9-2015

Generating a Multipliciy of Policies for Agent Steering in Crowd Simulation

Cory D Boatright
University of Pennsylvania

Mubbasir Kapadia
University of Pennsylvania

Jennie Shapira
University of Pennsylvania

Norman I. Badler
University of Pennsylvania, badler@seas.upenn.edu

Recommended Citation

Boatright, C., Kapadia, M., Shapira, J., & Badler, N. I. (2015). Generating a Multipliciy of Policies for Agent Steering in Crowd Simulation. *Computer Animation and Virtual Worlds, 26* (5), 483-494. http://dx.doi.org/10.1002/cav.1572

Generating a Multipliciy of Policies for Agent Steering in Crowd Simulation

Abstract
Pedestrian steering algorithms range from completely procedural to entirely data-driven, but the former grossly generalize across possible human behaviors and suffer computationally, whereas the latter are limited by the burden of ever-increasing data samples. Our approach seeks the balanced middle ground by deriving a collection of machine-learned policies based on the behavior of a procedural steering algorithm through the decomposition of the space of possible steering scenarios into steering contexts. The resulting algorithm scales well in the number of contexts, the use of new data sets to create new policies, and in the number of controlled agents as the policies become a simple evaluation of the rules asserted by the machine-learning process. We also explore the use of synthetic data from an "oracle algorithm" that serves as an as-needed source of samples, which can be stochastically polled for effective coverage. We observe that our approach produces pedestrian steering similar to that of the oracle steering algorithm, but with a significant performance boost. Runtime was reduced from hours under the oracle algorithm with 10 agents to on the order of 10 frames per second (FPS) with 3000 agents. We also analyze the nature of collisions in such a framework with no explicit collision avoidance.

Keywords
machine learning, synthetic data, crowd simulation, steering

Disciplines
Computer Sciences | Engineering | Graphics and Human Computer Interfaces

Generating a Multiplicity of Policies for Agent Steering in Crowd Simulation

Cory D. Boatright* Mubbasir Kapadia Jennie M. Shapira

Norman I. Badler

Grove City College

100 Campus Drive

Grove City, PA 16127

Tel. +1(724)264-4622 Fax. +1(724)450-1550

email: cdboatright@gcc.edu

Work done while all four authors were at the University of Pennsylvania

Abstract

Pedestrian steering algorithms range from completely procedural to entirely data-driven, but the former grossly generalize across possible human behaviors and suffer computationally while the latter are limited by the burden of ever-increasing data samples. Our approach seeks the balanced middle ground by deriving a collection of machine-learned policies based on the behavior of a procedural steering algorithm through the decomposition of the space of possible steering scenarios into steering contexts. The resulting algorithm scales well in the number of contexts, the use of new data sets to create new policies, and in the number of controlled agents as the policies become a simple evaluation of the rules asserted by the machine-learning process. We also explore the use of synthetic data from an “oracle algorithm” that serves as an as-needed source of samples which can be stochastically polled for effective coverage. We observe that our approach produces pedestrian steering similar to that of the oracle steering algorithm, but with a significant performance boost. Runtime was reduced from hours under the oracle algorithm with 10 agents to on the order of 10 FPS with 3,000 agents. We also analyze the nature of collisions in such a framework with no explicit collision avoidance.

Keywords: machine learning, synthetic data, crowd simulation, steering

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. Assumptions such as the reciprocity of steering algorithm are not sound in the presence of human input, suggesting the need to rethink how to handle this diversity. The problem of predicting *a priori* the possible situations an agent will encounter is rapidly becoming intractable as users are given more freedom in their virtual worlds, and thus we need algorithms that are scalable not only in agent count, but circumstance as well.

Data-driven steering algorithms are a natural fit for expanding virtual pedestrians' capability to handle new problems, but current approaches use a single policy as a "one size fits all" approach and are data-bound in their ability to handle general steering. A leading problem for machine learning in crowd steering is the feature space itself, especially for a single-policy system. For behavior as complex and diverse as human steering, increasing amounts of samples can lead to contradictory data which can require an increase in features to try to accommodate the new factor causing the difference. Furthermore, the source of data is often observations of the real world which poses logistical challenges for gathering samples from an inability to control the observed environment. This ultimately leads to a lack of scenario coverage in the training data itself. Poor semantic understanding of particular steering choices also inhibits fully robust usage of this data, which can manifest as what appear to be poor steering decisions.

In this paper we define the concept of steering contexts, which are collections of situations selected for their qualitative similarity. By identifying such contexts, we divide the problem space, which limits the necessary scope of the data-driven solution and avoids the single-model problem described above. We propose a pipeline which leverages these contexts through the use of a collection of machine-learned models trained on synthetic data from a space-time planner. This development pipeline is visualized in Figure 2.

