Robust Space-Time Footsteps for Agent-Based Steering

Glen Berseth
University of British Columbia

Mubbasir Kapadia
Rutgers University

Petros Faloutsos
York University

Abstract

Recent agent-based steering methods abandon the standard particle abstraction of an agent's locomotion abilities and employ more complex models from timed footsteps to physics-based controllers. These models often provide the action space of an optimal search method that plans a sequence of steering actions for each agent that minimize a performance criterion. The transition from particle-based models to more complex models is not straightforward and gives rise to a number of technical challenges. For example, a disk geometry is constant, symmetric and convex, while a footstep model maybe non-convex and dynamic. In this paper, we identify general challenges associated with footstep-based steering approaches and present a new space-time footstep planning steering model that is robust to challenging scenario configurations.

Keywords: Crowd Simulation, Footsteps, Planning and Analysis

1 Introduction

While traditionally sliding particle models have been the standard for crowd simulation, there has been recent interest in footstep-based steering. Footstep-based models do not suffer from the *footskate* issue, where feet turn or slide while in contact with the ground. They can easily incorporate dynamic representations of the agent's body and thus achieve denser crowd packing [1].

While footstep-based steering is known to remove many artifacts from the problem of mapping human motion to the steering behaviour, there are still a number of challenges researchers face when adopting more complex agent models. In a standard footstep-based steering approach, an A* algorithm is used to find optimal paths from the agent's current location to the agent's target location as a sequence of footsteps. Yet, how do we know the types of footsteps to use or what is a good stepping distance range, or even how should we initially configure an agent to ensure it can reach its target?

We focus on the issue of making a footstep-based steering algorithm resilient to environment configuration. Specifically, we present a robust footstep-based steering algorithm to avoid invalid initial configuration and to prune undesirable and potentially unsound short term goal states.

This is done in two steps. First, geometric checks are used when adding an agent to a scenario, ensuring the agent can make an initial step. Second, we add constrained random footsteps to handle cases where pre-defined step intervals can result in an inability to find a plan. These, together with the properties of the A* search method, construct a more robust steering strategy.

2 Related Work

Sliding particle methods [2, 3] model the agent as a disk centred at the particle's location. There are a number of issues when driving bipedal characters with only position information. Sliding disks can instantly change their forward direction, which is not natural for a biped. For people, complex interactions occur at doorways where tendency is to step aside for oncoming traffic, both disc models may try to push through the doorway at the same time. Resulting in an unusual sliding contact motion between the agent. Footstep-based models do not suffer from these issues and have been used in dynamic environments [4].

Footstep-based planning is used by both the computer animation community [1] and the robotics community. This work uses a similar method to both of these works but focuses on creating an algorithm that can endure random environment layouts and agent state configurations.

3 Initial Agent Placement

Extracting a valid plan to for a footstep-based steering agent is still a poorly understood problem; an infinite number of possible plans exists that could lead the agent to its target. To understand the conditions necessary to ensure a plan can be found we must analyze the problem inductively.

A scenario s is a collection of agents **A** and obstacles **O**. In traditional crowd simulation the geometry of an agent is a disk. In footstep-based models the geometry for an agent is dynamic depending on the current state of the agent. This dynamic geometry suffers from complex configurations that can result in an invalid state where the agent can not proceed without colliding with an obstacle.

To add agents to a scenario, particular locations could be hand selected but the most versatile method would be to add agents randomly. In order to add an agent to a scenario two properties must be satisfied. The first, which is true for any type of crowd simulation algorithm, is that the agent must not overlap any items in the scenario. A footstep-based model needs an additional check to make sure the agent can make a footstep from its initial configuration. Given the forward direction of the agent, a rectangular region can be traced out in front of the agent. An example initial geometry check is illustrated in Figure 1(b) that detects overlap with nearby obstacles. These two properties together ensure that the agent does not start intersecting any geometry and will be able to make an initial step.

4 Robust Footstep Planning

In a footstep-based steering method, a plan is computed between two locations that is free of collisions. The actions in the plan can be understood as a sequence of footstep actions in space-time $\langle step_0 \ldots step_n \rangle$.

