	0.13 (0.01)
6	1.33 (0.18)	0.43 (0.07)	0.4 (0.2)	0.67 (0.22)	1.39 (0.42)	**2.34 (0.05)**	0.13 (0.13)	0.14 (0.11)
7	**0.45 (0.11)**	0 (0.07)	0.6 (0.18)	0.03 (0.04)	0.36 (0.08)	0.07 (0.03)	0.22 (0.15)	0.08 (0)
8	0.16 (0.03)	0.28 (0.04)	0.09 (0.07)	**0.29 (0.18)**	0.21 (0.03)	0.08 (0.1)	0.1 (0.09)	0.13 (0)

TABLE II

THE FITNESS OF EACH CONTROLLER ON EACH TRACK. EACH ROW REPRESENTS THE PERFORMANCE OF THE BEST CONTROLLER OF ONE EVOLUTIONARY RUN WITH FIXED SENSORS, EVOLVED THE TRACK WITH THE SAME NUMBER AS THE ROW. EACH COLUMN REPRESENTS THE PERFORMANCE OF THE CONTROLLERS ON THE TRACK WITH THE SAME NUMBER AS THE COLUMN. EACH CELL CONTAINS THE MEAN FITNESS OF 50 TRIALS OF THE CONTROLLER GIVEN BY THE ROW ON THE TRACK GIVEN BY THE COLUMN. CELLS WITH BOLD TEXT INDICATE THE TRACK ON WHICH A CERTAIN CONTROLLER PERFORMED BEST.

Track	1	2	3	4	5	6	7	8
Fitness/sd	1.66 (0.08)	1.48 (0.25)	2.56 (0.2)	2.49 (0.15)	2 (0.25)	2.02 (0.42)	0.4 (0.21)	0.16 (0.07)

TABLE IV

FITNESS OF AN INCREMENTALLY EVOLVED GENERAL CONTROLLER WITH FIXED SENSOR PARAMETERS ON THE DIFFERENT TRACKS. COMPOUND FITNESS OVER ALL 8 TRACKS IS 2.01 (0.11).

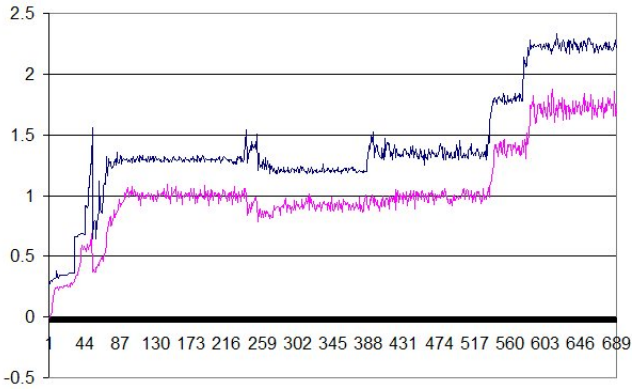

Fig. 3. A successful incremental run, producing a generally proficient controller. New tracks were added to the fitness function when fitness of the best controller reached 1.5; this happened at generations 53, 240, 253, 394 and 536. Maximum fitness continued to increase for approximately 50 generations after that. The graph show the fitness of the best controller (dark line) and the mean fitness of the population.

because these runs suffer from ”premature specialization” - after evolving a good controller for the first track, the sensor setup might not be suited for good driving on the second track, and changing the parameters would diminish fitness on the first track, thus creating a local optimum. That the first two tracks are, from the point of view of the car, mirror images of each other, adds plausibility to this hypothesis.