This paper makes the following contributions:

- We introduce and use the concept of steering contexts to separate data for easier machine learning and allow for scalability of circumstance as well as help mitigate contradictory training samples.

- 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.

- We analyze the performance of an “implicit” approach to collision avoidance with a data-driven technique.

Related Work

Following seminal work [1] 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 [2, 3].

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 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 [1, 4] 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 [5, 6].

Centralized techniques rely on a broad conformity amongst the population for best efficiency as seen the fluid-like approach of [7]. This is an acceptable premise for group-dynamic simulations as used in the study of crowd flows in religious pilgrimages [8] and emergency evacuations [9], 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 [10, 11] determine their next action based on which velocities may avoid a collision with another agent. This similar to the approach used by [12] which uses a synthetic sense of vision to determine information about other agents' trajectories and adjust accordingly. Agents have also used affordance fields [13] to try to find safe passage to a goal. A cognitive system was used in the seminal work [14] 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 [15] was proposed that switched between other steering algorithms to best suit an agent's needs.

Machine learning has been used [16] 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 [17] 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 [18] 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 [19] 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 [15]. 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 [20] 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 [21] as our contexts are egocentric, not scenario-wide. Another data-driven method seen in [22] 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 [19] 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.

Steering Contexts

A **scenario** in scenario space [20] $\mathbb{S}$ is the global configuration of obstacles, agents, and their goals in a virtual environment. The high dimensionality of scenario space makes it inherently intractable to exhaustively cache, so a more general model of steering behavior is needed. Each agent in a frame of simulation encounters its own **situation** S from its perspective. Situations which are similar, based on some feature space $\mathbf{F}^*$, are grouped together to form a **context**, C. The similarity-based grouping is performed to give a high-level perspective on the current situation, and to properly steer the agents require a policy for each context. While these could be handwritten rulesets, identifying contexts and creating policies for each one quickly becomes work-bound. We use machine learning to offset this burden and automatically generate models to serve as a policy for each context.

We can now give a formal definition of context space and of contexts themselves. For a given feature space $\mathbf{F}^*$, context space $\mathbb{C}$ is a projection of $\mathbb{S}$ onto the coordinate system of $\mathbf{F}^*$ and may consist of many overlapping contexts each with boundaries defined by various values of the features. An individual context $C_i \subseteq \mathbb{C}$ is defined in Equation 1 with respect to the success of steering policy i in handling situations. A policy is successful if it can produce a valid action from action space $\mathbb{A}$ for the situation, which is one where a collision does not occur and the overall scenario does not deadlock. A scenario then can be considered a sequence of situations and actions with some transition function $\delta\left(S, a\right)$.

$$C_i = \left\{ S \in \mathbb{C} \mid \exists a : \left\langle \mathbf{f}, a \right\rangle, \mathbf{f} \in \mathbf{F}^i, a \in \mathbb{A}, S \neq \delta\left(S, a\right) \right\} \qquad (1)$$

A situation is guaranteed membership in at least one context because in the worst case, it could have a special-case policy defined for it. This lets us redefine scenario space as $\mathbb{S} = \bigcup_i C_i$. This redefinition yields interesting insight into the pursuit of generalized steering. While it would be convenient to know if a set of policies exists that provide optimal behavior for all scenarios in $\mathbb{S}$, this requires the corresponding contexts partition $\mathbb{S}$ based on the “best” context and is thus a direct application of the exact cover problem. Furthermore, it is intractable to know if a set of contexts is sufficient to cover $\mathbb{S}$ as it is an example of the set cover problem. Both of these are known to be NP-Complete [23]. This important fact necessitates approximating contexts rather than strictly defining them.

These contexts express how different situations require different policies and improve scenario space by better characterizing regions of success and failure. By approximating contexts, we can also identify a more constrained domain for data-driven techniques. This allows for a more modular and thus extensible approach to building a model for general steering. Examples of contexts we defined by intuition are provided in Figure 3 with a full index in Table 2.

Initial Implementation

We now explain our pipeline 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.

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.

Our feature space for learning specialized policies are based on a circular neighborhood about the agent with discretized wedges that track the nearest agent or obstacle in that region. Our feature spaces can be seen in Figure 4 and are in part inspired by the state spaces of [18, 19]. In particular, the context classifier's state space is built of two values for each of the four regions and an additional value denoting the presence of obstacles in front of the agent for a 9-dimensional vector. The specialized feature space is a 29-dimensional vector broken down into three values for each slice: the distance, speed, and orientation of the nearest entity. The distance to the goal and its orientation are the final two values.

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.

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 [15]. 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 2 and is based on the distance to the goal and average expected energy cost to reach that goal.

$$h\left(\mathbf{p},\mathbf{g}\right) = \frac{\|\mathbf{p}-\mathbf{g}\| \cdot \mathrm{energy}_{\mathrm{avg}}}{\mathrm{stride}_{\mathrm{avg}}} \tag{2}$$

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.

