A Modular Framework for Adaptive Agent-Based Steering

Shawn Singh* Mubbasir Kapadia† Billy Hewlett‡ Glenn Reinman§ Petros Faloutsos¶

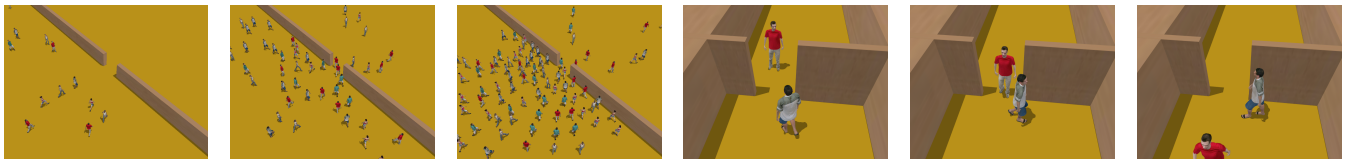

Figure 1: Snapshots of a bottleneck and a doorway scenario, showing progress from left to right. The hybrid approach efficiently handles large crowds and challenging space-time scenarios.

Abstract

Next-generation steering algorithms will need to support thousands of believable individual agents, capable of steering in very challenging situations with low-latency reactions. In this paper we propose a steering framework that offers three key contributions: (a) It integrates several models of steering into a single steering decision, (b) it employs a novel space-time planning approach to allow agents to steer during complex local interactions, and (c) it varies the frequency of update of each component (phase) of the framework to drastically improve performance. We demonstrate the versatility and robustness of our framework using a large number of test cases. We also show that the frequency of updates for each phase of the framework can be "decimated" by a surprisingly large amount before resulting steering behaviors degrade. This technique achieves more than a 5× performance improvement, allowing the use of better, more costly algorithms for robust steering, while supporting thousands of agents with low-latency reactions in real-time.

Keywords: Autonomous agents, steering behaviors, pedestrian simulation

1 Introduction

Steering is the layer of intelligence between cognition and locomotion that an agent uses to navigate through a virtual environment. Despite the many positive advancements to steering in research literature, the quality of steering in commercial applications is still unsatisfactory. There are several reasons for this disparity:

- The majority of steering research focuses on crowd behaviors, while many applications, such as games, need to have equally realistic local intelligence for *individual* agents up close.
- Applications tend to have rigid real-time constraints, allowing only 5-10% of computing resources to be dedicated to steering and artificial intelligence (AI) [Rabin 2005]. If AI is updated at 20 Hz, this leaves only 5 ms per frame to update all agents. Better algorithms are generally more costly, and it is not obvious whether some costly algorithms can still be used in real time.
- There are many ways to model real pedestrian steering, each with its own advantages and disadvantages. A robust steering framework needs to integrate these techniques, capitalizing on the advantages of each technique when appropriate. This is not a trivial issue, because each model of steering overlaps and potentially conflicts with other aspects.

In this paper, we explore the use of multiple steering techniques in combination, in order to achieve more versatile and robust behaviors in a single framework. We view the problem as a mapping from an agent's current situation to the steering technique (or techniques) that should be used. There are many possible ways to do this which are discussed in Section 3.1. We combine space-time predictions, reactive rules, and crowd-based steering decisions that are based on existing literature, and we also incorporate space-time planning that allows agents to form localized small-scale plans to steer through complicated situations. Space-time path planning can solve challenging local steering situations that other approaches cannot, but to our knowledge it has not been used in any previous real-time steering framework.

Our modular approach also creates a unique opportunity: various components of steering do not need to execute at the same rate of update. Existing real-world and research applications update steering decisions at slower frequencies than rendering and physics, but to our knowledge, no previous approach has explored running phases *within* steering at separate update frequencies. In fact, some steering algorithms are monolithic by nature and cannot be modularized. By choosing the right phases and running these phases at separate rates, not only can agents react more quickly in low-level steering, but agents can also afford to use more expensive high-level steering methods that produce better quality results. We call this technique *phase decimation*.

We demonstrate the effectiveness of these techniques with a broad variety of test cases and provide an in-depth analysis of how far we can push phase decimation. Results show that phases can be decimated quite drastically, with very little degradation in the quality of steering behaviors, achieving more than 5× performance improvement compared to common update rates. We show that a hybrid framework that integrates crowd-based, reactive, predictive, and space-time steering techniques can support thousands of agents in real-time using phase decimation.

*e-mail: shawnsin@cs.ucla.edu
†e-mail: mubbasir@cs.ucla.edu
‡e-mail: billyh@cs.ucla.edu
§e-mail: reinman@cs.ucla.edu
¶e-mail: pfal@cs.ucla.edu

I3D 2011, San Francisco, CA, February 18 – 20, 2011.
© 2011 ACM 978-1-4503-0565-5/11/0002 $10.00

Steering Technique	Local interactions	Crowd behaviors	Deadlock resolution	Performance	Role
Simple forces	poor	very good	poor	very fast	locomotion and dense crowds
Reactive steering	average	very good	poor	fast	low-latency reactions
Space-time predictions	very good	average	poor	average	believable local interactions
Following a space-time plan	very good	intractable	very good	very slow	resolves complex local interactions
Following a planned path	N/A	N/A	N/A	average	used when no interactions

Table 1: Taxonomy of different steering techniques used in our hybrid approach. Each technique has advantages that compensate for other technique's limitations.

2 Related Work

Particle and dynamics approaches. The seminal work of Reynolds [Reynolds 1987] modeled agents as particles that interact using dynamics. Most particle flow and dynamics based techniques, e.g., [Brogan and Hodgins 1997; Goldenstein et al. 2001; Treuille et al. 2006; Helbing et al. 2000], are well suited for modeling large crowds. They usually achieve natural behaviors at a macro-scale, relying on emergent properties of their interactions, but they are not intended to simulate local agent interactions.

