Watch Out! A Framework for Evaluating Steering Behaviors

Shawn Singh, Mishali Naik, Mubbasir Kapadia, Petros Faloutsos, and Glenn Reinman

University of California, Los Angeles

Abstract. Interactive virtual worlds feature dynamic characters that must navigate through a variety of landscapes populated with various obstacles and other agents. The process of navigating to a desired location within a dynamic environment is the problem of *steering*. While there are many approaches to steering, to our knowledge there is no standard way of evaluating and comparing the quality of such solutions. To address this, we propose a diverse set of benchmarks and a flexible method of evaluation that can be used to compare different steering algorithms. We discuss the challenges and criteria for objectively evaluating steering behaviors and describe the metrics and scoring method used in our benchmark evaluation. We hope that, with constructive feedback from the community, our framework will eventually evolve into a standard and comprehensive approach to debug, compare and provide an overall assessment of the effectiveness of steering algorithms.

1 Introduction

A fundamental requirement of nearly all agents in virtual worlds is *steering*: the ability of an agent to navigate to a goal destination, through an environment that includes static obstacles, such as buildings, and dynamic obstacles, such as other virtual characters. Steering is a challenging problem for autonomous agents. In reality, steering is a result of a complex process: An agent makes steering decisions based on sensory information, predictions of the motion of dynamic obstacles, social etiquette, personal experience, situation specific parameters, cognitive goals and desires. Even within the simplified environment of a virtual world, the state space of the steering problem is too large to allow for trivial solutions, such as pre-computed state-action tables.

There is a significant amount of research that tries to address the steering problem. Current solutions seem to address only a subset of the problem's challenges. For example, particle-based approaches are well suited for macroscopic crowd behaviors, while agent-based approaches work better for the local interaction of a small group of agents.

Given the importance of steering in modern applications and the growing number of steering algorithms, it is important and timely to ask the question, *how can we compare different steering approaches?* To our knowledge this paper makes the first attempt to answer this fundamental question.

A. Egges, A. Kamphuis, and M. Overmars (Eds.): MIG 2008, LNCS 5277, pp. 200–209, 2008.
© Springer-Verlag Berlin Heidelberg 2008

We propose a novel benchmarking tool for comparing steering behaviors. Our framework is based on two components that form the main contributions of this paper:

- *A diverse suite of steering benchmarks*: We propose a forward-looking set of scenarios that capture the broad range of situations a steering algorithm may encounter in practical applications.
- *A method and metrics of evaluation*: We propose a set of metrics that can be used to evaluate the behavior of steering algorithms. We also propose a method of scoring the behavior, so that two steering algorithms can be compared. We call this process *benchmark evaluation*.

1.1 Criteria for Effective Evaluation

The main challenge in designing steering benchmarks is how to evaluate a steering algorithm *objectively*. Determining the quality of the results of a steering algorithm depends on many factors, many of which are situation-specific and/or depend on cognitive decisions by the agents. For example, an agent may decide to push through a crowd or politely go around the crowd, depending on the agent's situation and personality. To remain objective even with seemingly ad hoc constraints, we propose that steering evaluation should satisfy the following criteria:

1. The test cases should be representative of a broad range of situations that are common in real-world scenarios.
2. The evaluation should be blind to the specifics of the steering algorithm.
3. The evaluation should be customizable to allow a user to test for certain expected behaviors over others.

To meet these criteria, we separate the concepts of providing test cases and evaluating a steering result. This allows users to specialize test cases to their needs, and our benchmark evaluation would still apply to the custom test cases. Our evaluation uses metrics computed from the position, direction, and the goal that the agent is trying to reach. The interpretation of these metrics is surprisingly meaningful, yet at the same time, they do not require any knowledge of the algorithm that produced the steering result.

Of course, no set of benchmarks and associated metrics could cover and evaluate every possible steering situation that can happen in the real world – and we do not claim that ours does. We cannot necessarily tell how well one algorithm does in absolute terms. Instead, our framework gives a good indication of the pros and cons of any steering algorithm and to effectively *compare* different approaches.

