A Reactive Mutli-agent Approach to Car Driving using Artificial Potential Fields

Tim Uusitalo, Stefan J. Johansson, *Member, IEEE*

***Abstract*— The Car Racing championship initiative is a platform for evaluating different car control solutions under competitive conditions. We introduce the novel combination of using artificial potential fields as the major control paradigm, and a multi-agent system to deal with the coordination of the different control interests. By combining fields for shortest path, and minimal curvature with a field that makes sure that the car stays on the track, we get a purely reactive car controller. The experimental evaluation of it shows that it performs well, ending at 2nd place in the time challenge, and in the top in the car races against three of the controllers of last year's championship.**

I. Introduction

The Open Racing Car Simulator (TORCS) is a multi platform car racing simulator. It has, since a couple of years back also been an arena for competitive AI research, where bots are created to illustrate the applicability of various solutions to the car control problem that the championship express [20], [14], [13], [12]. Our work is not an exception in that respect, although we believe that it is an exception in the way that we use novel techniques to address the problem.

In 1985 Ossama Khatib introduced the concept of *Artificial Potential Fields* (APFs) while he was looking for a real-time obstacle avoidance approach for manipulators and mobile robots. The technique moves a manipulator in a field of forces. The position to be reached is an attractive pole for the end effector (e.g. a robot) and obstacles are repulsive surfaces for the manipulator parts [10].

Many studies concerning APFs are related to spatial navigation and obstacle avoidance, see e.g. [2], [11], [15]. The technique is really helpful for the avoidance of simple obstacles even when they are numerous. Combined with an autonomous navigation approach, the result is even better, being able to surpass highly complicated obstacles [1]. However most of the premises of these approaches were only based on repulsive APFs of the obstacles and an attractive potential in some goal for the robot [22].

In the last decade some other interesting applications for APFs have been suggested. The use of APFs in architectures of multi agent systems (MAS) is giving quite good results defining the way of how the agents interact. Howard et al. developed a mobile sensor network deployment using APFs [8], and APFs have been used in robot soccer to choose actions based on the state of the game [9], [17]. Thurau et al. [19] has developed a game bot which learns reactive behaviors (or APFs) for actions in the First-Person Shooter (FPS) game Quake II through imitation and Hagelbäck and Johansson has applied APFs to Real Time Strategy (RTS) games to control hundreds of units in parallel in real time. They also presented a methodology for combining APFs with MAS, an approach that we will adapt to the current context.

One first attempt to make a car controller using either a MAS or APFs was made already in 1992 by Hassoun et al. [7]. They compared two implementations of car controllers navigating in a simulated city. During the last decade, the use of APFs for driving assistance purposes has been investigated by e.g. Gerdes and Rosetter [5] who studied lane keeping, and Wolf and Burdick, who proposed the use of APFs in highway overtaking [23].

There are several advantages of using a multi-agent approach. There are a number of different factors that needs to be taken into account when controlling the car. Where is the car placed on the road? What is the state of the car in terms of speed, gear, rotational speed, etc.? Where are the opponent cars? How does the track look ahead of the car, and so on. These factors constitute a complex environment in which a controller have to make real-time decisions. Multi-agent systems is a powerful design paradigm which enables a high degree of modularization. In this case, it means that the different factors may be distributed to different agents, improving the maintainability compared to monolithic solutions. Having an agent perspective also fits very well with the APF paradigm, where one may let each force be handled by an own agent.

Previous work related to the Car Racing championship include evolving optimal paths that balance the shortest path and the minimal curvature [4]. They showed that very good results were achievable for sections of a track where an evolved controller balanced between these two paths, see Fig. 1. Inspired by that paper, we also let both these paths have an impact on the control of the car, but without learning the balance between them as Cardamone does [4].

The outline of the report is as follows: First, in Section II, we will adapt the methodology of Hagelbäck and Johansson to the situation of car racing [6]. Then we describe a series of experiments in which we test the bot and describe their results (Section III). In Section IV we discuss the results and their implications in relation to the previous work in the field and we finish with the conclusions in Section V.

II. Building a Car Driver based on APFs

So, how do we best use the advantages of APFs in creating a competitive car driver? Previous work by Hagelbäck and

Both authors are at the School of Computing, Blekinge Institute of Technology, Sweden (phone: +46-455 385 831; email: tim@uusitalo.cc, sja@bth.se).

978-1-4577-0011-8/11/$26.00 ©2011 IEEE

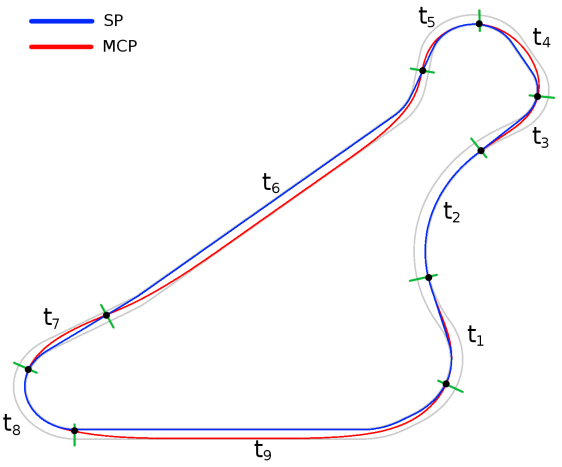

Fig. 1. Shortest Path (SP) and Minimal Curvature Path (MCP) of a track in TORCS. Figure from Cardamone et al. [4].

