Robust player imitation using multiobjective evolution

Niels van Hoorn, Julian Togelius, Daan Wierstra and Jürgen Schmidhuber
Dalle Molle Institute for Artificial Intelligence (IDSIA)
Galleria 2, 6298 Manno-Lugano, Switzerland
{niels, julian, daan, juergen}@idsia.ch

***Abstract*— The problem of how to create NPC AI for videogames that believably imitates particular human players is addressed. Previous approaches to learning player behaviour is found to either not generalize well to new environments and noisy perceptions, or to not reproduce human behaviour in sufficient detail. It is proposed that better solutions to this problem can be built on multiobjective evolutionary algorithms, with objectives relating both to traditional progress-based fitness (playing the game well) and similarity to recorded human behaviour (behaving like the recorded player). This idea is explored in the context of a modern racing game.**

I. Introduction

This paper concerns the creation of controllers for computer game agents which are able to play a game in a manner similar to a particular human player. We call this imitation of player behaviour, or (interchangeably) modelling of playing style. While research within computational intelligence and games is often concerned with learning to play a particular game as well as possible, there are many applications for imitating player behaviour in computer games, and many algorithms within computational intelligence that could be used to such ends.

A good example of player behaviour imitation from commercial computer games is the very successful racing game *Forza Motorsport* for the Microsoft XBox, where players can train "drivatars" that drive like the human playing the game. To create a drivatar, a player has to drive a sequence of tracks designed to contain a number of representative challenges; the game records the path taken on each track segment, and the behaviour of a drivatar on a new track consists of the splicing together of the recorded path on each track segment. This approach posits some rather serious restrictions on the design of the game, most notably that each new racing track must be composed of segments that were present in the test tracks [1]. However, the benefits of being able to imitate playing styles was so great that this was deemed acceptable. One of the chief uses of drivatars is to estimate the performance of a particular playing style on tracks that the player does not have the time to drive, such as endurance events, or to get an estimate of the likely difficulty of a new track given a particular playing style. Another use for drivatars is to send them over the Internet to friends, so that they can compare their playing style to a virtual copy of their friend driving on the same track.

Examples of behaviour imitation can be found in other genres of games as well. In the critically acclaimed commercial "god-game" *Black and White* by Lionhead Studios, the key NPC (non-player character) is a giant monster that imitates the actions taken by the player. This effectively amplifies the effects of the player's actions, as most of the effects the player has on the game world are interpreted through this monster. Player imitation can also be used in order to acquire *believable* behaviour, as both hard-coded and adaptive behaviour can come across as "mechanical" or "unnatural" if the designer is not careful (and sometimes even if the designer is careful). In [2], example-guided evolution is used to acquire controllers for agents in a board game-like strategy game, resulting in more believable agents than those acquirable through other evolutionary means.

Yet another use for player imitation was recently proposed: personalized automatic content creation [3], [4]. In a first realisation of this concept a simple car racing game was used, where the objective was to complete laps in the shortest possible time, on tracks of varying complexity. Human players drove test tracks designed to contain a number of different challenges, and the actions they took at each part of the track were recorded and used to construct controllers that mimicked their behaviour. These controllers were then used as part of the fitness function for evolving new racing tracks. The fitness of a particular racing track depended on how the human-like controller drives on this track: ideally, not too fast or too slow, with the right amount of variance between and within trials. (These criteria were inspired by theories of what makes computer games fun, such as those by Koster [5] and Malone [6]). Racing tracks were represented as b-splines, posing track design as a real-valued numerical optimization problem, and evolutionary algorithms were used to optimize tracks for being as fun as possible for the modelled player. The end result was an algorithm that produces novel tracks that are fun to drive for particular human players.

It is likely that many other uses of player imitation can be found, in many other game genres, if the requisite methods are refined and demonstrated (which is the purpose of this paper); for an example from FPS games see [7].

A. Direct and indirect modelling

With the usefulness of player behaviour imitation established, the question shifts to how to use computational intelligence techniques to imitate players. In [3], [4] a distinction and comparison was made between *direct* and *indirect* methods for modelling behaviour.

Direct modelling uses supervised learning to associate observations (sensor data) with actions, and then uses the trained function approximator directly as a controller. In the example given in [3], human test subjects were asked to

978-1-4244-2959-2/09/$25.00

drive a number of laps around a track in a simple racing game, and both the sensor data experienced by the car and the actions taken by the human were recorded at each time step. The sensor data consisted of speed and a number of rangefinder sensors giving the approximate distance to the edges of the track in various directions; this is the same sensor representation as is used as input to the controller when evolving controllers for that racing game.

