Technical Report #100023: Evaluating the Costs and Constraints Used to Make Footstep Decisions

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

Abstract

The majority of steering algorithms model very few locomotion constraints or capabilities, and output only a force or velocity vector to motion synthesis. This simplistic interface between navigation and motion synthesis lacks control and introduces ambiguities. This paper considers how *footsteps* can be used to navigate in crowds of characters. We propose a novel locomotion model that represents a character's stepping state and footstep actions using a 2D approximation of an inverted spherical pendulum model of bipedal locomotion. We use this model with a short-horizon planner to generate a timed sequence of footsteps that existing motion synthesis techniques can follow exactly. During planning, our locomotion model not only constrains characters to navigate with realistic locomotion, but more importantly, it also enables characters to control finer-grain subtle *navigation* behaviors that are possible with exact footsteps. Our approach can navigate crowds of hundreds of individual characters with collision-free, natural behaviors in real-time.

Keywords: steering behaviors, navigation, autonomous agents

1 Introduction

Autonomous characters that navigate in dynamic environments are an essential part of any living, breathing virtual world. Such characters are required in games, movies, urban planning simulations, training simulators, and architecture visualizations. Despite many advancements in crowd simulation and steering algorithms, autonomous characters still lack intelligent *local* steering interactions in dynamic crowds of characters. Intelligent local interactions require that characters can step in a variety of ways, have tighter, dynamically changing collision bounds, and exhibit the human-like ability to understand exactly how they can steer through a dynamic environment in the immediate short-term future, *i.e.*, local space-time planning, not just prediction. Very few steering algorithms have considered these aspects, and no previous algorithm has demonstrated the synergy of all three aspects combined.

The root of this problem is the limited types of actions that have been used to make steering decisions – more generally, the limited *locomotion model*. In our context, the locomotion model refers to the state and action spaces, constraints, and costs *used for making steering decisions*, which is different from the locomotion used by the underlying motion synthesis for animations. The vast majority of steering algorithms use one of two locomotion models: (1) an oriented particle model, where the character locomotes by choosing a force or velocity vector, or (2) a discrete set of locomotion actions where the character locomotes by choosing one of the actions. These locomotion models have the following disadvantages:

- *Limited locomotion constraints:* Very few navigation algorithms account for locomotion constraints. Trajectories may have discontinuous velocities, oscillations, awkward orientations, or may try to move a character during the wrong animation state, and these side-effects make it harder to animate the character intelligently. For example, a character moving forward cannot easily shift momentum to the right when stepping with the right foot, and a character would rarely side-step for more than two steps at a steady walking speed.
- *Limited navigation control:* Existing steering algorithms usually assume that motion synthesis will automatically know how to obey a vector interface. This is not the case – motion synthesis does not have enough information to choose appropriate subtle maneuvers, such as side-stepping versus reorienting the torso, stepping backwards versus turning around, stopping and starting, planting a foot to change momentum quickly, or carefully placing footsteps in exact locations. These details are critical to depicting a character's local steering intelligence, and thus it is appropriate for steering to have better control.

In this paper, we propose one possible solution to this problem: generating sequences of footsteps as the interface between navigation and motion synthesis. Footsteps, including location, orientation, and timing, are an intuitive and representative abstraction for most locomotion tasks. Footsteps offer finer control which allows navigation to choose more natural and nuanced actions. At the same time, the series of footsteps produced by navigation communicates precise, unambiguous spatial and timing information to motion synthesis. Since there exist several motion synthesis algorithms that can animate a character to follow timed footsteps exactly [Girard 1987; Ko and Badler 1992; Van de Panne 1997; Torkos and van de Panne 1998; Chung and Hahn 1999; Coros et al. 2008; Donikian 2009], the main challenge and focus of our work is how to *generate* footsteps as the output of a navigation algorithm.

We develop a novel locomotion model (used for making steering decisions, not animations) that represents a character's stepping state and possible footstep actions. Given this model, we use a space-time planning approach to dynamically generate a short-horizon sequence of footsteps. The output of our system is an efficient sequence of space-time footsteps that avoids time-varying collisions, satisfies footstep constraints for natural locomotion, and minimizes the effort to reach a goal. In its most general form, this is a nonholonomic optimization planning problem in continuous space where the configuration and action spaces must be dynamically re-computed for every plan. This is generally acknowledged to be a difficult type of planning problem [LaValle 2006]. However, by (a) using an *analytic* approximation of inverted pendulum dynamics, (b) allowing near-optimal plans, (c) exploiting domain-specific knowledge, and (d) searching only a limited horizon of nodes, our approach generates natural footsteps in real-time for large crowds of characters. Characters successfully avoid collisions with each other and produce dynamically stable footsteps that correspond to natural and fluid motion, with precise timing constraints.

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

Because the most significant biomechanics and timing constraints are already taken into account in our navigation locomotion model, integrating our results with an existing motion synthesis alorithm that follows footsteps is straightforward and results in animations that are often richer and less awkward than previous approaches.

Contributions. This paper presents a new approach to steering in dynamic crowds that uses space-time planning combined with a biomechanically-based footstep locomotion model. To our knowledge, this is the first crowd steering algorithm to generate footsteps and to use a locomotion model that is more detailed than the oriented particle model or discrete animation actions model. Our footstep planning approach enables characters to steer with natural variety, but also with spatial precision and temporal precision. All relevant biomechanical constraints are accounted for, such as stride length, foot orientation, momentum, center of mass, and step timing, to ensure that the output footsteps are biomechanically plausible. Our approach also incorporates *dynamic* collision bounds that are tighter than a single coarse collision radius. We also describe appropriate energy costs that allow characters to choose natural, intelligent footsteps for a variety of challenging steering situations. The amortized performance cost of planning a short sequence of footsteps is extremely low, which allows us to simulate large crowds of characters in real-time on a single thread with natural, collision-free, locally intelligent behaviors.

2 Related Work

Prior work in steering. A large amount of existing work has already successfully modeled various phenomena of pedestrian and crowd steering. Only a few representative works are described here, with emphasis on the locomotion models used. There are many more works on agent navigation, collision avoidance, global planning, and crowd simulation that are not listed here [Badler 2008].

Reynolds [Reynolds 1987] pioneered group and crowd behaviors in graphics by introducing the boids behavioral model. In addition to its other influences, this was one of the earliest works to demonstrate the *particle abstraction* - each agent represented as a particle that moves with a force or velocity vector. It quickly became a standard way to modularize the navigation and animation challenges from each other.

