

Optimized Sensory-motor Couplings plus Strategy Extensions for the TORCS Car Racing Challenge

Martin V. Butz and Thies D. Lönneker

Abstract—The TORCS simulated car racing competition was introduced over a year ago. It asks for the design of racing car control strategies that have to rely on local track and driving information only, such as distance sensors, angle-to-track axis, or velocity vectors. Thus, the competition asks for strategies that are sensory-motorically grounded rather than strategies that can be designed (online or even offline) by an external observer that has full track knowledge. Moreover, the competition enforces the development of rather general driving strategies since optimization is on driving success in general rather than driving success on one particular track. This paper describes the steps taken to develop COBOSTAR, an autonomous racing car strategy with several general, context-dependent behavioral modules and strategic advancements. Most of the behavioral parameters were optimized with covariance matrix adaptation evolutionary strategy techniques. COBOSTAR won the simulated car racing competition at the IEEE Congress of Evolutionary Computation (CEC 2009) and there is still lots of room for further optimizations and strategy additions. Apart from describing the COBOSTAR racer in detail, we also outline possible next steps and future challenges.

I. INTRODUCTION

The simulated car racing environment TORCS [1] (TORCS stands for “The Open Racing Car Simulator”) provides an open-source car racing environment with a highly realistic track and car simulation engine. TORCS is an advanced simulation environment that was derived and further developed from the robot auto racing simulator (RARS) [2]. The competition setup that enforces the usage of only local information was first designed for the 2008 competitions at the IEEE World Congress on Computational Intelligence (WCCI 2008) and the IEEE Congress of Evolutionary Computation (CEC 2008) [3]. Further information on the current simulated car racing championship can be found on the web [4]. A screen shot of the game is shown in Figure 1 to illustrate its full graphics support, besides the advanced underlying physics simulator that controls the driving behavior of the car.

Most previous approaches for solving the simulated racing car challenge were hand-crafted rule-based and only slightly optimized in several respects or, on the other hand, controlled with very general evolutionary neural network approaches [5], [4]. At WCCI 2008, a very general Neural Evolution of Augmenting Topologies (NEAT) [6] setup won [5], which was designed by Matt Simmeron. In this case the full

The authors are with the Department of Psychology, Cognitive Psychology III, COBOSLAB, and the Department of Computer Science, Chair of Artificial Intelligence and Applied Informatics, at the University of Würzburg, Germany, (phone: +49-931 888 82808; fax: +49-931-31 2815; email: butz@psychologie.uni-wuerzburg.de, thies.loenneker@stud-mail.uni-wuerzburg.de).



Fig. 1. The TORCS racing game is not only a fun game to play, but it offers also the possibility to design novel driving strategies.

sensory information was used and mapped onto the behavior outputs with a constraint NEAT approach. While this success certainly points to the general, powerful applicability of the NEAT algorithm, we decided to challenge this approach with a more constrained controller that maps distance and other sensory information only indirectly onto speed and steering behavior.

In the spirit of Braitenberg vehicles [7], we generate an advanced sensory-to-motor mapping without any internal representations. However, seeing that the body morphology of the racing car is not very well known (despite being generally deducible) and that the track properties are accessible only via the limited sensor capabilities, it is practically impossible to set the direct sensory-to-motor mapping parameters by hand. However, they can certainly be optimized by common computational optimization techniques.

The developed “COBOSTAR” racer (which is a loose acronym for “COgnitive BOdySpaces for Torcs-based Adaptive Racing”) maps sensory information onto actual motor behavior indirectly, mediated by a flexible function that converts sensory information, and particularly mostly the distance and angular information in the currently longest distance sensor, to desired steering angle and desired speed. Given the computed current desired speed and actual speed, actual decisions for accelerating or braking the car as well as for gear shifting are made. The mappings were optimized for on-track behavior as well as for off-track behavior, with a different mapping since distance sensor information is only available on-track. Moreover, the COBOSTAR racer was further optimized including partially hand-coded features such as anti-lock braking (ABS) and anti-slip regulation

(ASR), gear shifting, recovery when stuck, as well as strategy adjustments after the first lap was raced.

Before we provide the details on the COBOSTAR racing strategy, however, we give an overview of related work and the TORCS competition setup.

II. RELATED WORK

Previous approaches to car racing were already developed for the forerunner of TORCS, the robot auto racing simulator (RARS) [2]. For example, Stanley et al. [8] developed a car racing strategy that depended on range-finders and developed a sensory-motor mapping with the incremental evolution of augmenting topologies (NEAT) approach. In this case, NEAT evolved a growing neural network (NN) topology for the control of a racing car that perceived the track with eight range sensors and controlled speed and steering. Interesting behaviors emerged such as the behavior to approach a curve from the outside and then cut through on the inside. This evolved control strategy was then further evaluated to develop a crash warning system, again with the NEAT approach [9], [8]. While the reported results point into an interesting direction, it remains an open question if NEAT is the most appropriate algorithm for such car racing competitions. Most recently, a genetic programming approach was used to challenge this question introducing a highly unconstrained strategy evolution mechanism, the success of which is still to be determined [10].