2 Related Work

Benchmarking is a crucial process in many fields – ranging from business management to software performance. Benchmark suites have been developed for a

variety of purposes related for graphics and multimedia, including hardware [1], global illumination [2], animation for ray tracing [3], general-purpose architecture [4], and many more. In these fields, benchmarks have clear metrics for comparison: performance in seconds, signal-to-noise ratio, power consumption, area, monetary price, etc. Steering behaviors and other aspects of artificial intelligence do not have a clear objective metric – instead, much of the evaluation is inherently subjective.

Efficiency Metrics. Some works have proposed metrics that evaluate "believability" or "natural behavior." One major approach is to follow the rule that natural behaviors are usually efficient (e.g. [5]). This approach has been used effectively for animation, e.g. [6,7]. Our work applies this same principle to the evaluation of steering behaviors, while keeping the user in control of the final evaluation process.

Approaches to Steering. Most steering behaviors can be classified into two major categories: Dynamics-based and agent-based. Dynamics based models represent the environment with potential fields or flow maps, often treating each agent as a particle (e.g., [8,9,10]). Rule-based models use heavy branching to determine the exact scenario being described, and performs an action associated with that scenario (e.g., [11,12,13,14]). A problem is that such papers only have enough space to showcase their novel features, and it is difficult to assess how effective a steering algorithm will be on fundamental, common scenarios that were not the focus of the paper. A benchmark suite can illuminate these results in a compact form that future papers could very easily include.

3 Benchmark Suite

Our framework consists of two major parts: (1) a diverse set of test cases, and (2) a benchmark evaluation utility that uses several metrics to score a steering algorithm. In this section, we describe the suite of test cases.

Overview. Our current framework contains 36 scenarios. Several of the scenarios have many variations, resulting in a total of 50 test cases. The test cases can be classified in five major categories: (1) simple validation scenarios, (2) basic one-on-one interactions, (3) agent interactions including obstacles, (4) group interactions, and (5) large-scale scenarios. Many test cases can be classified in multiple categories – this classification is only for presentation, and does not limit the test cases or evaluation process in any way. These five categories are detailed in Section 3.1.

Each test case specifies a number of agents and static objects. Static objects are specified by a bounding box. Agents are specified by their size, initial position, initial direction, desired speed, and target location(s). Additionally, each test case specifies which agents should be examined during benchmark evaluation, and a set of weights that configure the evaluation process, discussed in Section 5. Dynamic objects can be specified as agents in a customized scenario, however, our current suite does not test agent's steering behaviors with dynamic objects because the expected behaviors in such cases are application-specific.

Test Case Design Choices. The main challenge of designing the benchmark suite is to cover the range of possible scenarios without having an inordinate number of test cases. With this in mind, we choose our test-cases to be common, frequently appearing scenarios, but with challenging, worst-case parameters. For example, most of the static obstacles have sharp corners which are generally more difficult to handle than smooth ones.

In our experience, an agent in a typical urban environment faces the following situations:

1. *Walking alone.* Most of the time, an agent interacts with only 2-4 nearby agents and a few obstacles at a time.
2. *Walking as part of a small group.* Often the agent moves in the same direction with 2-6 other agents and encounters other groups and obstacles.
3. *Walking as part of a crowd.* Groups of hundreds of agents tend to form less frequently, and only in specific situations, for example when a large number of agents exit or enter a building at the same time.

Furthermore, some of the situations involving many agents, can often be viewed as a sequence of smaller problems. For example, a lone agent that tries to go through a dense oncoming crowd of hundreds of agents, may only consider 4-5 people at a time. Based on these observations, we choose test cases that reflect these smaller problems which can represent a large number of composite real-world scenarios.

It is fair to ask the question, what is the effect of the specific number of agents in the large scale examples? We have selected default values for these parameters based on what we think are average cases in the real world. We have no reason to believe that the specific number of agents in the large scale examples is crucial. If an algorithm can handle a bottleneck example with exactly 500 agents, it should be able to handle around 500 as well. In any case, we would like to remind the reader that our goal is to provide an estimate of an algorithms performance, not a proof of its robustness or correctness.

Customizing the Suite. It is clearly not possible to cover every conceivable situation in our test cases. Therefore, we have designed our benchmarking approach to be flexible and customizable, so that users can quickly focus on the details of interest to their algorithm.