Rule-based systems. Rule-based approaches, *e.g.* [Reynolds 1999; Loscos et al. 2003; Lamarche and Donikian 2004; Metoyer and Hodgins 2004; Lee et al. 2007; Lerner et al. 2007; Pelechano et al. 2007; Rudomín et al. 2005; Shao and Terzopoulos 2005; Sud et al. 2007; van den Berg et al. 2008; Boulic 2008; Paris et al. 2009], consist of (1) interpreting the agent's environment, and (2) rules or heuristics to react to the interpreted information. Most rule-based systems perform a large amount of redundant computation per update, for example, updating an agent's visual field that is not likely to have changed since the last frame. In general, rule-based systems are very difficult to tune for more than approximately four agents interacting simultaneously, but for fewer interacting agents, rule-based behaviors can be highly believable.

Space-time prediction and planning. For rule-based systems, one promising technique is to use the space-time domain [Feurtey 2000; Paris et al. 2007; Kapadia et al. 2009]. So far, the space-time domain has only been used in crowd steering to make predictions. Full space-time *planning* is traditionally found in the context of robotics and motion planning, e.g. [Lau and Kuffner 2005; Shapiro et al. 2007], and to our knowledge it has not been applied to the goal of real-time robust crowd simulation.

Comparison to our work. While most previous works focus on only a few aspects of human steering, we aim to model a more diverse set of steering abilities. For performance, instead of trying to approximate or simplify the models of steering, we use an adaptive technique that reduces computation. Furthermore, most other works demonstrate only a small, focused set of scenarios. In this paper, we show results of more rigorous testing using a wide variety of test cases, benchmarking, and user studies.

A common technique in games is to decouple path planning from agent updates. A similar technique has been used in recent steering research, decoupling steering from the rest of the system. Paris et. al. [2007] perform their steering algorithm updates at 1-2 Hz. Treuille et al. [2006] perform updates at approximately 3-5 Hz. Our phase decimation technique goes one step further, and decouples phases *within* steering. This allows agents to react with low latency (high frequency) while updating state-machine, predictions, perceptions, and several types of planning only as frequently as necessary. This keeps amortized costs very low despite the high cost of phases such as perception.

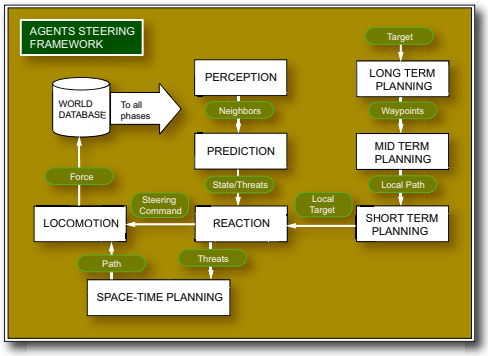

Figure 2: Overview of our steering framework: the reaction phase decides between predictive collision avoidance, reactive steering, or steering to follow the planned path. If necessary, space-time planning is invoked to briefly control locomotion directly.

3 Our Framework

Our framework has the following phases: planning, perception, prediction, reaction, space-time planning, and locomotion. Table 1 shows a taxonomy of the steering methods we combine, and Figure 2 shows the phases of our framework. Each approach has advantages that compensate for the other approaches' limitations. Steering with simple forces is very fast and works well in dense crowds where the agents have only a few steering options. Reactions serve as a "catch-all", reacting with low-latency to situations that are not successfully avoided by prediction or planning. However, by itself, reactive steering looks awkward, waiting until the last minute to steer around other agents, and without predictions, the agent can easily steer itself into deadlock. Predictive steering makes the agents aware of threats in advance, and naturally steers through local interactions. In crowded environments, where too many threats are predicted or the environment is too dense, the agent relies more on reactive and force-based behaviors anyway. Finally, even with a good set of predictive and reactive rules, some deadlocks are unavoidable without space-time planning, such as a narrow doorway where one agent must backtrack or sidestep away from its goal in order to let all agents progress. Space-time planning, however, is too computationally expensive to apply to all agents all the time. Essentially, all of these steering techniques are necessary and rely on each other to produce versatile behaviors.

Long-term and mid-term planning phases. The agent plans a path to its goal using the standard A-star algorithm [Hart et al. July 1968]. The graph used by A-star is a rectangular grid where each node is connected to its eight neighbors. The grid-based graph can result in costly A-star searches, but this choice avoids the need for manually creating A-star graphs, and the amortized cost of path planning still remains low.

Short-term planning phase. Given the planned path, the agent

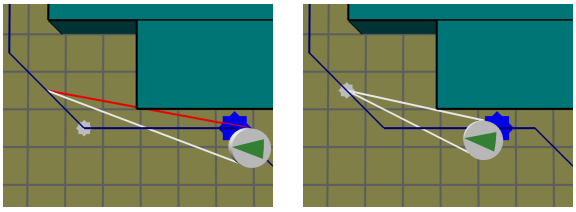

Figure 3: Short-term planning. The local target (white star) is chosen as the furthest point such that all path nodes between the agent's closest path node (blue star) and the local target have line-of-sight to the agent.

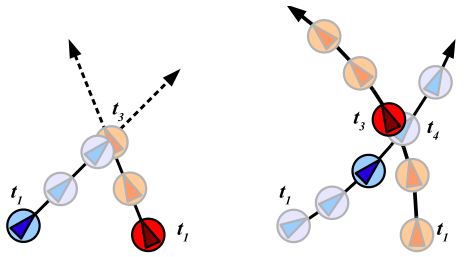

Figure 4: Space-time prediction. Left: agents predict a collision, knowing their trajectories will overlap at the same time, t_3. Right: agents steer to avoid the collision. Note that space-time prediction correctly avoids a false prediction between the blue agent at t_4 and the red agent at t_3, because they reach that point at different times.

chooses a point along the path to steer towards. This local target is chosen as the furthest point along the path such that all path nodes between the agent and the local target are visible to the agent. This criterion smooths the agent's path while enforcing that the agent follows the path correctly around obstacles.