Johansson [6] lined out a methodology for creating Multi-Agent APF-based players for Real Time Strategy (RTS) games using the following steps:

1) The identification of objects,
2) The identification of the driving forces (fields) of the game,
3) The process of assigning charges to the objects,
4) The granularity of time and space in the environment,
5) The agents of the system, and
6) The architecture of the MAS.

Although they applied it to RTS games, the description is actually general enough to be applicable also in other types of applications and we will follow their methodology. We have based our implementation on the C++ client outlined by Loiacono et al. [12].

A. The objects

In TORCS, there are not that many objects of interest. Basically the objects that are relevant for the decision of what to do are:

- *The track,* as interpreted by the distance sensor input,
- *the car* that the bot tries to control,
- *the competing cars,* as interpreted by the input of the opponent sensors,

B. The driving forces of TORCS

At a macro-level, the major driving force of TORCS is of course to finish first. However, to achieve that will require a number of sub-forces to take into account. We identify the following sub-forces:

- Drive as fast as possible in the right direction around the track.
- Avoid driving outside the track.
- Avoid losing the control of the car.
- Avoid crashing into other cars.

Each of these identified sub-forces is taken care of by a corresponding field. We will use the following fields:

- *Field of Shortest path* (FoSP) which will try to minimize the travelled distance around the track.
- *Field of Track* (FoT) is the field that try to keep the car on the track and avoid getting outside it.
- *Field of Curvature* (FoC) is trying to minimize the curvature in order to be able to maintain the highest possible speed through the curves.

C. The Assignment of Forces

In this step, we identify the different forces used in the different fields. That is, we assign the sizes of the forces and their placement (relative to the own car). Since this will be dealt with by the agents, we refer to that section for the detailed description of the assignment of the forces.

D. The Granularity of Time and Space

In car racing, things may happen fast and APF is a method that enable very fast decisions that may be done more or less every game tick of the game. This is of course no exception, so we will activate the decision making in every game tick. The next question then is of course how many decisions do we have the time to evaluate during one game tick?

Initial tests have shown that having 171 lookahead points, evaluating them in 150 frames per second is no problem. With a maximum speed of around 83 meters per second, that means evaluating positions that are at most appr. 55 centimeters away from the current position. We doubt that a higher granularity actually would improve the performance.

The last question to answer is exactly *what* actions to evaluate. We could potentially evaluate the outcomes of all different combinations of actuators (such as steering and gas/brake control). However, we restrict it to probing 9 different speeds (ranging from from full brake to full gas) in 19 different steering directions in the interval of $[-45, 45]$ degrees.

E. The Agents

Whereas it is quite clear what agents to use in a APF-based solution for RTS games (one per own unit in the game), it becomes less obvious in this case.

- The *Driver Agent* is the agent steering the car and controlling the speed of the car through the throttle and the breaks. Its goal is to maximize the potential in the position it predicts it will be at in the next game tick. This is done by finding and selecting the action that optimize the sum of potentials from the fields.
- The *Shortest Path Agent* tries to find the shortest path around the track. This is done through identifying the direction in which the car has the longest free sight.
- The *Curvature Agent* keeps track of the path that maximizes the minimal speed that the car can keep through a curve by maximizing the curvature radius.
- The *Track Agent* is responsible for the charges used to try to keep the car on the road. This is done by having heavy penalties in form of negative potentials for positions outside the track, making the driver agent choosing other positions if possible.

- The *Navigator Agent* helps the Curvature Agent to identify the direction of the next curve.
- Last but not least, the *Interface Agent* is the agent which interacts with the TORCS server, receiving sensory input, and sending the control commands.

We will now present the detailed description of our agents.

1) The Driver Agent: This is the core agent of the decision process. As an input, it gets the potential fields provided by the other agents. The output is the action that will be chosen to take in the following time step. The way it does that is that, given the current speed, s, and the current steering control, $\alpha \in [-1, 1]$, it calculates a number of lookahead points $L = \{l_1, l_2, ..., l_{n_a}\}$. These points range from sharp turn left (tires at 45 degrees left) to sharp turn right, and from full break, to full gas. Note here, that full gas will not give the same resulting lookahead points if the car stands still, as if it runs at a speed of 80 m/s since it takes the current speed into account. Depending on the current gear, the gas and the speed, the acceleration varies, and so does the expected speed at the end of the current time frame. The distance that we travel in the time interval $[t, t+1]$ is approximated to:

$$d = \frac{(v_t + v_{t+1}) \cdot \delta_t}{2} \quad (1)$$

Figures 2–3 illustrates the position of the lookahead points.