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.

The pipeline proposed by this paper is agnostic to the specific learning algorithms used at the different levels of the hierarchy, and different algorithms can even coexist on different levels of the hierarchy if particular contexts are better handled by different models. We have chosen to use two levels of boosted decision trees [24] for our instantiation of the pipeline based on the similar problem domain of [25] that showed success for learning different policies that both classified different types of soccer behavior and could be used to decide the actual action itself.

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 (`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 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.

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.

As shown earlier in the paper, it is NP-Complete to know if our contexts cover all possible scenarios. Furthermore, decision trees are susceptible to high variance depending on the dataset we generate through our stochastic sampling. This causes uncertainty in the decisions our agents will make. We account for this uncertainty through the use of a confidence threshold defined by the C5.0 algorithm. 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 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 [17] where training samples are used but thorough handling of collisions is required. As will be shown, the training data itself is sufficient to prevent many significant collisions from occurring with dense scenarios experiencing relatively few collisions per capita when all factors are considered.

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.

Classifier Accuracy

Figure 7 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 exists. 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.

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 6. 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.

Collisions

Recall our virtual agents navigate without an explicit collision avoidance stage to their navigation. Generally, the agents do not collide on the basis that their training samples contain no collisions, and thus they inherently steer around one another. However, as the models are not 100% accurate, collisions are to be expected.

We have run several medium-scale scenarios that are beyond the type of scenarios used for training the models. These scenarios were as follows:

Hallway Two opposing groups of 100 agents cross a hallway.

Random 500 randomly placed agents with 696 randomly placed obstacles throughout the environment.

Urban 2500 randomly placed agents in an environment simulating an urban area with obstacles as city blocks.

These tests were run for varying numbers of training scenarios, from 100 to 2000 in increments of 100 and each test was run for 3600 frames. Afterwards, we tabulated the number of collisions and created the graphs in Figure 8. The collisions were recorded by severity. Type A collisions have occlusions in the range $(0\%, 10\%]$ at the worst point. These collisions could be registered due to the circular profile of the agents' bounding volume and thus may not be visible when the simulation is rendered. Type B collisions have occlusion in the range $(10\%, 35\%]$ and while more severe than before, could be alleviated with a better anthropomorphic model with torso-rotation. This type of collision is often dealt with in real pedestrians by turning the shoulders to more easily pass one another in cramped conditions. Type C collisions occlude on the range $(35\%, 75\%]$ and are major collisions which require more tuning to the algorithm to avoid. Type D collisions complete the possibilities at $(75\%, 100\%]$ and would most likely need a fully reactive collision avoidance system to prevent.

The results were counterintuitive at first. As training samples grew in quantity, so did collisions and even the severity of the collisions. We hypothesize two main factors behind this increase. First, the oracle algorithm is collision-free. Thus a sort of "event horizon" was established in the training data where no reaction to an agent occurs once the agent is too close to another. This means once two agents are too close, there is no force to push them apart, which explains the increased amount of more serious collisions compared to the more minor offenses.

The second factor is that with increased sample counts, the models better attempt the mimicry of the oracle algorithm's behavior. The oracle has the ability to steer agents together in a very tight, close-call manner. While this is good for the oracle and such nearby passing can be accommodated by it, as the training data increases in size and the agents steer more like the oracle, a misstep is more likely to cause a collision. In essence, more training data made the agents attempt to steer in a more precise manner, but the inherent inaccuracy of any machine learning algorithm simultaneously leads to higher risk. Thus a collision avoidance algorithm is necessary for a data-driven approach to steering.

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. These contexts provide new insight into the task of creating a general steering controller capable of handling anything it encounters. Unless the controller can be independently proven to be general and thus consist of a single context, the algorithm will shatter scenario space into subsets which must each be handled by a separate policy. This creates a coverage uncertainty that is by nature NP-Complete and to our knowledge no realtime algorithm is unaffected by this discovery.

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, we 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

[1] Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH*, 21(4):25–34, August 1987.

[2] Daniel Thalmann and Soraia Raupp Musse. *Crowd Simulation*. Springer, 2007.

[3] Nuria Pelechano, Jan Allbeck, and Norman Badler. *Virtual Crowds: Methods, Simulation, and Control*. Morgan & Claypool, 2008.

[4] C. W. Reynolds. Steering behaviors for autonomous characters. In *GDC*, volume 1999, pages 763–782. Citeseer, 1999.

[5] D. Helbing and P. Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282, 1995.

[6] N. Pelechano, J. M. Allbeck, and N. I. Badler. Controlling individual agents in high-density crowd simulation. In *ACM SIGGRAPH/Eurographics SCA*, volume 1 of *SCA*, page 108, 2007.

[7] Adrien Treuille, Seth Cooper, and Zoran Popović. Continuum crowds. *ACM TOG*, 25(3):1160, July 2006.

