Mimicking Humanlike Movement in Open World Games with Path-Relative Recursive Splines

Emmett Tomai, Rosendo Salazar and Roberto Flores
Department of Computer Science, University of Texas – Pan American
Edinburg, TX, 78539, USA
tomaie@utpa.edu

Abstract

In this paper we explore the use of recursive cubic Hermite splines to mimic human movement in open world games. Human-like movement in an open world environment has many characteristics that are not optimal or directed towards clear, discrete goals. Using data collected from a simple MMORPG-like game, we use our spline representation to model human player movements relative to corresponding optimal paths. Using this representation, we show that simple distributions can be used to estimate control parameters to generate human-like movement across a population of agents in a novel environment.

Introduction

Online virtual worlds are an increasingly significant venue for human interaction. By far the most active virtual worlds are video games, headlined by the explosive growth of the Massively Multiplayer Online Role-Playing Game (MMORPG) genre in the last decade. With growing public awareness of the potential of embodied, virtual interaction, there has been increased interest in virtual worlds for education, training and scientific research (cf. Bainbridge, 2006; Dickey, 2005). Virtual agents play an important role in these online worlds, whether as characters or opponents in a game, or as virtual guides and assistants. In this paper, we consider the problem of creating agents that move in an open world environment in a way that could be considered human-like. Standard pathing algorithms allow an agent to reach a goal destination when a straight line path is not available. They choose paths that are subjectively reasonable, typically defined in terms of utility such as quickest travel time and attaching cost to certain areas (e.g. dangerous ones). In an open world environment, however, reasonable movement is not so easily defined in terms of goals and utility. First, human players do not always take optimal paths. Second, it is more difficult to model how player goals direct movement. For example, a player may have a goal of interacting with an entity in the world, and thus move to the location of that entity. However, they may veer to the left along the way, or wander off the path entirely before coming back, or take a suboptimal route to get there. This might still be a product of goal based movement, but the goals may be opaque, and possibly not even clear to the player. Were they avoiding an obstacle? Getting a different viewpoint? Not paying attention? Exploring the area? Or perhaps a combination of all those things. In this work we model human-like movement as divergence from the optimal path to a known goal, without knowing what unknown goals cause that divergence. We contribute a novel representation using *recursive cubic Hermite splines* that is portable to novel paths and maps. This representation can be used to generate a path to a goal location with human-like divergence, by sampling control parameters from distributions taken from actual human movement traces. We show these generated paths, for a population of agents, have similar qualities to human player paths, and that those qualities are maintained when sampling from one environment and testing in another.

Related Work

There has been prior work, primarily in the computer graphics community, using agents to simulate plausible movement among other agents. Shao argued for a full-agent approach combining reactive controllers, scripted behaviors and mental states with path-planning and goal setting (Shao & Terzopoulos, 2005). In contrast, Treuille proposed a deliberately simpler model with less individual variation but less cost in modeling and run-time resource requirements (Treuille et al., 2006). Lerner showed a technique for entirely avoiding rule development with agents that retrieve and mimic human samples taken from crowd videos (Lerner et al., 2007). Henry trained an agent

Copyright © 2013, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

controller using inverse reinforcement learning to navigate through crowded environments, making the distinction between shortest and human-like paths (Henry et al., 2010). Both the environment and the training data were generated by simulation. These approaches focus particularly on maintaining plausible behavior in crowded environments, as human pedestrians do. Players in an open virtual world do not move with that deliberative pace and flow, so the techniques involved may not transfer well. It is also worth noting that the majority of evaluations are qualitative visual examples, underlining the difficulty of quantitative analysis.

A number of evolutionary approaches have been evaluated for developing more human-like agent controllers. Graham used a genetic algorithm to evolve an artificial neural network that implements dynamic obstacle avoidance while following a direct path (Graham, 2005). Togelius evaluated several co-evolution strategies for creating car racing controllers with the aim of deploying a diverse population of human-like AI opponents in a car racing game (Togelius et al., 2007). These approaches are evaluated according to whether they effectively traverse space while avoiding obstacles and hitting checkpoints, making them human-like from a functional point of view.

