Footstep Navigation for Dynamic Crowds

Shawn Singh Mubbasir Kapadia Glenn Reinman
Petros Faloutsos
University of California, Los Angeles

Abstract

The majority of steering algorithms output only a force or velocity vector to an animation system, without modeling the constraints and capabilities of human-like movement. This simplistic approach lacks control over how a character should navigate. This paper proposes a steering method that uses *footsteps* to navigate characters in dynamic crowds. Instead of an oriented particle with a single collision radius, we model a character's center of mass and footsteps using a 2D approximation of an inverted spherical pendulum model of bipedal locomotion. We use this model to generate a timed sequence of footsteps that existing animation techniques can follow exactly. Our approach not only constrains characters to navigate with realistic steps but also enables characters to intelligently control subtle *navigation* behaviors that are possible with exact footsteps, such as side-stepping. Our approach can navigate crowds of hundreds of individual characters with collision-free, natural steering decisions in real-time.

1 Introduction

The majority of previous steering algorithms represent a character as an oriented particle that moves by choosing a force or velocity vector. Often, orientation is heuristically chosen to be the particle's velocity. This approach has the two key disadvantages:

Limited locomotion constraints: Most steering algorithms do not account for constraints of real human movement. Trajectories may have discontinuous velocities, oscillations, awkward orientations, or may try to move a character unnaturally, and these side-effects make it harder to animate the character intelligently.

Limited navigation control: It is common to assume that an animation system will know how to interpret a vector-based steering decision. In practice, a vector does not have enough information to indicate appropriate maneuvers, such as side-stepping versus reorienting the torso, stepping backwards versus turning around, planting a foot to change momentum quickly, or carefully placing steps in exact locations.

We propose to generate sequences of footsteps as the output of navigation. Since there are already several animation techniques that can animate a character to follow timed footsteps exactly, *e.g.* [1, 2, 3, 4, 5, 6], the main challenge and focus of our work is how to *generate* footsteps as the output of navigation. Footsteps are an intuitive abstraction for most locomotion tasks, and they provide precise, unambiguous spatial and timing information to animation.

In our system, each step is defined by a 2D parabolic trajectory that approximates the motion of a 3D inverted pendulum. The location, orientation, and timing of footsteps are derived from the these trajectories. We use a best-first search to plan a sequence of space-time parabolic trajectories and the associated footsteps that avoids time-varying collisions, satisfies footstep constraints for natural locomotion, and minimizes the effort to reach a local goal. Characters successfully avoid collisions with each other and choose steps that correspond to natural and fluid motion, including precise timing. Because the most significant biomechanics constraints are already taken into

account in our model, integrating our results with an existing animation algorithm that follows footsteps is straightforward and results in navigation that is often richer and less awkward than vector-based navigation.

Contributions. This paper presents a new approach to steering in dynamic crowds that uses a simple biomechanically-based footstep model combined with space-time planning. Our work demonstrates that a steering algorithm can have better navigation features than a vector interface, while still retaining fast performance. These features include: short-term space-time planning, dynamic collision bounds, appropriate movement constraints, and more precise navigation control. Because substantial work already exists to animate characters to follow exact footsteps including timing information, we focus on the navigation: how to generate biomechanically plausible footsteps for dynamic crowds.

2 Related Work

Two widely accepted strategies are (1) the social forces model [7], which associates a small force field around agents and obstacles, and (2) the steering behaviors model [8], where forces are procedurally computed to perform desired functions such as seek, flee, pursuit, evasion, and collision avoidance. Many works are extensions or elaborations of these two ideas, e.g., [9, 10, 11, 12, 13, 14, 15]. A more complete survey of collision avoidance, navigation, and crowd simulation work can be found in [16]. The common theme in these works is the use of force or velocity vectors as navigation decisions, which has the limitations described above.

Only a few steering techniques take into account locomotion constraints that an animation system will have. Paris and Donikian [17] demonstrate a framework where the animation module can potentially tell steering that an action is not plausible. Musse and Thalmann [18] and Shao and Terzopoulos [19] both address higher-level aspects of pedestrians, and their navigation modules output a choice from a set of navigation behaviors that correspond directly to animations the character can produce. Van Basten and Egges [20] discuss problems of interfacing navigation with animation, proposing abstractions that reduce such discrepancies.