Backpropagation was used to train standard MLPs to associate sensor data with the action chosen by the player. Despite reaching low errors on the training data, networks thus trained failed to perform very well when used as controllers in the game. They typically started to drive in the right direction, but then soon crashed into walls and remained stuck there, not being able to back away and recover. This behaviour points to two shortcomings of direct modelling: the complexity of the function to approximate (the human player), and the inability of the model to generalize to unseen situations (e.g. if the human player never crashed into a wall (the behaviour of backing away from a wall is not in the dataset) a controller based on direct modelling of the player's behaviour will not know how to back away from walls). Further experiments using a k-nearest neighbour classifier yielded better initial imitation but worse generalization.

It is instructive to compare these results to two examples where neural network-based direct modelling was apparently used successfully in racing games. In the successful commercial game *Colin McRae Rally 2.0*, all the computer-controlled cars were controlled partly by feedforward neural networks that had been trained on data from human driving using the RProp algorithm [8], [9]. However, not all aspects of the driving were controlled by the neural network; a number of human-developed rules switched between different states, such as overtaking or recovering from crashes.

In another example, neural networks were trained on data from human driving to control a motorcycle in *Motocross The Force*, a game that features a reasonably detailed physical simulation [10]. In some cases the trained networks performed almost as well as the human driver that they were modelled on, and also generalized to other tracks. However, the environments used in this game differ significantly from the tracks used in the other games due to the absence of sharp track borders, meaning that deviations from the best path simply result in longer lap times (not crashing).

Evolutionary algorithms have previously been used to evolve neural networks that are able to drive cars well on a large variety of tracks using an incremental distance-based fitness function [11]. These controllers, however, do not exhibit very human-like behaviour; to a human observer, evolved driving styles often appear weird and irrational, though effective. The idea of *indirect modelling* is to profit from evolution's ability to find good controllers that generalize well, while retaining some apparent human-likeness.

In [3], evolutionary runs were seeded with good general neural network-based controllers, and then further evolved using three different fitness measures: the difference between the controller's driving and the observed driving in terms of distance travelled, the variation between laps, and frequency of steering changes. Results were less than stellar: the controllers reproduced the performance of particular modelled humans, but did not look qualitatively similar.

The approach to indirect modelling taken in [4] is similar in that it starts from a good evolved controller, but the objectives were instead related to reproducing the human player's speed and lateral deviation from the centre of the track at a number of waypoints distributed around a test track. Though more successful, this approach suffered from problems with weighting the different objectives.

B. Multiobjective modelling

The problems outlined above boil down to two central problems. First, direct modelling is too hard (without incorporating considerable domain knowledge, as was done in the successful commercial examples above): it produces controllers that do not generalize well to new environments, and sometimes do not even even behave correctly in their original context. Second, indirect modelling produces controllers that do not resemble the modelled behaviour well enough to appear believably human, or to act as a proxy for the modelled human in automatic content generation.

Ideally, we would want to be able to combine the advantages of both methods into a superior modelling method. If this is not possible, and there is an inherent tradeoff, we would like to gain extensive insight into this tradeoff. The ideal tool for both of these tasks would seem to be an evolutionary multiobjective optimization algorithm (*MOEA*). MOEAs evaluate each solution according to a number of objectives (often two or three) and create a *Pareto front* of *nondominated* solutions. Solution X dominates solution Y if the fitness values for X are not worse than the fitness values for Y in all objectives, and for at least one objective the fitness of X is better than that of Y. A nondominated solution is one that is not dominated by any solution in the population.

With *multiobjective modelling* we mean using an MOEA to construct controllers that both reproduce aspects of human behaviour or playing styles, *and* perform well on the given task. The way this is done is by using one or more objectives related to performing a task (e.g. playing a game) well, and additionally one or more objectives related to performing the task in a way similar to the modelled human. The idea is that this approach to modelling can combine the generality of indirect modelling (through the performance-related objectives) with the more faithful reproduction of human behaviour of direct modelling (through the ability to pick solutions from the pareto front which are maximally similar to the modelled human while still performing acceptably well).

In this paper, we explore multiobjective modelling for the first time; the particular task is to model human driving styles in the TORCS racing game.

C. Research questions

To our best knowledge, this paper represents the first attempt to use multiobjective optimization to model be-

haviour, and also the first attempt to model driving behaviour in TORCS. It is also a contribution to the understudied area of multiobjective reinforcement learning. The particular questions we try to answer include:

- How well do the best evolved neural networks drive, compared to human driving? Will their driving styles resemble human driving?
- How well will networks trained to approximate human driving drive? Will their styles resemble human driving?
- When combining objectives related to driving well and to approximating human driving, will there be a tradeoff between fulfilling these objectives?
- Can human driving data help evolution create better driving behaviour?
- How well will the various controllers that are derived using these techniques generalize to other tracks?

Fig. 1. The TORCS game.