In the domain of video games, particular interest has been shown in developing human-like movement for agents in the *first-person shooter (FPS)* genre. Geisler notes the high predictability and manual labor involved in traditional AI scripting of game agent opponents (*bots*) as motivation for automatic learning of human-like behavior (Geisler, 2004). These behaviors included low-level movement primitives such as changing direction, changing speed and jumping, as well as basic game actions such as aiming and firing a weapon at opponents. Gamez showed that a global workspace architecture combining independent, hand-tuned neural networks can deliver human-like bot control (Gamez et al., 2013). Thurau used self-organizing maps and artificial neural networks to learn those primitive actions based on position and relative enemy positions (Thurau et al., 2003). Geisler evaluated both naïve Bayes and neural network approaches to this problem with promising results (Geisler, 2004). Schrum created a FPS bot architecture that learns combat behavior using neuroevolution (Schrum et al., 2012) and won the 2K Games' 2012 BotPrize while being judged as human more than 50% of the time (Karpov et al., 2012). These and numerous other results (cf. Galway et al., 2008) have demonstrated that machine learning and evolutionary computation are well suited to optimizing behavior control in the fps domain, where the problem has a reactive nature (e.g. positioning relative to other agents, strategic responses), a small number of output dimensions (e.g. movement and facing) and works at a single level of abstraction (Bakkes et al., 2012). However, the larger question of complex behavior requires working at multiple levels of abstraction (Bakkes et al., 2012). Ultimately, that is the goal for open world behavior as well.

Modeling Player Movement

In MMORPGs, players control avatar characters in a physically simulated virtual world. In contrast to more reactive and/or linear environments in other genres, players roam freely in the world, picking up tasks and completing them at their own discretion. At a high level, there is goal-directed travel between the regions where tasks are acquired, completed and turned in for credit. To collect player behavior data, we created a lightweight, research focused MMORPG. We gathered data from 37 human players playing together in a laboratory setting. The game collects a wide range of data for each player, including movement, avatar actions, UI actions and visibility of other entities. For the purposes of this study, we isolated traces of player movement between regions. For example, Figure 1 shows two such traces as yellow lines. They are identified from the collected data by interaction events (interacting with in-game characters, enemies or other entities) at the beginning and end. Most of the movement within regions consists of combat or other highly directed behavior, so was set aside for other study. The data set consists of 162 movement traces grouped into 8 region-to-region groups.

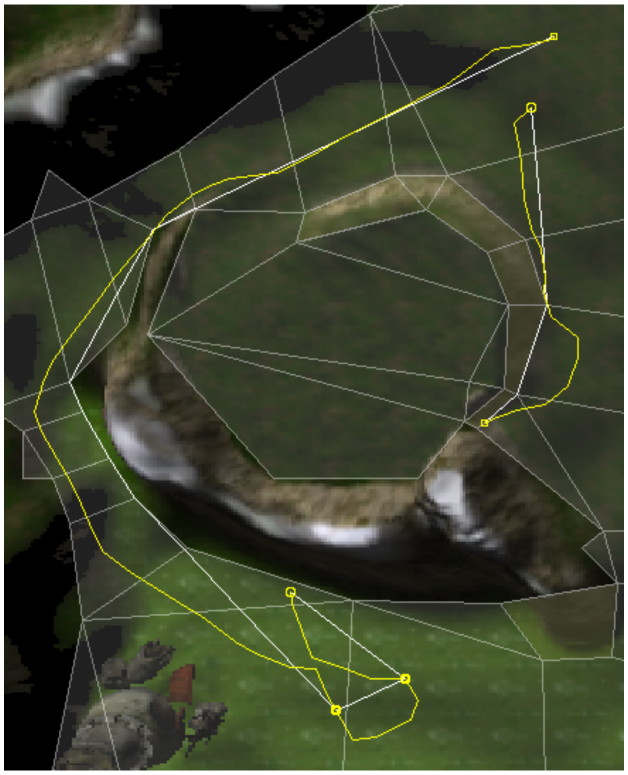
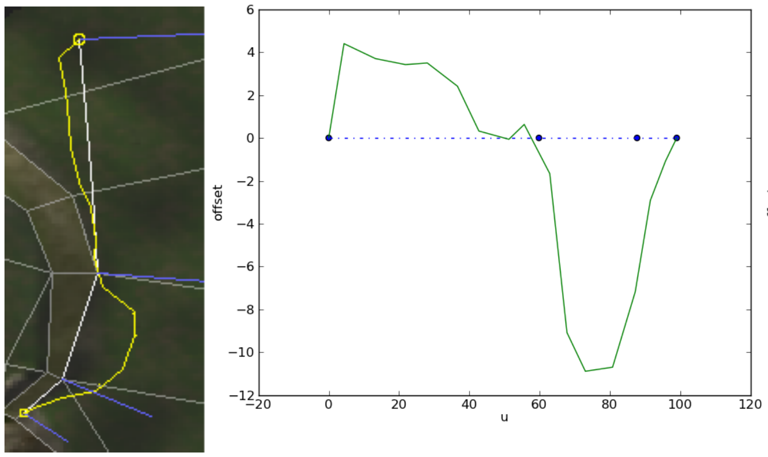