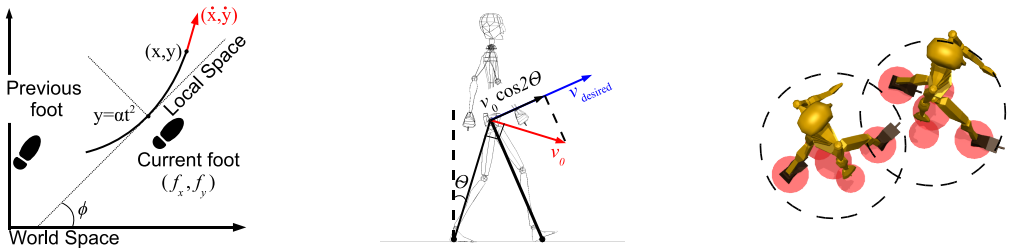
Another approach to navigation is to plan sequences of motion clips, *e.g.*, [21] demonstrated this is possible in real-time for crowds, by precomputing a tree of all possible sequences of motion clips. However, a large number of motion clips would be needed to emulate the versatility of far fewer stepping options. The technique of precomputing a search tree can also be applied to our footstep planner, but our approach is scalable even without this extension.

Footsteps. Several animation techniques, academic and commercial, can *follow* a given sequence of footsteps [1, 2, 3, 4, 5, 6], and more. Animation methods in these works include forward and inverse kinematics, physically based control, and motion capture.

The challenge of *generating* footsteps has so far only been explored for single characters in static environments. Research in robotics [22, 23, 24, 25, 26] explores autonomous footplacement to avoid obstacles while navigating towards a goal. Their focus is practical robot control, and so they do not consider issues of real human locomotion. Torkos and Van de Panne [1] generate footsteps to randomly wander, changing direction if nearby objects are too close, used to demonstrate their animation system. Chung and Hahn [2] input a trajectory, and generate footsteps by aligning eacj step to the orientation of the trajectory, with smaller footsteps around curves. Choi et al. [27] use roadmaps to plan sequences of steps, choosing from steps that are possible with the given motion clips and requiring costly roadmap construction and footstep verification. [28] propose a hierarchical planning approach that computes full-body motion including footsteps for tasks in highly constrained environments. Recently several papers have considered footsteps as a way of guiding controllers for physically-based character animations [3, 29].

3 Footstep Model

The primary data structure in our model is a *footstep*, which includes: (1) the position, velocity, and timing of the character's center of mass trajectory, (2) the location and orientation of the foot, and (3) the cost of taking the step. In this



(a) (b) (c)

Figure 1: Our footstep model. **(a)** Depiction of state and action parameters. **(b)** A sagittal view of the pendulum model used to estimate energy costs. **(c)** The collision model uses 5 circles that track the torso and feet over time, allowing tighter configurations than a single coarse radius.

section, we describe these aspects of a footstep, as well as the constraints for choosing footsteps.

Center of mass trajectory. The analogy between human locomotion and the inverted pendulum is well known [30]; the pendulum pivot represents a point on or near a footstep, while the pendulum mass represents a character's center of mass. We define a 2D analytical approximation to the dynamics of an inverted spherical pendulum using parabolas. Piecewise parabolic curves are enough to capture the variety of trajectories that a human's center of mass will have: varying curvature, speed, and step sizes. Each step is a parabola defined with the following parameters in local space:

$$(x(t), y(t), \dot{x}(t), \dot{y}(t)) = \left(v_{x_0} t, \alpha t^2, v_{x_0}, 2\alpha t\right), \quad (1)$$

such that both v_{x_0} and α are positive.

Equation 1 allows us to *analytically* evaluate the position and velocity of a character's center of mass at any time t. This makes it practical to search through many possible trajectories for many characters in real-time.

3.1 Footstep actions

The state of the character $s \in \mathcal{S}$ is defined as follows (Figure 1a):

$$s = \{(x, y), (\dot{x}, \dot{y}), (f_x, f_y), f_\phi, I \in \{\mathsf{L}, \mathsf{R}\}\},$$

where (x, y) and $(\dot{x}, \dot{y})$ are the position and velocity of the center of mass of the character at the *end* of the step, (f_x, f_y) and f_ϕ are the location and orientation of the foot, and I is an indicator of which foot (left or right) is taking the step. The state space $\mathcal{S}$ is the set of valid states that satisfy the constraints described below.