With this simplified model of an agent's locomotion capabilities, the navigation task is essentially a question of how to choose forces (or velocities) intelligently. Two widely accepted strategies are (1) the social forces model [Helbing and Molnár 1995], which associates a small force field around agents and obstacles, and (2) the steering behaviors model [Reynolds 1999], 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. The HiDAC framework [Pelechano et al. 2007] adds many carefully chosen psychological and social forces to support crowds of agents. Gayle et al. [Gayle et al. 2009] use adaptive roadmaps to plan navigation decisions on top of a social force model. Loscos et al. [Loscos et al. 2003] define specific rules that determine how an agent steers to avoid collisions. Boulic [Boulic 2008] extends the work Reynolds to consider the orientation of agents more carefully as well as supporting more general goals. Velocity obstacles [Fiorini and Shiller 1998] and reciprocal velocity obstacles [den Berg et al. 2008] focus on a mathematically elegant collision avoidance procedure that computes steering forces. Lamarche and Donikian [Lamarche and Donikian 2004] demonstrate a more complete agent steering framework, incorporating global planning and local reactive steering, outputting a velocity vector. Treuille et al. [Treuille et al. 2006] compute velocities of all agents globally, able to capture a variety of macroscopic crowd phenomena.

Several recent steering techniques have enhanced the vector based locomotion model in various ways. Goldenstein et al. [Goldenstein et al. 2001] use separate dynamical systems models to compute turning and speed of each agent. Kapadia et al. [Kapadia et al. 2009] use locally computed fields to determine speed and orientation separately. Another interesting enhancement is to use empirical data of real crowds. Paris et al. [Paris et al. 2007] use space-time predictive collision avoidance that is tuned based on real data. Lee et al. [Lee et al. 2007] and Lerner et al. [Lerner et al. 2007] both use a database of trajectories from real crowds to compute steering decisions. Note that even though Lee et al. use motion graphs to synthesize motions, the locomotion model used to make steering decisions is only a data-driven particle model.

A few steering techniques take into account the locomotion that an animation system will actually be capable of. Paris and Donikian [Paris and Donikian 2009] demonstrate a more complete framework for virtual characters, using a similar velocity vector output to an animation module. However, in their work, animation is part of the closed-loop system, and can potentially tell steering that an action is not plausible. Musse and Thalmann [Musse and Thalmann 1997] demonstrated psychological and social aspects of navigation, and they use a combination of vector-based steering and steering from a set of locomotion behaviors. Shao and Terzopoulos [Shao and Terzopoulos 2005] model many layers of autonomous pedestrians from cognition to navigation, and the steering decision is chosen from a set of parameterized navigation behaviors that correspond to animations the character can produce. Zhang et al. [Zhang et al. 2009b] propose a velocity-based model that takes into account locomotion properties from data collected about real human locomotion. Van Baster and Egges [van Baster and Egges 2009] discuss the problems that arise when interfacing navigation with motion synthesis, for example, the problem of using the pelvis as an exact reference point between navigation and motion synthesis is often problematic since the pelvis naturally oscillates during locomotion. They propose various abstractions that reduce such discrepancies. In comparison to these works, by using footsteps, we achieve properly constrained locomotions without limiting characters to a smaller set of motions, and using footseps as the interface to motion synthesis avoids the need for corrective abstractions between navigation and motion synthesis.

Nearly all the above works, whether using a vector-based particle model or a set of steering choices, have some fundamental limitations. The orientation of characters (if it is modeled at all) is always facing the direction of the agent's movement. As such, previous crowd simulations have not demonstrated natural u-turns or side-steps, and many local interactions between navigating characters cause too much turning while agents "fight" to get around each other. Second, previous crowd steering approaches have difficulty with tight spaces, where precise steering is necessary. Finally, the collision bounds of characters is usually a single coarse radius, which forces characters to remain artificially sparse. These issues are critical for characters to appear intelligent when viewed up close.

Locomotion models in other fields. There are many fully 3D locomotion models used in the field of motion synthesis and animation. This includes physically based models, *e.g.*, [Faloutsos et al. 2001; Hodgins et al. 1995], kinematic and data-driven approaches, *e.g.*, [Sturman 1994; Gleicher 1998; Gleicher 2001; Kovar et al. 2002], and detailed musculo-skeletal models, *e.g.*, [Lee et al. 2009].

Of particular interest in this context are motion planning techniques, which often overlap with pedestrian navigation; only a few are listed here. Treuille et al. [Treuille et al. 2007] demon-

strated low-dimensional controllers for motion synthesis to obey a vector command interface. Their work did demonstrate heuristic navigation behavior for dynamic crowds, but this type of navigation still has all the same problems as the vector interface described above, and increases the dimensionality of their approach. [Lau and Kuffner 2005] demonstrated crowds of real-time characters that plan sequence of states that correspond to motion clips. While their planning approach is similar to ours, their approach is essentially based on discrete animations, and such approaches would need far too many actions to cover the finer-grain space of footsteps. For only navigation, this full-body approach is unnecessary. Some works have also proposed parametrized motion synthesis techniques, however we note that such parameterization is essentially an "interface" between a controller and the underlying synthesis, and a vector-based parameterization suffers the same problems as described above.

Our work could be considered a parameterized motion planning approach, but the output of our system is a sequence of footsteps, not motion clips. By keeping navigation and motion synthesis separate, the dimensionality of both navigation and animation tasks remains lower, making it possible to scale footstep planning to hundreds, even thousands of characters in the space-time domain. This also makes it tractable to consider the continuous space of possible footsteps if necessary, instead of only supporting a limited discrete set of actions. At the same time, planning footsteps allows navigation to control animations better than a vector interface.

To improve performance of motion planning techniques, Lau and Kuffner [Lau and Kuffner 2006] demonstrated a technique of precomputing the search space, allowing an online process to quickly form space-time plans. They demonstrated crowds of characters able to navigate in dynamic environments using a state machine of discrete motion clips. It is not clear how their approach would scale when increasing the variety of motion clips, necessary for locally intelligent steering. Additionally, they use only spatial distance as the cost of actions for their space-time planner, which means that characters would try to minimize path length, but not path time, and this may cause unnatural results in tight or challenging situations such as doorways. Their precomputation methodology could be applied to our footstep planner as well, but so far our approach is scalable even without this extension.