II. Methods

A. Car racing game

The Open Racing Car Simulator (*TORCS*) is a modern racing game, complete with multi-player capacities, advanced car physics and first-person 3D graphics[1] (see figure 1). Being open source, TORCS has an open API that allows for the interfacing of custom software for driving the cars in the game. This capability was used for the simulated car racing competitions associated with the conferences IEEE CEC and CIG in 2008. A software framework was developed where the TORCS game was extended to become a server, where one or more cars could be controlled by clients connecting over TCP/IP. Sample clients and learning algorithms in C and Java were developed and supplied to competitors on a web page[2]. The goal of the competitions was to learn or otherwise develop the best-performing car controller. Results and a discussion were published as [12].

In the experiments described here, we used the CIG 2008 version of the competition software for interfacing our controllers to TORCS. As the series of TORCS-based car racing competitions is an ongoing project, this version of the software still has some technical limitations. One of these is that evolutionary processes cannot simply be set up to use more than one track simultaneously, precluding the approach to multi-track generalization demonstrated in [11]. Another limitation is that the behaviour of the simulation differs slightly between different setups, such as operating system and visual and non-visual modes, limiting the ability to judge the human-likeness of evolved driving styles through ocular inspection.

[1]Available at http://torcs.sourceforge.net

[2]http://cig.dei.polimi.it/

B. Tracks used

The tracks used are shipped with TORCS by default. They are picked on difference and difficulty level by hand and numbered 1 to 4. They are listed below and depicted in figures 2-5.

- **Track 1**: CG Speedway number 1 - a fairly easy track, with long straight pieces and few sharp turns. When leaving the track, getting on again is pretty easy.
- **Track 2**: Alpine 1 - a track with a lot of sharps turns, where there are barriers on the side of the track, so it's impossible to get off the track.
- **Track 3**: E-Track 1 - a pretty difficult track, with a lot of sharp turns. It's easy to end up in the grass at the sides of the track and hard to get back on.
- **Track 4**: CG track 3 - a difficult track with a bit of everything: sharp and smooth turns, long straight pieces, and barriers as well as grass alongside the track.

C. Sensors and controller architecture

The controllers are based on Elman-style recurrent neural networks with *tanh* activation functions. The motivation for this was that in initial experiments, recurrent neural networks were consistently able to evolve better-performing driving in fewer generations than simple MLP's with the same number of inputs, hidden neurons and outputs. As modelling human driving is likely to require taking observations gathered at previous time steps into account, it is a reasonable assumption that a recurrent architecture would work better than a reactive architecure for modelling as well.

Each network has 23 inputs, 8 hidden neurons and 3 outputs. The inputs consist of a constant bias, the current speed of the car, the angle between the direction of the car and the track axis, the lateral distance between the centre of the car and the centre of the track, and 19 track edge rangefinder sensors. Each rangefinder sensor returns the distance between the centre of the car and the edge of the track in a particular direction, which is relative to the frame of reference of the car; these sensors are distributed uniformly around the car.

D. Fitness measures

Three fitness measures were defined, one relating to how well the car drives on a particular track, and two related to how well it reproduces human driving behaviour.

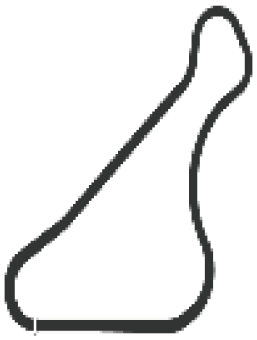

Fig. 2. Track 1: CG Speedway number 1

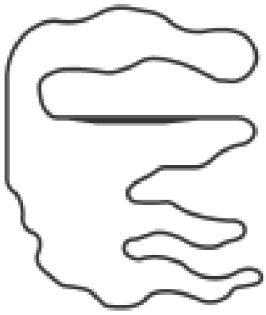

Fig. 3. Track 2: Alpine 1

Fig. 4. Track 3: E-Track 1

Fig. 5. Track 4: CG track 3

- **Distance**: Maximize the distance travelled by a car controlled by that controller on a particular track over 5000 time steps.
- **Steering**: Minimize the average squared difference between the steering command issued by a human player and the steering command issued by the controller when presented with the same situation as the human player. This fitness criterion is always defined relative to a particular log of human driving on one or several track. Each data point in such a log consists of the sensor inputs that would have been presented to a presumptive controller of a car in the same position as the human controlled, and the steering action taken by the human.
- **Acceleration**: Minimize the sum of the squared difference of the acceleration commands and the squared difference of the braking commands.

E. Multiobjective evolutionary algorithm

All the experiments in this paper were performed using the *NSGA-II* algorithm, which can safely be regarded as the industry standard for evolutionary multiobjective optimization [13]. Each experiment used either 2 or 3 objectives (in the "degenerate" case of only distance fitness both objectives were identical), and was run for 150 generations with a population of 100. For generating new offspring we mutate all the weights by adding a normally distributed value X, where $X \sim N(0, 0.05)$. No crossover was used. Unless otherwise specified, each experiment was repeated 5 times.