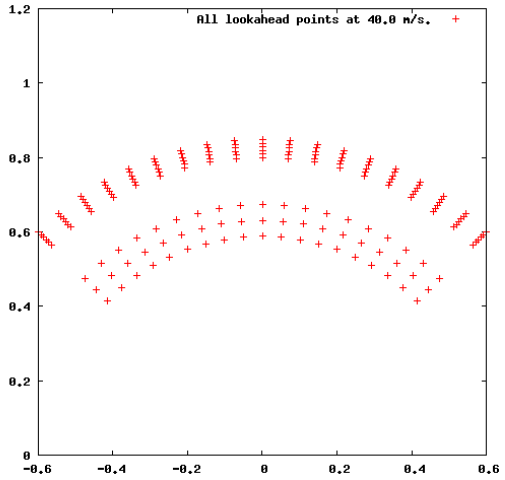

Fig. 2. The lookahead points used at the speed of 40.0 m/s. The scale in x and y is meters.

The driver agent now iterates over L to find the point p_{max} that have the highest potential which is generated by all charges c in all fields $pf \in PF$:

$$p_{max} = \max_{l \in L} \sum_{pf \in PF} \sum_{c \in pf} f_{pf}(dist(l, c), size(c)) \quad (2)$$

where $dist(l, c)$ is the Euclidean distance between the lookahead position l and the position of the charge c. The function f_{fp} defines how to calculate the impact of a certain charge, given its size and the distance to it. After p_{max} has been found, the action associated with that point is sent to the interface agent that in turn forward it to the TORCS server.

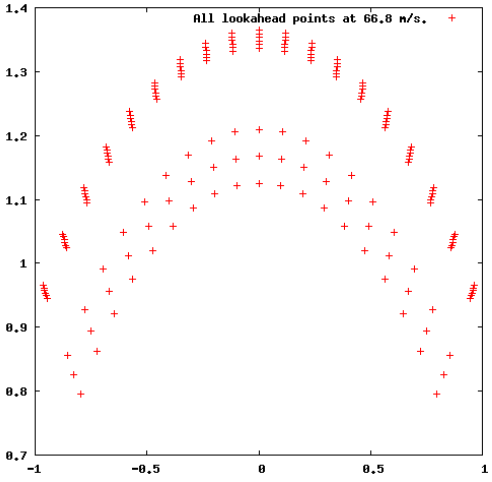

Fig. 3. The lookahead points used at the speed of 66.8 m/s.

2) The Shortest Path Agent: The Shortest Path Agent tries to get the car to follow the shortest path of the track. The idea followed is to let the car go in the direction (of the ones measured by the distance sensors) that has the longest measured distance (in the direction that we should go). By placing an attractive charge in that direction, the driver agent will be more willing to steer the car in the direction of the shortest path, than choosing other directions.

Our agent use all track sensors in the directions α between -90 and 90 degrees relative to the direction of the car, at every 5 degrees if $\alpha \in [-45, 45]$ else at every 15 degrees.

So, how far away should this charge be placed? The further away (along the longest sight line) we place it, the more speed is needed to get close to it. This is of course depending on the maximum distance measured. In order to deal with the noise, we use the average length measured by the 5 frontmost distance sensors and place the charge at the position reached by full gas within one game tick. If the distance is shorter, the attractive charge will be placed closer to the car, which in turn means that to maximize the potential in the position of the car in the next frame, the driver agent needs to start breaking. The function $g(msd)$ for deciding the distance between the car and the charge, given the maximum sight distance, msd is tricky to calculate exactly. We have thus used a simple approximation in:

$$g(msd) = \frac{msd_{-10} + msd_{-5} + msd_0 + msd_5 + msd_{10}}{5} \cdot \delta_t \quad (3)$$

Here, δ_t is the length of the time frame and msd_α is the maximum sight distance in direction α relative to the direction of the car.

So, we now have a direction of the shortest path charge, and also a distance, which gives us the position of the charge.

We then need a function $f_{FoSP}(d, c)$ that tells us how the potential in a position at distance d from a charge of size c

is calculated. We have chosen the following:

$$f_{FoSP}(d, c) = -k \cdot d^2 + c \quad (4)$$

We use $k = 24$ based on empirical testing. As we will show later in Section IV, this somewhat unorthodox definition of the APF function has its advantages.

3) The Curvature Agent: The task of the curvature agent is to try to get the car in a position where it maximizes its speed through the curves. This is done in a race by e.g. keeping at the right side of the track before a sharp left turn. By doing so, the driver does not have to turn as sharp as if it followed the shortest path, and can keep a higher speed through the curve.

The f_{FoC} function is calculated as follows:

$$f_{FoC}(d, c) = c \cdot (1 - d/d_{max}), \quad (5)$$

where, d_{max} is the width of the road.

The position of the charge in the FoC is clear. It tries to get the player to select the side of the road that will enable the car to maintain an as high speed as possible through the next curve, i.e. the opposite side of the direction in which the curve is turning. Thus its position is at that side of the road.

The size of c is depending both on the current speed v_t, and on the distance to the curve, as measured by msd_0, the maximum sight distance in front of the car. It is proportional to the speed, but described in terms of a second degree polynomial of the distance to the curve.

$$c = (-0.0002msd_0^2 + 0.06msd_0 - 2.5) \cdot v_t \quad (6)$$

4) The Track Agent: The track agent has a very simple but important task. It tries to make sure that the driver agent avoids getting outside the track. By placing repelling forces outside the track , the driver agent will avoid choosing actions that lead to lookahead points outside the track. The potential is defined as follows:

$$f_{FoT}(d, c) = c/\sqrt{d} \quad (7)$$

, where we use a repelling charge $c = -5$.