From a more general perspective, Togelius and Lucas [11] argue that simulated car racing provides a scalable and relevant testbed for evolutionary robotics. The authors particularly investigate ways to generate generally applicable strategies—those that are evolved on one track but yield high performance on many, possibly very different tracks. Moreover, they show that such generally applicable controllers can be further optimized for particular tracks by allowing sensor mutations. However, it remains unclear what was the best way to generate a general controller—Togelius and Lucas were using an incremental approach by training on progressively more complex tracks.

Togelius, Lucas, and De Nardi [12] provide an interesting overview on the diverse computational intelligence challenges that car racing games can pose. These include the challenge to optimize, that is, optimize a given setup, strategy, or the like, to innovate, that is, to develop new or emergent behavior from a given input-output space, or to imitate, that is, to mimic player strategies in an optimal way. Moreover, they point out that there are several challenges in these games including the obvious development of actual controllers—racing alone or in an actual competition—but also the generation of novel, interesting, challenging, fun tracks and the advanced modeling of driving behaviors or actual players. Our COBOSTAR racer adheres to the proposed optimization and innovation principles relying mostly on the available range-finder sensors, because these sensors are the ones that allow the best anticipation of the track properties ahead.

The challenge of imitating other players and optimizing driving behavior meanwhile was taken on with a multi-objective approach [13]. Although it remains a challenge to measure the believability in mimicking players behaviors, it was shown that good multi-objective strategies could be evolved. Nonetheless, the authors propose the incorporation of better sequence learners—such as advanced NNs—to be able to model the players' behavior more exactly. A first approach that uses Long Short-Term Memory recurrent neural networks [14] for robot car control can be found elsewhere [15].

In comparison to the mentioned approaches, the COBOSTAR racer optimizes parameters of a designed baseline strategy. Thus, the optimization process is more constrained than in most of the above approaches, which mostly use general purpose NN approaches combined with evolutionary optimization techniques. The COBOSTAR setup consequently does not enable the development of a large variety of strategies but may be able to optimize its own strategy to a higher degree, and, due to the imposed constraints, may be forced to develop a sufficiently general strategy simply because further strategy differentiations are not representable. Moreover, the parameters of the COBOSTAR racer were not only optimized, but modularly optimized for different driving challenges, such as on- and off-road driving, and additional behavioral features were added to yield the final driving strategy, which is composed of several behavior modules.

III. TORCS COMPETITION SETUP

The TORCS competition “Computational Intelligence in Games” [4] provides a two-component package, which consists of a server component, which modifies the general TORCS setup [1] and allows the evaluation of controllers on particular track setups, and a client component, which hosts the evolving strategy and some additional control capabilities that allow, for example, track restarts. The client module communicates with the server module by means of data packages.

The main novelty of the competition is the fact that there is no complete track information available, but only simulated local sensory information. In particular, the available sensors are an angle sensor, which specifies the current angle between the car direction and the track axis, the current speed in longitudinal and transverse axes of the car, 19 range sensors, which sample the free track space in front of the car and are only valid while on track, 36 opponent sensors, which notice opponents around the car, the current engine speed in rounds per minute, the current gear, the track position with respect to the track edges, and the current rotation speeds of the four wheels. Moreover, there was further racing information available including the current lap time, the damage of the car, the distance from the start line along the track line, the total distance raced, the amount of fuel remaining, the last lap time, and the standing in the race. For car control, there was a gas pedal and a brake pedal, gear shifting, and steering values available.

Thus, while the strategy cannot use any information on future track properties (such as which curve comes next, etc.) it can (implicitly) derive local track information (such as how narrow the current curve seems to be) from the track distance sensors. Note that the track distance sensors are the only ones that provide some predictive knowledge, which is essential for any anticipatory behavior approach [16]. Thus, future track knowledge can be implicitly used by exploiting the information in the distance sensors most effectively.

IV. COBOSTAR DESIGN

The COBOSTAR strategy may be divided into optimized on-track and off-track strategies as well as additional “goodies”, which yield further behavioral improvements. While the on-track strategy mainly relies on the distance and direction of the longest range sensor to the track boundaries, the off-track strategy relies on the angle-to-track axis as well as its relative position to the track center.

A. On-Track Strategy

The main component of the on-track strategy relies on the 19 distance sensors that scan the area in front of the car and reflect at the track borders. Figure 2 visualizes the distance sensors and the contained information. In order to prevent the consideration of distance measures that actually point backwards, first, those distance sensors are excluded from the 19 that point in the reverse track direction, which can be easily accomplished by considering the current angle-to-track axis. Next, the angle α' and distance d of the largest distance sensor are determined and used to compute target velocity v_t and current steering angle τ . However, before the computation the angle α' is further adjusted by considering the two neighboring distance measurements d_l and d_r . The adjustment is as follows:

$$\alpha = \alpha' - 0.5 + \frac{d - d_l}{2d - d_l - d_r} \quad (1)$$

In this way, the angle is further biased to point towards the direction of the larger neighboring distance.