C. Further evolution

Evolving sensor parameters can be beneficial, however, when this is done for a controller that has already reached general proficiency. We used one of the generally proficient

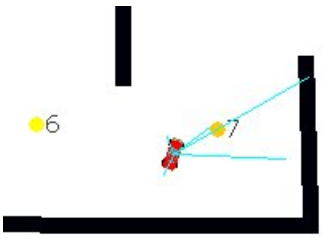

Fig. 4. Sensor setup of the further evolved general controller analysed in table V. Only three sensors seem to be long enough to of any use, and all of those point to the right or front-right. The asymmetry and ”waste” is somewhat surprising, as the controller performs well on all the first six tracks (but it does do slightly better on clockwise than on anti-clockwise tracks).

controllers evolved using the incremental method as the seed for a new evolutionary run, with sensor mutation turned on and controllers tested on all six tracks simultaneously. The results was an increase in mean fitness, as can be seen in V. Although the mean fitness does not increase on every single track, the best controller of the last generation races all the tracks more reliably, and is very rarely observed to crash into a wall in such a way that the car gets stuck. The evolved sensors of this controller showed little similiarity to the original sensor setup, described above - see figure 4 for an example.

VI. EVOLVING SPECIALIZED CONTROLLERS

In order to see whether we could create even better controllers, we used one of the further evolved controllers (with evolved sensor parameters) as basis for specializing

Track	1	2	3	4	5	6	7	8
Fitness (sd)	1.66 (0.12)	1.86 (0.02)	2.27 (0.45)	2.66 (0.3)	2.19 (0.23)	2.47 (0.18)	0.22 (0.15)	0.15 (0.01)

TABLE V

FITNESS OF A FURTHER EVOLVED GENERAL CONTROLLER WITH EVOLVABLE SENSOR PARAMETERS ON THE DIFFERENT TRACKS. COMPOUND FITNESS 2.22 (0.09).

Track	10	50	100	200	*Pr.*
1	1.9 (0.1)	1.99 (0.06)	2.02 (0.01)	2.04 (0.02)	10
2	2.06 (0.1)	2.12 (0.04)	2.14 (0)	2.15 (0.01)	10
3	3.25 (0.08)	3.4 (0.1)	3.45 (0.12)	3.57 (0.1)	10
4	3.35 (0.11)	3.58 (0.11)	3.61 (0.1)	3.67 (0.1)	10
5	2.66 (0.13)	2.84 (0.02)	2.88 (0.06)	2.88 (0.06)	10
6	2.64 (0)	2.71 (0.08)	2.72 (0.08)	2.82 (0.1)	10
7	1.53 (0.29)	1.84 (0.13)	1.88 (0.12)	1.9 (0.09)	10
8	0.59 (0.15)	0.73 (0.22)	0.85 (0.21)	0.93 (0.25)	0

TABLE VI

FITNESS OF BEST CONTROLLERS, EVOLVING CONTROLLERS SPECIALISED FOR EACH TRACK, STARTING FROM A FURTHER EVOLVED GENERAL CONTROLLER WITH EVOLVED SENSOR PARAMETERS.

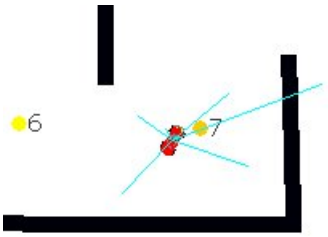

Fig. 5. Sensor setup of controller specialized for track 5. While more or less retaining the two longest-range sensors from the further evolved general controller it is based on, it has added medium-range sensors in the front and back, and a very short-range sensor to the left.

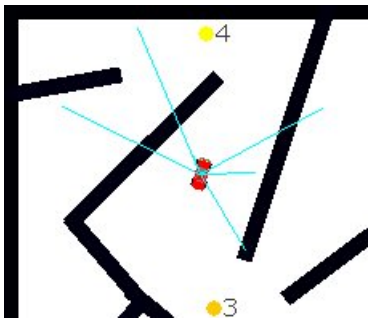

Fig. 6. Sensor setup of a controller specialized for, and able to consistently reach good fitness on, track 7. Presumably the use of all but one sensor and their angular spread reflects the large variety of different situations the car has to handle in order to navigate this more difficult track.

