12-2012

Parallelized Egocentric Fields for Autonomous Navigation

Mubbasir Kapadia
University of Pennsylvania

Shawn Singh

William Hewlett

Glenn Reinman

Petros Faloutsos

Recommended Citation

Kapadia, M., Singh, S., Hewlett, W., Reinman, G., & Faloutsos, P. (2012). Parallelized Egocentric Fields for Autonomous Navigation. *The Visual Computer, 28* (12), 1209-1227. http://dx.doi.org/10.1007/s00371-011-0669-5

Parallelized Egocentric Fields for Autonomous Navigation

Abstract

In this paper, we propose a general framework for local path-planning and steering that can be easily extended to perform high-level behaviors. Our framework is based on the concept of affordances: the possible ways an agent can interact with its environment. Each agent perceives the environment through a set of vector and scalar fields that are represented in the agent's local space. This egocentric property allows us to efficiently compute a local space-time plan and has better parallel scalability than a global fields approach. We then use these perception fields to compute a fitness measure for every possible action, defined as an affordance field. The action that has the optimal value in the affordance field is the agent's steering decision. We propose an extension to a linear space-time prediction model for dynamic collision avoidance and present our parallelization results on multicore systems. We analyze and evaluate our framework using a comprehensive suite of test cases provided in SteerBench and demonstrate autonomous virtual pedestrians that perform steering and path planning in unknown environments along with the emergence of high-level responses to never seen before situations.

Keywords

affordance, egocentric, steering, space-time planning

Disciplines

Computer Sciences | Engineering | Graphics and Human Computer Interfaces

Keywords affordance, egocentric, steering, space-time planning

Parallelized Egocentric Fields for Autonomous Navigation

Mubbasir Kapadia[1,2] **Shawn Singh**[1,4]
William Hewlett[1] **Glenn Reinman**[1]
Petros Faloutsos[1,3]
[1]**University of California, Los Angeles**
[2]**University of Pennsylvania**
[3]**York University**
[4]**Google Inc.**

the date of receipt and acceptance should be inserted later

Abstract In this paper we propose a general framework for local path-planning and steering that can be easily extended to perform high-level behaviors. Our framework is based on the concept of *affordances*: the possible ways an agent can interact with its environment. Each agent perceives the environment through a set of vector and scalar fields that are represented in the agent's local space. This egocentric property allows us to efficiently compute a local space-time plan and has better parallel scalability than a global fields approach. We then use these perception fields to compute a fitness measure for every possible action, defined as an affordance field. The action that has the optimal value in the affordance field is the agent's steering decision. We propose an extension to a linear space-time prediction model for dynamic collision avoidance and present our parallelization results on multi-core systems. We analyze and evaluate our framework using a comprehensive suite of test cases provided in SteerBench and demonstrate autonomous virtual pedestrians that perform steering and path planning in unknown environments along with the emergence of high-level responses to never seen before situations.

email: mubbasir@cs.ucla.edu

email: shawnsin@cs.ucla.edu

email: billyh@cs.ucla.edu

email: reinman@cs.ucla.edu

email: pfal@cs.ucla.edu

Address(es) of author(s) should be given

1 Introduction

Research in the area of pedestrian simulation has seen a dramatic rise in recent years. With the potential for this work being realized in a wide variety of areas, ranging from urban planning and training simulations to games, life-like steering motions for each individual have become critical for a truly immersive and realistic experience.

There are two major components involved in pedestrian navigation: path planning and a steering mechanism. They are often tackled as separate problems that need to be interfaced for a fully functional navigation system [10,42]. Approaches such as A* [16] and potential fields [61,45] are popular planning methods for pedestrian simulations. Given a target location and an obstacle-laden environment, these techniques compute a global path to the target. Then, a steering mechanism (*e.g.*, [38,28,36]) tries to follow the planned path while avoiding dynamic objects.

However, an important feature of realistic steering is missing in traditional approaches: humans constantly compute a local short-term space-time plan to steer through their immediate environment which includes dynamic objects and other agents. This short-term plan is essential for *natural* steering in crowded environments, as well as for resolving deadlock situations, for example two people arriving at a doorway from opposite directions. Most agent-based approaches do not perform space-time *planning*. Some works use space-time predictions (*e.g.* [33]) with an explicit time dimension and basic linear prediction. Most field-based approaches (*e.g.*, [45]) do compute plans on the required scale, but do not take into account time or dynamic obstacles. One notable fields-based method uses continuum dynamics to achieve dynamic collision avoidance [56], but at the expense of removing individuality in behaviors. In general, field-based approaches require storing large high-resolution fields for the entire environment. Updating these global fields cannot be easily parallelized, and spends storage and computation cost in places that may not affect any agents.

This paper presents a novel technique that bridges the gap between steering and planning by using information fields in an egocentric fashion, where the origin of the field is centered about the agent at all times. This allows us to implicitly account for time when planning, removes the storage and computation problems of global field-based approaches, and achieves linear scalability when parallelized.