Perception phase. To perform natural predictions and reactions, it is important to model what the agent actually sees. We model an agent's visual field as a 10 meter hemisphere centered around the agent's forward facing direction. The agent collects a list of objects using a range query in the spatial database, as described above. Furthermore, objects that do not have line-of-sight are not added to the list of objects the agent sees.

Prediction phase. The agent predicts possible collisions, only with agents in its visual field, using a linear space-time predictor based on [Paris et al. 2007]. Given an agent's position P, velocity V, and radius r, our linear predictor estimates the agent's position at time t as $P+t \cdot V$. A collision between agent a and b would occur at time t if the distance between their predicted positions becomes less than the sum of their radii:

$$\|(P_a + t \cdot V_a) - (P_b + t \cdot V_b)\| < r_a + r_b. \quad (1)$$

Solving this expression for time t results in a quadratic equation. The agents collide only if there are two real roots, and these two roots represent the exact time interval of the expected collision. Predicted threats are handled similar to [Shao and Terzopoulos 2005], where the agent that will reach the collision first speeds up and turns slightly outward, and the agent that will reach the collision later slows down and turns slightly inward (Figure 4).

Reaction phase. Our reaction phase implements both reactive steering and crowd-based steering. The agent traces three forward-facing rays, 1 meter to the front of the agent, and 0.1 meters to the side. If these rays intersect anything, the agent may need to react. When reacting, the agent takes into account the relative location

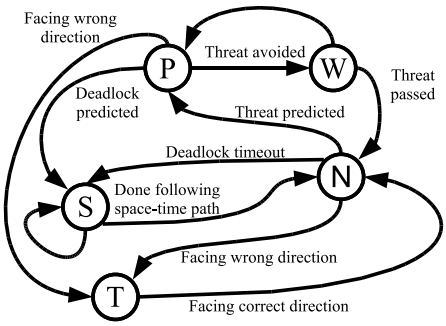

Figure 5: State machine used to integrate space-time planning, prediction, reaction, and state-dependent steering behaviors. The agent steers normally in state N, proactively avoids threats in state P, waits for avoided threats to pass in state W, follows a space-time path in state S, and re-orients itself towards the target in state T. These behaviors may be overridden by the reaction phase.

and orientation of the new obstructions. This results in a very long list of rules that account for all possible configurations: there can be up to three perceived obstructions (one per forward-facing ray), each obstruction may be an agent or an obstacle, and any obstructing agents can be classified as oncoming, crossing, or facing the same direction. For efficiency, the rules are implemented using a hierarchy of conditions instead of checking each rule one by one. This way, identifying any rule requires only a logarithmic number of conditional checks rather than linear. Once the specific rule has been identified, it is usually clear how the agent should steer, and so we do not enumerate this long list of rules here.

Locomotion phase. The locomotion phase receives an abstract steering command that tells an agent to turn, accelerate, aim for target speed, and/or scoot left or right. This allows orientation to be treated separately from movement. The command is converted into a force and angular velocity that moves the agent's position and orientation using simple forward Euler integration.

3.1 Integrating multiple steering techniques

The problem of using multiple steering techniques can be stated as follows: Given all possible situations that an agent can encounter, how can we map each situation to a decision about which steering technique(s) to use? Here we discuss four possible solutions:

- Multiple steering decisions can be blended using a weighted sum. The weights for this sum may even change depending on the agent's situation. When we tried this technique, however, the steering decisions would commonly conflict and cancel out. For example, an agent may reactively decide to turn right, but predict a threat further away that should be avoided by turning left.
- The mapping from an agent's situation to a single steering technique could be a classifier, trained with machine learning techniques. Unfortunately, the training would require significant manual effort, and the classifier would need to be retrained every time the steering techniques are adjusted.
- The mapping from an agent's situation to a single steering technique can be manually approximated by a set of heuristics. Our implementation of this approach is described below.
- The agent can blindly compute steering decisions from all techniques before choosing one, and then choose the decision that maximizes some fitness criteria. This approach has the

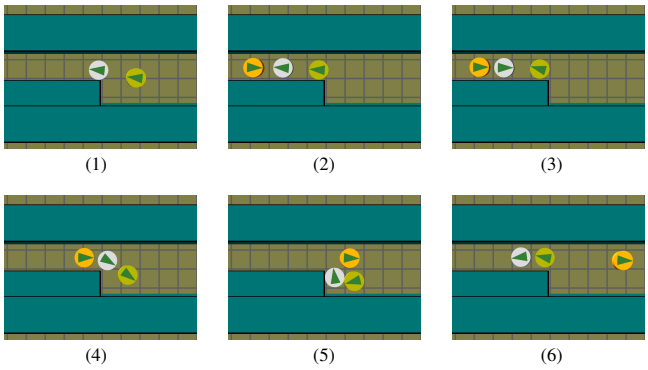

Figure 6: A challenging scenario that most steering frameworks cannot solve. Two agents encounter a surprise agent in a narrow pathway. To solve this properly, the agents must understand how to backtrack and get out of the way; this is the only way they can eventually progress through the pathway.

potential to be the most robust for the price of adding significant computational cost. Exploring this approach is left for future work.