III. Results

Our experiments were structured as follows:

1) We drove the three first test tracks manually a number of times using a keyboard interface, and saved logs of sensor inputs and actions at each time step. We selected two logs, each of one lap's length, on each track for further experimentation: one of careful driving (with longer lap times and less damage taken) and one of aggressive driving (as fast as possible and reckless).
2) We evolved controllers for two of the four test tracks using only distance fitness. The controllers thus evolved were then tested on all four tracks.
3) We evolved controllers using only the two objectives that relate to replicating human driving behaviour, steering and acceleration fitness. This was done for logs of both careful and aggressive driving on three of the tracks and with data of specific tracks as well as the combined data of all tracks. These controllers, which were evolved without being tested on the actual driving task (just against logged user data), were then tested on all four tracks. Controllers that were evolved using data of one track were only tested on that track.
4) We evolved controllers using all three fitness measures on the first three tracks, and tested them on all tracks.

A. Human driving

The first experiments concerned driving the tracks manually, and recording logs of sensor data/action tuples for one

	track1	track2	track 3
aggressive	4880.361	5099.944	4321.396
careful	4244.142	4119.491	4004.494

TABLE I

PERFORMANCE OF HUMAN DRIVING ON THREE TEST TRACKS. NUMBER OF METRES DRIVEN IN THE FIRST 5000 TIME STEPS OF THE RECORDING.

	track1	track2	track 3	track4
track1	4706.342	1115.3244	412.62442	832.0062
track4	1565.66344	562.541984	308.04864	2945.494

TABLE II

PERFORMANCE OF CONTROLLERS EVOLVED ON TRACKS 1-4 WHEN TESTED ON TRACKS 1-4. HORIZONTAL: EVOLVED ON, VERTICAL: TESTED ON. DISTANCE TRAVELED IN 5000 TIMESTEPS AVERAGED OVER 5 RUNS EACH

whole lap. Table I displays the performance of the driving attempts we chose to use for further experimentation.

B. Evolving for distance

A number of evolutionary runs were performed with distance fitness being the sole objective. In table II, the performance of controllers evolved on track 1 or on track 4 are tested on all four tracks. It is clearly possible to evolve very well-performing controllers for both tracks. Not surprisingly, the controllers perform best on the tracks they were evolved on, much in line with the results in [11].

Table III displays the steering and average fitness of the same controllers, relative to aggregate user data of aggressive or careful driving. From the very high squared errors here, it is plain to see that the driving is very unlike the particular human driving that was recorded (which does not necessarily mean that the driving is not human-like).

C. Modelling player behaviour

A number of evolutionary runs were performed with only the two objectives related to reproducing player behaviour, steering fitness and acceleration fitness. Table IV lists the performance of controllers created through this evolutionary modelling process on all track. The controllers are of four types: those created through modelling aggressive driving data on the same track as it was tested, those created through modelling all the agressive driving data (from all tracks), those created from careful driving data for one track, and lastly those created from careful driving data from all tracks. From this table, it is clear that the controllers created using

	agressive steering/acc.	careful steering/acc.
track1	0.576 / 0.3604	0.490 / 0.625
track4	1.105 / 0.3584	1.124 / 0.496

TABLE III

MEAN SQUARED ERROR (AVERAGED OVER 5 RUNS) OF THE OUTPUT OF NETWORKS EVOLVED FOR DISTANCE FITNESS ON TRACK 1 AND 4 WHEN PRESENTED WITH THE USERDATA ON ALL TRACKS

	agressive	with all data	careful	with all data
track1	1530.762	1395.8452	2018.714	1206.0948
track2	2729.028	2856.918	2557.396	2225.898
track3	380.3576	527.983	694.4642	344.6784
track4		347.1984		399.5694

TABLE IV

DISTANCES REACHED BY CONTROLLERS CREATED THROUGH MODELLING EITHER AGGRESSIVE OR CAREFUL USER DATA ON EACH TRACK. SOME RESULTS OF TRACK 4 ARE ABSENT BECAUSE NO HUMAN DRIVING DATA WAS COLLECTED ON TRACK 4

	agressive steering/acc.	with all data steering/acc.	careful steering/acc.	with all data steering/acc.
track1	0.039 / 0.095	0.043 / 0.130	0.029 / 0.248	0.037 / 0.228
track2	0.103 / 0.086	0.046 / 0.125	0.248 / 0.262	0.080 / 0.234
track3	0.133 / 0.160	0.345 / 0.137	0.086 / 0.185	0.037 / 0.230
track4		0.074 / 0.116		0.050 / 0.237

TABLE V

MEAN SQUARED ERROR OF THE NETWORKS TRAINED ON THE AGRESSIVE AND CAREFUL USERDATA ON SPECIFIC AND ALL TRACKS

modelling only do not drive the tracks as well as the human players they were modelled on, nor do they perform as well as the controllers evolved with distance as the only objective. In fact, for the harder tracks (3 and 4) they perform very badly. In general, the controllers that are trained only on the data for the particular tracks they are driving perform better than those trained on all tracks, with the notable exception of controllers trained on careful driving data for track 2.