While example-driven locomotion models are very popular in animation, biomechanics has extensively studied analytical models intended to capture the subtleties of human locomotion. For normal human walking, the two most common models are the six determinants of gait model [Saunders et al. 1953] and the inverted pendulum model [Kajita et al. 2001]. The work in [Kuo 2007] compares the kinematic model using six determinants of gait with the inverted pendulum model for bipedal locomotion. Running has been modeled using a foot-spring model [Meghdari et al. 2008]. All these models generally have high dimensionality, complex analytical representations, and are impractical models for crowd simulations. Our approach is to simplify an inverted spherical pendulum model, keeping the aspects that are necessary to make decisions about where to place footsteps for navigation, and removing other complexities related to biomechanics and animation.

Footsteps. Several motion synthesis approaches have been demonstrated to follow a given sequence of footsteps. The work of [Girard 1987] addresses how a user can specify desired animations and how to integrate this with natural limb motion models, and serves as the basis of modern bipedal animation tools [Autodesk 2010]. The work in [Van de Panne 1997; Torkos and van de Panne 1998] generates animations from footprints for bipeds and quadrupeds. Combining motion clips and biomechanical constraints, [Wu et al. 2008] proposes a simple and effective method for synthesizing controlled stepping motions. StepSpace [van Basten and Egges 2010]is a recent example-based method that can follow footsteps with precise timing constraints. Biomechanical and gait analysis has also been used in the development of models that follow footsteps naturally [Chung and Hahn 1999]. FootSee [Yin and Pai 2003] supports user-controlled avatars using foot-pressure sensor pads as input.

The challenge of *generating* precise footsteps has been explored in robotics and, to a limited extent, animation. Torkos and Van de Panne [Torkos and van de Panne 1998] generated footsteps to randomly wander, changing direction if nearby objects were too close; this was primarily used as a way to demonstrate their motion synthesis system. Chung and Hahn [Chung and Hahn 1999] generated footsteps by aligning the next step to the orientation of the trajectory, using smaller footsteps around curves. Choi et al. [Choi et al. 2003] use roadmaps to compute sequences of footsteps that are possible with the given motion clips. This technique requires costly roadmap construction and footstep verification preprocessing, and was also demonstrated for only a single character in a static environments. [Coros et al. 2008] demonstrate a physically-based character with the ability to carefully place footsteps in exact locations. [Zhang et al. 2009a] propose a hierarchical planning approach that compute stable footsteps and upper body motion to solve manipulation tasks in highly constrained environments. In all these works, the focus was on animations for individual characters rather than navigation of crowds.

Research in robotics [Chestnutt et al. 2005; Nishiwaki et al. 2001; Kuffner et al. 2003] explores autonomous foot-placement to avoid obstacles while navigating towards a goal. These works, however focus on a single character, and limit the state space to statically stable states. Furthermore, their focus is on practical robot control, and so they do not model the subtle ways that humans use (i.e. choose) footsteps while navigating in dynamic crowded situations, involving many characters.

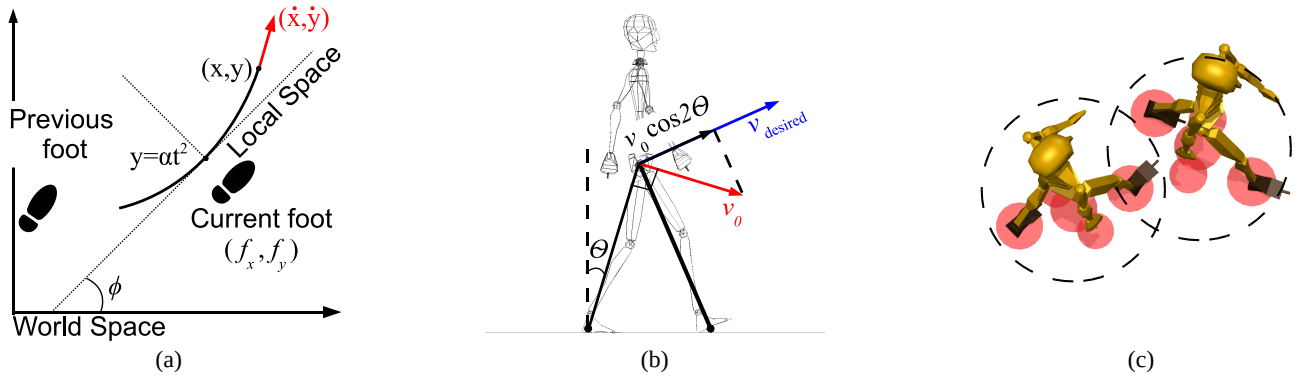
Comparison to our work. Our work demonstrates that a steering algorithm can have better navigation features without adverse effects to performance, including short-term space-time planning, dynamic collision bounds, and a better locomotion model. Unlike many motion planning techniques which focus on high dimensional full-body tasks for a single character, our work focuses on demonstrating large numbers of interacting characters in dynamic crowds. We achieve this by modularizing the navigation and animation tasks with a footstep interface. This makes it possible to keep both tasks low-dimensional without reducing the potential quality. Because enough work already exists to animate characters to follow exact footsteps including timing information, we focus on the steering aspect, how to generate footsteps, which has not been deeply explored yet.

3 Locomotion Model

The primary data structure in our locomotion model is a *footstep*. Each footstep includes the following information: (1) the trajectory of the character's center of mass, including position, velocity, and timing information, (2) the location and orientation of the foot itself, and (3) the cost of taking the step. In this section, we describe how the model computes these aspects of a footstep, as well as the constraints associated with creating a footstep.

3.1 Inverted pendulum model

The analogy between human locomotion and the inverted pendulum is well known; the pivot of the pendulum represents a point on or near a footstep, while the weight of the pendulum represents a character's center of mass. To automatically generate footsteps,



(a) (b) (c)

Figure 1: Our footstep locomotion model. **(a)** Depiction of state and footstep action parameters. The state includes the 2D approximation of an inverted spherical pendulum trajectory. **(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.

we define a 2D approximation to the dynamics of inverted spherical pendulum.

Our approach is inspired by the *linear inverted pendulum* model [Kajita et al. 2001]. In their work, linear equations of a 3D pendulum are derived by constraining the motion of the center of mass to a horizontal 2D plane. Using their model, the passive dynamics of a character's center of mass pivoting over a footstep located at the origin is described by:

$$\ddot{x}(t) = \frac{g}{l}x, \quad (1)$$

$$\ddot{y}(t) = \frac{g}{l}y, \quad (2)$$

where g is the positive gravity constant, l is the length of the character's leg from the ground to the center of mass (*i.e.*, the length of the pendulum shaft), and (x, y) is the time-dependent 2D position of the character's center of mass (*i.e.*, the pendulum weight). It is interesting to note that both x and y have the same dynamics as the basic planar inverted pendulum when using the small-angle approximation, $x \simeq \sin x$. The small-angle approximation is valid for the region of interest; the angle between a human's leg and the vertical axis rarely exceeds 30 degrees.