A *footstep action* determines the next parabolic trajectory, defined as $a \in \mathcal{A}$:

$$a = \{\phi, v_{\text{desired}}, T\},$$

where ϕ is the desired orientation of the parabola, v_{desired} is the desired initial speed of the center of mass, and T is the desired time duration of the step. The action space $\mathcal{A}$ is the set of valid footstep actions, where the input and output states are both valid. Note that when the character's previous step is fixed, varying ϕ directly affects the width of the parabolic trajectory, thus allowing a large variety of step choices.

A key aspect of the model is the transition function, s'= **createFootstep**(s, a). This function receives a desired footstep action a and a state s and returns a new state s' if the action is valid. It is implemented as follows. First, ϕ, which indicates the orientation of the parabola, is used to compute a transform from world space to local parabola space. Then, the direction of velocity $(\dot{x}, \dot{y})$ from the end of the previous step is transformed into local space, normalized, and re-scaled by the desired speed v_{desired}. With this local desired velocity, there is enough information to solve for α, and then Equation 1 is used to compute (x, y) and $(\dot{x}, \dot{y})$ at the end of the next step. In local space, the foot location is always located at $(f_x, f_y) = (0, -d)$, where d is describes the distance between a character's foot and center at rest. Finally, all state information is transformed back into world space, which serves as the input to create the next footstep.

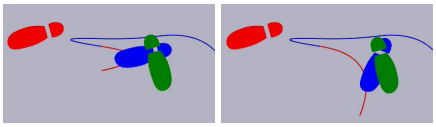


Figure 2: An interval of valid foot orientations (the blue and green feet) is maintained for each step, constrained by the previous step (red foot) and the chosen trajectory (red line).

3.2 Locomotion constraints

Biomechanical properties. Several properties of human locomotion are automatically enforced by the definition of our model. The piecewise parabola will be G-1 continuous, and the center of mass will remain between the two feet by enforcing the local-space parabola remains positive. Our footstep model offers a number of intuitive parameters with meaningful defaults and well-defined physical meaning. These parameters include the height of the character's center of mass, the min, max, and preferred step timing and stride length, the preferred and max velocities of the character's walk, the interval of valid foot orientations, et al. If these constraints are violated, the footstep is considered invalid. A user can modify these parameters to create new locomotion styles. For example, restricting the valid range of step timing and output velocity for one foot results in asymmetric limping, like an injured character.

Footstep orientation. Intuitively, it may seem that footstep orientations must be an additional control parameter when creating a footstep. However, the choice of footstep orientation has no direct effect on the dynamics of the center of mass trajectory; the foot orientation only constrains the options for current trajectory and future footsteps. This is a key aspect to our model's efficiency – instead of increasing the dimensionality of our search space to include foot orientation, we use orientation to constrain the search space of a lower dimensional system.

To implement this constraint, we compute an interval $[f_{\phi\text{inner}}, f_{\phi\text{outer}}]$ of valid foot orientations. This interval is constrained by the same interval from the previous step, and further constrained by the parabola orientation ϕ used to create the next footstep (Figure 2):

$$[f_{\phi\text{next\_inner}}, f_{\phi\text{next\_outer}}] = [f_{\phi\text{prev\_outer}}, f_{\phi\text{prev\_inner}} + \frac{\pi}{2}] \cap [\phi, \text{atan2}(\dot{y}, \dot{x})].$$

If this intersection becomes an empty set, that implies that no foot orientation can satisfy the step constraints, so the step is invalid. Note the ordering of bounds in these intervals; the next foot's outer bound is constrained by the previous foot's inner bound. In words, the interval $[\phi, \text{atan2}(\dot{y}, \dot{x})]$ describes two constraints: (1) the character would not choose a foot orientation that puts his center of mass on the outer side of the foot, (2) a human would rarely orient the next step more outwards than the direction of momentum; violating this constraint would put the character's center of mass on the wrong side of the foot. The exact orientation is chosen as a fast postprocess, described below.

Space-time collision model. For any given footstep, our model computes the *time-varying* collision bounds of the character at any exact time. To determine if a footstep causes a collision, we iterate over several time-steps within the footstep and query the collision bounds of nearby characters for that time. The collision bounds are five circles, depicted in Figure 1c. Each circle associated with a foot exists while the foot is planted on the ground. The three circles associated with the torso are placed on the center of mass, which moves along the parabola over several time-steps. If any of these circles collide with an obstacle or another character's circles, the footstep is considered invalid.

