Context-Sensitive Data-Driven Crowd Simulation

Cory D. Boatright[1,2*] Mubbasir Kapadia[1†] Jennie M. Shapira[1‡] Norman I. Badler[1§]

[1]University of Pennsylvania [2]Grove City College

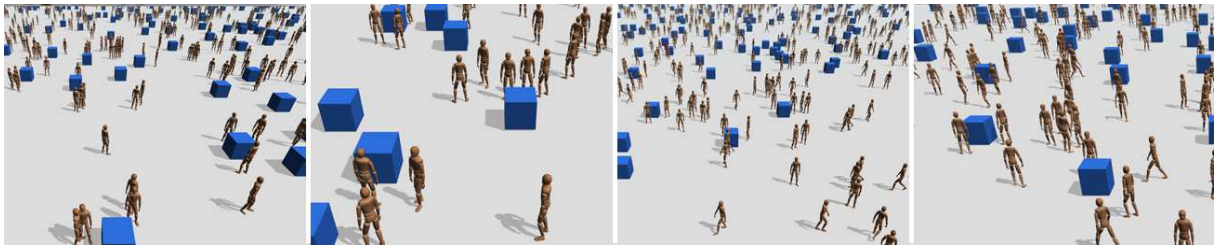

Figure 1: *Multiple views of a 3,000 agent simulation with high quality rendering.*

Abstract

In terms of computation, steering through a crowd of pedestrians is a challenging task. The problem space is inherently high-dimensional, with each added agent giving yet another set of parameters to consider while finding a solution. Yet in the real world, navigating through a crowd of people is very similar regardless of the population size. The closest people have the most impact while those distant set a more general strategy. To this end, we propose a data-driven system for steering in crowd simulations by splitting the problem space into coarse features for the general world, and fine features for other agents nearby. The system is comprised of a collection of steering contexts, which are qualitatively different overall traffic patterns. Due to their similarity, the scenarios within these contexts have a machine-learned model fit to the data of an offline planner which serves as an oracle for generating synthetic training data. An additional layer of machine-learning is used to select the current context at runtime, and the context's policy consulted for the agent's next step. We experienced speedup from hours per scenario with the offline planner and 10 agents to an interactive framerate of 10FPS for 3,000 agents using our data-driven technique.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence—Multiagent Systems I.5.2 [Pattern Recognition]: Design Methodology—Classifier Design and Evaluation;

Keywords: crowd simulation, steering, machine learning

1 Introduction

Crowd simulations are increasingly called upon for realtime virtual experiences. This push also includes a component of dynamic interaction with a user which adds additional unpredictability to the agents' decision-making process. The problem of predicting *a priori* the possible situations an agent will encounter rapidly becomes intractable in the face of increasing parameters in the form of users given more freedom in their virtual worlds, and thus we need algorithms that are scalable not only in agent count, but circumstance as well.

We introduce the concept of steering contexts—collections of situations selected for their qualitative similarity. By identifying such contexts we also divide the problem space, which limits the necessary scope of data-driven solutions. which were previously data-bound due to the complexity of human nature, data collection methods, and environmental limitations. Furthermore, we introduce a pipeline that leverages these contexts through the use of machine-learned models trained on synthetic data from a space-time planner to counter the logistical challenges of attaining sufficient scenario coverage from real-world observations. These models take as input nearby agents and obstacles and return the next footstep the agent should take.

This paper makes the following contributions:

- We introduce and use steering contexts to separate data for easier machine learning.
- We demonstrate the efficacy of synthetic training data from stochastically generated samples for better control over data collection resulting in more universal coverage of possible situations.
- Our pipeline produces a fast runtime algorithm with similar steering characteristics to a slower, more optimal algorithm.

2 Related Work

Following seminal work [Reynolds 1987] on flocking behaviors using particle systems, the field of crowd simulation has grown into a well-developed, multi-faceted area of study. In this section we review other publications most applicable to this work and for a broader survey of the field we refer the reader to the reviews in [Thalmann and Musse 2007; Pelechano et al. 2008].