In this work we further assume that l remains constant – a reasonable approximation within the small-angle region. Then, given initial position (x_0, y_0) and initial velocity (v_{x_0}, v_{y_0}), the solution for $x(t)$ is a general hyperbola:

$$x(t) = \frac{(kx_0 + v_{x_0})e^{kt} + (kx_0 - v_{x_0})e^{-kt}}{2k}, \quad (3)$$

and $y(t)$ has a similar solution, where $k = \sqrt{\frac{g}{l}}$.

We use the second-order Taylor series approximation of this hyperbola:

$$x(t) = x_0 + v_{x_0}t + \frac{x_0k^2}{2}t^2, \quad (4)$$

$$y(t) = y_0 + v_{y_0}t + \frac{y_0k^2}{2}t^2. \quad (5)$$

Finally, this parabola is translated, rotated, reflected, and reparameterized from world space into a canonical parabola in local space (Figure 1a):

$$x(t) = v_{x_0}t, \quad (6)$$

$$y(t) = \alpha t^2, \quad (7)$$

$$\dot{x}(t) = v_{x_0}, \quad (8)$$

$$\dot{y}(t) = 2\alpha t, \quad (9)$$

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

Evaluating the trajectory. Equations 6-9 allow us to *analytically* evaluate the position and velocity of a character's center of mass at any time. The implementation is extremely fast: first, the local parabola and its derivative are evaluated at the desired time, and then the local position and velocity are transformed from the canonical parabola space into world space. This makes it practical to search through many possible trajectories for many characters in real-time.

3.2 Footstep actions

The state of the character $s \in \mathcal{S}$ is defined as follows:

$$s = \langle (x, y), (\dot{x}, \dot{y}), (f_x, f_y), f_\phi, I \in \{\texttt{left}, \texttt{right}\} \rangle, \quad (10)$$

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 of the locomotion model described below.

A *footstep action* transitions a character from one state to another. An action $a \in \mathcal{A}$ is given by:

$$a = \langle \phi, v_{\text{desired}}, T \rangle, \quad (11)$$

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 momentum and orientation of the character's previous step is fixed, varying ϕ directly affects the width of the parabolic trajectory, thus allowing for a large variety of stepping 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

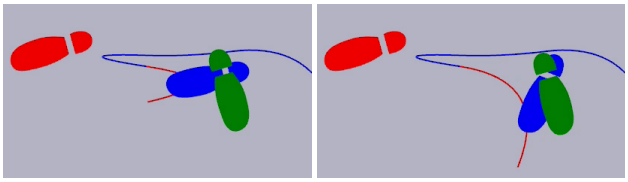


Figure 2: The orientation of the swing foot is constrainted by the previous step and the chosen trajectory (red line).

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 Equations 6-9 are used to compute (x, y) and $(\dot{x}, \dot{y})$ at the end of the next step. The foot location is always located at $(f_x, f_y) = (0, -d)$ in local space, where d is a user-defined parameter that describes the distance between a character's foot and center at rest. Additionally, for every footstep, an interval of valid foot orientations is maintained, and the exact foot orientation within this interval is chosen with a fast, simple post-process. (Foot orientations are discussed further below.) Finally, all state information is transformed back into world space, which serves as the input to create the next footstep. During the entire process, checks are made to verify that the transition and new state are valid based on constraints described below.

3.3 Locomotion constraints

Biomechanics properties. Several properties of human locomotion are automatically enforced by the definition of our model. Center of mass trajectories are guaranteed to be at least G-1 continuous. We can detect whether the center of mass will remain between the two feet by making sure the local-space parabola remains positive. The error between parabolic trajectories and more accurate inverted pendulum trajectories is negligible; specifically, parabolic trajectories have higher eccentricity than hyperbolas, and the error introduced by a parabola approximation can only push a character more quickly towards the next step, never away from it.

Footstep orientation. Intuitively, it would seem that footstep orientations must be an additional control parameter when creating a footstep. However, the choice of footstep orientation seems to have no direct effect on the dynamics of the center of mass trajectory; in our experience 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 can in fact use orientation to constrain the search space into a lower dimensional system.

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

$$[f_{\phi\text{inner}}, f_{\phi\text{outer}}] = [f_{\phi\text{outer}}, f_{\phi\text{inner}} + \frac{\pi}{2}]_{prev} \cup [\phi, (\dot{x}, \dot{y})]. \tag{12}$$

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 constraining interval $[\phi, (\dot{x}, \dot{y})]$ describes that the character would not choose a foot orientation that puts his center of mass on the outer side of the foot. Similarly, a human would rarely orient their next footstep more outwards than the orientation of their momentum – this constraint is unintuitive because it is easy to overlook momentum. Finally, parameters can be defined that allow a user to adjust these constraints, for example, to model a slow character that has difficulty with sharp, quick turns.

Space-time collision model. For any given footstep, our model can compute the *time-varying* collision bounds of the character at any exact time during the step. Thus, to determine if a footstep may cause a collision, we iterate over several time-steps within the footstep and query the collision bounds of other nearby characters for that time. The collision bounds are estimated using 5 circles, as depicted in Figure 1c. If a potential footstep action causes any collision, that footstep is considered invalid. In our system, a character can only test collisions with other characters that overlap his visual field.

User parameters. Our locomotion model offers a number of intuitive parameters that a user can modify. These parameters include the preferred value, as well as min and max limits of the step timing and stride length, the desired velocity of the center of mass and its limits, the interval of admissible foot orientations described above and its bounds, et al. By modifying these parameters a user can practically create new locomotion models. For example, restricting the range of some of the parameters results in a locomotion model can result in asymmetric stepping like an injured character.

Our model uses common default values for these parameters. For example, the desired velocity is set based on the Froude ratio, Fr. The Froude ratio provides a simple way to estimate the desired velocity of locomotion. For a typical human-like walking gait, an average value of the Froude ratio is 0.25 [Biknevicius and Reilly 2006]. The associated desired velocity is $v = \sqrt{glFr}$, where g is the acceleration of gravity and l is the height of the pelvis.