We chose to use a set of rules that approximately classify the agent's current situation. A state machine is used to track the agent's situation for predictive, space-time, and normal steering circumstances, and this can be overridden for reactive or crowd-based steering. The states and their transitions are shown in Figure 5, and the rules are described as follows, in order starting with the highest priority rule:

1. If the agent is following a space-time path (state S) and encounters an un-predicted agent with its reactive feelers $\Rightarrow$ use reactive/crowd steering.
2. If the agent is following a space-time path that is already planned $\Rightarrow$ use space-time-plan steering (i.e. closely follow the space-time path).
3. If the agent is facing the wrong direction from its planned path (state T) $\Rightarrow$ the agent should briefly slow down and quickly reorient itself towards the short-term local target (special case).
4. If not following a space-time path, and if the agent encounters un-predicted agents with its reactive feelers $\Rightarrow$ use reactive/crowd steering.
5. If there are any predicted threats (state P and W) $\Rightarrow$ use predictive steering.
6. If there are no threats (state N), $\Rightarrow$ use normal steering (i.e. follow the planned path)

Results. The results shown in the supplementary video indicate that this approach works effectively, seamlessly transitioning between normal steering, predictive decisions, reactive decisions, crowd behaviors, and space-time planning.

4 Space-time Planning

Figure 6 and several examples in the supplementary video show challenging local interactions that are rarely addressed in crowd steering literature. Real human pedestrians usually have a good understanding of exactly how they would proceed through an environment over the next few seconds, even in the presence of other moving pedestrians. In other words, real pedestrians tend to form very localized space-time plans. Here we propose a space-time planning technique that can be seamlessly integrated into a steering framework.

The space-time graph. To compute the space-time path, we use A-star over a dynamically generated graph. The graph represents an on-the-fly discretization of the space-time domain starting at the agent's current position. Each node of the graph represents a particular point in space-time. Children nodes represent the possible spatial positions that the agent can reach at the next point in time. These children nodes essentially define the locomotion style of the space-time path, so for realistic behaviors we found it was necessary to give the agent many options. The agent can be stopped, can move in 8 different directions at slow speed, and 16 directions at normal speed. This results in at most 25 children nodes, but there may be fewer nodes if the corresponding point in space-time is obstructed by any predicted threats, which implies the agent cannot steer that way. Nodes without obstructions are added to the graph in a lazy manner, only when A-star tries to expand those nodes. Because A-star finds optimal paths, agents will side-step, stop, and backtrack only when necessary.

The A-star cost function. It was surprisingly tricky to find a good cost function for this particular type of graph. We first tried to use Euclidian distance in the space-time domain. This did not work, however, because time and spatial units are on different scales, and the agent kept searching for ways to minimize path length in cases where it should have minimized the time to reach its goal. As a result, the agent often chose a space-time path that remained stationary for too long while continuing to be an obstacle for other agents. We then tried to use only time as the cost function, so that the agent minimizes the time it takes to reach its goal. In this case, the agent had erratic behavior because there was no constraint on how the agent should steer while waiting for a path to clear. Our solution is a novel cost function that represents both time and spatial distance concurrently, so that when A-star is deciding which node to expand next, it first chooses nodes that would minimize the time to goal, and if the two time estimates were equal, then it chooses the node with the shorter spatial distance estimate. This is implemented by placing time (in units of frames) in the integer portion of the real-valued cost, and placing the spatial distance (scaling it by a constant factor so it is always less than 1) in the fractional portion.

Conditions to invoke space-time planning. An agent invokes space-time planning if a deadlock is predicted or perceived. Deadlocks are predicted when there are too many predicted threats in close proximity to each other. Deadlocks are perceived if the agent has been stuck at zero speed due to reactive steering. Additionally, if an agent must react to another agent performing space-time planning, that agent also computes a space-time plan. To avoid a catastrophic ripple-effect of costly space-time plans, space-time planning is not invoked when agents have more than some user-defined number of agents in their visual field. We currently set this value to 3. This may seem arbitrary, but it is arguably a good model of human-like steering as well, since real pedestrians are more likely to use basic crowd steering behaviors in crowded environments instead of space-time planning.

Performance. Dynamically generating the graph and running A-star with a branching factor of 25 are both extremely costly, typically taking hundreds of microseconds. However, in our implementation, space-time planning is only invoked on-demand when a deadlock is perceived, which occurs infrequently. Furthermore, once the space-time path is computed, the agent follows the computed path with trivial force computations. For these reasons, the space-time planning phase does not adversely affect our real-time performance.

		Benchmark		User Study			
Test case name	A versus B decimation (Hz) vs. (Hz)	Score for A	Score for B	A is more intelligent (%)	Equal intelligence (%)	B is more intelligent (%)	p-value
Frogger	4 vs 20	291	274	16.1	67.8	16.1	0.00
	20 vs 2	274	297	45.2	6.45	48.4	0.11
	2 vs 1.5	297	365	51.6	16.1	25.8	0.02
3-way confusion	4 vs 20	391	350	19.4	51.6	29.0	0.048
	20 vs 1.17	350	403	41.9	25.8	32.3	0.4
4-way confusion	20 vs 6.7	369	332	9.68	35.5	54.8	0.02
	20 vs 1.5	369	380	45.2	29.0	25.8	0.22
	1.17 vs 1	343	417	48.4	3.23	48.4	0.11
Double squeeze	20 vs 2	442	411	3.23	80.7	16.1	0.00
	20 vs 1	442	434	0.0	67.7	32.3	0.00
	1 vs 0	434	529	67.7	25.8	6.55	0.00
Oncoming groups	20 vs 1.3	856	868	51.6	45.2	3.23	0.048
	1.3 vs 1.17	868	880	80.7	9.68	9.68	0.00
Squeeze	20 vs 2	432	474	16.1	77.4	6.45	0.00
	20 vs 1	432	454	16.1	77.4	6.45	0.00
	1 vs 0	454	812	96.8	3.23	0.0	0.00

Table 2: Comparative user study, also used to validate the benchmark scoring process. Lower benchmark scores are better; the benchmark correctly captures user opinions. Each row compares two different cases of decimating the prediction phase; 0 Hz means that predictions were disabled.