The method that converts the distance d and angle α into the target velocity v_t first determines a target velocity v'_t :

$$v'_t = p_1 + p_2 d + p_3 \max\{0, \frac{d - \theta_1}{\theta_2 - \theta_1}\}^{p_4} - p_5 \left(\frac{\alpha - 9}{9}\right)^{p_6}, \quad (2)$$

as long as $d < \theta_2$. Otherwise, v'_t is set to 1000. The equation can be tuned by six parameter values p_i and additional two threshold values θ_i . In essence, the equation determines the target speed out of a constant, linear, and polynomial component, which depend on the distance measure and as an additional subtraction component, which takes the measured angle into account. Finally, the target velocity v_t is set to the minimum speed defined in parameter p_7 given the determined target speed from Equation 2 yields a velocity below p_7 or to v'_t otherwise. The constant 9 comes from the fact that the angle α is a value in the range of $[0, 18]$ with the value 9 pointing straight ahead. Figure 3 shows the mapping from

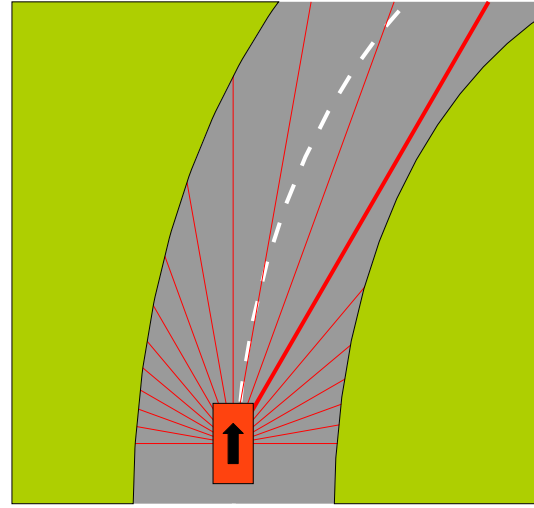


Fig. 2. The distance sensors are used to determine heading direction and current target speed.

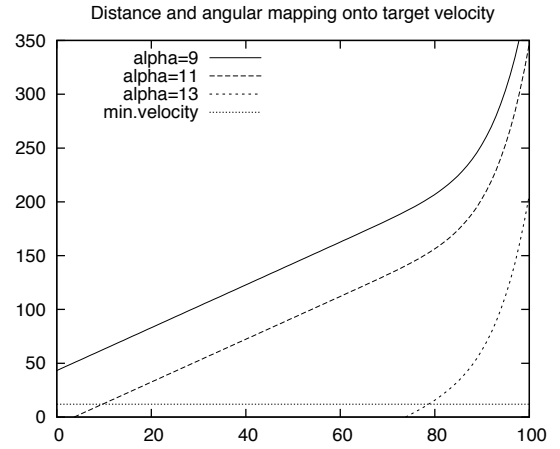


Fig. 3. The mapping function from Equation 2 shows how offset, linear, and polynomial component shape the mapping from the maximal distance sensor to the target speed. Moreover, the different curves show the influence of the angle α of the maximum distance sensor on the target speed. Note that the distance sensors have a limited range of $[0, 100]$.

the distance sensor to the velocity for several angle cases for the parameter setting used in the optimized strategy of COBOSTAR.¹

The target velocity v_t is then compared with the current velocity v_c . If the current car velocity is smaller than the current target velocity ($v_c < v_t$) then the car accelerates maximally. Otherwise, the strategy still scales the braking level. Neither brake nor acceleration are applied if the current velocity v_c is smaller than $p_8 v_t$ (that is, $v_t \leq v_c < p_8 v_t$, where p_8 was optimized to $p_8 = 1.13$). If the current velocity is even larger, then the brake is applied increasingly strongly

¹The exact optimization values are $p_1 = 43.23$, $p_2 = 1.99$, $p_3 = 104.7$, $p_4 = 9.38$, $p_5 = 907.6$, $p_6 = 1.92$, and $p_7 = 11.89$. The thresholds were optimized to $\theta_1 = 36.50$ and $\theta_2 = 97.33$.

using the following scaling equation:

$$b = \min\{1; p_9(v_c - p_8 v_t)\}, \quad (3)$$

where slope parameter p_9 evolved to $p_9 = 0.70$. In this way, gradual braking is applied. This was expected to yield particularly effective car control in curves that become increasingly tight, since in this case immediate full braking may lead to uncontrollable car behavior.

The steering is accomplished with a rather straight-forward mapping, by converting the target angle into the steering direction using:

$$\tau = p_{10}(9 - \alpha), \quad (4)$$

where parameter p_{10} was optimized to $p_{10} = 0.39$.

The final two parameters that were optimized on track were the two parameters for anti-lock braking (ABS), which came with the basic competition code. Slip and range for ABS were optimized on track to 11.7 and 10.18, respectively. Anti-slip regulation (ASR) was not applied while driving on track.