The user can easily create custom scenarios to use with our existing benchmark evaluation. This allows our benchmark evaluation process to be useful even for applications that cannot use our provided test cases. For example, steering behaviors found in a sports game will have unique steering scenarios that should be evaluated with unique criteria.

The initial conditions and parameters can also be re-defined by the user. In its current form, our test cases are intended to roughly approximate typical humans: agents have a diameter of 1 meter (roughly the distance from elbow to elbow of an average human) and an average walking speed of 1.3 meters per second [15]. The user can use slightly modified initial conditions to simulate cars, bicyclists, or any dynamic objects.

3.1 Description of the Scenarios

Here we describe the scenarios used in the current version of our benchmark suite. These scenarios can be seen in Figure 1. Many of these test cases are very challenging and forward-looking, and we expect that initially very few algorithms, if any, will be able to successfully handle all scenarios gracefully. Scenarios annotated with an asterisk (*) are in our opinion more difficult scenarios.

Simple Validation Scenarios. These scenarios are designed to test very basic, fundamental abilities of a steering agent. While such behaviors are trivial, it is still important for an algorithm to successfully handle these cases.

- *Simple:* one agent steering towards a target located to the left, right, or behind.
- *Simple-obstacle:* one agent steering towards a target located behind an obstacle.
- *Simple-wall:* two agents steering around a wall to reach a target.
- *Curves:* one agent steering through an S-curve to reach the target.

Basic One-on-one Interactions. These scenarios test the ability of agents to steer around each other. In this category, the emphasis is on natural interactions without other threats to distract the agents.

- *Oncoming:* two agents traveling in opposite directions towards a head-on collision.
- *Crossing:* two agents crossing paths at various angles.
- *Oncoming-trick:* two oncoming agents that will not collide because their targets are closer.
- *Crossing-trick:* two crossing agents that will not collide because their targets are closer.
- *Similar-direction:* two agents with slightly different goals traveling in a similar direction.

Agent-agent Interactions Including Obstacles. These scenarios test the ability of an agent to navigate around static objects while interacting with other agents.

- *Oncoming-obstacle:* two oncoming agents, with an additional obstacle in the way.
- *Crossing-obstacle:* two crossing agents, with an additional obstacle in the way.
- *Surprise:* two agents that do not see each other until the last minute because of large obstacles.
- *Squeeze:* two oncoming agents walking through a narrow hallway.
- *Doorway-one-way:* two agents enter a doorway from the same side.
- **Doorway-two-way:* two oncoming agents walk through a doorway from opposite sides.
- *Overtake:* one agent is expected to overtake the other agent in a hallway.
- **Overtake-obstacle:* one agent is expected to overtake the other agent, with obstacles in the hallway.