The state of a footstep-based agent is defined as

$$st = \{(x,y),(\dot{x},\dot{y}),(f_x,f_y),f_\phi,I \in \{L,R\}\}. \quad (1)$$

Where (x,y) and $(\dot{x},\dot{y})$ are the position and velocity of the centre of mass. The current footstep is described by the location (f_x,f_y), orientation f_ϕ, and foot $I \in \{L,R\}$. Potential actions are created, using an *inverted pendulum* model, between states by considering an action with orientation ϕ, velocity and time duration. Each step has a cost related to step length and ground reaction forces. The heuristic function is then a combination of the expected cost of the step and the number of steps left to reach the goal.

4.1 Improved Footstep Sampling

The A* planning algorithm in a footstep-based steering model is used to compute safe navigation decisions during simulation. The successor states that are generated using an A* model can be ad-hoc, with different footstep angles and durations at fixed intervals [1]. However, there is an infinite number of geometry combinations in a scenario and it is simple to construct an example scenario where an ad-hoc method can not find a valid step when many exist. To make the planning system more robust, *randomized* footsteps are introduced. The randomized angle orientations (in radians) and step time lengths are limited to be between 0.3 and 1.3. An example is shown in Figure 1(c), where an agent starts in a corner and can escape after a feasable random step is generated. By adding randomized step angles and step distances, the algorithm achieves better theoretical properties for steering in any possible scenario configuration.

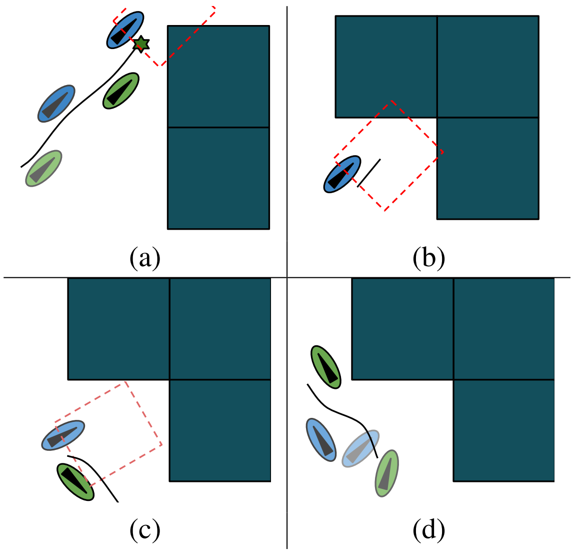

Figure 1: Corner cases that are avoided by the *robust* algorithm. The left foot is blue, the right green, the dashed box is the geometry overlap check and the green star is a generated waypoint. The navy blue squares are obstacles.

4.2 *Finite Horizon Planning*

When planning is used, the path found is guaranteed to be sound from beginning to end. However, it is common to mix long range global planning with a more dynamic finite horizon planing between waypoints. When using finite horizon planning, the final state of the composed plan can put the agent in a state were the agent can not make a step. An example of an agent getting stuck straddling an obstacle is illustrated in Figure 1(a). These invalid states can be avoided by applying the same configuration checks used when an agent is randomly placed in a scenario, at the end of every short term plan.

Additional optimizations can be placed on short term planning to improve efficiency and fitness. The first of these is to never execute the final action in a short term plan unless that action is a final goal state. The geometry configuration validation ensures there will be a possible footstep, but re-planning one step earlier results in a more realistic plan.

4.3 *Additional Footstep Types*

The last feature of the planning system is a new footstep style. In lieu of common forward stepping at different angles, footsteps that simulate in-place turning can be used. In-place turning is done by allowing the agent to take steps where the agent's heels are close, with the feet being nearly perpendicular or the next stance foot is placed pointing inward, as shown in Figure 1(d).

5 Analysis and Results

A group of metrics similar to [5] are used to compare footstep-based algorithms. These metrics use the concept of a *reference-agent*, Other agents added to the scenario make the scenario more challenging for the *reference-agent*. Scenario specific metrics are defined and then aggregate metrics over a sizable sampling of $10,000$ scenarios ($\mathbf{S}$) are used.