Figure 1: Navigation mesh with human traces (yellow) and corresponding optimal paths (white).



Figures 2a and 2b: A human trace (yellow) projected into path-relative space u,offset based on the normals (blue) of the corresponding path (white).

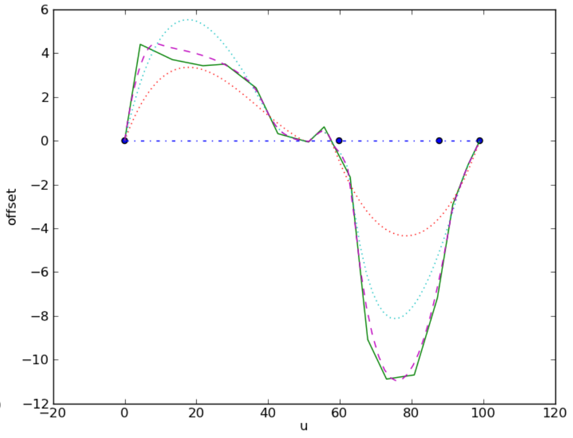


Figure 3: Human trace (solid, green) fit by a recursive Hermite spline (first-order: dotted, red; second-order: dotted, blue; third-order: dashed, purple).

Path Modeling

The first step in our approach is to identify a path to a goal location. Many of the systems discussed in the related work section train and test in the same environment (game map), because they learn movement relative to the topology of that map. To allow our model to apply to novel environments, we instead model movement relative to paths that can be automatically generated using fast, standard algorithms. Variations on A* search have long been the standard way to compute paths in research and the games industry. In some games, this can be implemented on a grid in a very straightforward way. However, in sophisticated virtual worlds, it is common to use a *navigation mesh*. This is a mesh of edge-connected polygons (typically triangles and/or quadrilaterals) that is manually overlaid by a designer on the game world. Figure 1 shows part of the navigation mesh in our lightweight MMORPG. Because the navigation mesh is still a graph, graph search can be used to find paths. However, once a sequence of mesh nodes is identified as a path, specific waypoints must be chosen within those nodes. Using centroids, corners or edge midpoints creates very unnatural, jagged paths. A common solution is the *funnel* or *string-pulling* algorithm, which finds the optimal waypoints through a sequence of mesh nodes (Lee & Preparata, 1984). Figure 1 shows two human movement traces in yellow. First consider the shorter trace on the right. The corresponding straight white lines show the path generated using the navigation mesh and funnel algorithm between the start and end points (yellow circles) of the human trace. The corners of the white paths are *funnel points*.

To represent the human movement in a path-relative way, we project the points in the trace onto the path. Each point becomes a tuple (*u*,*offset*), where *u* is the distance along the path and *offset* is the distance along the normal from the path to the point. Those normals are computed at each funnel point (shown in blue in Figure 2a) as the average of the perpendicular normals of each adjacent path edge, and linearly interpolated between the funnel points. Figure 2b shows the projection of the human movement trace from Figure 2a into path space (*u*,*offset*). The large blue dots along the x-axis indicate the positions of the funnel points in *u*. In these experiments, the sampling rate for the project is equal to the sampling rate used in the data collection, but it does not have to be.