[8] Jens Schneider, Dina Garatly, Madhusudhanan Srinivasan, S. J. Guy, Sean Curtis, Steven Cutchin, Dinesh Manocha, M. C. Lin, and Alyn Rockwood. Towards a Dig-

ital Makkah—Using Immersive 3D Environments to Train and Prepare Pilgrims. In *DMACH*, pages 1–16, 2011.

[9] Rahul Narain, Abhinav Golas, Sean Curtis, and Ming C. Lin. Aggregate dynamics for dense crowd simulation. *ACM TOG*, 28(5):1, December 2009.

[10] Jur van den Berg and Dinesh Manocha. Reciprocal Velocity Obstacles for real-time multi-agent navigation. *ICRA*, pages 1928–1935, May 2008.

[11] Stephen J. Guy, Jatin Chhugani, Changkyu Kim, Nadathur Satish, Ming Lin, Dinesh Manocha, and Pradeep Dubey. Clearpath: highly parallel collision avoidance for multi-agent simulation. In *ACM SIGGRAPH/Eurographics SCA*, pages 177–187, 2009.

[12] J. Ondřej, J. Pettré, A. H. Olivier, and S. Donikian. A synthetic-vision based steering approach for crowd simulation. *ACM TOG*, 29(4):123, 2010.

[13] Mubbasir Kapadia, Shawn Singh, William Hewlett, and Petros Faloutsos. Egocentric affordance fields in pedestrian steering. In *ACM SIGGRAPH I3D*, pages 215–223, 2009.

[14] Wei Shao and Demetri Terzopoulos. Autonomous pedestrians. *Graphical Models*, 69(5-6):246–274, September 2007.

[15] Shawn Singh, Mubbasir Kapadia, Glenn Reinman, and Petros Faloutsos. Footstep navigation for dynamic crowds. *CAVW*, 22(2-3):151–158, 2011.

[16] Ronald A. Metoyer and Jessica K. Hodgins. Reactive pedestrian path following from examples. In *CASA*, volume 20, pages 149–156, November 2003.

[17] Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. Crowds by Example. *CGF*, 26(3):655–664, September 2007.

[18] K. H. Lee, M. G. Choi, Qyoun Hong, and Jehee Lee. Group behavior from video: a data-driven approach to crowd simulation. In *ACM SIGGRAPH/Eurographics SCA*, volume 1, pages 109–118, 2007.

[19] Paul Torrens, Xun Li, and William A. Griffin. Building Agent-Based Walking Models by Machine-Learning on Diverse Databases of Space-Time Trajectory Samples. *Transactions in GIS*, 15:67–94, July 2011.

[20] Mubbasir Kapadia, Matt Wang, Shawn Singh, Glenn Reinman, and Petros Faloutsos. Scenario space: Characterizing coverage, quality, and failure of steering algorithms. In *2011 ACM SIGGRAPH/Eurographics SCA*, pages 53–62, 2011.

[21] Alon Lerner, Yiorgos Chrysanthou, Ariel Shamir, and Daniel Cohen-Or. Context-Dependent Crowd Evaluation. *CGF*, 29(7):2197–2206, 2010.

[22] Nicolas Courty and Thomas Corpetti. Data-driven animation of crowds. In *Int. Conf. on Computer vision/computer graphics collaboration techniques*, MIRAGE, pages 377–388, 2007.

[23] R. M. Karp. Reducibility among combinatorial problems. *50 Years of Integer Programming 1958-2008*, 2010.

[24] J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[25] Ruck Thawonmas, Junichiro Hirayama, and Fumiaki Takeda. RoboCup Agent Learning from Observations with Hierarchical Multiple Decision Trees. *PRIMA*, 2002.

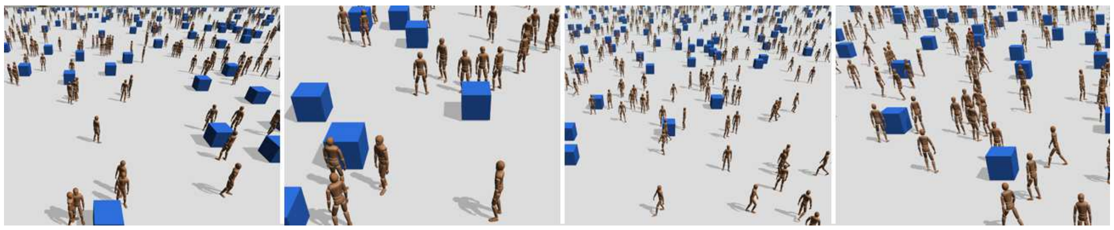


Figure 1: Multiple views of a 3,000 agent simulation with high quality rendering.