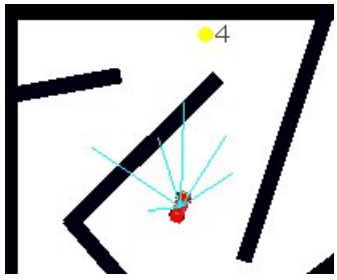

Fig. 7. Sensor setup of another controller specialized for track 7, like the one in figure 6 seemingly using all its sensors, but in a quite different way.

controllers. For each track, 10 evolutionary runs were made, where the initial population was seeded with the general controller and evolution was allowed to continue for 200 generations. Results are shown in table VI. The mean fitness improved significantly on all six first tracks, and much of the fitness increase occured early in the evolutionary run, as can be seen from a comparison with table V. Further, the variability in mean fitness of the specialized controllers from different evolutionary runs is very low, meaning that the reliability of the evolutionary process is very high. Perhaps most surprising, however, is that all 10 evolutionary runs produced proficient controllers for track 7, on which the general controller had not been trained (and indeed had very low fitness) and for which it had previously been found to be impossible to evolve a proficient controller from scratch.

Analysis of the evolved sensor parameters of the specialized controllers show a remarkable diversity, even among controllers specialized for the same track, as evident in figures 5, 6 and 7. Sometimes, no similarity can be found between the evolved configuration and either the original sensor parameters or those of the further evolved general controller the specialization was based on.

VII. OBSERVATIONS ON EVOLVED DRIVING BEHAVIOUR

It has previously been found that the evolutionary approach used in this paper can produce controllers that outperform human drivers[4]. To corroborate this result, one of the authors measured his own performance on the various tracks, driving the car using keyboard inputs and a suitable delay of 50 ms between timesteps. Averaged over 10 attempts, the author's fitness on track 2 was 1.89, it was 2.65 on track 5, and 1.83 on track 7, numbers which compare rather unfavourably with those found in table VI. The responsible author would like to believe that this says more about the capabilities of the evolved controllers than those of the author.

Traces of steering and driving commands from the evolved controllers show that they often use a PWM-like technique, in that they frequently - sometimes almost every timestep - change what commands they issue. For example, the general controller used as the base for the specializations above employs the tactic of constantly alternating between steering left and right when driving parallell to a wall, giving the appearance that the car is shaking. Frequently alternating

between neutral and forward drive is also used a way of keeping a certain speed; an approach many engineers would use when designing a controller for a vehicle that can only be controlled with discrete inputs. Doing so is however practically impossible for a human driver, and analysis of action traces for human drivers shows much fewer changes to the commands given; as an example, one of the authors changes drive command only four times per lap when racing on track 3.

VIII. Conclusions and future work

We believe that the results presented above answer, at least in part, the several questions posed in section I-B. Different tracks can indeed be constructed that have various difficulty levels, in terms of the probability that an evolutionary run starting from scratch will produce a proficient controller within a given number of generations, and the mean fitness of evolved controllers. Difficulty levels range from very easy, where the evolutionary algorithm almost always succeeds, to very hard, for which no successful controllers have been found, and agree with intuitive human difficulty ratings of the same tracks. These skills are, however, not transferable: a controller evolved from scratch to perform well on a given track usually performs very poorly on all other tracks. Evolving sensor parameters along with network weights makes for fewer proficient controllers (probably because of more local optima), a results which is not inconsistent with the good controllers that do emerge being slightly superior to fixed-sensor ones, as found in [4].

As for the question on whether we can automatically create controllers with driving skills so general that they can proficiently race all tracks in our training set, this can be done by using incremental evolution, going from simpler to more complex tracks, with sensor mutation turned off. Attempts to evolve general controllers with sensor mutation turned on failed, as did attempts to evolve controllers on all tracks simultaneously. Once a general controller has been created, its fitness can be increased through continued evolution with sensor mutation turned on. Specialized controllers can be created by further evolving a general controller, using only one track in the fitness function. These specialized controllers invariably have very high fitness. Much to our surprise, this was true even for one hard track which the general controller had not been evolved on and which it had very low fitness on, and for which we have not been able to evolve proficient controllers from scratch. Apparently, the general controller is somehow closer in search space to a proficient controller for that track, even though it has no proficiency itself on that track. Exactly how this works remains to be found out.