B. Off-track Strategy

When the car leaves the track due to a crash, or exaggerated speed in a curve, or even a simple steering mistake, the situation significantly changes. First, the distance sensors to the track borders are not available any longer, seeing that the car is off-track. Moreover, steering becomes much more error-prone and also the wheel slippage is much stronger. Thus, a very different driving strategy needs to be applied.

The target speed off-track was designed to rely on a constant offset plus a linear scaling depending on the difference between a target angle β and the currently sensed angle to track value γ . The target angle was computed from the current relative position to the track middle axis t as follows:

$$\beta = \text{sgn}(t)(\text{abs}(t) - q_1)q_2 \quad (5)$$

The target speed was then determined by combining this target angle with the current angle to track and a constant offset yielding:

$$v_t = q_3 + q_4(\max\{0, 1 - q_5 \text{abs}(\gamma - \beta)\}), \quad (6)$$

where the optimized parameters were $q_1 = 0.392$, $q_2 = 0.150$, $q_3 = 117.5$, $q_4 = 123.6$, and $q_5 = 34.56$. Thus, the target speed was determined by offset q_3 and further scaled by the angle of heading: the faster, the more the car is heading towards the track. The target velocity v_t is then handled as in the on-track strategy to determine if the car should accelerate or brake.

Moreover, anti-slip regulation (ASR) was applied off-road by measuring the slip of the wheels. To do so, the average speed of all wheels was computed out of the four currently sensed wheel velocities. The slip of the wheels was then defined as the difference between the average wheel velocity and the sensed current velocity of the car. If the difference between the resulting negative slip was larger than parameter $q_6 = -10.01$, then the acceleration of the car was decreased

by the subtracter $(\text{slip} - q_6)/q_7$, where parameter q_7 evolved to $q_7 = 155$.

Finally, if nothing else works anymore, that is, if the car is totally stuck and not moving much at all, then a hard-coded stuck-behavior was applied, which backs up the car by switching to reverse gear and inverting the usual steering. The car will stay in this mode until either the angle to track axis is halved, compared to the value measured when entering stuck mode, or it gets stuck again while in reverse gear. The number of iterations to wait until the stuck behavior is applied was evolved to parameter $q_8 = 53.3$, whereby the stuck counter was increased when the current speed was below an evolved stuck speed parameter $q_9 = 2.03$.

In sum, the off-road strategy chooses either to backup the car if nothing else works anymore or to try and head towards the track again with a speed that mostly prevents excessive slippage or even spinning. On-road and off-road behaviors were evolved by evolving the mentioned parameters p_i and θ_i on-road as well as the parameters q_i off-road. The following section details how this was done.

C. Parameter Optimization

Parameter optimization was accomplished by the covariance matrix adaptation (CMA) evolution strategy [17], using the available online Java code [18]. The optimization underwent first an on-track, then an off-track, and finally another very extensive on-track optimization turn. Further optimization turns could certainly yield even better controllers, but time (time to setup further turns rather than computational time) prevented further optimization efforts. Moreover, all parameters were optimized on various available tracks in TORCS and then compared on the other tracks for their generality. The most general parameter set was finally chosen and entered into the CEC 2009 competition [4].

1) *On-Road Optimization*: Including the ABS parameters, there were a total of 14 parameters to optimize for the on-road optimization. For the on-road optimization, the fitness was the inverse of the distance raced (plus one to avoid divisions by zero), which had to be minimized (in this case fitness may also be termed a ‘‘cost’’). We simply used the standard setting from the CMA code, with an evolution strategy of $(\lambda = 11 + \mu = 1)$.

To run CMA successfully, important additional choices had to be made for the initialization of the initial means and standard deviations. Respecting the expectable value range and behavioral sensitivity, we set the parameters as follows: $p_1 = 2.0$, $p_2 = 2.6$, $p_3 = 110$, $p_4 = 1.6$, $p_5 = 25$, $p_6 = 2.0$, $p_7 = 1.2$, $p_8 = 0.79$, $p_9 = 0.55$, $p_{10} = 0.11$, $\theta_1 = 47$, $\theta_2 = 90$, $\text{abs}_1 = 7.5$, and $\text{abs}_2 = 3.2$ for the respective means and $\sigma p_1 = 10.0$, $\sigma p_2 = 0.5$, $\sigma p_3 = 30.0$, $\sigma p_4 = 1.5$, $\sigma p_5 = 100$, $\sigma p_6 = 1.0$, $\sigma p_7 = 10.0$, $\sigma p_8 = 0.4$, $\sigma p_9 = 0.3$, $\sigma p_{10} = 2.0$, $\sigma \text{abs}_1 = 20.0$, and $\sigma \text{abs}_2 = 5.0$ for the standard deviations. Despite careful considerations it is clearly impossible to exclude the possibility that even more diverse parameter settings could have yielded even better results. Nonetheless, the resulting best parameter settings achieved during each track were very diverse indeed. Figure 4