3.4 Cost function

We define the cost of a given step as the energy spent to execute a 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 a footstep action transitioning a character from s to s' is given by:

$$c(s, s') = \Delta E_1 + \Delta E_2 + \Delta E_3. \tag{13}$$

Fixed energy rate. The user defines a fixed rate of energy spent per second, denoted as P. For each step, this energy rate is multiplied by the time duration of the step T to compute the cost:

$$\Delta E_1 = P \cdot T. \tag{14}$$

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 P are roughly proportional to the character's mass.

Ground reaction forces. We model three aspects of ground reaction forces that are exerted on the character's center of mass, based on knowledge from biomechanics literature [Kuo 2007]. The geometry and notation of the cost model is shown in Figure 1b. First, at the beginning of a new step, some of the character's momentum dissipates into the ground. We estimate this dissipation with straightforward trigonometry, reducing the character's speed from v_0 to $v_0 \cos(2\theta)$. Second, in order to resume a desired speed, the character actively exerts additional work on his center of mass, computed

as:

$$\Delta E_2 = \frac{m}{2} \left| (v_{\text{desired}})^2 - (v_0 \cos(2\theta))^2 \right|. \quad (15)$$

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, he 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 also the elasticity of human joints can significantly reduce the amount of energy lost per step, thus reducing the required work of a real human versus this inverted pendulum model. However, while the model is not an accurate measurement, it is more than enough 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 apply the force over the length of the step, weighted by constant w:

$$\Delta E_3 = w \cdot F \cdot \text{length} = w \cdot m\alpha \cdot \text{length}. \quad (16)$$

Note that α is the same coefficient in Equation 7, 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. Therefore, minimizing this cost corresponds to preferring straight steps when possible, and preferring to go slower (and consequently, smaller steps) when changing the direction of momentum significantly. The weight w can be adjusted to prioritize whether the character prefers to stop (i.e. it costs less energy to avoid turning) or walk around an obstacle (i.e. it costs more energy to stop). In general we found good values of weight w to be between 0.2 and 0.5. The interpretation is that twenty to fifty percent of the curvature is due to the character's effort.

4 Planning Algorithm

Our autonomous characters navigate by planning a sequence of footsteps using the locomotion model described above. In this section we detail the implementation and integration of the short-horizon planner that outputs footsteps for navigation.

Long-term path planning. To navigate through large environments, we first compute a long-term path using A*. A local footstep goal is repositioned just before planning footsteps, placed along the long-term path, approximately 10 meters ahead of the character.

Generating discretized footstep actions. The choices for a character's next step are generated by discretizing the action space $\mathcal{A}$ 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. 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 velocity $(\dot{x}, \dot{y})$ at the end of the previous footstep, and the discretization ranges from almost straight to almost U-turns. We note that the first choice that humans would usually consider 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.

Procedure **SHBFS**(s_s,s_g)
Input: s_s= Start state of agent
Input: s_g= Goal state of agent
Output: $\mathsf{P}_k^s = \{s_s, s_1, s_2 \ldots s_k\}$ where P_k^s is complete if $s_k = s_g$
$\mathtt{OPEN} = \{s_s\}$
$\mathtt{CLOSED} = \{\phi\}$
while *OPEN* $\neq \{\phi\} \vee |\mathit{CLOSED}| \geq \mathsf{N}_{max}$ **do**
  $s_c = \arg_s \min(f(s))$ where $s \in \mathtt{OPEN}$
  if $s_c = s_g$ **then**
    $\mathsf{P}_g^s =$ **generatePath**$(s_s, s_g, \mathtt{CLOSED})$
    return P_g^s
  end
  $\mathtt{OPEN} = \mathtt{OPEN} - \{s_c\}$
  $\mathtt{CLOSED} = \mathtt{CLOSED} \cup \{s_c\}$
  $\mathsf{A} =$ **generateDiscretizedActions**(s_c, s_g)
  foreach a *in* A **do**
    $s' =$ **createFootstep**(s_c, a)
    if $g(s') > g(s_c) + c(s_c, s')$ **then**
      $g(s') = g(s_c) + c(s_c, s')$
    end
    $f(s') = g(s') + h(s')$
    $\mathtt{OPEN} = \mathtt{OPEN} \cup \{s'\}$
  end
end
$s_h = \arg_s \min(h(s))$ where $s \in \mathtt{CLOSED}$
$\mathsf{P}_h^s =$ **generatePath**$(s_s, s_h, \mathtt{CLOSED})$
return P_h^s

Algorithm 1: Short Horizon Best First Search

Short-horizon best-first search. Our planner uses a short-horizon best-first search algorithm that returns a near-optimal path, $\mathsf{P}_k^s = \{s_s, s_1, s_2 \ldots s_k\}$, from a start state s_s towards a goal state s_g. The implementation is shown in Algorithm 1.

The cost of transitioning from one state to another is given by $c(s, s')$, described by Equations 13-16. The heuristic function, $h(s)$ estimates the energy along the minimal path from s to s_g:

$$h(s) = \mathsf{c}_{expected} \times \min |\mathsf{P}_g^c|, \quad (17)$$

where $\mathsf{c}_{expected}$ is the energy spent in taking one normal footstep action, and $\min |\mathsf{P}_g^c|$ is the number of steps along the shortest distance to the goal.

The *horizon* of our planner is the maximum number of nodes that can be expanded for a single search. The path returned will be complete, *i.e.* $s_k = s_g$ if the goal s_g lies within the horizon of the planner. As $\mathsf{N}_{max} \to \infty$, the path returned by the planner will always find the goal if it exists. For efficiency reasons, however, we limit the horizon N_{max} to reasonable bounds. If the planner reaches the horizon without reaching the goal, it instead constructs an path to a state from the closed list that had the best heuristic. Intuitively, this means that if no path is found within the search horizon, the planner returns a path to the node that had the most promise of reaching the goal.

	Egress 50 agents	2-way traffic 200 agents	700 obstacles 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.

User parameters. Higher level steering decisions can be modeled as hard constraints or soft constraints on the planner. Hard constraints can essentially prune the search space, while soft constraints affect the cost of making certain decisions. For example, in a city landscape, sidewalks can be considered less costly to traverse than roads.