The question of whether these controllers actually drive more human-like than the ones evolved only for distance fitness is partly answered by table V. In general, these errors are much lower than those found in table III. Thus, the controllers evolved using only the steering fitness and acceleration fitness objectives reproduce the recorded driving behaviour better than those evolved only for fitness - at least from a mathematical perspective. From a human perspective, it is hard to tell. The differences between the behaviour of TORCS in visual and non-visual modes (see section II-A) is currently hindering us from drawing firm conclusions on this matter, though initial observations support this hypothesis.

Figures 6 through 9 show scatter plots of the "super-pareto fronts": pareto fronts of the nondominated solutions from multiple combined pareto fronts. In this case, each pareto front was made up of the nondominated solutions from five separate runs. From these plots, it is clear that there is a tradeoff between steering fitness and acceleration fitness. One should notice that the color of these pareto front is the *evaluated* distance of the datapoints in the pareto front. This means that the pareto front on which the NSGA-II evolved is two-dimensional, but the graphs shows the performance of the networks in the third dimension: distance. This is different from the pareto fronts depicted in Figures 10-13 that show purely three-dimensional pareto fronts.

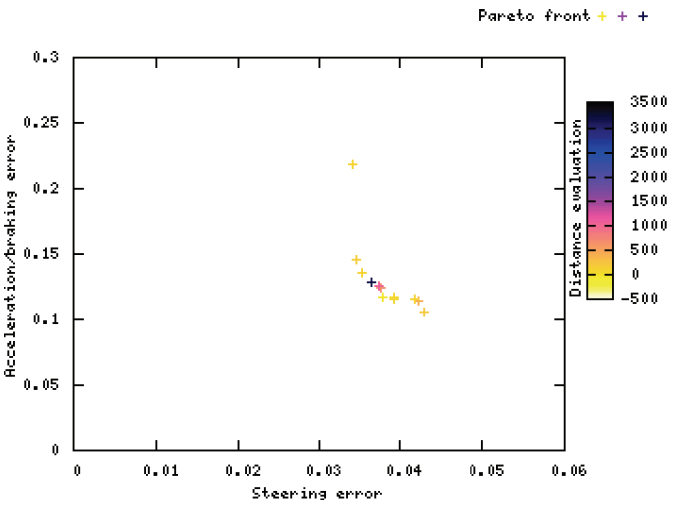

Fig. 6. 3D pareto front of the 3 objectives learned on track 1 with agressive userdata of all tracks, best of all runs

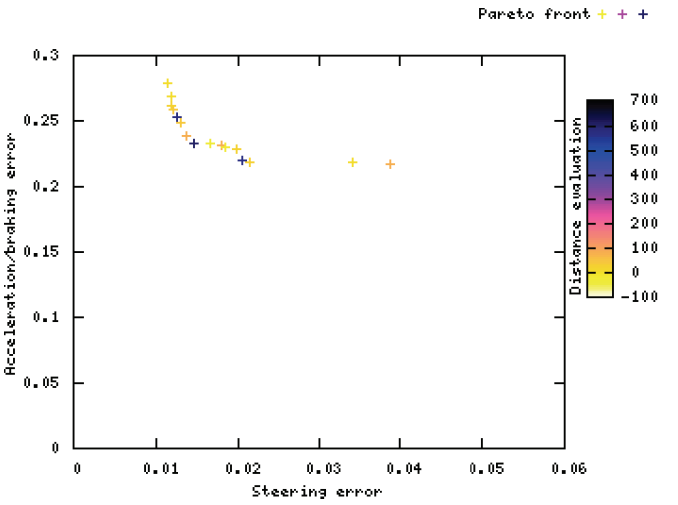

Fig. 7. 3D pareto front of the 3 objectives learned on track 1 with careful userdata of all tracks, best of all runs

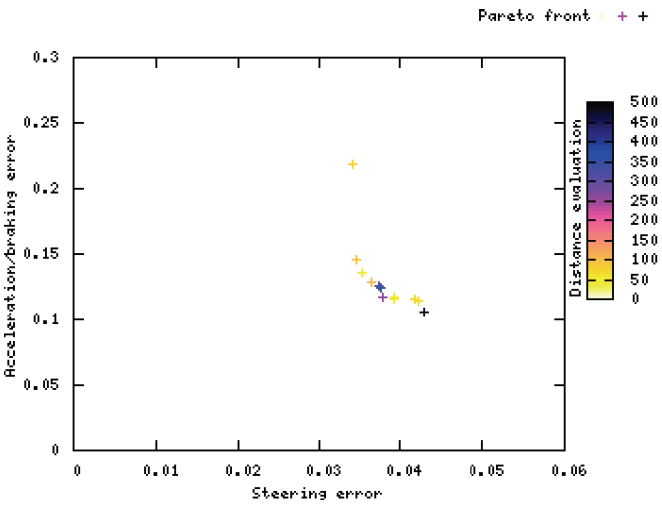

Fig. 8. 3D pareto front of the 3 objectives learned on track 4 with agressive userdata of all tracks, best of all runs