TABLE I

THE TABLE SHOWS THE DISTANCES COVERED ON PARTICULAR TEST TRACKS GIVEN THE OPTIMIZED PARAMETER SETTINGS FROM OPTIMIZATION RUNS ON PARTICULAR (NOT NECESSARILY SAME) TRACKS. BEST VALUES FOR EACH TRACK ARE IN BOLD, VALUES THAT WERE OPTIMIZED FOR A PARTICULAR TRACK HAVE AN ADDITIONAL '*' SYMBOL. THE TEST TRACK ROWS ARE ORDERED BASED ON THE MAXIMUM DISTANCE VALUE ACHIEVED BY A PARAMETER SETTING. THE COLUMNS ARE ORDERED BY THE MEAN POINTS ACHIEVED IN THE TEST TRACKS. THE MEAN AND MEDIAN INFORMATION CONFIRMS THAT THE PARAMETER SET THAT WAS OPTIMIZED ON E-TRACK 6 YIELDS THE BEST RESULTS OVER ALL TRACKS. THE STRONG DIFFERENCES IN MINIMUM AND MAXIMUM PERFORMANCE VALUES NOURISH THE SUSPICION THAT PARAMETER SETTINGS BECOME OPTIMIZED FOR PARTICULAR TRACKS OR TRACK TYPES. THE TESTS OF OPTIMIZED PARAMETERS ON OTHER TRACKS REVEAL THAT NOT EVEN NECESSARILY THOSE PARAMETERS YIELD BEST RESULTS ON A TRACK THAT WERE PARTICULARLY OPTIMIZED FOR THAT TRACK.

TestTrack	track optimized on										statistics			
	Aalb.	Stre1	Ruud.	E-Tr2	Forza	Spring	E-Road	Wheel1	E-Tr4	E-Tr6	mean	dev.	min	max
Aalb.	6812*	5482	4062	5488	1699	5482	5904	2896	4905	5975	4870	1476	1699	6812
Alpi2	4788	7759	7479	7847	7410	7732	7839	7702	7395	7668	7362	872	4788	7847
CG t3	2247	8123	6923	7600	7594	7300	6576	7967	7761	7641	6973	1635	2247	8123
E-Tr2	3563	7968	6578	8189*	3617	6084	6085	7799	6716	7863	6446	1608	3563	8189
Spring	775	5926	6880	7538	5965	8711*	8507	8065	7080	8246	6769	2205	775	8711
Stre1	1506	9109*	8556	8777	8540	8900	8833	8688	8511	8754	8017	2177	1506	9109
Ruud.	510	7884	8274*	8008	7980	8244	7846	7286	8429	9293	7375	2339	509	9293
Alpi1	8439	8784	8447	8849	9246	8743	8769	8817	8902	8688	8768	217	8439	9246
E-Tr6	1059	7321	7164	6566	7001	7156	7352	6331	8100	9932*	6798	2135	1058	9932
OLR1	8222	9658	9610	9349	10006	9093	9208	9547	9830	9577	9410	473	8222	10006
Wheel2	2844	7718	7769	9322	9065	10022	9940	9962	9667	9908	8622	2094	2844	10022
E-Road	1855	3979	3991	8525	8761	9592	10342*	10060	9925	9996	7703	2998	1855	10342
CG Sp1	3324	9300	7665	8871	9685	8884	10033	10105	9646	10450	8796	1974	3324	10450
E-Tr3	7934	6248	8303	8549	9125	8485	9153	9546	9196	10730	8727	1108	6248	10730
Wheel1	3522	9202	8703	8402	8445	9465	6753	11027*	10009	9840	8537	1993	3522	11027
CG t2	10147	2903	10955	11201	11787	11585	11795	11790	11734	11439	10534	2590	2903	11795
Forza	9844	9541	9906	10351	12035*	10494	10657	10839	10597	10916	10518	663	9541	12035
E-Tr4	10303	8651	12070	5635	12809	8133	12067	12834	13157*	12574	10823	2423	5635	13157
mean	4872	7531	7963	8282	8376	8561	8759	8959	8976	9416	8169	1212	4872	9416
median	3542	7926	8022	8463	8651	8727	8801	9181	9049	9709	8207	1633	3542	9709

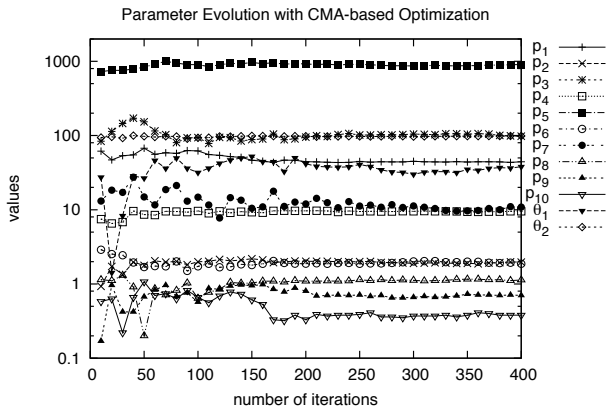


Fig. 4. The parameter evolution of the most generally applicable parameter set shows how CMA typically yields a strong parameter search early on and then converges to one particular parameter setting.

shows the evolution of the parameters of the best individual over successive generations of the run that yielded the finally used parameter values over 400 generations. Initially, the variance is quite high so that strong parameter changes can be seen in the successively best individuals. Later on, the values stabilize and, if at all, small local parameter changes enforce the system to climb towards a local driving strategy optimum in the track.