5) The Navigator Agent: The Navigator Agent does not have any APF to handle. It works in two phases. During the first lap of the training, it monitors whether the track turns right, turns left or goes forward, given the traveled distance so far. During the races, it gives valuable information to the Curvature Agent about in what direction the next turn is. Although using learning in this way is in contrast to the reactive paradigm that APFs represent, the course grained information we get from the distance sensors, especially when the noise is turned on, interferes with the control in an undesirable way.

6) The Interface Agent: The interface agent is simply the interface between the MAS and the TORCS server. It both receives and distributes the sensory data from the TORCS server to all the agents and forward the decisions of the driver agent to the server. The Interface agent also handles the gears. It uses a version of the gear change implementation of the client provided by the competition as well as the standard recovery code that trigger if it gets off the track.

F. The MAS Architecture

The agents share a number of common fields; the curvature agent use the lookahead points calculated by the driver agent, and all field producing agents send their fields to the driver agent that makes the final decision taking all the fields into account.

There is however no need to let the agents pass information dynamically; the ways they communicate are the same, frame after frame. We will therefore use a hardcoded architecture where the information is passed directly from agent to agent as illustrated in Fig. 4.

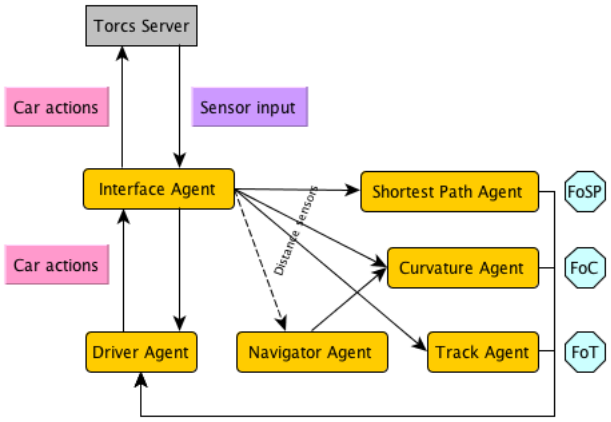

Fig. 4. The MAS architecture of Powaah.

III. THE EXPERIMENTS

We will use the previous year's competitors as a benchmark to measure the feasibility of our approach in terms of tournament rules competition with opponent cars at four different tracks. The versions of the opponents we chose to compete against were the official downloadable entries of last year's car racing championship at CIG 2010 (see Table I).

A. The Tracks

TABLE I
THE TRACKS TESTED (LEFT), AND THE OPPONENT BOTS USED FOR BENCHMARKING (RIGHT).

Tracks tested
CG Speedway 1
CG Speedway 2
Ruudskogen
Wheel 2

Opponent controllers
Autopia
Neil
COBOSTAR

The tracks are of variable length and difficulty. Both CG Speedways are relatively fast tracks, Wheel 2 is a technically challenging track with lots of curves and Ruudskogen is a narrow track mixing sharp curves with long straight stretches, see Fig. 5.

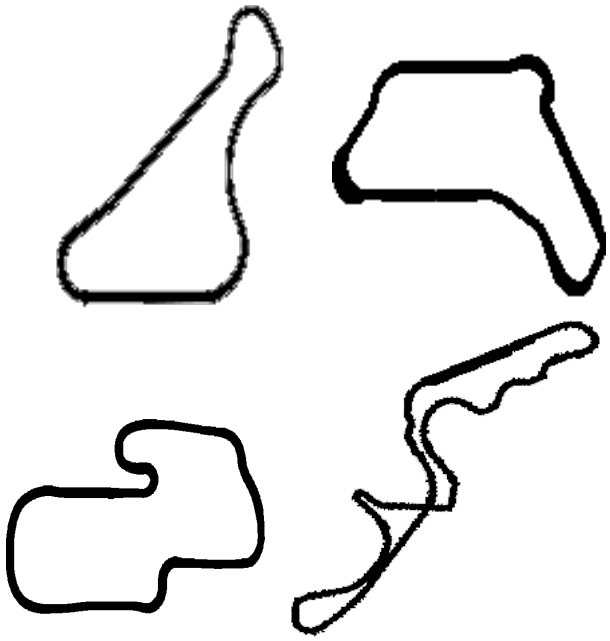

Fig. 5. The tracks tested (from upper left to lower right): CG Speedway 1, CG Speedway 2, Ruudskogen, and Wheel 2. The tracks are zoomed to fit the figure, so their dimensions are not comparable.

B. The opponent controllers

Three opponent controllers were used as benchmark in our experiments, the winner of 2010 — Autopia, the winner of 2009 — COBOSTAR, and one of the mid-ranked controllers of 2010, namely Neil. The controller of Cardamone was not available at the time we did our experiments, otherwise, that would have been a good benchmark as well.