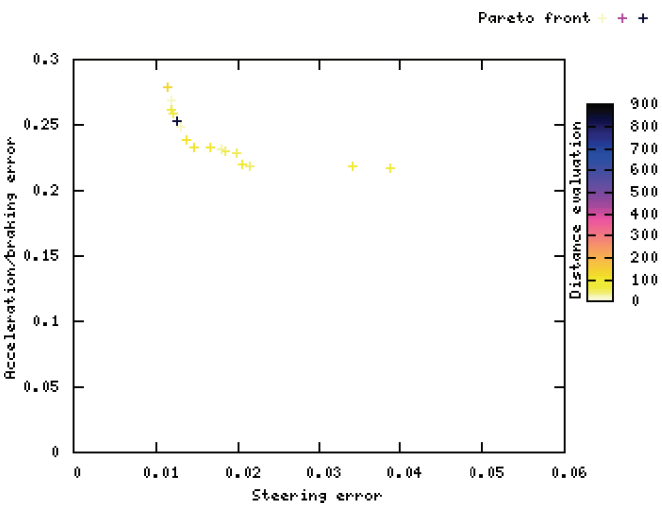

Fig. 9. 3D pareto front of the 3 objectives learned on track 4 with careful userdata of all tracks, best of all runs

D. Multiobjective evolution for performance and modelling

The final set of experiments concern the multiobjective modelling, where all three objectives were used. The results are here split up according to what sort of user data was used. In table VI results are shown for controllers evolved with distance fitness defined for each track, but user data for aggressive driving on all tracks taken together. Comparing these results with those for evolving with only distance fitness in table II, it seems that the addition of user data does not significantly affect either the maximum attainable distance fitness or the generalization ability of the controllers. The results in table VI are overall similar, meaning that the driving style that is being modelled seems to have no systematic effect on the driving fitness. It is interesting to note that the distance fitness for controllers evolved on track 1 and 2 are quite similar to the distance traversed by human drivers on the same tracks, whereas the evolved controllers for track 3 has clearly subhuman performance.

Table VII lists the average acceleration and steering fitnesses for these controllers. As expected these controllers do not outperform the controllers of table V in steering fitness, but they are not significantly higher either. This shows that adding a third objective (distance) to the multi-objective evolution does not remove the ability of the networks to model the userdata.

Figures 6 through 9 show scatter plots of the super-pareto fronts of these controllers, plotting distance fitness against steering fitness on the two main axes. Again, we see a tradeoff: either a controller drives well, or has low error on reproducing logged human steering. However, the fronts are very steep, with most solutions having similarly low steering error and moderate acceleration error. It is therefore possible

agressive	track1	track2	track 3	track4
track1	4697.062	1066.271	727.196	491.0446
track2	503.476	4252.355	250.447	179.671
track3	2142.521	911.939	2436.913	1416.731
track4	2354.070	514.454	1366.115	2685.888
careful	track1	track2	track 3	track4
track1	4252.408	930.7194	869.273	353.29252
track2	701.03544	4101.199	315.19009	164.43662
track3	2061.25154	1054.1768	2644.488	460.8872
track4	2127.8258	737.9388	814.658	2522.9

TABLE VI

PERFORMANCE OF CONTROLLERS MULTIOBJECTIVELY EVOLVED ON TRACKS 1-4 WITH AGRESSIVE AND CAREFUL USERDATA OF ALL TRACKS, WHEN TESTED ON TRACKS 1-4. HORIZONTAL: EVOLVED ON, VERTICAL: TESTED ON. DISTANCE TRAVELED IN 5000 TIMESTEPS AVERAGED OVER 5 RUNS EACH

agressive	steering	acceleration/braking
track1	0.160	0.165
track2	0.063	0.212
track3	0.092	0.380
track4	0.137	0.280
careful	steering	acceleration/braking
track1	0.035	0.291
track2	0.026	0.520
track3	0.052	0.294
track4	0.115	0.287

TABLE VII

MEAN SQUARED ERROR (AVERAGED OVER 5 RUNS) OF THE OUTPUT OF THE MULTIOBJECTIVELY EVOLVED NETWORKS WHEN PRESENTED WITH THE USERDATA ON ALL TRACKS