These observations plus the general suspicion that the parameter-to-strategy-to-performance mapping is highly non-linear and the fitness is noisy lead us to the independent optimization of many parameter settings on many tracks. In

fact, even on the same track, the final optimized parameters were not necessarily similar in successive evolutionary runs. It appears that CMA, despite identical initialization except for the random number generator, runs each time on a different search path to a different local optimum, as expected. We evolved more than twenty parameter settings on different TORCS tracks. Table I shows the performance of the best ten strategies on a number of test tracks. The table orders the columns by overall mean performance on the test tracks and the test track rows by the maximum performance achieved on the respective track.

Several observations can be made. First, the best performance on each track is achieved by different parameter settings. The most wins (4) are achieved by the right-most strategy, which is the one that was finally used. However, the second most wins (3) are achieved by the 6th best strategy when considering the overall means. Thus, it is clear that different behavioral strategies are particularly suitable for particular track types. Interestingly, the strategy that was optimized in E-Track 6 was the most generally applicable one, which is also the one with points always on or above the mean track performance values. Second, the best strategy for a particular track is not necessarily the one that was also optimized on that track. For example, for the Ruudskogen track, the strategy that was optimized on the track (column 3, value=8274) does perform rather badly compared to the best strategy on the track (column 10, value=9293) — a strong sign for a premature convergence to a local optimum. Third, also the difference between worst (min) and best (max) performance on each track point out that the resulting

TABLE II

THE DIFFERENT CMA OPTIMIZATION RUNS PARTIALLY YIELD RADICALLY DIFFERENT PARAMETER SETTINGS, EACH OF WHICH APPEAR PARTICULARLY OPTIMIZED BUT NOT NECESSARILY OPTIMALLY OPTIMIZED FOR THE TRACK TRAINED ON.

Parameter	track optimized on										statistics			
	Aalb.	Street1	Ruud.	E-Tr2	Forza	Spring	E-Road	Wheel1	E-Tr4	E-Tr6	mean	dev.	min	max
p_1	-6	30.76	7.63	35.57	26.79	18.22	12.61	8.64	56.25	43.23	23.37	17.78	-6	56.25
p_2	2.73	0.58	2.7	2.04	1.38	2.24	2.44	1.48	2.33	1.99	1.99	0.64	0.58	2.73
p_3	303.5	95.48	74.55	33.88	83.45	168.	267.5	118.4	2.42	104.8	125.3	91.11	2.42	303.5
p_4	7.66	5.08	4.26	6.95	0.3	9.3	11.88	1.19	1.29	9.38	5.73	3.76	0.3	11.88
p_5	234.9	922.4	537.2	873.4	738.0	1148	573.7	558.6	524.7	907.7	701.9	251.8	234.9	1147
p_6	3.33	2.26	1.95	2.5	1.66	1.92	1.6	1.27	1.07	1.92	1.95	0.61	1.07	3.33
p_7	-23.8	66.2	18.42	25.87	101.0	20.92	7.5	23.16	16.98	11.89	26.81	32.29	-23.8	101.0
p_8	1.27	1.12	0.63	1.1	1.1	1.19	1.16	1.1	1.08	1.13	1.09	0.16	0.63	1.27
p_9	1.07	0.42	1.03	1.26	0.25	1.07	1.1	0.56	0.38	0.7	0.78	0.34	0.25	1.26
p_{10}	1.86	0.34	0.91	0.36	0.46	0.46	0.42	0.37	0.28	0.39	0.58	0.46	0.28	1.86
θ_1	16.72	-197.4	-73.73	-38.52	-7	0.19	4.96	-43.78	50.48	36.5	-25.16	67.61	-197.4	50.48
θ_2	98.57	62.16	85	74.72	99.41	84.8	95.25	94.28	96.7	97.33	88.82	11.63	62.16	99.41
abs_1	22.22	16.64	11.86	6.49	15.22	7.76	3.09	5.49	8.14	11.7	10.86	5.56	3.09	22.22
abs_2	15.94	16.7	15.48	12.7	5.98	2.79	8.93	2.01	15.67	10.18	10.64	5.28	2.01	16.7

strategy parameters do not guarantee particularly general performance.

Table II shows the generated optimized parameter settings. Here, the high standard deviations of the resulting parameters and large differences between minimum and maximum values point out that the parameters are rather specialized for particular tracks. For example, the strong differences in offset values (p_1) suggest that some strategies realized the velocity gain more with the linear component (p_2), while others used a larger offset and a smaller linear component. Overall, the diversity in the optimized parameter sets shows that similarly good strategies were achievable with a variety of parameter combinations. Each of the sets results in a specialized strategy with its particular advantages and disadvantages. As a consequence, with the optimization and strategy mapping function chosen, it seemed a very essential step to choose that parameter set that yields the most robust performance overall.