The long human movement trace on the left side of Figure 1 shows multiple yellow circles along the trace that are treated as *intermediate target locations*. In plain terms, any time the player is not moving in a way consistent with the path to the end goal (the end of the trace), they must be moving towards some other target location within the trace. These intermediate target locations are assumed to be motivated by some unknown player goal (e.g. went over to look at a flower along the way). The human movement trace is automatically segmented by those locations such that the projection of each trace segment is monotonic along the generated path from the beginning to the end of that segment. The segmentation is done by working backwards along the trace, adding points to a segment until it results in a non-monotonic projection, which identifies an intermediate target location at that point. There are two intermediate target locations in the example trace. This work only uses intermediate target locations, and does not extend to predicting their occurrences.

Recursive Splines

To facilitate learning and mimicking these movement traces, we present a compact representation based on *recursive cubic Hermite splines*. These splines (and subsets such as Catmull-Rom splines) are commonly used to represent curving paths in computer animation. A Hermite spline defines a curve as a linear interpolation between two points based on a tangent leaving the first point and a tangent arriving at the second point. This is a natural fit for an agent leaving the path and curving back to it, as seen in the example traces. In order to fit the human movement trace, we represent the coarse, low-frequency component of the trace using what we will call the first-order recursive splines. The first-order recursive splines for the example trace shown in Figure 2a and 2b are shown as a red dotted line in Figure 3. Each spline leaves the path and returns to it, so for the single crossing of the human movement trace in this example, there are 2 first-order recursive splines. Each spline is defined by its start and end points in *u* (the start and end of the path and the single crossing point), and the tangents at those points. The tangents are estimated by taking the derivative of the cubic B-spline interpolation at those points using the scipy[1] implementation of Dierckx's spline fitting algorithms (Dierckx, 1993). Those estimated tangents are then refined using a recursively finer grained grid search over the angle and magnitude to minimize the mean square distance (MSD) between the spline curve and the human trace points.

Once the first-order splines have been created, the human trace in (*u,offset*) is projected against those splines to create a new projection of the remaining error. That projection is used to recursively create second-order recursive splines that make higher-frequency crosses of the parent first-order splines. This process continues until a minimum threshold MSD is reached, or a specified depth limit is reached. The dotted blue line in Figure 3 shows the second-order spline fit (with the second-order splines added to the first-order), and the dashed purple line in Figure 3 shows the third-order spline fit. Table 1 shows the mean square distances at different depths of recursion, averaged across all the human movement traces being used in this work. As shown, the gains in fit fall off sharply after third-order splines, so that limit is used.

Table 1: Average mean square distances between the recursive splines and the human traces they are fitting.

depth	**MSD**	**STDDEV**
0	83.77601	101.13
1	14.34542	18.37096
2	2.851349	3.767416
3	0.77223	0.776236
4	0.460432	0.44196
5	0.412963	0.430128

[1] http://www.scipy.org

By modeling the human movement traces as recursive splines relative to the path in (*u,offset*), we have created a small number of parameters that control the shape of the movement in a portable way. Given a set of training traces, a model can be trained to sample from the distribution of appropriate values for each parameter.

Path Generation

To generate a human-like path to a goal, an agent first must compute the optimal path as described in the prior section. It then generates a sequence of recursive splines (to a given depth) for human-like movement along that path by using a model to sample controlling parameter values. For each path segment (separated by the funnel points), the number of *crossing points* is sampled from the model. These values are modified according to two constraints. 1) If a segment runs along an impassable edge of the navigation mesh, there can be no crossing points in that segment. 2) If a crossing causes the path to end up on an impassable side in a later segment, another crossing is added. The locations of the crossings within each segment are also sampled from the model. Once the set of crossing points is determined, it defines the beginnings and ends of the first-order splines. The starting and ending tangent parameters are similarly sampled from the model. It then recursively generates higher-order splines in the same manner, to the specified depth. The only difference is that the higher-order splines do not take the path segments into account, but predict crossing points along their parent spline instead. Once the full recursive splines have been created, they can be followed by evaluating it and its children along *u* (using the agent's movement speed) and summing the resulting offsets, then projecting that (*u,offset*) from the path back into world space. The cost of the algorithm is polynomial in the number of crossings generated at each level, which in this data set is predominantly in the range of 1-3 at all levels. Even this cost could be amortized over the entire movement by only generating the recursive spline up to the next crossing at the deepest level.

Figure 4 shows human-like paths (blue) generated by this algorithm using the same target locations as the human movement traces in the running example. Figure 5 shows a complete set of region-to-region human movement traces (yellow) and a corresponding set of generated paths (purple).