Crowd simulation strives to replicate the pedestrian behavior of a group of people as realistically as possible while remaining computationally tractable. Due to this pull between two extremes—human complexity and processing speed—algorithms have been formulated as an abstraction to human behavior. These abstractions vary

*e-mail:cdboatright@gcc.edu
†e-mail:mubbasir@seas.upenn.edu
‡e-mail:jshapira@seas.upenn.edu
§e-mail:badler@seas.upenn.edu

VRCAI 2013, November 17 – 19, 2013, Hong Kong.

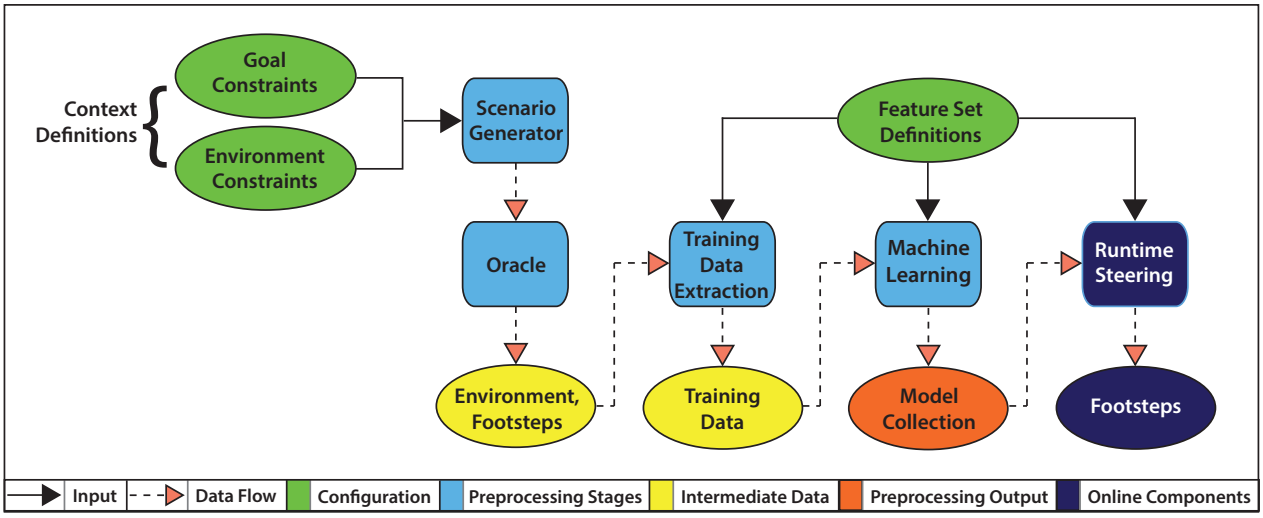

Figure 2: *Our pipeline for using steering contexts to develop a machine-learned model for use at runtime. The majority of the pipeline is offline processing. A collection of models is trained on data extracted from an oracle algorithm's solution to steering situations, which are stochastically generated. Then each model is a boosted decision tree with its own specialization. The action space consists of footsteps as an advantageous discretization which permits direct control and modeling of human locomotion.*

in how they approach the problem of moving so many agents.

Centralized Techniques. This category of approaches looks at the agents as pieces, either discrete or part of a continuous entity, on a board and moves each agent in accordance with a desired global outcome. Since a centralized process is planning their actions, agents appear to have an omniscient knowledge of their environment. Particle system approaches [Reynolds 1987; Reynolds 1999] replace the Newtonian physics of a typical n-body simulation with social forces. These particle approaches are further refined in the social force models of [Helbing and Molnar 1995; Pelechano et al. 2007].