3.3 Cost function

We define the cost of a given step as the energy spent to execute the footstep action. We model three forms of energy expenditure for a step: (1) ΔE_1, a fixed rate of energy the character spends per unit time, (2) ΔE_2, the work spent due to ground reaction forces to achieve the desired speed, and (3) ΔE_3, the work spent due to ground reaction forces accelerating the center of mass along the trajectory. The total cost of a footstep action transitioning a character from s to s' is given by:

$$c(s, s') = \Delta E_1 + \Delta E_2 + \Delta E_3. \qquad (2)$$

Fixed energy rate. The user defines a fixed rate of energy spent per second, denoted as R. For

each step, this energy rate is multiplied by the time duration of the step T to compute the cost:

$$\Delta E_1 = R \cdot T. \quad (3)$$

This cost is proportional to the the amount of time it takes to reach the goal, and thus minimizing this cost corresponds to the character trying to minimize the time it spends walking to his goal. We found that good values for R are roughly proportional to the character's mass.

Ground reaction forces. As a character pushes against the ground, the ground exerts equal and opposite forces on the character. We model three aspects of ground reaction forces that are exerted on the character's center of mass, from the study of biomechanics. The geometry and notation of the cost model is shown in Figure 1b. First, at the beginning of a new step (heel-strike), some of the character's momentum dissipates into the ground. We estimate this as an instantaneous loss of momentum along the pendulum shaft, reducing the character's speed from v_0 to $v_0 \cos(2\theta)$. In order to resume a desired speed, the character actively exerts additional work on his center of mass, computed as:

$$\Delta E_2 = \frac{m}{2} \left| (v_{\text{desired}})^2 - (v_0 \cos(2\theta))^2 \right|. \quad (4)$$

This cost measures the effort required to choose a certain speed. At every step, some energy is dissipated into the ground, and if a character wants to maintain a certain speed, it must actively add the same amount of energy back into the system. On the other hand, not all energy dissipates from the system after a step, so if the character wants to come to an immediate stop, the character also requires work to remove energy from the system. Minimizing this cost corresponds to finding footsteps that require less effort, and thus tend to look more natural. Furthermore, when walking with excessively large steps, $\cos(2\theta)$ becomes smaller, implying that more energy is lost per step.

It should be noted that there is much more complexity to real bipedal locomotion than this cost model. For example, the appropriate bending of knees and ankles and the elasticity of human joints can significantly reduce the energy lost per step, reducing the required work for a real human. While the model is not an accurate measurement of energy spent, it is sufficient for *comparing* the effort of different steps.

ΔE_2 captures only the cost of changing a character's momentum at the beginning of each step. The character's momentum may also change during the trajectory. For relatively straight trajectories, this change in momentum is mostly due to the passive inverted pendulum dynamics that requires no active work. However, for trajectories of high curvature, a character spends additional energy to change his momentum. We model this cost as the work required to change momentum (denoted as P) over the length of the step, weighted by constant w:

$$\Delta E_3 = w \cdot \frac{d\text{P}}{dt} \cdot \text{length} = w \cdot m\alpha \cdot \text{length}, \quad (5)$$

Note that α is the same coefficient in Equation 1, the acceleration of the trajectory. α increases if the curvature of the parabola is larger, and also if the speed of the character along the trajectory is larger. Minimizing this cost corresponds to preferring straight steps when possible, and preferring to go slower (and consequently, taking smaller steps) when changing the direction of momentum significantly. The weight w can be adjusted to change whether it costs more energy to walk around an obstacle or to stop and wait for the obstacle to pass. We found good values of w to be between 0.2 and 0.5, meaning that twenty to fifty percent of the curvature is due to the character's active effort, and the rest due to the passive inverted pendulum dynamics.