4.1 Interfacing with motion synthesis

Interfacing with our footstep navigation method is very flexible. Our method can output properly timed footsteps, center of mass trajectories and velocity or momentum of the character at any given time. Of course, to take advantage of the full intelligence of our approach, the motion synthesis should ideally be able to follow properly timed sequences of footsteps, including footstep orientation. For offline, high-quality animation systems, our method can be modified to output more accurate center of mass trajectories by using the hyperbola described in Equation 3 instead of the parabolic approximation, and by modifying the locomotion model and planner accordingly. The hyperbolic form is slower to evaluate, but it and its derivatives are still analytic, and the center of mass trajectories more accurately represent the inverted pendulum dynamics – in particular, the way a human's center of mass "lingers" at the apex of the motion.

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. To compute the exact foot orientation for a given step, we introduce a similar constraint that the *next* step's trajectory constrains the orientation of the current step, now that we have this information. This computation is 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 process, that would imply that no orientation of the current step could have produced a valid interval of the next step, and this would have caused an invalid interval during the planning process. The purpose of this is to further narrow down the orientations so that the final heuristic choice will be reasonable with respect to the surrounding trajectory. The final 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 next step's trajectory.

5 Results

To demonstrate our footstep navigation, we used 3D Studio Max [Autodesk 2010] scripting to automatically create animations from our footsteps. Note that these animations are not motion capture; to our knowledge, the bipedal model in 3D Studio Max is based on [Girard 1987], using inverse kinematics and a basic gait model to follow the footsteps. As described above, there already exist many motion synthesis techniques that can follow footsteps, so our focus was to evaluate the quality of the footsteps chosen by our planner.

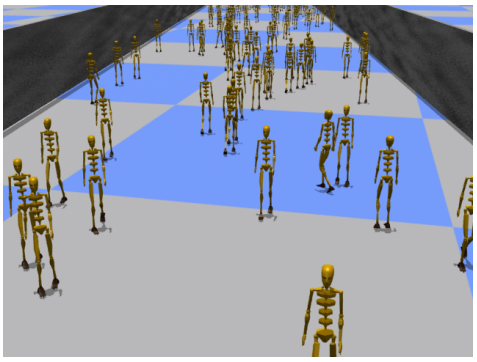


Figure 8: Two hundred characters in a crowded corridor, navigating in real-time both ways.

Even though we visualized the animations offline with this method, our planner works online, in real-time. Performance and search statistics are shown in Table 1. One reason that our planner is fast is because of the scope of footsteps: a short horizon plan of 5-10 footsteps takes several seconds to execute but only a few milliseconds to compute. The amortized cost of updating a character at 20Hz is shown in the table. These performance numbers were measured on a Core 2 process, using a single thread. With this performance, it is practical to use footstep navigation in future games. Figure 8 shows two hundred characters walking in both directions in a crowded corridor. We were able to simulate hundreds, even thousands of characters, but 3d Studio Max ran out of memory if we tried to visualize more than 200 characters simultaneously.

Our short-horizon planner can solve challenging situations such as potential deadlocks in narrow spaces. Figures 3 and 4 depict challenging doorway situations a character side steps allowing the other pedestrian to go through first. The space-time aspect of our planner helps the character to exhibit predictie, cooperative behaviors. Figure 4 is especially interesting, becase the doorway is barely wide enough to fit a single pedestrian. Many previous navigation techniques would rely on collision prevention at the walls until the character eventually finds the doorway.

Our time-varying collision model, shown ealier in Figure 1, follows more realistically the changes in bounds of a character than fixed size disks, and allows much tigher spacing in crowded conditions. An example of this is shown in Figure 5, where a group of characters tightly squeeze through a narrow glass door.

The planner and locomotion model offer a number of intuitive and fairly detailed parameters to interactively modify the behavior of the virtual humans. Figure 6 shows two characters with the same goals but different desired velocity. In Figure 7, restricting the range of footstep orientations results into additional turning steps. For additional results and associated animations, we refer the reader to the accompanying video.

6 Discussion and Future Work

Footsteps are a very appropriate form of control since they are the major contact point between a bipedal system and the external environment. By choosing to generate footsteps for navigation, using a space-time planning approach, and by using tighter dynamic collision bounds, our approach is able to control characters more precisely.

The ability of characters to stop is actually a specialized action in our planner. Being based on a general planning approach, our tech-

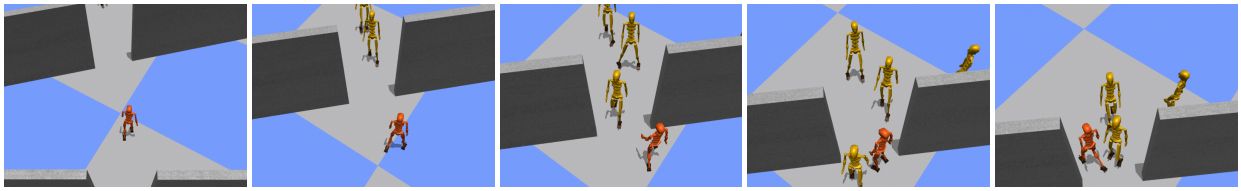


Figure 3: A character navigates through oncoming traffic at a doorway.

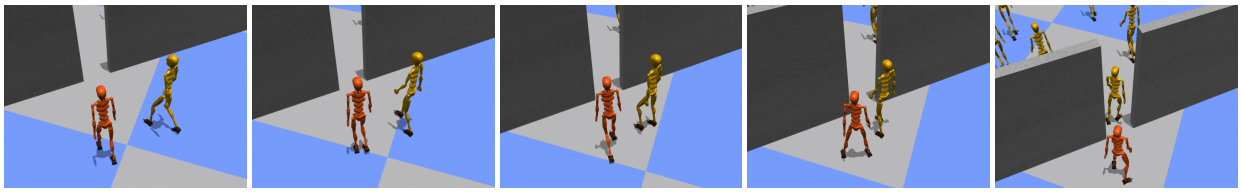


Figure 4: A character side-steps and yields to the other pedestrian, and then precisely navigates through the narrow doorway.

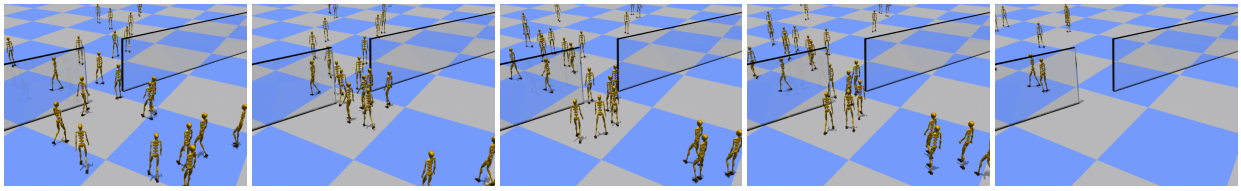


Figure 5: An egress simulation. Characters do not get stuck around the corners of the glass door.