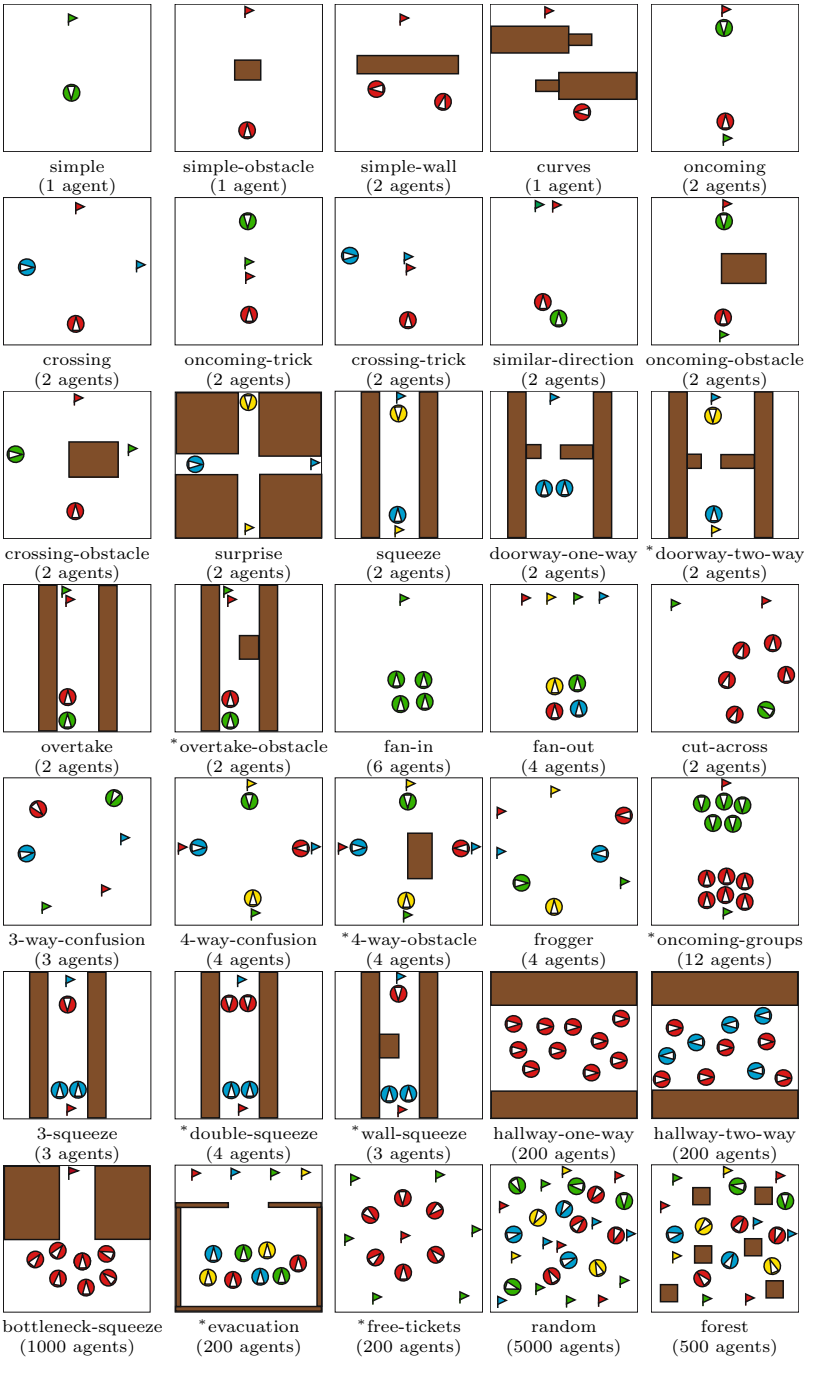

Fig. 1. Approximate depiction of each scenario in the current version of our benchmark suite. The number in parentheses indicates how many agents are specified in the scenario, and asterisks indicate more difficult scenarios.

Group Interactions. Group interactions are composed of several agents and static objects, intended to test an algorithm's ability to handle a variety of common situations.

- *Fan-in:* a small group of agents aiming for the same target. This tests how agents cooperate while contending for the same space.
- *Fan-out:* a small group of agents aiming for slightly separated targets. This tests whether agents will unnaturally stick to the crowd when their goal is in a different direction.
- *Cut-across:* one agent cutting across a small group.
- *3-way-confusion:* three agents traveling in different directions, meeting at nearly the same time.
- *4-way-confusion:* four agents traveling in four opposing directions, meeting at nearly the same time.
- **4-way-obstacle:* four agents crossing paths with a static object in the way.
- *Frogger:* one agent encounters many perpendicular crossing agents.
- **Oncoming-groups:* a small group of agents encounters another small group of agents traveling in opposite direction.
- *3-squeeze:* two agents facing the same direction encounter an oncoming agent in a narrow hallway.
- **Double-squeeze:* two agents facing the same direction encounter two oncoming agents in a narrow hallway.
- **Wall-squeeze:* two agents facing the same direction encounter an oncoming agent in a narrow hallway with an obstacle.

Large Scale Scenarios. These scenarios are designed to stress-test the ability of an algorithm to handle macroscopic situations, and to scale to a large number of agents.