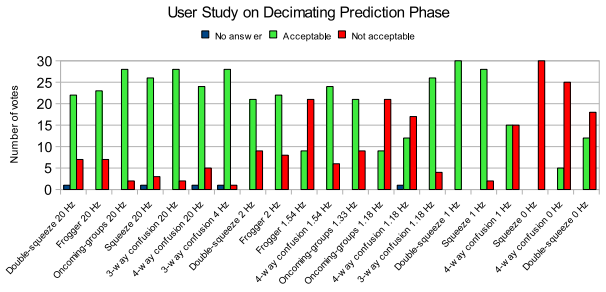

Figure 7: Acceptability user study. Test cases are shown here in order of increasing decimation of prediction phase. Results are statistically significant if 20 or more users had the same preference.

5 Phase Decimation

In this section we turn our focus to performance. We study how much we can reduce the frequency of updates for each phase independently – a technique we call *phase decimation* – while still achieving acceptable behavior. We perform user studies and use a benchmark suite provided by [Singh et al. 2009] to show that we can decimate a surprisingly large amount for all phases; even conservative decimation has a significant performance improvement.

5.1 Evaluation Method

All forms of path planning in our framework – long-term, mid-term, and space-time path planning – execute only when needed, on-demand. Therefore, we only need to study the effects of decimating short-term planning, perception, and predictions. To isolate the effects of decimation, space-time planning is disabled for these tests. In all cases, we kept reaction phase running at 20 Hz. Where possible, we discuss decimation in terms of frequency (Hz), but with high decimation these frequencies are small fractions, and in those cases it is clearer to discuss the number of frames skipped (Updating once per n frames is $20/n$ Hz).

We conducted two user studies, a comparative study and an acceptability study, using the same participants in the same session. All participants were willing volunteers, with no background in steering behaviors. For the comparative study, participants were shown 16 comparisons, in random order. For each comparison, we showed two examples of agents steering the same situation with varying levels of decimation of the predictive phase. We showed example A, then example B, and then examples A and B side by side, and we asked them to "compare the intelligence of A and B." The participants were not told the purpose of the study, did not know anything about the concept of phase decimation, and did not know how the examples were constructed. For the acceptability study, participants were shown 21 video examples in random order and asked to evaluate whether it was "acceptable" or not. Prior to the study we defined "acceptable behavior" to the participants, orally and in writing, as a "behavior that could be seen in a real world situation with normal pedestrians that are paying attention to where they are going," and we made sure they understood what was requested of them. Participants were given enough time to answer between examples, and they were given the opportunity to re-watch the videos if they desired. In previous user studies, we tried to use a Likert scale instead of binary choices, but this consistently resulted in statistically insignificant results because the participants have varied criteria for evaluation. We found that asking binary questions resulted in consistently statistically significant results, even if the results were unfavorable.

A complete description of the test cases used can be found in [Singh et al. 2009]. The benchmark technique we used was slightly different than their published paper. The scoring method uses a weighted sum of four metrics collected for each agent: number of collisions, total time, total kinetic energy, and a new metric, total acceleration. The benchmark score is computed as a weighted sum of metrics, with a weight of 150.0 for the number of collisions, and a weight of 1.0 for the other three metrics. Note that benchmark scores cannot be compared across test cases. Scores can only be compared for the same test case. We also used the test cases provided by the benchmark suite, which are described in their paper. Table 2 shows that the benchmark scores matched the user study, and so we felt comfortable enough using the benchmark score as a method of evaluating phase decimation in more detail.