2) *Off-Road Optimization*: The biggest challenge for the off-road optimization part was to generate a useful fitness signal. If the car is controlled by a reasonably good strategy then the car will hardly ever reach “off-road”. Thus, the fitness the controller achieves hardly depends on the off-road strategy. To generate a more useful and meaningful fitness signal, we thus used an evolved strategy from the settings described above and added a “crash strategy”. This detrimental driving strategy was applied every 300 meters and caused the car to drive in alternate trials completely to the left or right. Thus, the distance raced mainly depended on the quality of crash recovery and thus on the capability of the car to drive between crash behaviors as fast as possible. Thus, the fitness value stayed the same as in on-road optimization but the strategy had the enforced crash behavior activated every 300 meters.

We did not analyze the optimization of the off-road behavior in as much detail, but generally the results showed similar parameter variations as the optimized parameters for the on-road strategy. For off-road optimization, the parameters for CMA were initialized to means of $q_1 = 0.3$, $q_2 = 0.2$, $q_3 = 70$, $q_4 = 100$, $q_5 = 20$, $q_6 = 10$, $q_7 = 20$,

$q_8 = 100$, and $q_9 = 1$ and deviations of $\sigma_{q_1} = .1$, $\sigma_{q_2} = .2$, $\sigma_{q_3} = 10$, $\sigma_{q_4} = 20$, $\sigma_{q_5} = 4.0$, $\sigma_{q_6} = 3.0$, $\sigma_{q_7} = 2.0$, $\sigma_{q_8} = 10.0$, and $\sigma_{q_9} = 1.5$. These yielded the parameter values reported above for the off-road strategy. The off-road strategy optimization was done *before* the on-road optimization reported above. During on-road optimization, the off-road strategy stayed fixed.

D. Further Strategy Modifications

A simple not yet mentioned strategy modification is gear-shifting. Here, we stayed on the simple side and shifted gears dependent only on the rounds-per-minute engine rotations currently measured. This was the standard procedure also provided with the basic controller given by the organizers. Tests on a few tracks suggested, however, that the car’s engine worked best when using the full spectrum of its gears. We shifted up each time when the engine reached a speed of 9500 rounds-per-minute and shifted back down when the rounds-per-minute sensor dropped below 3300, 6200, 7000, 7300, and 7700 for gears 2, 3, 4, 5, and 6, respectively. No further evaluations nor optimizations were done for gear-shifting.

Furthermore, we added a few additional strategy components, which need to be optimized in the future to gain their full potential. Nonetheless, we consider the ideas worthwhile to share:

1) *Large Track Adjustments*: As mentioned above, the angle estimate of the furthest distance is interpolated between the maximal distance sensors and its neighbors (see Equation 1). Experiments on very wide speed-way tracks showed that this modification resulted in slightly irregular driving behavior, that is, the car would drive on a slightly wavy trajectory rather than straight. To avoid this, we had the car measure the track width with its distance sensors upon the start of the race, assuming that the track width at the start reflects the track properties throughout the race. If the measured distance exceeded the hand-set threshold 28 the steering factor parameter p_{10} was set to 0.1 and the angular adjustment, which was usually set to 0.5, was set to 0.1 (see Equation 1). In this way, excessive steering was

prevented and irregularities in the neighboring sensors were not taken as strongly into account. The result was a much more straight driving behavior reaching maximum speeds that were impossible without these adjustments.

2) *Second Lap Adjustments*: Several additional adjustments were made when the car entered the second and possibly successive laps. Seeing the best performance on the Aalborg track was also the one optimized on that track and seeing further more that this performance excelled the others by far, we decided to analyze the general properties in the first lap and adjusted our strategy parameters according to these measurements in the second lap. Consequently, we adjusted to the Aalborg strategy (column 1 in tables I and II) if the first lap average speed was below 29—indicating a really curvy track. We adjusted to the Street 1 strategy (column 2 in tables I and II), if the average speed of the first lap was below 43. Moreover, for very fast tracks, the Wheel 1 strategy (column 8 in tables I and II) was chosen if the minimum speed (after the initial speed-up) was never below 61. In this way, we switched to a more aggressive strategy if the track appeared easy and to a more conservative strategy if the track appeared difficult, that is, curvy. This strategy enhancement resulted in a distance gain of 1.67% averaged over all tested tracks.

Besides this strategy adjustments, we also added a crash-point strategy, remembering crash points on the road. If a crash occurred in the first lap somewhere and if furthermore this crash appeared not to be caused by an opponent (no opponent in immediate vicinity during crash), then the relative point on the track with respect to the start was remembered and 100 meters before this location the strategy was converted into a passive strategy, leading to a much slower driving style. While this strategy seemed to have big potential, so far the definition of the passive strategy was very ad-hoc—the car simply drove very slow around the crash area. This often lead to overly passive behavior, so that in the submitted controller this strategy was used only in very severe crash situations.