- *Hallway-one-way:* many agents traveling in the same direction through a hallway.
- *Hallway-two-way:* many agents traveling in either direction through a hallway. Agents should form lanes.
- **Bottleneck-squeeze:* all agents begin on one side of the arena, and must enter and traverse a hallway to reach the target. Note that hard corners at the bottleneck are much more challenging than rounded corners.
- **Evacuation:* all agents must exit a crowded room that has only one exit.
- **Free-tickets:* all agents are aiming for the same target in the middle of the arena, and have a random secondary goal once the middle target is reached. This scenario is particularly difficult because agents that reach the middle goal must then turn to face a dense oncoming crowd. This scenario requires both natural individuals and natural crowds simultaneously.
- *Random:* each agent is placed randomly in the arena and has an individual random target. Here, stress is placed on handling a large number of agents. Our default test case specifies 5000 agents for this scenario.
- *Forest:* each agent is placed randomly in an arena filled with small obstacles.
- *Urban:* each agent is placed randomly in an arena filled with building-sized obstacles.

4 Metrics of Evaluation

Given the suite of benchmarks, the next question is how to evaluate the result of a steering algorithm. We propose that a set of meaningful metrics can be measured directly from the output of a steering algorithm, without any knowledge about the algorithm itself. This section describes the metrics that we compute.

Our benchmark evaluation works as follows. First, the user records the position, direction, and goal target of every agent for every frame. Second, our evaluation tool will read this recorded information to compute a variety of statistics about each agent's behavior. Users can see these detailed statistics or automatically computed scores based on three primary metrics.

Primary Metric 1: Number of Collisions. The first primary metric is the number of collisions that occur for a given agent. In most cases, fewer collisions indicates more realistic steering behavior. One notable exception is the *Surprise* scenario, where it would be natural for two agents to collide because they do not see each other soon enough. Our evaluation tool computes the number of unique collisions that occur as well as the total number of frames that each agent spent in a collision with other objects.

Primary Metrics 2 and 3: Time and Effort Efficiency. The second and third primary metrics measure two forms of efficiency. These, and most of the detailed metrics as well, are based on the idea that *efficient* behaviors are very often natural behaviors. Section 2 describes many references that use the same principle.

Time efficiency measures how quickly the agent is able to reach its goal destinations. Of course, the quicker the agent reaches its goal, the more time efficient the agent is. Our evaluation tool measures time efficiency as the total time (in seconds) that an agent spent to reach its goal.

Effort efficiency measures how much effort an agent spent to reach its goal destinations. The less effort an agent spent, the more effort efficient the agent is. Our evaluation tool measures effort efficiency as the integral (sum total) of the magnitude of acceleration that an agent used to reach its goal. Note that acceleration includes changes in speed and changes in direction, and its magnitude is always positive.

The combined interpretation of time and effort efficiency provides insight into a spectrum of behaviors. Some agents may desire to reach their destinations quickly, willing to spend more effort. Other agents may desire to save effort and slowly, politely progress towards their goals.

Detailed Metrics. The rest of the metrics measure variations of: (1) speed, (2) turning rate (angular velocity), and (3) change in speed. Specifically, for each of these three, we measure the total (integral), max instantaneous, average, and max/min over a sliding window. The sliding window in our current implementation is an integral over an interval of 20 frames. We compute a new window integral every next frame, and finally store the max/min of these values.

These detailed metrics are interesting to examine when a user knows the expected behavior of a scenario. A user will usually be able correlate one or two of the metrics with a clear indication of unnatural behavior in the scenario. For example, if a character is expected to go straight towards its goal with very little turning, then the "integral of turning rate," which describes the total amount of turning, should be close to zero. Similarly, if an agent is expected to have one abrupt turn in the scenario, the "integral of turning rate in a window interval" should be somewhat large, while the "total average turning rate" should be small.

5 Benchmark Scoring

After computing the metrics described in the previous section, the final task is to compute a meaningful score that represents the quality of the steering behavior. A score can be computed for (1) a single agent in a test case, (2) all agents in a test case, or (3) across all test cases.

A Note About Benchmark Scores. It is important to note that scoring is not intended to be a proof of an algorithm's effectiveness. Instead, the purpose of scoring is to create *a simple number* that allows for a quick, intuitive estimate of evaluation, especially when comparing two approaches. To do a more rigorous analysis of the pros and cons of an algorithm, users need to manually examine the detailed metrics, and perhaps even tailor their own test cases.