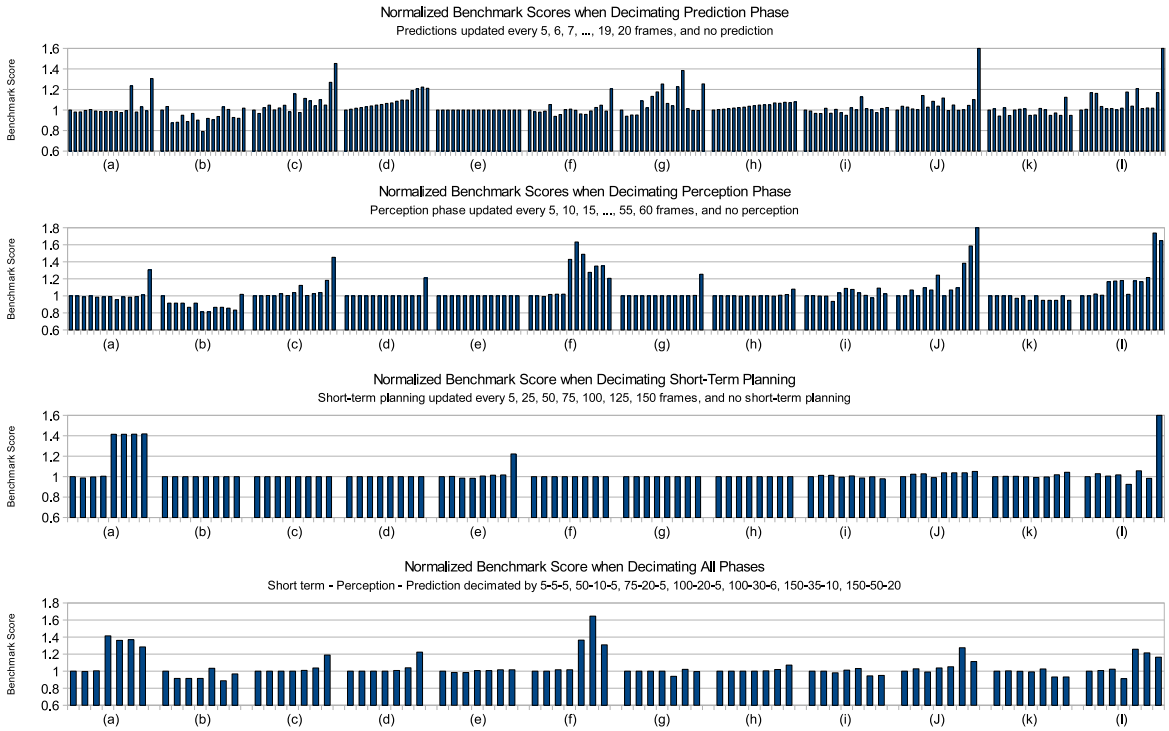

Figure 8: Benchmark results for static decimation. Each cluster of results is one test case with various amounts of decimation, i.e., various frequencies of updating phases. The test cases shown are: (a) 3-squeeze, (b) 3-way confusion, (c) 4-way confusion, (d) crossing, (e) doorway-one-way, (f) double-squeeze, (g) frogger, (h) oncoming, (i) oncoming-groups, (j) squeeze, (k) surprise, and (l) wall-squeeze. Lower benchmark scores are better, and scores are normalized with respect to the 4 Hz reference cases.

5.2 Static Phase Decimation

Results for the user studies are given in Table 2 and Figure 7. These results indicate that viewers preferred behaviors with moderate amounts of decimating prediction (2-10 Hz), felt ambivalent about behaviors with heavy decimation of predictions (below 2 Hz), and disliked behaviors that skipped predictions entirely.

The results from test cases of the benchmark suite fell into three categories: (1) large-scale simulations, where the benchmark tool could only count the number of collisions, (2) results that used purely reactive steering, and therefore were not affected by decimation, and (3) interesting test cases for decimation. Test cases from category (2) all worked subjectively well and had no collisions.

Figure 8 shows the benchmark results for (3) the interesting decimation cases. For each test case, benchmark scores were normalized against the same test case using an un-decimated update frequency of 4 Hz. Lower benchmark scores are better. These results are consistent with the user studies indicating that moderate amounts of decimation are acceptable. At the same time, when any phase (such as prediction) was completely disabled, the benchmark scores were notably worse.

Decimating the prediction phase had the most direct effect on steering behaviors, and we conclude that predictions can be safely decimated to 3-4 Hz (skipping every 5-6 frames). Our framework was more tolerant to decimating perceptions, which only indirectly affect steering behaviors by delaying when predictions can be made. We conclude that perceptions can be decimated as far as 0.6 Hz (skipping every 30 frames). Short-term planning could be decimated even further, and can even be used on-demand without breaking the steering behavior. However, even a very slow rate of update did smooth the path better than on-demand, so we conclude that short-term planning can be decimated up to 0.20-0.25 Hz (skipping every 75-100 frames). Finally, we benchmarked several combinations of decimating predictions, perceptions, and short-term planning simultaneously. The benchmark scores indicate that we can decimate each phase simultaneously at approximately the same rates that we could decimate them independently.

5.3 Adaptive Phase Decimation

One issue with our analysis in the previous section is that the amount of decimation is clearly dependent on the scenario the agent is encountering. Thus, there can be cases where decimating is not the right thing to do, and other cases that could be decimated more drastically. Furthermore, with increasing decimation, the reaction phase becomes more a significant cost per update. It would be nice to have a way of decimating the reaction phase as well. Decimating reactions can be risky, and we would need a way to revert back to full 20 Hz reactions if necessary.

To address this, we added support in our framework for *adaptive phase decimation* – throttling the frequencies of updating each phase at run-time, based on a heuristic assessment of the scenario. Each agent manages its own adaptive frequencies independently of all other agents. Because we identify cases where each phase needs to be updated more frequently or less frequently, we can do even more decimation than described in the previous section. The heuristics we use are listed as follows.

- **Short-term planning.** The key factor that determines how often to re-compute the short-term plan is the agent's distance to the local goal. If the local goal is close to the agent, short-

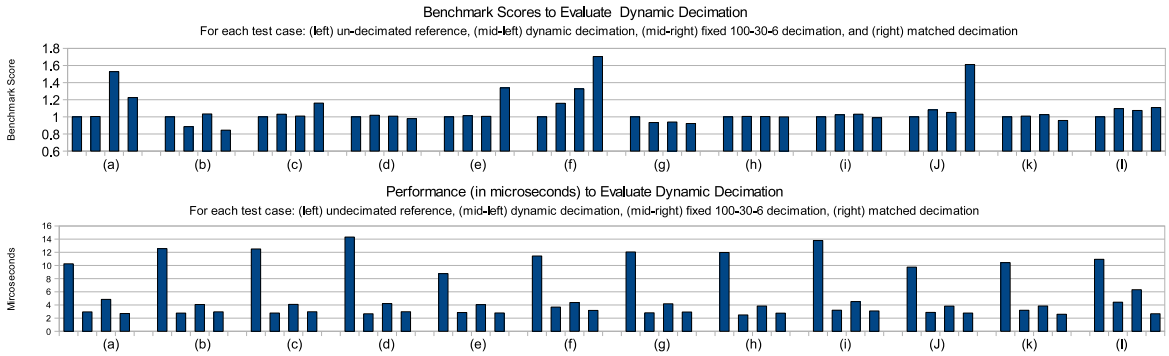

Figure 9: Comparison between (1) a 4 Hz reference, (2) adaptive decimation, (3) static decimation, and (4) matched decimation. In this round of tests, space-time planning was enabled. Top: normalized benchmark scores to evaluate the quality of results; lower scores are better. Bottom: performance results, measured in microseconds as the average cost of a single agent update. The test cases shown are: (a) 3-squeeze, (b) 3-way confusion, (c) 4-way confusion, (d) crossing, (e) doorway-one-way, (f) double-squeeze, (g) frogger, (h) oncoming, (i) oncoming-groups, (j) squeeze, (k) surprise, and (l) wall-squeeze. Adaptive decimation scores better than static and matched decimation on the benchmark, and consistently performs better.