3) *Avoiding Opponents*: It came to our attention that the racing cars suffer the most damage if they continuously crash into other cars. In fact, this damage increases even upon really soft touches, so that it seems a good idea to avoid these. However, our time ran out so that we did not include any explicit opponent avoiding strategies but were rather hoping that the opponent trajectories would be sufficiently different from the one of our racer so that our racer would overtake automatically when the trajectory ahead was temporarily not blocked. In future strategies, we intend to improve this behavior and particularly avoid continuous touches with cars immediately in front.

V. CONCLUSIONS

The TORCS racing car competition [4] is still rather young and still a very big challenge with lots of room for improvement. Since only local track information is available, it is necessary to generate a general behavioral strategy that does not assume too much about the track, and particular about

its curves. The strategy needs to drive not too aggressively but also not too passively when averaged over all relevant tracks.

By using CMA evolution strategies, we could expect that local strategy optima are found reliably. However, it was clear from the beginning that these will not necessarily be generally suitable. Thus, the evolutions of multiple strategies on multiple tracks and finally the choice of the most generally successful parameter set was necessary to generate a generally competitive control strategy. Moreover, the off-road optimization strategy ensured that the car can also behave reasonably off-road and consequently recover from a crash successfully. The add-ons to the basic sensory-to-motor mappings completed our COBOSTAR controller and still leave lots of room for further improvements.

It is hoped that the descriptions and ideas conveyed in this document will not only be useful to be able to re-design the current COBOSTAR controller but will also be useful to actually create more advanced controllers that combine several sensory-to-motor mappings, additional strategies, and even online adaptation mechanisms (such as the ideas for second lap behavior adjustments). COBOSTAR will be around and may serve as a useful competitor. But be aware, the work on COBOSTAR is not completed and we hope to generate even more robust COBOSTAR controllers for future TORCS competitions.

ACKNOWLEDGMENTS

The authors acknowledge funding from the Emmy Noether program of the German research foundation (grant BU1335/3- 1) and like to thank the COBOSLAB team as well as Prof. Puppe and the whole chair of artificial intelligence and applied informatics of the computer science department at their university.

REFERENCES

- [1] (2007) Torcs, the open racing car simulator. [Online]. Available: <http://torcs.sourceforge.net/>
- [2] (2006) Robot auto racing simulator. [Online]. Available: <http://rars.sourceforge.net>
- [3] D. Loiacono, J. Togelius, P. L. Lanzi, L. Kinnaird-Heether, S. M. Lucas, M. Simmeron, D. Perez, R. G. Reynolds, and Y. Saez, "The WCCI 2008 simulated car racing competition," *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, 2008.
- [4] (2009) Simulated car racing championship. [Online]. Available: <http://cig.dei.polimi.it/>
- [5] (2008) Wcci simulated car racing results. [Online]. Available: <http://cig.dei.polimi.it/?cat=4>
- [6] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10(2), pp. 99–127, 2002.
- [7] V. Braitenberg, *Vehicles: Experiments in Synthetic Psychology*. Cambridge, MA: MIT Press, 1984.
- [8] K. Stanley, N. Kohl, R. Sherony, and R. Miikkulainen, "Neuroevolution of an automobile crash warning system," *Genetic and Evolutionary Computation Conference, GECCO 2006*, pp. 1977–1984, 2006.
- [9] N. Kohl, K. Stanley, R. Miikkulainen, M. Samples, and R. Sherony, "Evolving a real-world vehicle warning system," *Genetic and Evolutionary Computation Conference, GECCO 2006*, pp. 1681 – 1688, 2006.
- [10] M. Ebner and T. Tiede, "Evolving driving controllers using genetic programming," *Computational Intelligence in Games*, vol. IEEE CIG 2009, in press.

- [11] J. Togelius and S. M. Lucas, "Evolving robust and specialized car racing skills," *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. CEC 2006, pp. 1187–1194, 2006.
- [12] J. Togelius, S. M. Lucas, and R. D. Nardi, "Computational intelligence in racing games," in *Advanced Intelligent Paradigms in Computer Games*, N. Baba, L. C. Jain, and H. Handa, Eds. Berlin Heidelberg: Springer-Verlag, 2007, pp. 39–70.
- [13] N. van Hoorn, J. Togelius, D. Wierstra, and J. Schmidhuber, "Robust player imitation using multiobjective evolution," *Proceedings of the Congress on Evolutionary Computation (CEC-09)*, pp. 652–659, 2009.
- [14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, 1997.
- [15] D. Wierstra, A. Foerster, J. Peters, and J. Schmidhuber, "Recurrent policy gradients," *Journal of Algorithms*, in press.
- [16] M. V. Butz, O. Sigaud, and P. Gérard, Eds., *Anticipatory Behavior in Adaptive Learning Systems: Foundations, Theories, and Systems (LNAI 2684)*. Berlin Heidelberg: Springer-Verlag, 2003.
- [17] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, pp. 159–195, 2001.
- [18] N. Hansen. (2008) CMA evolution strategy source code. [Online]. Available: http://www.lri.fr/~hansen/cmaes/_inmatlab.html