We hope that our results are relevant to both game AI development, as it suggests a way of using already evolved solutions as bases for further evolution to quickly and reliably produce specialized solutions for particular tasks, and to evolutionary robotics, as it goes some way to demonstrate the scalability and generality of car racing as an ER testbed.

Currently, our efforts are focused on extending the model to allow competitive coevolution between several vehicles on the same track. Further, we are planning to use this model to investigate controller architectures permitting internal state, such as plastic networks[15] or recurrent networks. We are also planning to compare evolutionary learning to other forms of reinforcement learning, such as TD-learning, which has been shown to be considerably faster in some domains.

However, to be able to handle really complex environment, the controller will need high-bandwidth sensor data of some kind, without which complex object recognition and resolution of perceptual aliasing is impossible. We are therefore working towards incorporating visual or visual-like input, using either the current 2D simulation, or some other 3D-enabled simulator. We have previously tried the "naive" approach of connecting high-bandwidth (on the order 10,000 inputs) 2D vision directly to evolvable single- or multi-layer perceptrons, but only had limited success. An integral part of the ongoing project is therefore to develop a modular architecture which can reuse weights so as to reduce the dimensionality of space in which to search for such controllers.

References

[1] DARPA, "Grand challenge web site," http://www.grandchallenge.org/, 2005.

[2] S. Nolfi and D. Floreano, *Evolutionary robotics*. Cambridge, MA: MIT Press, 2000.

[3] B. F. Skinner, *Science and Human Behavior*. The Free Press, 1953, ch. Behaviorism.

[4] J. Togelius and S. M. Lucas, "Evolving controllers for simulated car racing," in *Proceedings of the Congress on Evolutionary Computation*, 2005, pp. 1906–1913.

[5] K. O. Stanley, N. Kohl, R. Sherony, and R. Miikkulainen, "Neuroevolution of an automobile crash warning system," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2005)*, 2005.

[6] "Rars (robot auto racing simulator) homepage," http://rars.sourceforge.net/, 1995.

[7] D. Floreano, T. Kato, D. Marocco, and E. Sauser, "Coevolution of active vision and feature selection," *Biological Cybernetics*, vol. 90, pp. 218–228, 2004.

[8] M. Hewat, "Carworld driving simulator," http://carworld.sourceforge.net/, 2000.

[9] I. Tanev, M. Joachimczak, H. Hemmi, and K. Shimohara, "Evolution of the driving styles of anticipatory agent remotely operating a scaled model of racing car," in *Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC-2005)*, 2005, pp. 1891–1898.

[10] K. Wloch and P. J. Bentley, "Optimising the performance of a formula one car using a genetic algorithm," in *Proceedings of Eighth International Conference on Parallel Problem Solving From Nature*, 2004, pp. 702–711.

[11] D. A. Pomerleau, "Neural network vision for robot driving," in *The Handbook of Brain Theory and Neural Networks*, 1995.

[12] "Forza motorsport drivatars," http://research.microsoft.com/mlp/forza/, 2005.

[13] D. M. Bourg, *Physics for Game Developers*. O'Reilly, 2002.

[14] M. Monster, "Car physics for games," http://home.planet.nl/ monstrous/tutcar.html, 2003.

[15] D. Floreano and F. Mondada, "Evolution of plastic neurocontrollers for situated agents," in *From Animals to Animats IV: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, P. Maes, M. Mataric, J. Meyer, J. Pollack, H. Roitblat, and S. Wilson, Eds. Cambridge, MA: MIT Press-Bradford Books, 1996.