Centralized techniques rely on a broad conformity amongst the population for best efficiency as seen the fluid-like approach of [Treuille et al. 2006]. This is an acceptable premise for group-dynamic simulations as used in the study of crowd flows in religious pilgrimages [Schneider et al. 2011] and emergency evacuations [Narain et al. 2009], but such an approach does not handle low-level micro-management well, which is expected when a user is an active participant in the virtual world rather than a passive observer.

Agent-based Techniques. To introduce more individuality in a simulation's agents, we can make steering an integral part of the agents' abilities.

Geometric algorithms such as [van den Berg and Manocha 2008; Guy et al. 2009] determine their next action based on which velocities may avoid a collision with another agent. This similar to the approach used by [Ondřej et al. 2010] which uses a synthetic sense of vision to determine information about other agents' trajectories and adjust accordingly. Agents have also used affordance fields [Kapadia et al. 2009] to try to find safe passage to a goal. A cognitive system was used in the seminal work [Shao and Terzopoulos 2007] which included utility functions for desires, an attentional system to limit perception of the environment, and a motor system to carry out actions. Recently, a rule-based adaptive system [Singh et al. 2011] was proposed that switched between other steering algorithms to best suit an agent's needs.

Machine learning has been used [Metoyer and Hodgins 2003] which takes designer suggestions for how agents should steer in their world and fits a model. Additionally, samples of real-life steering behavior can be used with the machine learning to fit better models.

Data-Driven Techniques. Work in data-driven steering has focused primarily on generating local-space samples from observations of real people. In [Lerner et al. 2007] video samples were compiled into a database which was queried at runtime and trajectories were copied and used by the agents based solely on the similarity of the agents' surroundings to the video examples. The work of [Lee et al. 2007] used a more constrained state space of discretized slices around an agent and focused more on recreating group dynamics than individual steering. A similar state space is used by [Torrens et al. 2011] as one of two state spaces. A separate state space consisting of a discretized view frustum was used for environmental navigation. In common to all these techniques is using one collection of samples for all navigation under a single model.

Comparison to Related Work. Our work builds on the adaptive use of algorithms in [Singh et al. 2011]. While the adaptive algorithm swapped between policies based on hand-coded rules, we employ machine learning to fit a model that determines which policy to use for a given decision. We also expand on the idea of failure sets from [Kapadia et al. 2011] by taking the concept further with the use of their inverse to create contexts for steering. Our use of "contexts" is different from that found in [Lerner et al. 2010] as our contexts are egocentric, not scenario-wide. Another data-driven method seen in [Courty and Corpetti 2007] focuses on capturing the dynamics of the overall crowd, while we focus on the individual agents. The closest data-driven system compared to our pipeline is that of [Torrens et al. 2011] which uses interchangeable state spaces but also uses clustering to try to separate data after the fact where we separate the data from the beginning of the process. Our exclusive use of an oracle algorithm in lieu of real-data is also unique to this paper.

3 Technique

We now explain our pipeline shown in Figure 2 for the integration of various contexts into a unified steering algorithm. First, training data must be collected, which we generate by means of an oracle algorithm. Next, the various machine-learned models must be fit to the data. Finally, these models are used at runtime to decide where an agent's next footstep should be placed.

3.1 Steering Contexts

We identified 24 qualitatively different types of steering challenges, which we call steering contexts. These steering contexts were defined intuitively based on overall traffic pattern, agent density, and the presence or lack of obstacles. Each context has a policy especially fit to it using decision trees. These trees take in a feature vector consisting of the local environment and return the next footstep location for the agent.

As seen in Figure 3, this allows for the rapid creation of steering policies for a diverse range of scenarios while still abstracting at a high level to manage the learning process and keep the number of features needed low. Agents distinguish which context their environment best represents through the use of another classifier using more abstract features. The multiplicity of steering policies allows for a more robust total system, as the agents' steering automatically adjusts as their environment evolves.