4 Generating Sequences of Steps

Discretizing action space. The choices for a character's next step are generated by discretizing the action space $\mathcal{A}$ described above, in all three dimensions and using the **createFootstep**(s, a) function to compute the new state and cost of each action. We have found that v_{desired} and T can be discretized extremely coarsely, as long as there are at least a few different speeds and timings. Further optimizations are made by observing that speed v_{desired} and step timing T have a slight inverse correlation, and so not all combinations of v_{desired} and T need to be generated. Most of the complexity of the action space lies in the choices

for the parabola orientation, ϕ. The choices for ϕ are defined relative to the orientation of the velocity vector $(\dot{x}, \dot{y})$ from the end of the previous footstep, and the discretization of ϕ ranges from almost straight to almost U-turns. We note that the first choice that real humans would consider when navigating is to step directly towards the local goal. To address this, we create a special option for ϕ that would orient the character directly towards its goal. With this specialized goal-dependent option, we found it was possible to give fewer fixed options for ϕ, focusing on larger turns. Without this option, even with a large variety of choices for ϕ, the character appears to steer towards an offset of the actual goal and then takes an unnatural corrective step.

Short-horizon best-first search. We use a best-first search planner for a sequence of footsteps that minimizes energy cost. The implementation of our planner is the same as an A* search, except for the *horizon*, described below.

The cost of taking a step is computed using Equations 2-5. The heuristic function used by the best-first search, $h(s)$, estimates the energy cost from the current state to a local goal:

$$h(s) = \mathsf{c}_{expected} \times n, \tag{6}$$

where $\mathsf{c}_{expected}$ is the energy spent in taking one normal footstep action based on the character's user-defined parameters, and n is the number of steps it would take to travel directly to the goal.

The *horizon* of our planner is the maximum number of nodes to be expanded for a single search. In most cases, a path is found before this threshold. We limit the horizon so that difficult or unsolvable situations will not cause a significant delay. If the planner searches too many nodes without reaching the goal, we instead construct a path to a node from the closed list that had the best heuristic value (the same closed list used in A*). Intuitively, this means that if no path is found to the goal within the search horizon, the planner returns a path to the reachable state that had the most promise of reaching the goal. The short-horizon approach guarantees that we will have at least some path for the character to use, even in difficult or unsolvable planning problems. In worst case, if no good solution is found, the path will simply be a sequence of "stop" actions. For example, this can occur when a character is stuck dense environment. Eventually when the density clears, the character will continue.

Local goals and collision avoidance. To navigate through large environments, we first plan a path using A* (a traditional spatial path, not footsteps). Whenever a character needs to plan more footsteps, a local footstep goal is chosen, placed approximately 10 meters ahead on the spatial path. This 10-meter requirement is not strict; we experimented with other methods of choosing a local footstep goal, and they all worked decently well. Characters that are visible to each other can read each other's plans in order to predict their dynamic collision bounds at any given time. Visibility is determined by (1) having line-of-sight between the two characters, and (2) being within the character's visual field, modeled as a hemisphere centered around the character's forward-facing direction. This knowledge is analogous to the unspoken communication that occurs between real human pedestrians that makes human steering very robust. When a character re-plans, it does not try to avoid characters that it does not see, and therefore other characters, who are still executing old plans, may collide. The number of collisions can be drastically reduced by re-planning n steps in advance, before the previous plan is fully completed. This way there is always a "buffer" of 2 or 3 steps that are guaranteed to be correct when a character predicts how to steer around another character. While deadlocks and collisions are still possible with this scheme, collisions are very rare, and we have not yet encountered a deadlock in our experiments.

Choosing exact footstep orientation. As described above, the planner maintains an interval of valid foot orientations for every step, constrained by the previous step's interval, as well as the trajectory of the current step. Once a sequence of footsteps has been planned, it is possible to choose exact footstep orientations. We constrain the interval of valid orientations once more using the *next* step's trajectory, now that this information is available. This computation relies on the same interval arithmetic described in Section 3. It is easy to see by contradiction that this process will not cause an invalid interval of orientations: if the interval becomes invalid during this postprocess, that would imply

that no orientation of the current step could have produced a valid interval of the next step – but if this is true, that option would have already been pruned during planning and would not be encountered here. The exact orientation can be any value within this final interval; we found a good heuristic is to orient the foot as closely as possible to the orientation of the step's trajectory, with a special case for large turns.

5 Results

For most results, characters are modeled with a center of mass 1 meter above the ground, with a step length between 0.1 meters and 1.0 meters, step timing between 0.2 seconds and 0.8 seconds, and torso width of 60 cm.