term planning should be more frequent in order to have the desired smoothing effect on the mid-term path. On the other hand, if the local goal is far away, the agent's path will already be smooth, until it gets closer. Thus, our heuristic is to choose a frequency so that short-term planning is executed conservatively frequently at 0.25 Hz (every 75 frames) when the agent reaches within 5 meters of its local goal, and less frequently as the distance to the local goal increases.

- **Perception.** The frequency of updating perception is dependent on how likely the visual field will change. We choose the frequency of perception based on the speed of the agent: at normal walking speed of 1.3 m/s, we update perceptions at 0.67 Hz (every 30 frames). When the agent travels slower than 1.1 m/s, we slow perceptions to 0.3 Hz (every 65 frames). The reason we can do this is that an agent slows down when it is responding to a threat, which may take several seconds to pass anyway. We can also observe that when an agent's reactions override other steering behaviors, it is likely to slow down for several more seconds. In this case, we decimate perceptions to 0.25 Hz (every 80 frames).

- **Prediction.** Predictions, in our current framework, are about the same cost as reactions, and already can be safely decimated to 3-4 Hz. Thus, to facilitate decimating reactions, which can be a bigger performance gain, we do not decimate prediction dynamically.

- **Reaction.** It is possible to carefully avoid computing reactions at a full 20 Hz. The frequency of reactions should be dependent on how close the agent might be to a collision. Therefore, the first heuristic is to maintain a full 20 Hz reactions if the agent touches anything with its "feelers". The second heuristic is that we can decimate reactions a little bit if there are no perceived threats in the visual field. In this case we decimate to 10 Hz (every 2 frames). Finally, if the agent's visual field is completely empty, we can risk updating reactions at 5 Hz.

Figure 9 shows benchmark scores for several test cases, comparing undecimated reference behaviors, static decimation behaviors, and adaptive decimation behaviors. In these results, "matched" decimation are tests where we tried to use the same average frequencies achieved by adaptive decimation, but using those frequencies in static decimation mode. In general, adaptive decimation scored better on the benchmark than static and matched cases, yet it consistently performed better. This shows that our adaptive decimation method was able to exploit decimation for good performance when appropriate, while scaling back the decimation when needed for proper steering.

5.4 Performance Results

The primary implication of being able to decimate is improved performance. For the results described in this paper, we used a single thread on a 2.66 GHz Intel Core 2 Duo. A detailed performance profile is shown in Table 3. These performance numbers were acquired using the *Random* test case, consisting of 4000 agents randomly placed in a dense environment with random targets. The simulation was run for 1 minute (1200 frames). We use a base-line comparison where prediction, perception, and short-term planning all run at 5 Hz. This represents recent methods such as [Treuille et al. 2006] and [Paris et al. 2007], whose algorithms run this frequency on similar processors. Figure 9 shows performance results for more test cases. The corresponding benchmark scores indicate that adaptive decimation achieves this performance without degrading the quality of our steering results.

Recall in the introduction that practical applications can usually only support tens of agents, mainly because they can allot only 5-10% of computing resources to AI. Conservatively, at 5% of one second (50 milliseconds), and processing steering behaviors at 20 Hz, we would have merely 2.5 milliseconds to update all agents. In that amount of time, we can update approximately 400 agents with robust steering. We can simulate up to 4000 agents with adaptive decimation in real-time, using approximately 50% of computing resources.

There are two elegant ways that computations are balanced in our framework. First, more costly phases can generally be updated less frequently. This is clearly seen in Table 3. The second balance is more subtle, and it is related to the density of the environment. In dense, crowded situations, agents rely on the fast reaction phase, and we can aggressively decimate perceptions and predictions (when using adaptive decimation). On the other hand, in less dense environments, there are fewer items to perceive and predict, and so perception and predictions automatically become proportionally faster. In other words, both dense and sparse environments offer a way to achieve better performance with our framework. All

Un-decimated Reference

Phase	Frequency (Hz)	Profile (%)	Avg. time per update (μsec)
space-time planning	on-demand	0.001	216
mid-term path planning	on-demand	1.96	427
short-term planning	5	33.4	44.0
perception	5	49.2	65.0
prediction	5	2.73	3.61
reaction	20	10.7	3.54
locomotion	20	2.01	0.66
Amortized time for a single update at 20 Hz			**33.0 μs**

Static Decimation

Phase	Frequency (Hz)	Profile (%)	Avg. time per update (μsec)
space-time planning	on-demand	0.001	185
mid-term path planning	on-demand	7.93	425
short-term planning	0.2	6.00	45.6
perception	0.67	26.9	65.6
prediction	3.33	7.47	3.6
reaction	20	43.6	3.5
locomotion	20	8.07	0.66
Amortized total for a single update at 20 Hz			**8.1 μs (4×)**

Adaptive Decimation

Phase	Avg. Freq. (Hz)	Profile (%)	Avg. time per update (μsec)
space-time planning	on-demand	0.001	318
mid-term path planning	on-demand	9.95	431
short-term planning	0.168	5.67	42.6
perception	0.288	13.9	61.0
prediction	3.33	9.38	3.66
reaction	17.09	50.6	3.74
locomotion	20	10.5	0.67
Amortized total for a single update at 20 Hz			**6.3 μs (5×)**

Table 3: Detailed performance comparison, comparing a 5 Hz reference, static decimation, and adaptive decimation. These statistics were collected for the *Random* test case from the benchmark suite, where 4000 agents in a dense environment steer towards random (the same in all three comparisons) goals.

these properties contribute to keeping amortized costs low, and ultimately this makes it possible for our framework to support thousands of robust agents in real-time.