3.2 Training Data Generation

We define two orthogonal features for the area in each cardinal direction about the agent for a total of 8 features, with a ninth feature special to the region ahead of the agent. The components of each area are agent density and the net flow of agents in that area, with the area directly in front of the agent detecting the presence or lack of obstacles. **Agent density** is a rough approximation of overall crowding in the cardinal directions and includes obstacles. **Net flow** is the average velocity direction of agents in a particular area. This helps determine whether or not the general crowd is moving with or against the agent, which requires different care for such things as collision avoidance.

A data-driven approach relies on the quality and coverage of its training samples. Real-world data is often used as a source because humans empirically solve any presented steering challenges and we wish to create virtual representations of humans. However, we cannot completely control the steering scenarios or know all the variables in the decision-making process of the people observed. To enforce artificial limitations on the scenarios would impact the integrity of the data through the influences of the observer effect. Second, we have no way of knowing *a priori* whether the data set collected has adequate sample coverage for the situations the agents will need to handle. The problem of this potential incompleteness is compounded by the overhead—or impracticality—of collecting additional data. For these reasons, our pipeline uses synthetic data from which we can be conveniently gather additional samples and know all the influences in advance.

3.3 Oracle Algorithm

Our oracle algorithm is based on a memory-bounded A* planner with a discrete footstep action space similar to the action space in [Singh et al. 2011]. We choose a footstep action space because our machine learning can use classifiers instead of being constrained to regression. When the oracle is run on the generated scenarios, each agent uses the memory-bounded A* planner to calculate the optimal path from its current location to the goal. The bound on the memory is raised if a path is not found, as a last resort Iterative Deepening A* (IDA*) is used. The oracle planner's overall algorithm is given in Algorithm 1, and the heuristic used is in Equation 1 and is based on the distance to the goal and average expected energy cost to reach that goal.

Algorithm 1: Oracle Planner

```
Data: Start, goal, low memory bound, max memory
      bound, memory increment size.
Result: The path from start to goal.
for i ← memMin to memMax do
    path ← BoundAStar (start, goal, i)
    if path.size = 0 then
        i ← i + memBlock
    else
        return path
// Could not find path with BoundAStar
path ← IDAStar (start, goal)
return path
```

$$h(\mathbf{p}, \mathbf{g}) = \frac{\|\mathbf{p} - \mathbf{g}\| \cdot \text{energy}_{\text{avg}}}{\text{stride}_{\text{avg}}} \quad (1)$$

Each agent has full knowledge only of the obstacles and agents within the horizon of its field of view. Since other agents may enter or leave this field of view, each agent must monitor its path for new collisions and invoke the planner again if such a problem is found. We chose this limitation on the oracle because of the radius of the feature spaces used to sample the data, and the human-factors nature of the feature space designs.

The simulations using the oracle are recorded for later extraction of training samples. As the oracle does not use any feature spaces, the same oracle recordings can be used to extract data with different feature spaces, allowing for future exploration of such possibilities. We extract a state-action pair $\langle \mathbf{f},\ a \rangle$ where $\mathbf{f}$ is a vector from feature space $\mathbf{F}$ and a is the parameters of the agent's current step, and use it as a sample for training.

3.4 Decision Trees

Avoiding the requirement that the learned policy be a monolithic, universal solution has several key benefits. First, the policies can be simpler and thus executed faster at runtime. Second, we avoid the catastrophically high dimensionality common to such approaches, which are held back by all the factors that can influence every potential action. Finally, we do not need to relearn the entire system to assimilate new data. By using one model to select more specialized models, new data requires only the specialized model it belongs to be relearned. Even the creation of a new context only requires the top-level model be recomputed while the other models are still valid and will not be harmed by potentially contradictory data.