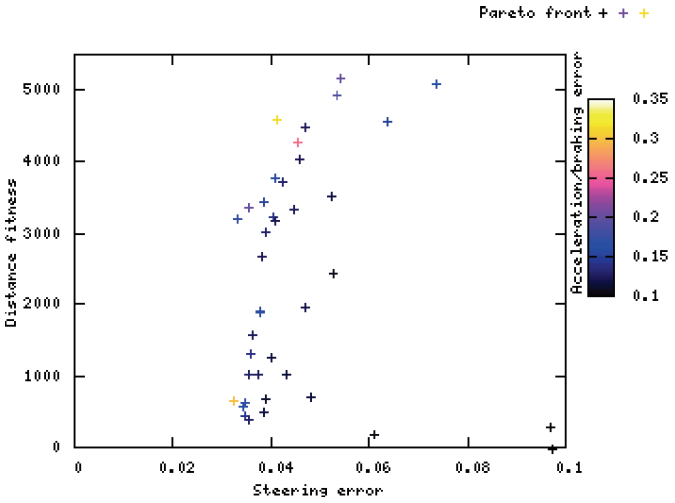

Fig. 10. Evaluation on track 1 of Multi-objective learning with aggressive userdata of all tracks, best of all runs

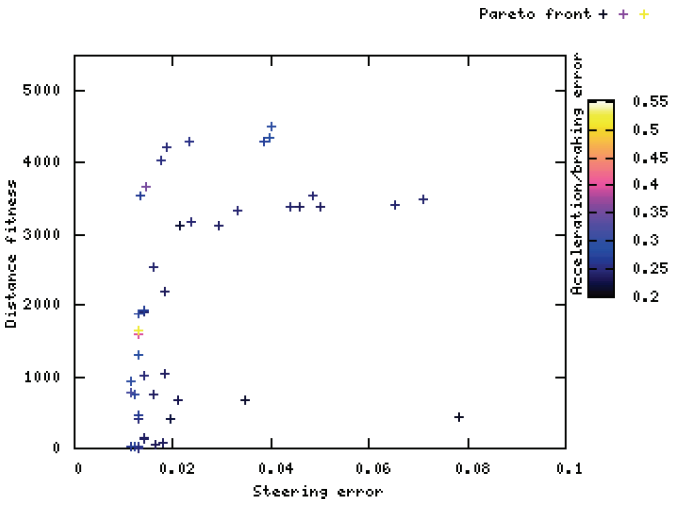

Fig. 11. Evaluation on track 1 of Multi-objective learning with careful userdata of all tracks, best of all runs

to find controllers that have high distance fitness but still close to the lowest possible steering error for any data set.

Finally, we performed a miniature *behavioural Turing test*. New controllers were evolved using a setup where small amounts of noise were added to all sensor readings, in order to reduce the differences between and non-visual modes. A small group of observers would look at the driving behaviour of some handpicked evolved controllers and give a rating from 1 to 10 in three categories: careful or agressive driving, human-likeness and driving quality. Although these results are far from statistically significant, some interesting trends were evident. Controllers evolved on aggressive user-data were consistently rated more agressive than controllers evolved on careful userdata. Furthermore, controllers evolved on userdata, especially those trained on careful userdata, were judged more human-like than controllers trained solely on distance. The driving quality of the latter controllers was rated higher than those where training involved userdata. Although more tests are required to confirm these statements, they indicate that our approach comes up with the expected results: controllers that have a lower driving performance, but seem more human-like.

IV. DISCUSSION

In trying to synthesize the results above, a number of observations stand out in particular. One is that there is indeed a tradeoff between being able to drive a track well and being able to accurately model human driving behaviour. Further, the NSGA-II MOEA is capable of exploring this tradeoff, as can be seen in the numerous scatterplots above.

The second observation is that while evolving controllers that drive particular tracks well is easy, modelling human driving behaviour is very hard, maybe too hard for the methods employed in this paper. In fact, as this problem can be seen as a sequence learning problem, it could be argued that standard methods for training recurrent networks on sequences should be tried, such as *Backpropagation Through Time (BPTT)*. As BPTT is not an evolutionary algorithm, it is not obvious how to combine it with the distance fitness measure into an evolutionary algorithm. One solution could be to first train a network on the user data using BPTT, and use it to seed the multiobjective evolutionary process.

Another way to combine the multiobjective approach with

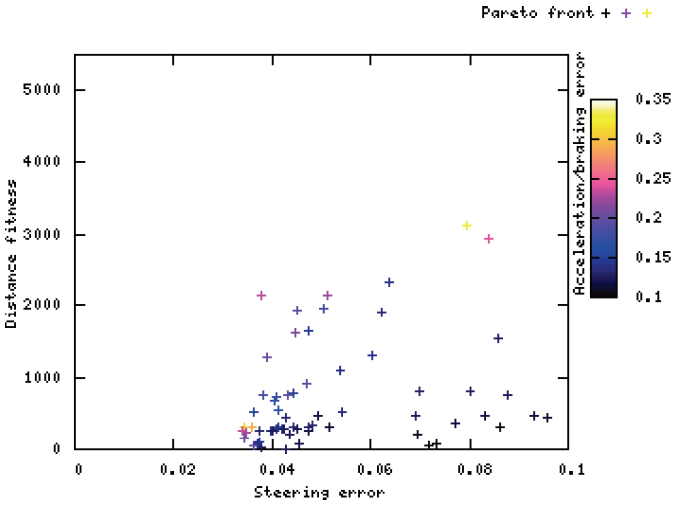

Fig. 12. Evaluation on track 4 of Multi-objective learning with agressive userdata of all tracks, best of all runs