1) Autopia: The Autopia competitor turned out to be a very fast driver, with best racing times as well as having good race results making this driver one of the hardest competitors. Autopia is using a Fuzzy Architecture which bases the driving from three basic modules which is from what gear we should be in, how fast we should go and steering control. In the warm-up stage it initializes a vector to 1.0, and tries to maintain that vector with as many real values as track length in meters. In case of any damage occurs while driving around in the warmup, or if the car goes outside of the track the vector positions are multiplied by 0.95, from 250 meters before the current position. To maintain a cautious function this vector is multiplied by a function $F = 1 - 0.02 \cdot \lceil damage/1000 \rceil$ to make the driver more cautious.

2) Neil: The Neil driver is a careful driver, using a Feed Forward Multi Layer Neural Network, with imitation learning to control the car. To train this classifier the controller is provided with human data from the author, which differs a bit from the rest of the competitors since the other drivers somehow takes the track into account to decide their behaviors.

3) COBOSTAR: COBOSTAR [3], which was one of the best drivers from the 2009 championship, is very good on fast tracks. However, on more challenging tracks (like Ruudskogen and Wheel 2 that we use in our experiments), it becomes more careful. COBOSTAR uses a parameterized controller which is optimized with a technique called CMA-ES (Covariance Matrix Adaption Evolution Strategy). For knowing where on the track it goes off the road during the warm-up, COBOSTAR saves the crash points dynamically, so it can be more careful next time it enters those curves.

C. Preparations

Before each test, all participants were allowed 100000 game control steps of warmup on the current track. This mode is not part of the actual competition, but is used to give the competitors a chance to learn the track (that is unknown to them). We do not calibrate the cars in any other way than letting them run this training.

D. Experiment 1: Single car race qualification

The test starts by placing the car in pole position, start the race and the car then run 3 laps. We measured the average time it took for the cars to finish and the tests were carried out both with and without noise in the distance measures. The results of the races without noise is presented in Table II.

TABLE II
THE RESULT OF 3 LAPS ON THE SPECIFIED TRACKS *without* NOISE. CGS = CG SPEEDWAY

	CGS1	CGS2	Ruudskogen	Wheel2	Sum
Autopia	129.8	169.4	207.4	392.5	899.1
Powaah	126.2	178.6	217.4	381.5	903.7
Neil	147.9	201.8	249.1	432.2	1031.0
COBOSTAR	147.6	174.8	275.8	459.6	1057.8

We then added noise to the distance sensor systems of the cars (the same as is added in the car racing championship) and repeated the experiments. The results are shown in Table III. The results of the noisy qualification runs are used in deciding the starting order of the first runs of each track in Experiment 2.

TABLE III
THE RESULT OF 3 LAPS ON THE SPECIFIED TRACKS *with* NOISE. CGS = CG SPEEDWAY

	CGS1	CGS2	Ruudskogen	Wheel2	Sum
Autopia	129.8	174.5	209.2	400.4	913.9
Powaah	126.9	192.2	216.7	390.5	926.3
Neil	146.1	203.7	247.8	428.4	1026.0
COBOSTAR	136.3	181.0	288.8	487.2	1093.3

E. Experiment 2: Races with multiple cars

In Experiment 2, we ran as many races on each track as we had competitors, namely four. The starting order of the first race was decided by the result of the qualification for that track (see Table IV) The starting order of the rest of the

races was made by letting the car starting last get the pole position in the next race, so that in four races, all competitors get one pole position each.

TABLE IV
THE STARTING POSITIONS IN THE CAR RACES OF EXPERIMENT 2 FOR RACE 1,2,3 AND 4.

Position	Powaah	Autopia	COBOSTAR	Neil
CGS1	1,2,3,4	2,3,4,1	3,4,1,2	4,1,2,3
CGS2	3,4,1,2	1,2,3,4	2,3,4,1	4,1,2,3
Ruudskogen	2,3,4,1	1,2,3,4	4,1,2,3	3,4,1,2
Wheel2	1,2,3,4	2,3,4,1	4,1,2,3	3,4,1,2

Table V shows the results of each of the races and a sum of the scores is presented in Table VI.

IV. DISCUSSION

This demonstration of the use of APFs shows that it indeed is a viable paradigm to apply in the domain of car racing. Our proposed solution is almost as fast as the winner of last year's competition in the time qualification of Experiment 1, but seem to be a bit more robust in terms of staying on track. Experiment 1 also shows that it handles the noisy conditions well without losing too much time.

Previous work by Uusitalo has also clearly show the impact of the *Field of Curvature.* Once we got that working, it improved the performance with up to 9 per cent compared to using only the *Field of Shortest Path* [21]. Tuning the FoC with the present FoSP was crucial. It was not until we combined the linearly formulated f_{FoC} with a squared f_{FoSP}, that we could start to tune it to do what we wanted. We are confident though that there are further improvements to make here, i.e. to adjust the strength of the FoC-charge to the expected radius of the curve. Let us return to the details here. Normally, APFs use potential-functions such as Coulombs law that describes the physics of real potential fields (for two idealized point charges of size q_1 and q_2 at distance r from each other):

$$F = k\frac{q_1 q_2}{r^2} \tag{8}$$

Basically what it says, is that the attractive (or repelling) force gets four times as strong as the distance is halved. In many applications of APFs this is indeed a good and useful potential-function. However, in our case, we would like to balance two forces; FoC and FoSP in a way that FoC may move the focus of attention created by the FoSP. Returning to the definition of FoSP, we see that its size is decreasing proportional to the squared distance, rather than being inversely proportional to the squared distance ($-d^2$ vs. $1/d^2$). When we now add the linear potential function of FoC, the effect is that the maximum of the total is moved from the position of the FoSP charge, towards the position of the FoC charge (at the side of the road), see Fig. 6.