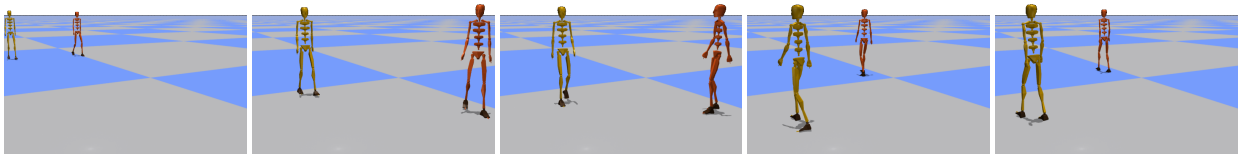


Figure 6: Two characters performing a U-turn with different, user-defined, desired velocity.

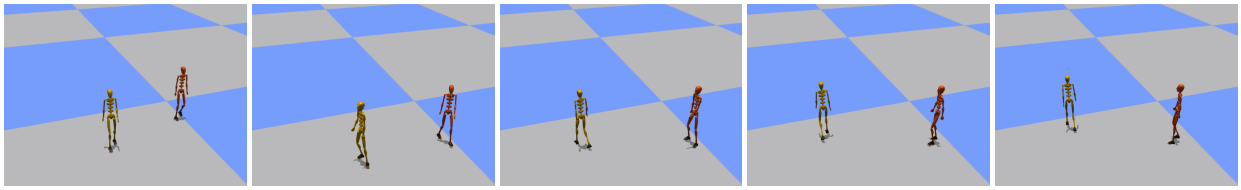


Figure 7: Two charactes performing a U-turn, with different user-defined ranges of foot orientations. The character on the right needs more steps to turn.

nique can be easily extended with other specialized actions, such as running, jumping, or specialized motion capture clips, as long as the action has well defined transitions, costs, and constraints according to our state space. Similarly, our locomotion model can be integrated with existing steering techniques to include higher level behaviors. For example, we can exploit the property that there always exists a scalar field whose gradient is a given force field. This means that any force-based steering technique can be translated into a potential function that can be used with our planner.

There are some prominent aspects of bipedal locomotion that are important in specialized situations, which our model does not consider. For example, bending knees and ankles are fundamental to capture more details about starting/stopping movements as well as going up stairs. Furthermore, the center of pressure (i.e. pendulum pivot) shifts from heel-to-toe during a step, and often times humans can rotate their foot orientations in-place; these factors have an impact on efficiency that we also do not model. Angular momentum is yet another important physical quantity that has importance for dancing and athletic activities. All these aspects may be important for a "universal" virtual character, and so it is appropriate to consider how to model these aspects in future work.

Similarly, not all costs that a character evaluates are related to effort or energy. It would be useful to characterize costs related to social and cognitive behaviors, where sometimes a character's objective is not necessarily to minimize effort. We hope to address this in future work, as well.

We found it very interesting that foot orientation did not have to be a parameter for the planner to find natural walking footsteps. This brings up very interesting biomechanics questions: what are the actual abstractions and parameters real humans use to decide one footstep over another? What are the fewest degrees of freedom we need to express the widest possible range of stepping actions? Our work is one candidate answer to this question, but of course much more rigorous study should be done in this direction.

References

AUTODESK, 2010. 3d studio max.

BADLER, N. 2008. *Virtual Crowds: Methods, Simulation, and Control (Synthesis Lectures on Computer Graphics and Animation)*. Morgan and Claypool Publishers.

BIKNEVICIUS, A., AND REILLY, S. 2006. Correlation of symmetrical gaits and whole body mechanics: debunking myths in locomotor biodynamics. *Journal of Experimental Zoology Part A: Comparative Experimental Biology 305A*, 11, 923–934.

BOULIC, R. 2008. Relaxed steering towards oriented region goals. In *Motion in Games, First International Workshop*, 176–187.

CHESTNUTT, J., LAU, M., CHEUNG, K. M., KUFFNER, J., HODGINS, J. K., AND KANADE, T. 2005. Footstep planning for the honda asimo humanoid. In *Proceedings of the IEEE International Conference on Robotics and Automation*.

CHOI, M. G., LEE, J., AND SHIN, S. Y. 2003. Planning biped locomotion using motion capture data and probabilistic roadmaps. *ACM Trans. Graph. 22*, 2, 182–203.

CHUNG, S.-K., AND HAHN, J. 1999. Animation of human walking in virtual environments. In *Computer Animation, 1999. Proceedings*, 4–15.

COROS, S., BEAUDOIN, P., YIN, K. K., AND VAN DE PANNE, M. 2008. Synthesis of constrained walking skills. *ACM Trans. Graph. 27*, 5, 1–9.

DEN BERG, J. P. V., LIN, M. C., AND MANOCHA, D. 2008. Reciprocal velocity obstacles for real-time multi-agent navigation. In *ICRA*, 1928–1935.

DONIKIAN, S., 2009. Golaem.

FALOUTSOS, P., VAN DE PANNE, M., AND TERZOPOULOS, D. 2001. Composable controllers for physics-based character animation. In *Proceedings of ACM SIGGRAPH '01*, ACM, New York, NY, USA, 251–260.

FIORINI, P., AND SHILLER, Z. 1998. Motion Planning in Dynamic Environments Using Velocity Obstacles. *The International Journal of Robotics Research 17*, 7, 760–772.

GAYLE, R., SUD, A., ANDERSEN, E., GUY, S. J., LIN, M. C., AND MANOCHA, D. 2009. Interactive navigation of heterogeneous agents using adaptive roadmaps. *IEEE Trans. Vis. Comput. Graph. 15*, 1, 34–48.

GIRARD, M. 1987. Interactive design of 3d computer-animated legged animal motion. *IEEE Comput. Graph. Appl. 7*, 6, 39–51.

GLEICHER, M. 1998. Retargetting motion to new characters. In *Proceedings of ACM SIGGRAPH '98*, ACM, New York, NY, USA, 33–42.

GLEICHER, M. 2001. Motion path editing. In *I3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, ACM, New York, NY, USA, 195–202.

GOLDENSTEIN, S., KARAVELAS, M., METAXAS, D., GUIBAS, L., AARON, E., AND GOSWAMI, A. 2001. Scalable nonlinear dynamical systems for agent steering and crowd simulation. *Computers and Graphics 25*, 6, 983 – 998.