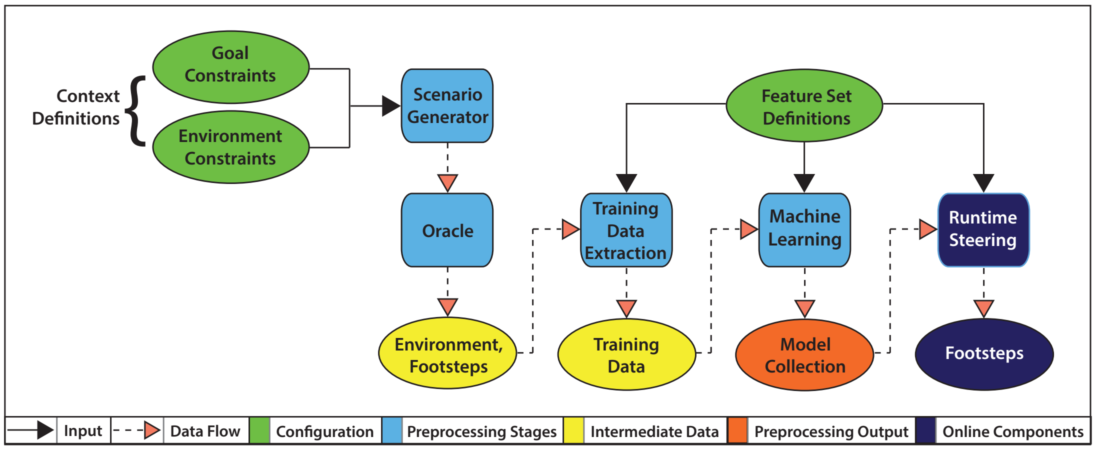


Figure 2: Our pipeline for using steering contexts to develop a machine-learned model for use at runtime. The majority of the pipeline is offline processing. A collection of models is trained on data extracted from an oracle algorithm's solution to steering situations, which are stochastically generated. 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.

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.

Context ID	**Obstacles**	**North**		**South**		**East**		**West**	
		Flow	**Density**	**Flow**	**Density**	**Flow**	**Density**	**Flow**	**Density**
0	Yes	Neutral	Light	Neutral	Light	Neutral	Light	Neutral	Light
1	Yes	Towards	Light	Neutral	Light	Neutral	Light	Neutral	Light
2	Yes	Towards	Medium	Neutral	Light	Neutral	Light	Neutral	Light
3	Yes	Towards	High	Neutral	Light	Neutral	Light	Neutral	Light
4	Yes	Towards	Medium	Towards	Medium	Neutral	Light	Neutral	Light
5	Yes	Towards	Light	Towards	High	Neutral	Light	Neutral	Light
6	Yes	Neutral	Light	Neutral	Light	Towards—Away	Light	Away—Towards	Light
7	Yes	Neutral	Light	Neutral	Light	Towards—Away	Medium	Away—Towards	Medium
8	Yes	Neutral	Light	Neutral	Light	Away—Towards	High	Away—Towards	High
9	Yes	Neutral	Light	Towards	Medium	Away—Towards	Medium	Away—Towards	Medium
10	Yes	Neutral	Light	Towards	High	Away—Towards	Light	Away—Towards	Light
11	Yes	Towards	High	Towards	High	Towards	High	Towards	High
12	No	Neutral	Light	Neutral	Light	Neutral	Light	Neutral	Light
13	No	Towards	Light	Neutral	Light	Neutral	Light	Neutral	Light
14	No	Towards	Medium	Neutral	Light	Neutral	Light	Neutral	Light
15	No	Towards	High	Neutral	Light	Neutral	Light	Neutral	Light
16	No	Towards	Medium	Towards	Medium	Neutral	Light	Neutral	Light
17	No	Towards	Light	Towards	High	Neutral	Light	Neutral	Light
18	No	Neutral	Light	Neutral	Light	Towards—Away	Light	Away—Towards	Light
19	No	Neutral	Light	Neutral	Light	Towards—Away	Medium	Away—Towards	Medium
20	No	Neutral	Light	Neutral	Light	Away—Towards	High	Away—Towards	High
21	No	Neutral	Light	Towards	Medium	Away—Towards	Medium	Away—Towards	Medium
22	No	Neutral	Light	Towards	High	Away—Towards	Light	Away—Towards	Light
23	No	Towards	High	Towards	High	Towards	High	Towards	High

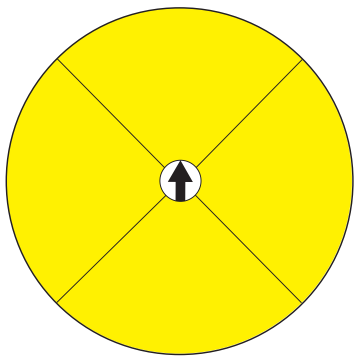
Table 2: Parameters which define the 24 contexts we use to prototype our pipeline.