In Experiment 2, we let them race each other. Note that we do not use any kind of opponent tracking or avoidance. That might have had a major impact on the results of race

TABLE V
THE RESULTS OF EXPERIMENT 2. POINTS IS THE F1 SCORE USED AS PART OF THE SCORING IN THE CAR RACING CHAMPIONSHIP.

	Bot	Total time	Best lap	Damage	Points
CG Speedway 1	Powaah	207.28	40.52	0	14
	Autopia	214.47	41.51	6	8
	COBOSTAR	255.46	49.71	914	6
	Neil	256.3	49.63	1267	5
	Powaah	209.62	40.81	1318	14
	Autopia	240.03	46.58	3817	8
	COBOSTAR	244.6	43.37	2817	6
	Neil	258.43	46.69	6023	5
	Powaah	208.23	40.74	416	12
	Autopia	219.99	41.61	4	8
	COBOSTAR	221.94	41.6	381	6
	Neil	243.18	46.38	0	7
	Autopia	213.52	41.5	0	14
	Powaah	220.21	41.85	6736	8
	COBOSTAR	252.77	49.48	4292	6
	Neil	255.39	48.47	4929	5
CG Speedway 2	COBOSTAR	293.53	55.32	16	12
	Powaah	307.23	58.74	2323	8
	Neil	336.86	65.37	0	8
	Autopia	337.84	56.96	2045	5
	Autopia	290.71	56.48	1874	10
	COBOSTAR	292.13	55.37	467	12
	Powaah	306.61	57.7	1950	6
	Neil	338.01	65.94	674	5
	COBOSTAR	294.65	55.73	1907	12
	Autopia	304.53	56.8 5	759	8
	Neil	335.04	65.27	0	8
	Powaah	+ 2 laps	60.35	7205	5
	COBOSTAR	292.94	55.29	18	12
	Powaah	303.37	56.75	1184	8
	Neil	335.77	65.5 0	0	8
	Autopia	336.51	56.37	1307	5
Ruudskogen	Powaah	363.51	71.07	5892	12
	Autopia	373.88	68.72	2422	8
	Neil	409.79	80.39	0	8
	COBOSTAR	+ 1 Lap	94.09	2527	5
	Autopia	350.63	68.11	4320	12
	Neil	409.20	80.44	0	10
	COBOSTAR	+ 1 Lap	94.33	7599	6
	Powaah	DNF	120.72	10169	0
	Powaah	360.86	70.69	1976	12
	Neil	408.46	80.32	0	10
	Autopa	409.25	71.24	640	6
	COBOSTAR	+ 1 Lap	94.5	1518	5
	Powaah	361.14	70.80	5338	12
	Neil	408.03	80.25	0	10
	Autopia	408.64	75.51	153	6
	COBOSTAR	+ 1 Lap	93.02	3517	5
Wheel 2	Autopia	663.87	130.34	0	12
	Neil	705.70	139.11	0	10
	Powaah	742.49	126.14	1165	8
	COBOSTAR	+ 1 Lap	167.08	834	5
	Powaah	640.92	125.89	0	14
	Autopia	663.61	130.56	0	10
	Neil	712.86	140.15	804	6
	COBOSTAR	+ 1 Lap	167.76	569	5
	Powaah	646.89	127.61	4131	12
	Autopia	660.11	129.84	0	10
	Neil	710.62	139.76	0	8
	COBOSTAR	+ 1 Lap	166.95	2724	5
	Autopia	663.58	130.58	490	10
	Neil	707.01	139.66	0	10
	Powaah	712.7	126.34	1978	8
	COBOSTAR	+ 1 Lap	166.23	955	5

TABLE VI

THE SUM OF SCORES OF THE RACES IN EXPERIMENT 2. 1ST PLACE GIVES 10 POINTS, 2ND 8, 3RD 6, 4TH 5 POINTS AND IF THE CONTROLLER CANNOT FINISH, IT GETS NO SCORE. FL = FASTEST LAP AND LD = LEAST DAMAGE. 2 POINTS WERE GIVEN TO THE BEST CONTROLLER IN EACH OF THESE CATEGORIES. IN THREE RACES, TWO CONTROLLERS TIED IN LEAST DAMAGE AND BOTH GOT 2 POINTS.