Each of our policies consists of two boosted decision trees; one for each foot. We use a Windows port of the GPL release of the C5.0 decision tree system (`http://www.rulequest.com`). We chose ten trees as the amount of boosting empirically based on cross-validation. In total 2500 scenarios were sampled from each context and each scenario was generated with respect to a central agent, which provided a variable number of steps per scenario. These steps then became the situations representative of the context

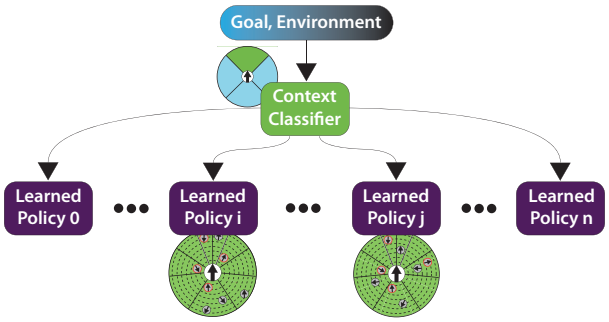

Figure 3: *The multilevel decision trees used by our models. At runtime the agent gives the model information about its current goal and environment in local-space. This data is used to calculate* **f** *for each model used. First the context classifier informs the agent of its current context, and the corresponding policy is used to determine the next footstep.*

Algorithm 2: Agent Decision at Runtime

Data: The environment with respect to the agent.
Result: The next footstep action.

```
1 fStar ← ObserveEnvironment ()
2 contextID ← ContextClassifier (fStar)
3 f ← ObserveLocalSpace (contextID)
4 action ← Classifier (f,contextID)
5 if action.confidence ≤ threshold then
6     action ← StopInPlace
7 return action.step
```

for the specialized classifier. A context classification sample was only generated for the first five steps of each recording due to the total number of scenarios that were sampled, all of which supplied data to the context classifier.

3.5 Steering At Runtime

At runtime the agent generates feature vectors corresponding to both the context classifier's feature space and the corresponding specialized model's feature space and receives parameters used to derive its next footstep. These parameters include a relative offset and rotational angle to the next step's location, while specifics such as stride length are calculated on the fly based on the agent's inherent characteristics. This step is validated and if found to be unfit, a default "emergency action" takes place, wherein the agent immediately stops. This allows the agent to try again after a short cool-down period. This safety net was implemented to account for the worst-case where a returned action is outside of the parameters permitted by the agents' walking such as two steps in a row from the same foot or too wide a turn. The models cannot be expected to be 100% accurate, which is the source of these potential errors. Pseudocode for the agents' runtime is listed in Algorithm 2.

To account for the imprecision in identifying steering contexts, we use a confidence threshold. This rating is roughly defined as the number of correct classifications made by the leaf nodes divided by the total number of classifications made by the same node, making it a static quantity once the tree is learned. If the confidence threshold is not met by the classification the agent stops with the ability to resume as conditions change. This confidence value is not a direct reflection on the technique itself, but is instead heavily affected by

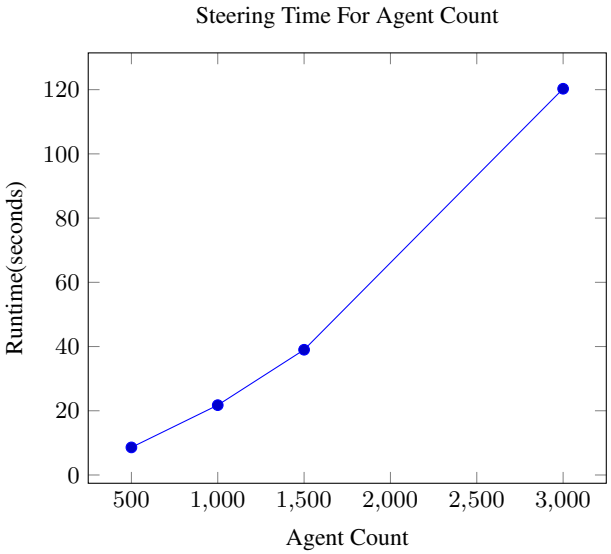

Figure 4: *Total time taken for computing the steps of a simulation 1,200 frames long for varying numbers of agents with randomly generated obstacles and an overall small area. Overhead was mostly incurred from a naïve implementation of agent density measurement which is* $O\left(n^2\right)$ *where* n *is the number of agents.*

pruning the decision trees to yield a more general model.