Parameter Sampling

Path generation as described above requires a model of the controlling parameter distributions. The crossing point parameters are the number of times a path segment or spline is crossed by a higher-order spline, and the relative locations of those points as percentages of the extent of the segment or spline in *u*. The spline tangent parameters are the (*u,offset*) components of the tangent, normalized by the path extent in *u* covered by the spline.

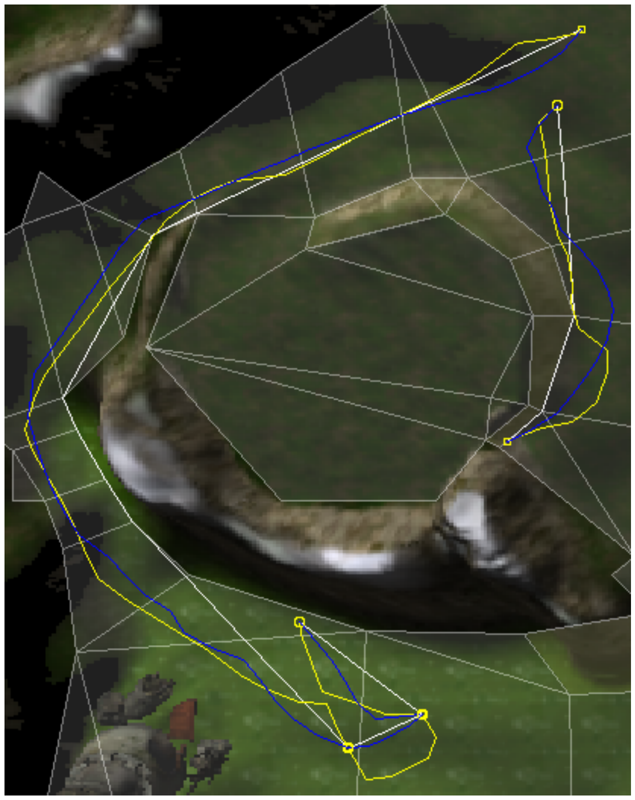


Figure 4: Generated paths (blue) over the same target locations as the human traces (yellow).

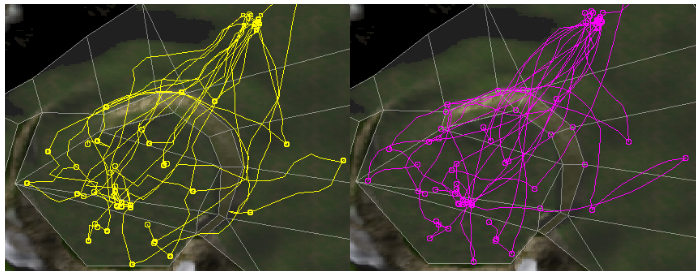


Figure 5: Human traces (yellow) and generated paths (purple) over the same target locations.

In this work, we investigate two distributions for parameter estimation. These are taken globally over the set of training traces. The first, intended to be a baseline, is the Gaussian normal distribution of the parameter given the mean and standard deviation of the training data (hereafter *normal*). These parameters do not follow a normal distribution in the training data, but it is the obvious first choice for introducing randomness into agent movement. The second, intended to verify the feasibility of this approach and give insight into evaluation metrics, is *inverse transform sampling* from the training distribution (hereafter *training*). Given a parameter as a random variable *X* and its *cumulative distribution function (CDF) F*, a random number *x* in *X* can be generated by generating a random number *y* from a uniform distribution such that *F(x) = y* (cf. Devroye 1986). This is simple to implement for finite, discrete data, as the CDF is simply the integration of the *probability density function* of *X*, and the inverse transform can be done as a lookup table, essentially the same as common game industry *hit tables*.

Evaluation

Evaluating behavior that is not intended to be optimal or narrowly task focused is quite difficult. The problem is further complicated because the goal is not to mimic a particular human movement trace, but to generate a population of agents have the same qualities in their movement as the human players. Various metrics have been tested (cf. Gamez et al., 2013), but their utility remains an open question. Here we consider distributional similarity: for a given metric, to what extent does the distribution generated by an agent population match the distribution generated by a human player population? As in many curve matching approaches, the similarity is computed as the mean square distance (MSD) between the values in the distributions, evaluated in discrete bins over the union of their ranges. We experimented with using cubic smoothing over the distributions in the MSD calculation, but found no notable difference in the results.