Scoring One Agent in One Test Case. An agent's score can be computed by combining the three major metrics described in Section 4: number of collisions, time efficiency, and effort efficiency. The user describes their relative importance as numerical weights, specified in the test case. Each agent in the test case has its own unique set of weights.

The score is simply a weighted sum of the metrics:

$$S_i = w_c C + w_t T + w_e E, \tag{1}$$

where S_i is the score of the i^{th} agent, w_c, w_t, and w_e are the user-specified weights, C is the number of collisions, T is time efficiency, and E is effort efficiency.

Scoring all Agents in One Test Case, and Across all Test Cases. To evaluate all agents in one test case, we simply compute the average over n agent scores:

$$S_A = \sum_{i=0}^{n} S_i. \tag{2}$$

Because each agent can have its own set of weights, the user can easily control the relative importance of agents in the scoring process.

Finally, to score an algorithm across all test cases, we can compute a sum total of the scores from each test case.

$$S_m = \sum_{A} S_A. \tag{3}$$

6 Conclusion

In this paper we present a framework for evaluating steering behaviors. The framework includes a diverse suite of test cases and an objective method of evaluation. The framework covers a broad range of common scenarios, is blind to the specifics of the steering algorithm, and is extensible and customizable. In future work, we plan to continue growing the suite of test cases, make the framework available online, and demonstrate its effectiveness on recordings of real humans steering through our test cases. We envision that this framework can grow into a standard for steering evaluation.

References

1. Futuremark: 3DMark (2008), http://www.futuremark.com
2. RealStorm: RealStorm Global Illumination Benchmark (2008), http://www.realtimeraytrace.de/
3. Lext, J., Assarsson, U., Moller, T.: A benchmark for animated ray tracing. IEEE Computer Graphics and Applications 21(2), 22–31 (2001)
4. Henning, J.L.: Spec cpu2006 benchmark descriptions. SIGARCH Comput. Archit. News 34, 1–17 (2006)
5. Reitsma, P.S.A., Pollard, N.S.: Evaluating motion graphs for character animation. ACM Trans. Graph. 26(4), 18 (2007)
6. Wu, J.c., Popović, Z.: Realistic modeling of bird flight animations. ACM Trans. Graph. 22(3), 888–895 (2003)
7. Tu, X., Terzopoulos, D.: Artificial fishes: physics, locomotion, perception, behavior. In: SIGGRAPH 1994: Proceedings of the 21st annual conference on Computer graphics and interactive techniques, pp. 43–50. ACM Press, New York (1994)
8. Brogan, D.C., Hodgins, J.K.: Group behaviors for systems with significant dynamics. Auton. Robots 4(1), 137–153 (1997)
9. Goldenstein, S., et al.: Scalable nonlinear dynamical systems for agent steering and crowd simulation. Computers and Graphics 25(6), 983–998 (2001)
10. Treuille, A., Cooper, S., Popović, Z.: Continuum crowds. In: SIGGRAPH 2006: ACM SIGGRAPH 2006 Papers, pp. 1160–1168. ACM, New York (2006)
11. Paris, S., Pettré, J., Donikian, S.: Pedestrian reactive navigation for crowd simulation: a predictive approach. In: EUROGRAPHICS 2007, vol. 26, pp. 665–674 (2007)
12. Lamarche, F., Donikian, S.: Crowd of virtual humans: a new approach for real time navigation in complex and structured environments. Computer Graphics Forum 23(10), 509–518 (2004)
13. Loscos, C., Marchal, D., Meyer, A.: Intuitive crowd behaviour in dense urban environments using local laws. In: TPCG 2003: Proceedings of the Theory and Practice of Computer Graphics 2003, p. 122. IEEE Computer Society, Washington (2003)
14. Shao, W., Terzopoulos, D.: Autonomous pedestrians. In: SCA 2005: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 19–28. ACM, New York (2005)
15. Knoblauch, R.L., Pietrucha, M.T., Nitzburg, M.: Field studies of pedestrian walking speed and start-up time. Transportation Research Record 1538, 27–38 (1996)