Note in Algorithm 2 there is no explicit collision detection or avoidance. In our system, runtime collision detection and avoidance is handled implicitly through the training data itself. This is different from other techniques such as [Lerner et al. 2007] where training samples are used but thorough handling of collisions is required. Empirically, even without explicit collision detection the amount of per-capita collisions is relatively low.

4 Results

We generated approximately 2,500 samples for each of our initial 24 contexts. The oracle algorithm required two weeks of continuous computation to return paths for all of the sample scenarios. Those scenarios which were shown to require IDA* were culled in the interest of time. All results were generated on a desktop with 16.0GB of RAM, Intel Core i7 860 CPU at 2.8GHz, and an NVIDIA GeForce GTX 680.

4.1 Classifier Accuracy

Figure 5 plots the error rate for the classifiers used in our experiments. Simulations were run using models trained on amounts of data ranging from 100 to 2000 scenarios per context. A separate validation set of 200 scenarios per context were kept back to calculate the error rate of the resulting trees.

Error rates were high but did decrease as data size increased, showing improvement in generalization and not simply noise. Additionally, the average number of steps used for each context was approximately 12, which sets random guess accuracy at 8%, which we clearly overcame. Furthermore, random guess accuracy of 24 contexts is 4% which we also surpassed. The error rate seen in the context classifier is likely a result of how the training data was generated in a noisy manner, for instance some overlap in density between a high density scenario and a medium density scenario ex-

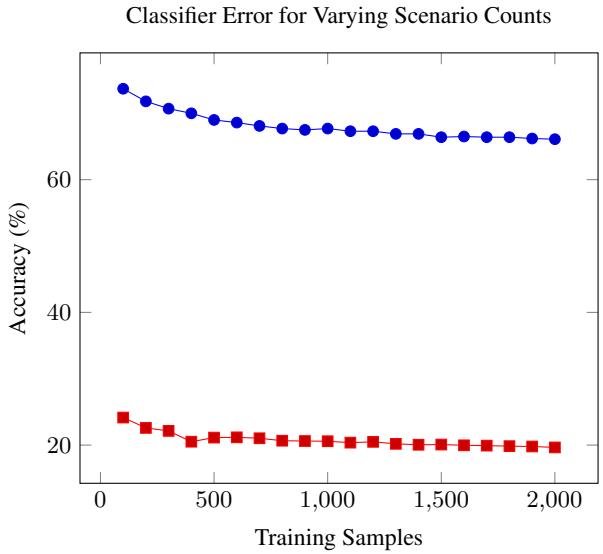

Figure 5: *Classifier error rates for both context classifier (blue) and an average over the specialized classifiers (red). While the context classifier has a high error rate, a 96% error rate is random chance given the large number of classes to choose from.*

ists. A large burden is also placed on the decision trees to distinguish the `Chaos` context from other contexts but this by its nature adds a lot of noise and has no structure, making it difficult to define hyperplanes to separate such scenarios.

4.2 Runtime

Our initial instantiation of a context-sensitive pipeline is much faster at runtime than the oracle. As seen in Table 1, all contexts experienced speedup, especially significant for the most challenging scenarios involving obstacles. The `Chaos` context, both with and without obstacles, was the most challenging for the oracle and resulted in skewed performance data due to the number of scenarios which were culled. Our method showed an extremely constant amount of time across the different contexts owing to its dynamic model-swapping.

To test the robustness of our collection of models, we created a large-scale simulation consisting of randomly generated obstacles, agents, and goals, as seen in Figure 1. We measured the time to generate the paths for varying numbers of agents to simulate 1,200 frames, with the results given in Figure 4. All tests were run using a single-threaded implementation and realtime framerates were experienced at 1,500 agents and interactive framerates of about 10FPS were experienced with as many as 3,000 agents.