Algorithm 1: Oracle Planner

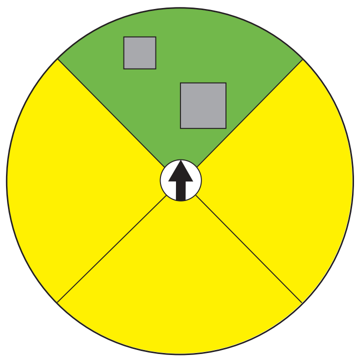
```
Data: Start, goal, low memory bound, max memory bound, memory
      increment size.
Result: The path from start to goal.
1 for i ← memMin to memMax do
2     path ← BoundAStar (start, goal, i)
3     if path.size = 0 then
4         i ← i + memBlock
5     else
6         return path
  // Could not find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

Algorithm 2: Agent Decision at Runtime

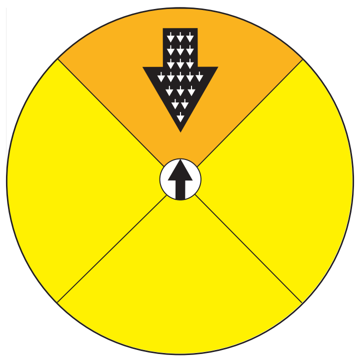
```
Data: The environment with respect to the agent.
Result: The next footstep action.
1 fStar ← ObserveEnvironment ()
2 contextID ← ContextClassifier (fStar)
3 f ← ObserveLocalSpace (contextID)
4 action ← Classifier (f,contextID)
5 if action.confidence ≤ threshold then
6     action ← StopInPlace
7 return action.step
```



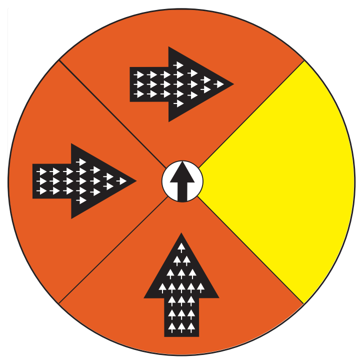
(a) Clear View



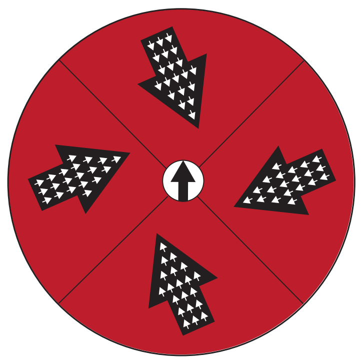
(b) Obstacles Ahead



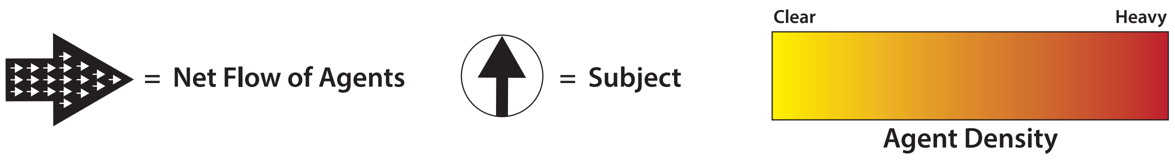
(c) Light Oncoming



(d) Groups Crossing

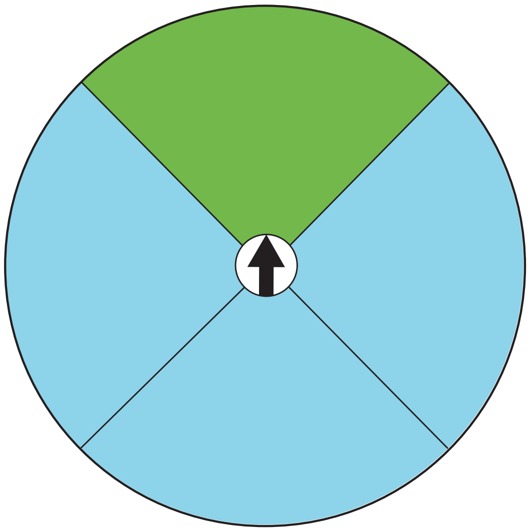


(e) Chaos



(f) Symbols Key