Controller	1st 10	2nd 8	3rd 6	4th 5	DNF 0	FL 2	LD 2	$\sum$
Powaah	8	3	3	1	1	10	3	153
Autopia	5	7	2	2	-	2	4	140
Neil	-	5	6	5	-	-	11	123
COBOSTAR	3	1	5	7	-	4	1	113

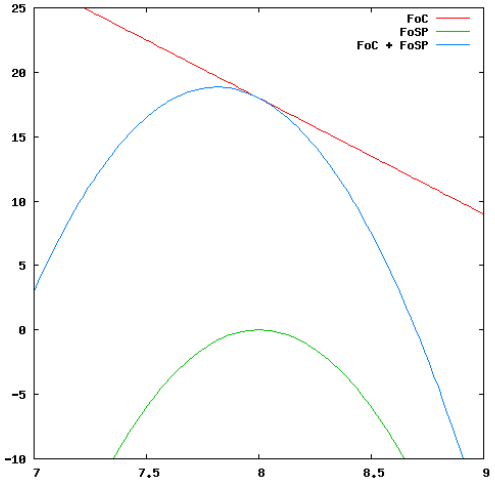

Fig. 6. An example of a FoSP function whose maximum is moved by the FoC field as the fields are added by the driver agent. The inclination of the linear FoC function decides how far the maximum of the sum of FoSP and FoC will be moved. In this example, the car is going in 60m/s with 100m left to the curve.

2 in Ruudskogen where we first got knocked off the track, then crashed right into another car. We got so damaged that we actually could not finish the race. In the rest of the races, Powaah won eight and got three 2nd and three 3rd places. To our surprise, Powaah was by far the most successful controller in running fast laps, achieving bonus points for that in 10 out of 16 races, despite being the fastest in only half of the qualification races and being beaten by Autopia in the total time comparison.

Autopia and Powaah were outstanding at all of the chosen tracks, except the CG Speedway 2, where COBOSTAR beat them both with its three first places. Neil was extremely careful, which paid off in the *Least damage* bonus score. In 11 out of the 16 races, it ended up as the least damaged car, but the numerous last places destroyed its chances to be competitive in the overall comparison.

A. Related Work

Chin Hiong Tan and Kay Chen Tan's entry from 2008 uses the largest free distance (as indicated by the distance sensors) to decide the direction. They also use the distance as such to set the speed of the car. Although this is similar to our shortest path agent, there are some important differences. First, we use a simple function based on the average distance to the next curve to set the speed, whereas Tan and Tan used a more sophisticated hyperbolic tangent speed regulator [18]. Second, the shortest path agent does not control the actuators directly, but indirectly by placing an attractive charge in the look-ahead physical state space. This charge will then just be one among three that will effect the choice of action. Third, the bots were made for quite different applications. Although the CEC 2007 Simulated Car Racing Competition was a predecessor of the TORCS based competition of today, it was a simple 2D environment with explicit waypoint passing, whereas we now have a full 3D environment.

V. CONCLUSIONS AND FUTURE WORK

We have introduced the novel combination of multi-agent control and multiple potential fields in the context of car racing. Our findings are that the approach is feasible and in some respects, it outperforms the state of the art so far. However, it requires careful and non-trivial tuning to address the problems of the multiple goal optimization that is present in our solution.

As this is a novel approach in this context, we are convinced that there is still quite a lot of things that may improve the performance of Powaah.

1) *Opponent tracking* could be used for a number of purposes. We could have an *overtaker agent* that choose the best side to overtake an opponent car. The obvious *opponent avoidance agent* could by adding repelling charges on opponent cars avoid that we crash into them, and last but not least a *Roadblock agent* that position the car in front of cars trying to get past it would possibly create problems for faster cars that get behind it. Previous work by e.g. Onieva et al. and Loiacono et al. has addressed the importance of opponent tracking and overtake control in TORCS [16], [13] and Wolf and Burdick has applied APFs to highway overtaking [23]. We believe that the total lack of opponent tracking cost Powaah several points in the final score as it did not avoid, or even see the opponent cars in front of or at the side of it, resulting in one race where it did not finish due to the damage caused by the crashes.
2) *Improve the methodology* (of Hagelbäck and Johansson) so that it better fits the problem at hand. Although many of the steps are similar, it seems as if there are additional steps that have to be taken in order to get it to work at its full potential.
3) *Better lookahead* may improve the spread of decision outcomes. Since it is the outcomes that are of interest for the driver agent, the more spread they are, the more of the action space it will explore. Instead of using an even distribution of actions (leading to an uneven distribution of lookahead positions), we could use an even distribution of lookahead positions, and calculate the corresponding actions leading to that state.

4) An *Improved track learning* could possibly further improve the performance. We do only store the direction of the road as a function of the travelled distance. By storing more accurate information about the curvature radius, etc., we believe that further improvements may be done to increase the speed through the curves. It would also be interesting to study how far we may reach without any learning at all. Would it be enough to use the information collected from the sensors in real time to decide the information that we now collect during the warmup?
5) An *Extended study* would improve the knowledge about how Powaah work in different contexts. We would also like to test the performance at other tracks and with new opponents. These initial experiments tell us that the approach is feasible in the used context, but more experiments are needed to raise the conclusions to the general level. We are therefore participating in the Car Racing Championship Tour this year to learn more about our approach and to tune it.