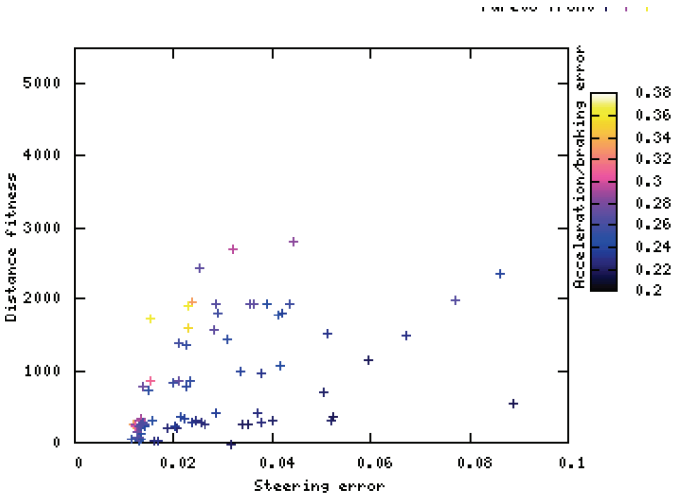

Fig. 13. Evaluation on track 4 of Multi-objective learning with careful userdata of all tracks, best of all runs

a state-of-the-art sequence learning algorithm could be to incorporate key elements of the *Evolino* algorithm, which is an evolutionary algorithm capable of training recurrent neural networks for supervised learning tasks [14]. The key idea is to evolve only the non-output weights, and replace evolving the weights from the hidden neurons to the output neurons by using a fast learning procedure such as linear regression at every network evaluation. This would ensure reasonably human-like outputs even in the beginning of the evolutionary search. And since this is an evolutionary method, it can be satisfactorily incorporated into the multiobjective *NSGA-II* framework as the human modelling component.

The three objectives propose here are not the only conceivable ones. A further objective to consider is the amount of damage taken during a lap, which reflects the carefulness or recklessness of a driver.

V. Conclusions

This paper proposed to use multiobjective evolutionary optimization to produce controllers that are similar to human players in particular respects, but which also perform well on a given task: *multiobjective modelling*. The main motivation for this was the need for at the same time robust and believable models of human game players. We explored this new concept in the context of modelling human driving in a racing game. Results were mixed. While we found that the MOEA was capable of finding and clarifying the tradeoff between similarity to recorded human behaviour and playing the game, we also found that the particular combination of sequence approximator and learning algorithm was not powerful enough to model the human behaviour as well as desired (though it was quite enough for evolving good game-playing behaviour). We believe that incorporating elements of a more powerful sequence learning algorithm into the framework of multiobjective modelling can solve this problem. Further, unexpected variability of the game platform made it hard to judge the human-likeness of evolved controllers.

Acknowledgements

This research was supported in part by the SNF under grant number 200021-113364/1.

References

[1] R. Herbrich, "(personal communication)," 2006.
[2] B. D. Bryant, "Acquiring visibly intelligent behavior with example-guided neuroevolution," in *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2008, pp. 801–808.
[3] J. Togelius, R. De Nardi, and S. M. Lucas, "Making racing fun through player modeling and track evolution," in *Proceedings of the SAB'06 Workshop on Adaptive Approaches for Optimizing Player Satisfaction in Computer and Physical Games*, 2006.
[4] ——, "Towards automatic personalised content creation in racing games," in *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, 2007.
[5] R. Koster, *A theory of fun for game design*. Paraglyph press, 2005.
[6] T. W. Malone, "What makes things fun to learn? heuristics for designing instructional computer games," in *Proceedings of the 3rd ACM SIGSMALL symposium and the first SIGPC symposium on Small systems*, 1980, pp. 162–169.
[7] S. Priesterjahn, "Online imitation and adaptation in modern computer games," Ph.D. dissertation, University of Paderborn, Paderborn, Germany, 2008.
[8] J. Matthews, "Interview with jeff hannan," http://www.generation5.org/content/2001/hannan.asp, 2001.
[9] M. Buckland, "Interview with jeff hannan," Publication date unknown. [Online]. Available: http://www.ai-junkie.com/misc/hannan/hannan.html
[10] B. Chaperot and C. Fyfe, "Improving artificial intelligence in a motocross game," in *IEEE Symposium on Computational Intelligence and Games*, 2006.
[11] J. Togelius and S. M. Lucas, "Evolving robust and specialized car racing skills," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2006.
[12] 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 *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, 2008.
[13] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 182–197, 2002.
[14] J. Schmidhuber, D. Wierstra, M. Gagliolo, and F. Gomez, "Training recurrent networks by evolino," *Neural Computation*, vol. 19, no. 3, pp. 757–779, 2007.