Figure 3: Examples from our set of contexts. Net flow is represented by the arrow in each region, density of the region is depicted by darker shades of red, and obstacles are gray boxes. Each of these contexts was stochastically generated with overlap in the permissible values for density. Chaos was generated randomly without regard to any structure as seen in the other contexts. We used a total of 24 contexts.


(a) $\mathbf{F}^*$

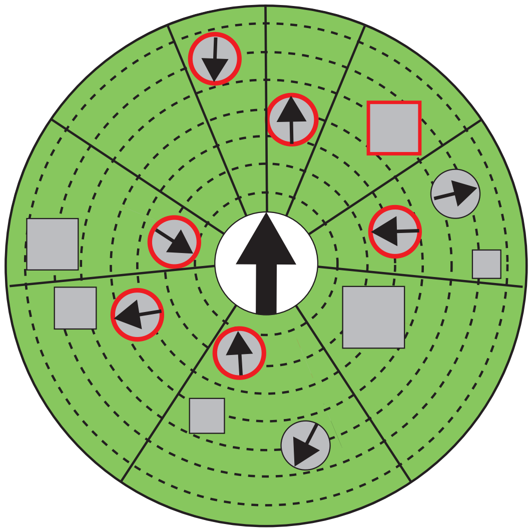

(b) $\mathbf{F}$

Figure 4: The feature sets used in our pipeline, where other agents are circles and static obstacles are depicted as boxes. $\mathbf{F}^*$ is used by the context classifier to dynamically choose the best model based on high-level features, while $\mathbf{F}$ is used to choose the agent's next step based on the local neighborhood.

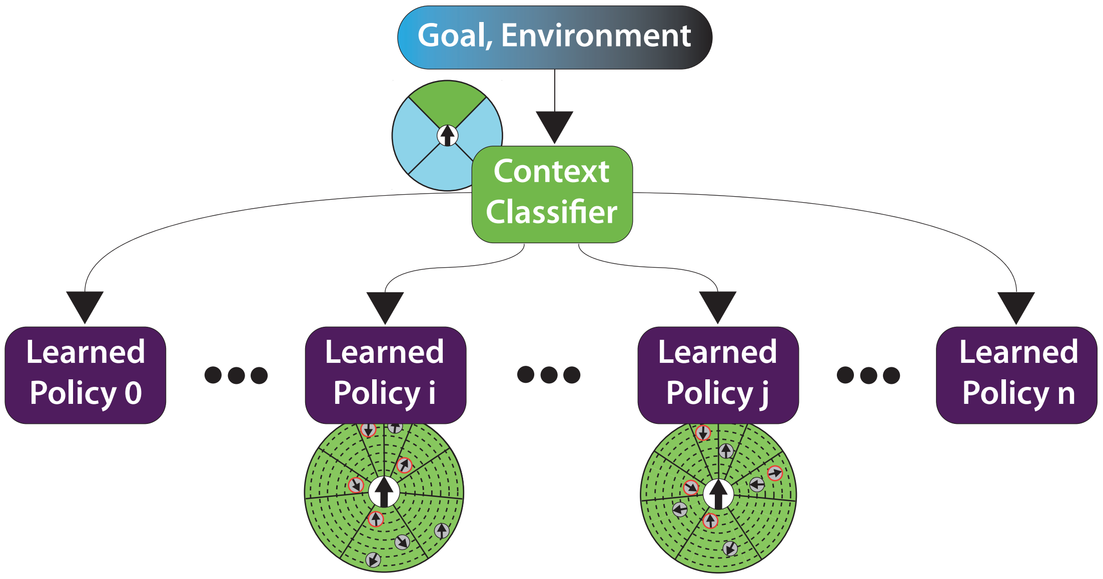


Figure 5: The multilevel decision trees used by our models. At runtime the agent gives the model information about its current goal and environment in local-space. This data is used to calculate **f** for each model used. First the context classifier informs the agent of its current context, and the corresponding policy is used to determine the next footstep.

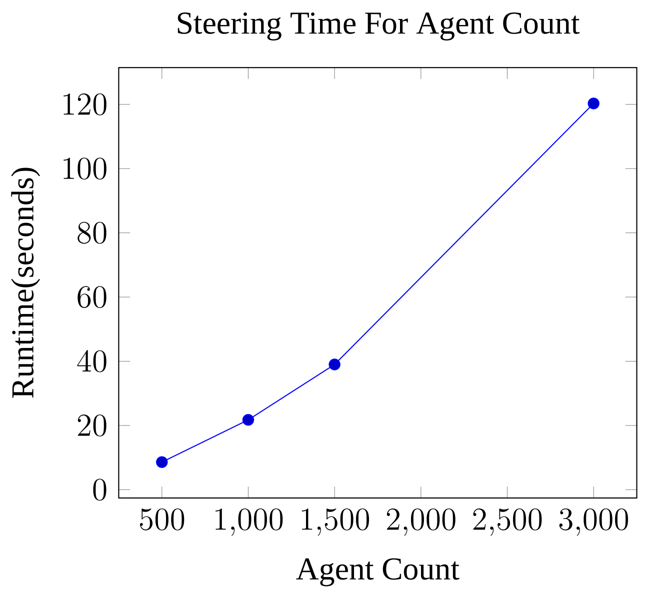