Our short-horizon planner can solve challenging situations such as potential deadlocks in narrow spaces. Figure 3 depicts a challenging doorway situation. In many previous algorithms, characters would "fight" at the doorway and may reach deadlock. In our method, the characters exhibit predictive cooperation, where one character steps aside. The doorway, 70 cm wide, is barely wide enough to fit a single pedestrian. In this tight situation, vector-based techniques would rely on collision prevention at the walls until the character eventually finds the door.

Our collision model allows tighter spacing in crowded conditions. An example is shown in Figure 3, where a group of characters squeeze through a glass door. With a single coarse collision radius, there would be many false-positive collisions. Instead, like real humans, these characters are comfortable placing their feet and shoulders close to others in the dense crowd.

Our planner works online, in real-time. Performance is shown in Table 1, measured on a Core 2 processor, using a single thread. Planning is fast is because of the scope of footsteps: a short horizon plan of 5-10 footsteps takes seconds to execute but only a few milliseconds to compute. The amortized cost of updating a character at 20 Hz is also shown in Table 1.

	Egress 50 agents	2-way hall 200 agents	700 boxes 500 agents
Avg. # nodes generated	137	234	261
Avg. # nodes expanded	82	190	192
Planner performance	1.6 ms	4.4 ms	3 ms
Amortized cost 20 Hz	0.037 ms	0.1 ms	0.11 ms

Table 1: Performance of our footstep planner for a character. The typical worst case plan generated up to 5000 nodes and expanded about 3000 nodes.

6 Discussion and Future Work

Footsteps are an appropriate form of control since they are the major contact point between a bipedal system and the external environment. By generating space-time sequences of footsteps, and by considering tighter dynamic collision bounds, our approach is able to control characters more precisely than existing crowd navigation techniques.

A "stop" step is a specialized action in our planner. Being based on general planning, our technique can extend to use other specialized actions, such as running, jumping, even motion capture clips, as long as the action has well defined transitions, costs, and constraints. Existing steering techniques can also be emulated, for example, social forces models can be mapped to cost functions used by our planner.

There are some prominent aspects of bipedal locomotion which should be addressed in future work. Knee joints, ankle joints, muscles, angular momentum, and the center of pressure (pendulum pivot) shifting from heel-to-toe during a step – all of these affect the energy cost of real footsteps. We would also like to explore social and cognitive costs, where a character's objective may not necessarily be to minimize effort.

Acknowledgements

We would like to thank Intel Corp. for their generous support through equipment and grants.

References

[1] Michiel Van de Panne. From footprints to animation. *Computer Graphics Forum*, 16(4):211–223, 1997.

[2] Shih-Kai Chung and J.K. Hahn. Animation of human walking in virtual environments. In *Computer Animation*, pages 4–15, 1999.

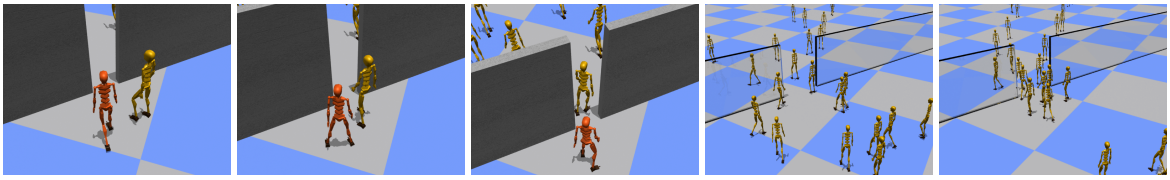


Figure 3: (Left) A character side-steps and yields to the other pedestrian, then precisely navigates through the narrow doorway. (Right) An egress simulation. Characters do not get stuck around the corners of the glass door.

[3] Stelian Coros, Philippe Beaudoin, Kang Kang Yin, and Michiel van de Panne. Synthesis of constrained walking skills. *ACM Trans. Graph.*, 27(5):1–9, 2008.

[4] Chun-Chih Wu, Jose Medina, and Victor B. Zordan. Simple steps for simply stepping. In *ISVC (1)*, pages 97–106, 2008.

[5] Ben van Basten and Arjan Egges. The stepspace: Example-based footprint-driven motion synthesis. Wiley, 2010.

[6] Autodesk. 3ds max, 2010.

[7] Dirk Helbing and Péter Molnár. Social force model for pedestrian dynamics. *Phys. Rev. E*, 51(5):4282–4286, May 1995.