Our approach is based on the concept of *affordances* – the ways in which an agent can interact with its environment [13]. Affordances have been applied to agent systems [30,59], but not explicitly for steering. We define *fitness* to be a measure of how appropriate the associated affordance (associated action) will be. An *affordance field* is a scalar field that has a fitness associated with every affordance in the space of all possible actions. A final decision is the affordance associated with the optimal fitness.

To evaluate our approach, we report extensive benchmarking results using SteerBench [47]. Testing, evaluating, and analyzing steering simulations

is known to be a difficult task, primarily due to the lack of automated tools. SteerBench proposes a transparent method of objectively evaluating and comparing the quality of steering solutions. We present these results for analysis and as a basis for evaluation and comparison to other steering methods.

This paper presents a novel egocentric fields steering framework with the following contributions:

- We represent the fields in an egocentric local-space, as opposed to a global world-space, gaining the benefits described above.
- We propose affordance fields as a powerful way to combine sensory information, giving more meaningful data than potential fields.
- Our discretized model has variable resolution, where information accuracy decreases with increase in distance from the origin. This avoids wasteful computation and storage cost further away, where a plan would be recomputed before it is used.
- Our approach performs short-term planning, accounting for dynamic objects, making it possible to steer naturally in challenging agent-agent interactions such as deadlocks.
- We discuss the issues involved in parallelizing our algorithm using inter-agent communication and present our performance results exhibiting near-linear speedup on multi-core architectures.
- We present the analysis results of our steering framework using SteerBench to serve as the basis of comparison with related work.

The rest of this paper is organized as follows: Section 2 provides a brief overview of the current state of the art in the field of crowd simulation. Then, Section 3 gives a theoretical overview of our approach, including our egocentric information representation and the concept of affordance fields. Section 4 describes the discretization of these fields for implementation, and Section 5 describes the specific fields that we use to demonstrate our approach. Section 6 discusses the integration of our field-based steering technique into our complete pedestrian simulation framework. We then provide extensive results of our method running on many different benchmark scenarios in Section 7, and evaluate the parallelization of our framework on multi-core architectures in Section 8. Finally, Section 9 concludes and discusses future work.

2 Related Work

Since the seminal work of [39,38], there has been a growing interest in crowd simulation with a wide variety of techniques being tested and implemented. Excellent reviews of prior work can be found here [54,35]. We broadly classify steering approaches into the following two categories:

Centralized Approaches. Centralized techniques such as continuum dynamics [8,56], regression [31], route choice models [21] and markov chains [29] model the macroscopic phenomena exhibited in crowds (e.g. stadium evacuation scenarios, urban simulations etc). However, these approaches are unable

to model specific agent-agent interactions which are crucial in a microscopic view of crowd simulations that are prevalent in todays applications.

De-centralized Approaches. These approaches model the agent as an independent entity which performs collision avoidance with static obstacles, reacts to dynamic threats in the environment while steering towards the target. Particle dynamics [39,38], social force models [20,19] and cellular automata models [53,7,55] are simple and efficient but cannot simulate realistic pedestrian behaviors. Rule based systems [24,28,41,36,50,4] limit the steering functionality to conditions that have been foreseen and do not react well to irregularities in the environment that inevitably arise in a high density crowd simulation. Predictive approaches [33,11,3,32] steer in complex environments by predictively avoiding dynamic threats. The work in [48] presents a hybrid approach that combines reaction, prediction, and planning into a single steering framework. Example based approaches [25,26] use video segments of real pedestrian crowds as input to a learning system that generates natural looking crowd behaviors. The work of [42,43] integrates motor, perceptual, behavioral, and cognitive components within a comprehensive model of pedestrians as individuals. [5] offers the ability of reaching a goal with a prescribed direction by extending the funneling behavior.

There are two popular methods to path planning in crowd simulation. The A* algorithm and its derivatives [16,17,9,57] tend to produce non-realistic routes and require smoothing techniques in addition to a steering mechanism for dynamic collision avoidance. The approach of potential fields [60,61,45, 14,2] generates a global field for the entire landscape where the potential gradient is contingent upon the presence of obstacles and distance to goal. Since a change in target or environment requires significant re-computation, these navigation methods are generally confined to systems with non-changing goals and static environments. A solution is proposed by [51] in which path-finding data is pre-computed and stored in a connectivity table which offsets this issue at the cost of a considerable memory overhead. [27] uses a segregated local and global planner to perform path planning in a layered environment. The work of [52] employs the notion of attractiveness of objects for path planning and its use in human animation.

Space-time models (e.g., [22,58,44]) combine space and time into a single construct by representing space as three dimensions and time as the fourth dimension. Space time planning exploits the inherent advantage of having information in time to predict collisions in the future. These models improve steering behaviors at the cost of an additional dimension which greatly increases the search space, incurring a considerable overhead.

There has been previous work in the realm of egocentric based navigation [1,6,12]. These techniques use egocentric maps for simple static obstacle avoidance but do not address the issue of larger environments where the goal falls outside the local egocentric map. Moreover, this egocentric model does

not address space-time planning for dynamic object avoidance or real-time performance.