Figure 6: Total time taken for computing the steps of a simulation 1,200 frames long for varying numbers of agents with randomly generated obstacles and an overall small area. Overhead was mostly incurred from a naïve implementation of agent density measurement which is $O\left(n^2\right)$ where n is the number of agents.

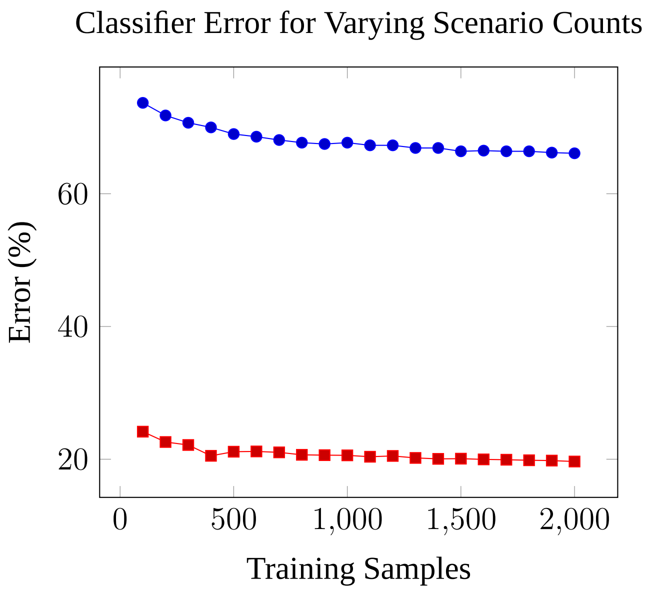
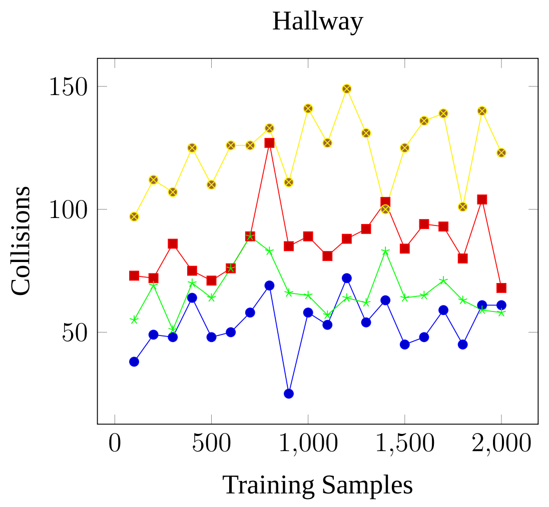
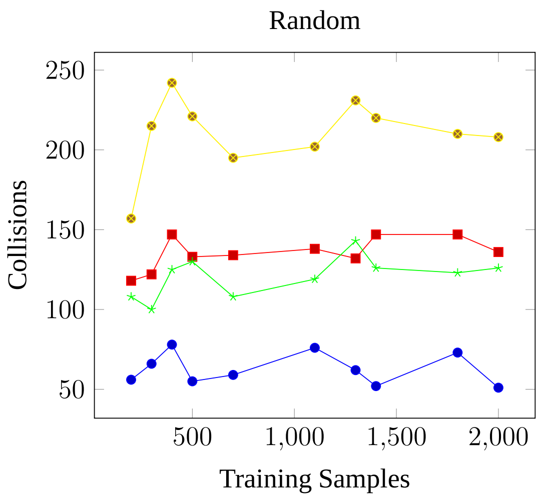


Figure 7: Classifier error rates for both context classifier (blue) and an average over the specialized classifiers (red). While the context classifier has a high error rate, a 96% error rate is random chance given the large number of classes to choose from.





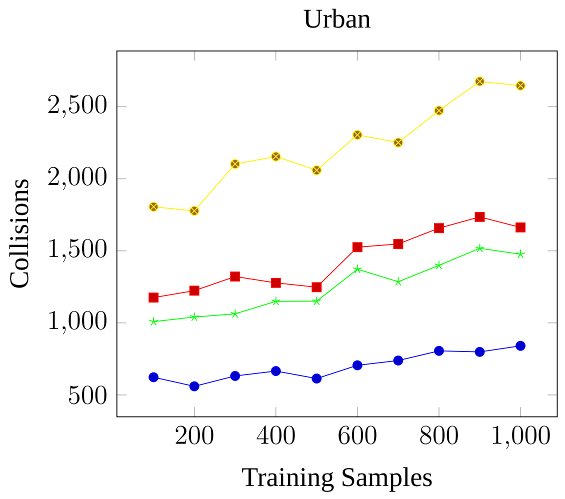


Figure 8: Counts for collisions in 3 minute simulations in different test scenarios. Type A collisions are blue, Type B collisions are red, Type C are yellow, and Type D are green. Once collisions occurred, there was little pressure for agents to move apart as the training data was collision-free, thus no samples existed for overlapping agents. Note that while high, per capita an agent in each of these simulations is only likely to encounter around 1-3 collisions with approximately one third of them minor in nature in spite of the lack of any explicit collision avoidance.