5 Conclusion and Future Work

In this paper, we have defined steering contexts, a new view on the space of possible scenarios an agent may encounter as it steers through its virtual world. We have also proposed a pipeline for constructing a steering algorithm that is both context-sensitive and scalable to circumstance. Through the use of a multiplicity of models fit to steering contexts, machine learned can be combined for better, and more structured, coverage of the space of possible scenarios than would otherwise be possible by a single-model approach generalizing to all situations. We used an oracle algorithm to get high quality, on-demand training data which can be used for new contexts without the overhead or uncertainty of real-world data. This training data was then broken into contexts based on intuition and policies fit for each context using machine learning.

Our technique has shown a massive increase in efficiency as realtime simulation was achieved with far higher population counts than the oracle algorithm could handle. Furthermore, training on this data resulted in relatively small numbers of collisions, many of them minor. This system would be ideal for populating a space with "extras" which are not the focus of an end-user's attention. In such a background application, the infrequent collisions would be more likely to go unnoticed.

Future Work.

The decision tree models used to prototype our pipeline are too restricting if the chosen action is incorrect. A naïve Bayesian approach would allow a better "next best" progression of footstep selection rather than the current all-or-nothing approach. Multiple algorithms can coexist throughout the collection of policies allowing each context to be fit as needed for better overall accuracy. Furthermore the contexts themselves could be defined from a collection of data using unsupervised clustering, further removing the human element from the problem.

Currently we decide the next step an agent should take and deciding multiple steps would require an exponential increase in the size of the action space if done naïvely. However, we postulate that analysis of step sequences would reveal that not all step combinations need to be learned, drastically decreasing the overhead. Maneuvers such as overtaking other pedestrians or rounding corners could then be encapsulated, rather than depending on each step in the process being decided accurately. Even with 90% decision accuracy, a 5-step sequence has a probability of being correct of only about 60%. Furthermore, rather than such a short horizon of a single step, this machine learning approach could tackle navigation instead and plot a waypoint, while a fast but reactive algorithm such as RVO moves the agent through the waypoints.

Finally, this data-driven approach is highly amenable to parallelization, and the results in this paper only for single-threaded performance. Exploring scalability with increased thread count would further show the strength of our technique.

Acknowledgements

The research reported in this document/presentation was performed at the University of Pennsylvania in connection with Contract Number W911NF-10-2-0016 with the U.S. Army Research Laboratory. The views and conclusions contained in this document/presentation are those of the authors and should not be interpreted as presenting the official policies or position, either expressed or implied, of the U.S. Army Research Laboratory, or the U.S. Government unless so designated by other authorized documents. Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation heron.

References

COURTY, N., AND CORPETTI, T. 2007. Data-driven animation of crowds. In *Int. Conf. on Computer vision/computer graphics collaboration techniques*, MIRAGE, 377–388.

GUY, S. J., CHHUGANI, J., KIM, C., SATISH, N., LIN, M., MANOCHA, D., AND DUBEY, P. 2009. Clearpath: highly paral-

Context	0	1	2	3	4	5	6	7	8	9	10	11
Oracle	0.73	13.84	5.11	15.53	12.35	9.26	1.68	67.27	101.56	19.90	14.71	1.20
Models	0.07	0.07	0.06	0.06	0.06	0.06	0.07	0.06	0.06	0.06	0.06	0.06
Context	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**
Oracle	123.95	785.0	1945.24	365.25	565.43	574.52	916.30	462.53	3384.10	577.54	396.79	64.78
Models	0.15	0.15	0.17	0.18	0.17	0.18	0.13	0.13	0.15	0.16	0.17	0.16

Table 1: *Total time for step planning for all contexts in seconds to calculate steps over short scenarios. The first 12 are contexts without obstacles and are based on oncoming and cross traffic patterns with varying levels of agent density. Contexts 12 and above have obstacles and agent patterns matching the upper 12.*

lel collision avoidance for multi-agent simulation. In *ACM SIGGRAPH/Eurographics SCA*, 177–187.