HELBING, D., AND MOLNÁR, P. 1995. Social force model for pedestrian dynamics. *Phys. Rev. E 51*, 5 (May), 4282–4286.

HODGINS, J. K., WOOTEN, W. L., BROGAN, D. C., AND O'BRIEN, J. F. 1995. Animating human athletics. In *Proceedings of SIGGRAPH '95*, ACM Press, New York, NY, USA, 71–78.

KAJITA, S., KANEHIRO, F., KANEKO, K., YOKOI, K., AND HIRUKAWA, H. 2001. The 3d linear inverted pendulum mode: a simple modeling for a biped walking pattern generation. In *Proceedings of IEEE/RSJ (IROS)*, vol. 1, 239–246.

KAPADIA, M., SINGH, S., HEWLETT, W., AND FALOUTSOS, P. 2009. Egocentric affordance fields in pedestrian steering. In *I3D '09: Proceedings of the 2009 symposium on Interactive 3D graphics and games*, ACM, New York, NY, USA, 215–223.

KO, H., AND BADLER, N. I. 1992. Straight line walking animation based on kinematic generalization that preserves the original characteristics, technical report.

KOVAR, L., GLEICHER, M., AND PIGHIN, F. 2002. Motion graphs. *ACM Trans. Graph. 21*, 3, 473–482.

KUFFNER, J., NISHIWAKI, K., KAGAMI, S., KUNIYOSHI, Y., INABA, M., AND INOUE, H. 2003. Online footstep planning for humanoid robots. In *IEEE Int'l Conf. on Robotics and Automation (ICRA'2003)*, IEEE.

KUO, A. 2007. The six determinants of gait and the inverted pendulum analogy: A dynamic walking perspective. *Human Movement Science 26*, 4, 617 – 656. European Workshop on Movement Science 2007.

LAMARCHE, F., AND DONIKIAN, S. 2004. Crowd of virtual humans: a new approach for real time navigation in complex and structured environments. *Comput. Graph. Forum 23*, 3, 509–518.

LAU, M., AND KUFFNER, J. J. 2005. Behavior planning for character animation. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM, New York, NY, USA, 271–280.

LAU, M., AND KUFFNER, J. J. 2006. Precomputed search trees: Planning for interactive goal-driven animation. In *2006 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 299–308.

LAVALLE, S. M. 2006. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K. Available at http://planning.cs.uiuc.edu/.

LEE, K. H., CHOI, M. G., HONG, Q., AND LEE, J. 2007. Group behavior from video: a data-driven approach to crowd simulation. In *Proceedings of ACM SIGGRAPH/EG Symposium on Computer Animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 109–118.

LEE, S.-H., SIFAKIS, E., AND TERZOPOULOS, D. 2009. Comprehensive biomechanical modeling and simulation of the upper body. *ACM Trans. Graph. 28*, 4, 1–17.

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 *TPCG '03: Proceedings of the Theory and Practice of Computer Graphics 2003*, IEEE Computer Society, Washington, DC, USA, 122.

MEGHDARI, A., SOHRABPOUR, S., NADERI, D., TAMADDONI, S. H., JAFARI, F., AND SALARIEH, H. 2008. A novel method of gait synthesis for bipedal fast locomotion. *Intelligent Robotics Systems 53*, 2, 101–118.

MUSSE, S., AND THALMANN, D. 1997. A Model of Human Crowd Behavior. In *Proc. CAS'97, Springer Verlag, Wien*, 39–51.

NISHIWAKI, K., SUGIHARA, T., KAGAMI, S., INABA, M., AND INOUE, H. 2001. Online mixture and connection of basic motions for humanoid walking control by footprint specification. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 4, 4110–4115.

PARIS, S., AND DONIKIAN, S. 2009. Activity-driven populace: A cognitive approach to crowd simulation. *IEEE Computer Graphics and Applications 29*, 4, 34–43.

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

PELECHANO, N., ALLBECK, J. M., AND BADLER, N. I. 2007. Controlling individual agents in high-density crowd simulation. In *Proceedings of ACM SIGGRAPH/EG Symposium on Computer Animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 99–108.

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

REYNOLDS, C., 1999. Steering behaviors for autonomous characters.

SAUNDERS, J., INMAN, V., AND EBERHART, H., 1953. The major determinants in normal and pathological gait.

SHAO, W., AND TERZOPOULOS, D. 2005. Autonomous pedestrians. In *Proceedings of ACM SIGGRAPH/EG Symposium on Computer Animation*, ACM, New York, NY, USA, 19–28.

STURMAN, D., 1994. A brief history of motion capture for computer character animation.

TORKOS, N., AND VAN DE PANNE, M. 1998. Footprint-based quadruped motion synthesis. *Proceedings of Graphics Interface '98*, 151–160.

TREUILLE, A., COOPER, S., AND POPOVIĆ, Z. 2006. Continuum crowds. *ACM Trans. Graph. 25*, 3, 1160–1168.

TREUILLE, A., LEE, Y., AND POPOVIC, Z. 2007. Near-optimal character animation with continuous control. *ACM Trans. Graph. 26*, 3, 7.

VAN BASTEN, B., AND EGGES, A. 2010. The stepspace: Example-based footprint-driven motion synthesis. Wiley. in press.

VAN BASTER, B., AND EGGES, A. 2009. Path abstraction for combined navigation and animation. *Lecture Notes in Computer Science, Proceedings of MIG'09 5884/2009*, 182–193.

VAN DE PANNE, M. 1997. From footprints to animation. *Computer Graphics Forum 16*, 4, 211–223.

WU, C.-C., MEDINA, J., AND ZORDAN, V. B. 2008. Simple steps for simply stepping. In *ISVC (1)*, 97–106.

YIN, K., AND PAI, D. K. 2003. Footsee: an interactive animation system. In *Proceedings of ACM SIGGRAPH/EG Symposium on Computer Animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 329–338.

ZHANG, L., PAN, J., AND MANOCHA, D. 2009. Motion planning and synthesis of human-like characters in constrained environments. *Lecture Notes in Computer Science, Proceedings of MIG'09 5884/2009*, 138–145.

ZHANG, Y., PETTRÉ, J., PENG, Q., AND DONIKIAN, S. 2009. Data based steering of virtual human using a velocity-space approach. *Lecture Notes in Computer Science, Proceedings of MIG'09 5884/2009*, 170–181.