6 Conclusion

We have presented three contributions for agent-based steering: (1) a way to integrate multiple steering approaches into a single framework, (2) a space-time planning technique that solves complicated steering scenarios, and (3) phase decimation that allows us to significantly reduce the amortized cost of each phase. We demonstrated a framework with these contributions that performs robustly and efficiently on a wide variety of scenarios, ranging from challenging local interactions to dense crowds of thousands of agents. We envision that future commercial applications will benefit greatly from the experiments and results discussed in this paper.

Acknowledgements. The work in this paper was partially supported by an Intel Visual Computing Grant. We would also like to a thank Intel Corp. and Microsoft Corp. for their generous support through equipment and software grants.

References

BOULIC, R. 2008. Relaxed steering towards oriented region goals. *Lecture Notes in Computer Science 5277, MIG 2008*, 176–187.

BROGAN, D. C., AND HODGINS, J. K. 1997. Group behaviors for systems with significant dynamics. *Auton. Robots 4*, 1, 137–153.

FEURTEY, F., 2000. Simulating the collision avoidance behavior of pedestrians. Master's Thesis.

GOLDENSTEIN, S., ET AL. 2001. Scalable nonlinear dynamical systems for agent steering and crowd simulation. *Computers and Graphics 25*, 6, 983–998.

HART, P. E., NILSSON, N. J., AND RAPHAEL, B. July 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE TSSC 4*, 2, 100–107.

HELBING, D., FARKAS, I., AND VICSEK, T. 2000. Simulating dynamical features of escape panic. *Nature 407*, 487.

KAPADIA, M., SINGH, S., HEWLETT, W., AND FALOUTSOS, P. 2009. Egocentric affordance fields in pedestrian steering. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*.

LAMARCHE, F., AND DONIKIAN, S. 2004. Crowd of virtual humans: a new approach for real time navigation in complex and structured environments. *Computer Graphics Forum 23*, 509–518(10).

LAU, M., AND KUFFNER, J. J. 2005. Behavior planning for character animation. In *2005 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 271–280.

LEE, K. H., CHOI, M. G., HONG, Q., AND LEE, J. 2007. Group behavior from video: a data-driven approach to crowd simulation. In *SCA '07*, Eurographics Association, 109–118.

LERNER, A., CHRYSANTHOU, Y., AND LISCHINSKI, D. 2007. Crowds by example. *Computer Graphics Forum 26*, 3 (September), 655–664.

LOSCOS, C., MARCHAL, D., AND MEYER, A. 2003. Intuitive crowd behaviour in dense urban environments using local laws. In *IEEE TPCG '03*, 122.

METOYER, R. A., AND HODGINS, J. K. 2004. Reactive pedestrian path following from examples. *The Visual Computer 20*, 10, 635–649.

PARIS, S., PETTRÉ, J., AND DONIKIAN, S. 2007. Pedestrian reactive navigation for crowd simulation: a predictive approach. In *EUROGRAPHICS 2007*, vol. 26, 665–674.

PARIS, S., GERDELAN, A., AND O'SULLIVAN, C. 2009. Calod: Collision avoidance level of detail for scalable, controllable crowds. In *Motion in Games*, vol. 5884 of *LNCS*. Springer, 13–28.

PELECHANO, N., ALLBECK, J. M., AND BADLER, N. I. 2007. Controlling individual agents in high-density crowd simulation. In *SCA '07*, 99–108.

RABIN, S. 2005. *Introduction to Game Development*. Charles River Media, Inc.

REYNOLDS, C. W. 1987. Flocks, herds and schools: A distributed behavioral model. In *ACM SIGGRAPH '87*, 25–34.

REYNOLDS, C. 1999. Steering behaviors for autonomous characters. In *Game Developers Conference*.

RUDOMÍN, I., MILLÁN, E., AND HERNÁNDEZ, B. 2005. Fragment shaders for agent animation using finite state machines. *Simulation Modelling Practice and Theory 13*, 8, 741–751.

SHAO, W., AND TERZOPOULOS, D. 2005. Autonomous pedestrians. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM, New York, NY, USA, 19–28.

SHAPIRO, A., KALLMANN, M., AND FALOUTSOS, P. 2007. Interactive motion correction and object manipulation. In *I3D '07: Proceedings of the 2007 symposium on Interactive 3D graphis and games*, 137–144.

SINGH, S., KAPADIA, M., FALOUTSOS, P., AND REINMAN, G. 2009. Steerbench: a benchmark sutie for evaluating steering behaviors. *Computer Animation and Virtual Worlds*.

SUD, A., GAYLE, R., ANDERSEN, E., GUY, S., LIN, M., AND MANOCHA, D. 2007. Real-time navigation of independent agents using adaptive roadmaps. In *VRST '07*, ACM, New York, NY, USA, 99–106.

TREUILLE, A., COOPER, S., AND POPOVIĆ, Z. 2006. Continuum crowds. In *SIGGRAPH '06*, ACM, New York, NY, USA, 1160–1168.

VAN DEN BERG, J., PATIL, S., SEWALL, J., MANOCHA, D., AND LIN, M. 2008. Interactive navigation of multiple agents in crowded environments. In *ACM I3D '08*, ACM, New York, NY, USA, 139–147.