HELBING, D., AND MOLNAR, P. 1995. Social force model for pedestrian dynamics. *Physical review E 51*, 5, 4282.

KAPADIA, M., SINGH, S., HEWLETT, W., AND FALOUTSOS, P. 2009. Egocentric affordance fields in pedestrian steering. In *ACM SIGGRAPH I3D*, 215–223.

KAPADIA, M., WANG, M., SINGH, S., REINMAN, G., AND FALOUTSOS, P. 2011. Scenario space: Characterizing coverage, quality, and failure of steering algorithms. In *2011 ACM SIGGRAPH/Eurographics SCA*, 53–62.

LEE, K. H., CHOI, M. G., HONG, Q., AND LEE, J. 2007. Group behavior from video: a data-driven approach to crowd simulation. In *ACM SIGGRAPH/Eurographics SCA*, vol. 1, 109–118.

LERNER, A., CHRYSANTHOU, Y., AND LISCHINSKI, D. 2007. Crowds by Example. *CGF 26*, 3 (Sept.), 655–664.

LERNER, A., CHRYSANTHOU, Y., SHAMIR, A., AND COHEN-OR, D. 2010. Context-Dependent Crowd Evaluation. *CGF 29*, 7, 2197–2206.

METOYER, R. A., AND HODGINS, J. K. 2003. Reactive pedestrian path following from examples. In *CASA*, vol. 20, 149–156.

NARAIN, R., GOLAS, A., CURTIS, S., AND LIN, M. C. 2009. Aggregate dynamics for dense crowd simulation. *ACM TOG 28*, 5 (Dec.), 1.

ONDŘEJ, J., PETTRÉ, J., OLIVIER, A. H., AND DONIKIAN, S. 2010. A synthetic-vision based steering approach for crowd simulation. *ACM TOG 29*, 4, 123.

PELECHANO, N., ALLBECK, J. M., AND BADLER, N. I. 2007. Controlling individual agents in high-density crowd simulation. In *ACM SIGGRAPH/Eurographics SCA*, vol. 1 of *SCA*, 108.

PELECHANO, N., ALLBECK, J., AND BADLER, N. 2008. *Virtual Crowds: Methods, Simulation, and Control*. Morgan & Claypool.

REYNOLDS, C. W. 1987. Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH 21*, 4 (Aug.), 25–34.

REYNOLDS, C. W. 1999. Steering behaviors for autonomous characters. In *GDC*, Citeseer, vol. 1999, 763–782.

SCHNEIDER, J., GARATLY, D., SRINIVASAN, M., GUY, S. J., CURTIS, S., CUTCHIN, S., MANOCHA, D., LIN, M. C., AND ROCKWOOD, A. 2011. Towards a Digital Makkah—Using Immersive 3D Environments to Train and Prepare Pilgrims. In *DMACH*, 1–16.

SHAO, W., AND TERZOPOULOS, D. 2007. Autonomous pedestrians. *Graphical Models 69*, 5-6 (Sept.), 246–274.

SINGH, S., KAPADIA, M., REINMAN, G., AND FALOUTSOS, P. 2011. Footstep navigation for dynamic crowds. *CAVW 22*, 2-3, 151–158.

THALMANN, D., AND MUSSE, S. R. 2007. *Crowd Simulation*. Springer.

TORRENS, P., LI, X., AND GRIFFIN, W. A. 2011. Building Agent-Based Walking Models by Machine-Learning on Diverse Databases of Space-Time Trajectory Samples. *Transactions in GIS 15* (July), 67–94.

TREUILLE, A., COOPER, S., AND POPOVIĆ, Z. 2006. Continuum crowds. *ACM TOG 25*, 3 (July), 1160.

VAN DEN BERG, J., AND MANOCHA, D. 2008. Reciprocal Velocity Obstacles for real-time multi-agent navigation. *ICRA* (May), 1928–1935.