ACKNOWLEDGMENTS

The authors would like to thank Blekinge Institute of Technology for their support throughout the work, the car racing championship organizers for providing an interesting and challenging application, and the anonymous reviewers for their valuable and constructive feedback.

REFERENCES

[1] J. Borenstein and Y. Koren. Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 19:1179–1187, 1989.

[2] J. Borenstein and Y. Koren. The vector field histogram: fast obstacle avoidance for mobile robots. *IEEE Journal of Robotics and Automation*, 7(3):278–288, 1991.

[3] M. Butz and T. Lonneker. Optimized sensory-motor couplings plus strategy extensions for the torcs car racing challenge. In *Proceedings of Computational Intelligence and Games (CIG)*. IEEE, 2009.

[4] L. Cardamone, D. Loiacono, P. L. Lanzi, and A. P. Bardelli. Searching for the optimal racing line using genetic algorithms. pages 388 –394, aug. 2010.

[5] J. C. Gerdes and E. J. Rossetter. A unified approach to driver assistance systems based on artificial potential fields. *Journal of Dynamic Systems, Measurement, and Control*, 123(3):431–438, 2001.

[6] J. Hagelbäck and S. J. Johansson. Using multi-agent potential fields in real-time strategy games. In L. Padgham and D. Parkes, editors, *Proceedings of the Seventh International Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*, 2008.

[7] M. Hassoun, Y. Demazeau, and C. Laugier. Motion control for a car-like robot: Potential field and multi-agent approaches. In *Proc. of the Int. Workshop on Intelligent Robots and Systems. IEEE*, pages 1457–1463. IEEE, 1992.

[8] A. Howard, M. Matarić, and G. Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In *Proceedings of the 6th International Symposium on Distributed Autonomous Robotics Systems (DARS02)*, 2002.

[9] S. Johansson and A. Saffiotti. An electric field approach to autonomous robot control. In *RoboCup 2001*, number 2752 in Lecture notes in artificial intelligence. Springer Verlag, 2002.

[10] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1):90–98, 1986.

[11] O. Khatib. Human-like motion from physiologically-based potential energies. In J. Lenarcic and C. Galletti, editors, *On Advances in Robot Kinematics*, pages 149–163. Kluwer Academic Publishers, 2004.

[12] D. Loiacono, L. Cardamone, and P. L. Lanzi. Simulated car racing championship: Competition software manual. Technical report, Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy, 2011.

[13] D. Loiacono, P. L. Lanzi, J. Togelius, E. Onieva, D. A. Pelta, M. V. Butz, T. D. Lnneker, L. Cardamone, D. Perez, Y. Saez, M. Preuss, and J. Quadieg. The 2009 simulated car racing championship. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(2):131–147, June 2010.

[14] D. Loiacono, J. Togelius, P. L. Lanzi, L. Kinnaird-heether, S. M. Lucas, M. Simmerson, D. Perez, R. G. Reynolds, and Y. Saez. The wcci 2008 simulated car racing competition. In *IEEE Symposium on Computational Intelligence and Games*, pages 119–126, 2008.

[15] M. Massari, G. Giardini, and F. Bernelli-Zazzera. Autonomous navigation system for planetary exploration rover based on artificial potential fields. In *Proceedings of Dynamics and Control of Systems and Structures in Space (DCSSS) 6th Conference*, 2004.

[16] E. Onieva, L. Cardamone, D. Loiacono, and P. L. Lanzi. Overtaking opponents with blocking strategies using fuzzy logic. pages 123 –130, aug. 2010.

[17] T. Röfer, R. Brunn, I. Dahm, M. Hebbel, J. Homann, M. Jüngel, T. Laue, M. Lötzsch, W. Nistico, and M. Spranger. GermanTeam 2004 - the german national Robocup team, 2004.

[18] C. H. Tan, J. H. Ang, K. C. Tan, and A. Tay. Online adaptive controller for simulated car racing. In *IEEE Congress on Evolutionary Computation*, pages 2239–2245, 2008.

[19] C. Thurau, C. Bauckhage, and G. Sagerer. Learning human-like movement behavior for computer games. In *Proc. 8th Int. Conf. on the Simulation of Adaptive Behavior (SAB'04)*, 2004.

[20] J. Togelius, S. Lucas, H. D. Thang, J. M. Garibaldi, T. Nakashima, C. H. Tan, I. Elhanany, S. Berant, P. Hingston, R. M. Maccallum, T. Haferlach, A. Gowrisankar, and P. Burrow. The 2007 ieee cec simulated car racing competition. *Genetic Programming and Evolvable Machines*, 9:295–329, December 2008.

[21] T. Uusitalo. A first approach in applying artificial potential fields in car games. Technical report, Blekinge Institute of Technology, Bachelor's thesis, 2011.

[22] P. Vadakkepat, K. Chen Tan, and W. Ming-Liang. Evolutionary artificial potential fields and their application in real time robot path planning. In *Proceedings of the 2000 Congress on Evolutionary Computation*, pages 256–263. IEEE Press, 2000.

[23] M. T. Wolf and J. W. Burdick. Artificial potential functions for highway driving with collision avoidance. In *ICRA*, pages 3731–3736. IEEE, 2008.