In the first experiment, the normal and training distributions are drawn from all the region-to-region traces in the data set. Then, for all the traces in that set, the algorithm generates a population of paths, over the same target locations, for each of the two distribution models. The purpose of this self-test experiment is to calibrate the similarity metrics for an agent population vs. a human population.

In the second experiment, the region-to-region traces are divided into two sets. Each set contains half of the start/end region pairings, selected randomly. The normal and training distributions are taken from one set, then used to generate a population of paths according to the traces in the other set. The purpose of this experiment is to verify that this approach can use data from one set of paths with a certain topology to generate paths for novel locations in a different topology. The generated movement should show the same distribution similarity to human traces as the self-test. In both experiments we hypothesize that the human-generated training distribution should be more effective at parameter estimation than the normal distribution.

Four evaluation metrics are used. The number of *path crossings* is an indicator of broad (low-frequency) back-and-forth movement off the optimal path. The number of *high-frequency crossings* is counted from the higher-order splines crossing the first-order splines, an indicator of how smooth or bumpy the path is. The *path area* is the mean square distance between the generated path and the optimal path, an indicator of how far off the optimal path the generated path moves. Finally, the *path slack* is the length of the generated path divided by the length of the optimal path, an indicator of how direct the movement is. Each metric is calculated for the human traces, the paths generated with the normal distribution and the paths generated with the training distribution. The distribution of each metric for the generated paths is compared to the distribution of that metric for the human traces, and the MSD is calculated. Smaller MSD values indicate more similarity to the human traces, as a population. Each experiment was run 5 times and the mean and standard deviation reported.

Results

The results of the experiments are shown in Table 2. In path crossings and path slack, the training distribution outperforms the normal distribution as predicted in both experiments (best values in bold). However, in path area the values show no difference relative to their standard deviations, and in high-frequency crossings the normal distribution outperforms the training distribution even in the self-test. It does not appear from these results that the training distribution has a notable advantage over the normal distribution. Comparing the two experiments, the distributions perform in a similar pattern and range in experiment 2 as experiment 1. This provides some evidence that even these distributions, with no sophisticated correlations to environmental variables, have some portability. However the magnitude of the difference in distribution similarity remains un-calibrated except by visual, qualitative assessment of the generated paths.

Table 2: Results for self-test (Exp1) and region-to-region (Exp2) experiments for normal and training distributions. HF crossings = high-frequency crossings.

Metric	Dist	Exp. 1		Exp. 2	
		mean	std	mean	std
Path crossings	normal	0.316	0.055	0.532	0.285
	training	**0.115**	0.059	**0.305**	0.280
HF crossings	normal	**0.015**	0.001	**0.034**	0.016
	training	0.031	0.005	0.072	0.019
Path area	normal	0.001	0.001	0.007	0.005
	training	0.001	0.001	**0.006**	0.003
Path slack	normal	12.31	2.543	29.32	4.78
	training	**10.97**	2.031	**27.59**	4.60

Conclusion

We have presented an approach to modeling and generating human-like movements in an open-world virtual environment such as found in the MMORPG genre. We have shown how recursive splines can be used for path-relative modeling that translates easily to different locations and environments based on industry-standard path-finding techniques. We have also provided empirical evidence that simple distribution sampling of parameters from human traces can generate these splines on a novel path, in a way that satisfies some measures of human-likeness to the movement. Finally, we have presented a set of possible quantitative metrics for such measurement and demonstrated their use with a measure of distribution similarity.

While this technique generates what appear to be quite reasonable and useful movements, the use of global normal and training distributions is a substantial simplification. We have done preliminary tests with Bayesian support vector regression and Gaussian processes for parameter estimation, based on features such as the width of the navmesh funnel, the sharpness of path segment corners and the presence of obstacles. However, given the difficulty of evaluation, we felt that the strong performance of global distribution sampling warranted further investigation. As the results show, there is at least a reasonable level of fidelity that can be gained in this very general, straightforward way. We hope that this study will provide a clearer baseline to expand the work with machine learning.

The most challenging element continues to be finding ways to quantitatively evaluate a population of agents' behaviors for human likeness. Clear future work includes having humans make qualitative judgments, but we must continue to establish objective measures and hypotheses about them, or a lot of time may be wasted on expensive and undirected human subject evaluations.