[8] Craig Reynolds. Steering behaviors for autonomous characters, 1999.

[9] N. Pelechano, J. M. Allbeck, and N. I. Badler. Controlling individual agents in high-density crowd simulation. In *SCA*, pages 99–108. Eurographics Association, 2007.

[10] Russell Gayle, Avneesh Sud, Erik Andersen, Stephen J. Guy, Ming C. Lin, and Dinesh Manocha. Interactive navigation of heterogeneous agents using adaptive roadmaps. *IEEE Trans. Vis. Comput. Graph.*, 15(1):34–48, 2009.

[11] Ronan Boulic. Relaxed steering towards oriented region goals. In *MIG'08*, pages 176–187, 2008.

[12] Jur P. van den Berg, Ming C. Lin, and Dinesh Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *ICRA*, pages 1928–1935, 2008.

[13] Fabrice Lamarche and Stéphane Donikian. Crowd of virtual humans: a new approach for real time navigation in complex and structured environments. *Comput. Graph. Forum*, 23(3):509–518, 2004.

[14] Sébastien Paris, Julien Pettré, and Stéphane Donikian. Pedestrian reactive navigation for crowd simulation: a predictive approach. In *EUROGRAPHICS 2007*, volume 26, pages 665–674, 2007.

[15] F. Feurtey. Simulating the collision avoidance behavior of pedestrians. Master's thesis, The University of Tokyo, School of Engineering, 2000.

[16] Norman Badler. *Virtual Crowds: Methods, Simulation, and Control*. Morgan and Claypool Publishers, 2008.

[17] Sébastien Paris and Stéphane Donikian. Activity-driven populace: A cognitive approach to crowd simulation. *IEEE Computer Graphics and Applications*, 29(4):34–43, 2009.

[18] S.R. Musse and D Thalmann. A Model of Human Crowd Behavior. In *Proc. CAS'97, Springer Verlag, Wien*, pages 39–51, 1997.

[19] Wei Shao and Demetri Terzopoulos. Autonomous pedestrians. In *SCA*, pages 19–28, 2005.

[20] Ben J. H. van Basten and Arjan Egges. Path abstraction for combined navigation and animation. *MIG'09*, 5884/2009:182–193, 2009.

[21] Manfred Lau and James J. Kuffner. Precomputed search trees: Planning for interactive goal-driven animation. In *SCA*, pages 299–308, September 2006.

[22] Jr. Kuffner, J.J., K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue. Footstep planning among obstacles for biped robots. In *IEEE Intelligent Robots and Systems (IEEE/RSJ)*, volume 1, pages 500–505, 2001.

[23] K.h. Nishiwaki, T. Sugihara, S. Kagami, M.y. Inaba, and H. Inoue. Online mixture and connection of basic motions for humanoid walking control by footprint specification. In *ICRA*, volume 4, pages 4110–4115, 2001.

[24] James Kuffner, K. Nishiwaki, Satoshi Kagami, Y. Kuniyoshi, M. Inaba, and H. Inoue. Online footstep planning for humanoid robots. In *ICRA*. IEEE, September 2003.

[25] Tsai-Yen Li, Pei-Feng Chen, and Pei-Zhi Huang. Motion planning for humanoid walking in a layered environment. In *ICRA*, volume 3, pages 3421–3427, 2003.

[26] Joel Chestnutt, Manfred Lau, Kong Man Cheung, James Kuffner, Jessica K Hodgins, and Takeo Kanade. Footstep planning for the honda asimo humanoid. In *ICRA*, April 2005.

[27] Min Gyu Choi, Jehee Lee, and Sung Yong Shin. Planning biped locomotion using motion capture data and probabilistic roadmaps. *ACM Trans. Graph.*, 22(2):182–203, 2003.

[28] Liangjun Zhang, Jia Pan, and Dinesh Manocha. Motion planning and synthesis of human-like characters in constrained environments. *MIG'09*, 5884/2009:138–145, 2009.

[29] Jia-chi Wu and Zoran Popović. Terrain-adaptive bipedal locomotion control. *ACM Transactions on Graphics*, 29(4):72:1–72:10, Jul. 2010.

[30] Arthur D. Kuo. The six determinants of gait and the inverted pendulum analogy: A dynamic walking perspective. *Human Movement Science*, 26(4):617 – 656, 2007.