For a single scenario s, when the *reference-agent* reaches its target location before the max simulation time expires[1], *completed* is 1 and 0 otherwise. The second metric, *solved*, is 1 when *completed* is 1 and the *reference-agent* reaches its target location without any collisions, otherwise $solved = 0$. These metrics are aggregated over $\mathbf{S}$ with $completion = \sum_{s \in \mathbf{S}} completed(s)$ and $coverage = \sum_{s \in \mathbf{S}} solved(s)$. An average of *completed* is used over all agents in a scenario as *all-completed* that is equal to the percentage of agents in the simulation that have $completed = 1$. Similarly, *all-solved* is an average for *solved*. To measure computational performance, the time spent simulating $\mathbf{S}$ is computed, denoted as *simTime*.

Three versions of footstep-based algorithms are compared. The first version is a common footstep-based method *baseline* [1]. The second version, *baseline-with-randomization*, is the *baseline* method with randomized footstep actions. The final version of the algorithm, *robust*, includes both randomized footstep actions and geometric checks. Using a combination of metrics, comparisons are made as to the effectiveness of each footstep-based algorithm. The results of these comparisons can be seen in Figure 2.

Notably, the largest increase in fitness comes from adding randomness and in-place turning to the algorithm. These two features together al-

[1]We give an agent more than enough time to navigate around the boundary of the scenario twice.

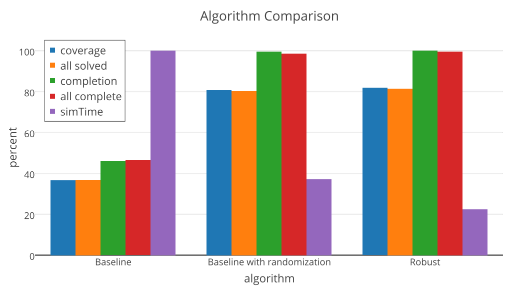

Figure 2: A comparison of three footstep-based algorithms, using five performance metrics.

low the algorithm to generate a wider variety of possible footsteps and increase *coverage* from 37% to 81%. Adding geometry checks to the algorithm increases the overall *completion* to a perfect 100% from an original 46%. The *robust* version of the algorithm is surprisingly capable. After simulating $\sim 45,100$ agents, only ~ 150 agents do not reach their target locations.

The *robust* algorithm has significant computational performance improvements over *baseline*. By using random footstep actions, the algorithm explores the search space more resourcefully, avoiding locally optimal regions in favour of smoother, lower effort, plans. With geometrical checks pruning undesired branches from the search space, the final algorithm simulates the $10,000$ scenarios in $\sim 1/5$ the time.

6 Conclusion

A more sound footstep-based steering method has been presented. This method has been analyzed and compared to a common version of the algorithm using numerical analysis with metrics for *completion* and *coverage*. The new method is found to be excellent at avoiding invalid states and almost perfect at completing simulations. The most significant improvement to the algorithm comes from adding random and in-place stepping features. These new features also increase the computational performance of the algorithm, as undesired search areas are avoided.

Limitations A simple rectangular geometry is used to validate initial configurations. It might be possible to achieve 100% *all-completed* by using a more complex geometry check. Other metrics could be used to compare the algorithms, such as ground truth similarity.

Future Work Any-time planning algorithms could be used to increase the acceptability of footstep-based steering methods. It is possible to further increase the *coverage* of the algorithm using a method such as [6] to optimize the parameters of the steering algorithm.

References

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

[2] Shawn Singh, Mubbasir Kapadia, Petros Faloutsos, and Glenn Reinman. An open framework for developing, evaluating, and sharing steering algorithms. In *MIG*, pages 158–169, 2009.

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

[4] Mubbasir Kapadia, Alejandro Beacco, Francisco Garcia, Vivek Reddy, Nuria Pelechano, and Norman I. Badler. Multi-domain real-time planning in dynamic environments. SCA '13, pages 115–124, New York, NY, USA, 2013. ACM.

[5] Mubbasir Kapadia, Matthew Wang, Shawn Singh, Glenn Reinman, and Petros Faloutsos. Scenario space: Characterizing coverage, quality, and failure of steering algorithms. In *ACM SIGGRAPH/EG SCA*, 2011.

[6] G. Berseth, M. Kapadia, B. Haworth, and P. Faloutsos. SteerFit: Automated Parameter Fitting for Steering Algorithms. In Vladlen Koltun and Eftychios Sifakis, editors, *Proceedings of ACM SIGGRAPH/EG SCA*, pages 113–122, 2014.