Because this work is applicable to real-time simulation, and shared virtual worlds in particular, the practical efficiency of the algorithm is another important question. It is worth noting that hit tables are well understood, and the spline evaluations are common in graphics applications and conducive to matrix-based implementation, creating possible synergy with consumer hardware acceleration.

Acknowledgements

This material is based upon work supported in part by the U.S. Army Research Laboratory and the U.S. Army Research Office under grant number W911NF-11-1-0150.

References

Bainbridge, W.S. (2007). The Scientific Research Potential of Virtual Worlds. *Science.* Vol. 317 no. 5837 pp. 472-476.

Bakkes, S., Spronck, P., and van Lankveld, G. (2012). Player Behavioral Modeling for Video Games. *Entertainment Computing*, Vol. 3, Nr. 3, pp. 71–79.

Devroye, L. (1986). *Non-Uniform Random Variate Generation*. New York: Springer-Verlag.

Dickey, M.D. (2005). Three-dimensional virtual worlds and distance learning: two case studies of Active Worlds as a medium for distance education. *British Journal of Educational Technology.* Volume 36, Issue 3, pages 439–451.

Dierckx, P. (1995). Curve and surface fitting with splines. *Monographs on Numerical Analysis*, Oxford University Press.

Gamez, D., Fountas, Z., Fidjeland, A.K. (2013). A neurally controlled computer game avatar with humanlike behavior. *IEEE Transactions on Computational Intelligence in Games*, vol 5, no 1, March 2013.

Galway, L., Charles, D. and Black, M. (2008). Machine learning in digital games: a survey. *Artificial Intelligence Review*. Volume 29, Number 2, 123-161.

Geisler, B. (2004). Integrated machine learning for behavior modeling in video games. In: Fu D, Henke S, Orkin J (eds) *Challenges in game artificial intelligence: papers from the 2004 AAAI workshop*. AAAI Press, Menlo Park, pp 54–62.

Graham, R. (2005). Realistic Agent Movement in Dynamic Game Environments. DiGRA 2005: Changing Views: Worlds in Play, 2005.

Henry, P., Vollmer, C., Ferris, B. and Fox, D. (2010). Learning to navigate through crowded environments. In *Proceedings ot the 2010 IEEE International Conference on Robotics and Automation*, Anchorage Convention District, May 3-8, 2010, Anchorage, Alaska, USA

Karpov, I.V., Schrum, J., Miikkulainen, R. (2012). Believable Bot Navigation via Playback of Human Traces. In Philip F. Hingston, editors, *Believable Bots*, 151--170, 2012. Springer Berlin Heidelberg.

Lee, D.T. and Preparata, F.P. (1984). Euclidean shortest paths in the presence of rectilinear barriers. *Networks*, 14:393--410.

Lerner, A., Chrysanthou, Y. and Lischinski, D. (2007). Crowds by Example. *EUROGRAPHICS*, Vol 26, no 3.

Schrum, J., Karpov, I.V. and Miikkulain, R. (2012). Humanlike Combat Behavior via Multiobjective Neuroevolution. In Philip F. Hingston, editors, *Believable Bots*, 119--150, 2012. Springer Berlin Heidelberg.

Shao, W., and Terzopoulos, D. (2005). Autonomous pedestrians. In SCA ’05: *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM Press, New York, NY, USA, 19–28.

Thurau, C., Bauckhage, C., Sagerer, G. (2003). Combining self-organizing maps and multilayer perceptrons to learn bot-behavior for a commercial game. In Mehdi Q, Gough N, Natkin S (eds) *Proceedings of the 4th international conference on intelligent games and simulation*. Eurosis, pp 119–123.

Togelius, J., Burrow, P. and Lucas, S.M. (2007). Multi-population competitive co-evolution of car racing controllers. *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, 4043-4050.

Tomai, E., Salazar, R. and Flores, R. (2013). Simulating Aggregate Player Behavior with Learning Behavior Trees. In *Proceedings of the 22nd Annual Conference on Behavior Representation in Modeling & Simulation*.

Treuille, A., Cooper, S. and Popović, Z. (2006). Continuum Crowds. *ACM Trans. Graph*, vol 25, pg 1160-1168.