In most crowd simulation approaches, the underlying steering framework offers parameters that can be used by a higher lever framework to implement group behaviors. For example, in force based approaches [19], properly designed attractive forces can keep a group of people together or make them follow a leader. The work in [40] demonstrates realistic group behaviors using a biologically-motivated space colonization algorithm previously used in generating leaf venation patterns. Our framework allows group behaviors to be implemented by simply setting intermediate dynamic or static goals for the agents.

Comparison to previous work. Our work is most similar to centralized field-based approaches [60, 56] and de-centralized predictive approaches [33, 3]. However, the affordance fields in our method are represented in an egocentric manner with variable resolution, giving us the following benefits: (1) they are no longer bound to the resolution and scaling problems associated with global fields and continuum methods. (2) We can dynamically scale the resolution of our egocentric fields to get the highest resolution possible for the scale we need. (3) The computational cost of our approach is not dependent on the complexity of the environment. (4) Our approach naturally supports efficient space-time planning, which is difficult to integrate into global fields and predictive approaches.

Extension to prior work. This paper is a significant extension of [23]. The novel contributions over the original work are summarized as follows:

- The related work includes 18 additional references, better reflecting the current state of the art in relation to the new research extensions.
- Section 4.3 proposes an analytical solution of the discrete field representation, to dynamically vary the resolution of the fields.
- Section 6 provides a detailed overview of the integration of our steering technique into a complete pedestrian simulation framework. In particular, we discuss the following three aspects of our framework: (1) pathfinding, (2) steering and, (3) animation.
- We perform a rigorous evaluation of our proposal using SteerBench [47] by demonstrating our steering algorithm on a wide variety of scenarios. We also provide a detailed comparison between the proposed method and established prior art [3, 48] by reporting and analysing the scores of each algorithm.
- We harness the advantage of local fields in our framework to parallelize our steering algorithm. We investigate the use of different decomposition and scheduling strategies with results of our findings (Section 8).
- We provide additional illustrative examples of our framework successfully handling complicated scenarios and demonstrating group behaviors.

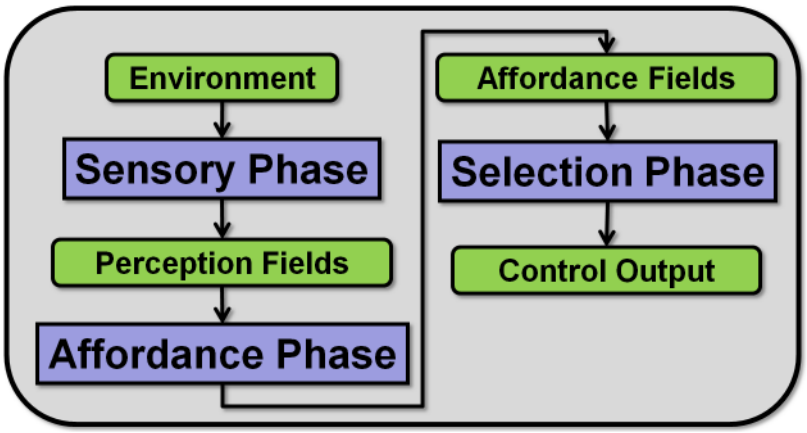

Fig. 1 Phases of our framework.

3 Overview

Figure 1 presents an overview of our framework, which consists of three phases: (1) sensory phase, (2) affordance phase and, (3) selection phase. The sensory phase gathers and interprets sensory information from the environment to compute perception fields. The affordance phase computes affordance fields which quantifies the relative strength of all affordances. Finally, the selection phases chooses the affordances having optimal fitness value, to produce the control decisions. There are two key aspects of our model. First, the data is represented using *variable-resolution egocentric fields*, where the origin is always the center of focus and the information accuracy decreases with increase in distance from the origin. Second, we use the concept of *affordances*, to quantify the different ways in which an agent cant interact with the environment and other agents. Sections 3.1, 3.2 and 3.3 describe each of these phases in detail.

3.1 Sensory Phase – Egocentric Perception Fields

The first phase of our method is to gather and interpret sensory information. An *egocentric perception field*, $P(\mathbf{X})$, is a vector or scalar field that quantifies a property of the environment. For example, a *traversability field* quantifies how easy it is to occupy a location in space – a high traversability value implies that it is easy for an agent to occupy that location, while a low traversability value implies that an agent would not be able to occupy that location, perhaps because another object already occupies that location. Other examples of perception fields include: velocity information of nearby objects, planned trajectories of other agents, or even more abstract quantities like the perception of other agents' state and attention. Egocentric perception fields can be

computed from a robot's sensors or by querying data structures of a virtual environment.

Time is naturally taken into account in this model, because of the egocentric representation. The agent is always located at the origin, and therefore the distance between any point and the origin can be used to estimate the time it would take to reach that point. We use this property to efficiently predict collisions and plan in the *space-time* domain without requiring an explicit additional dimension in the system.

When appropriate, the raw perception fields are combined to provide a more intuitive representation of perceptual sensory information:

$$P'(\mathbf{X}) = g(P_1, P_2, P_3, ...P_n, goal) \tag{1}$$

where $g(\cdot)$ is a function of one or more perception fields. For example, a linear combination of static and dynamic perceptual information provides information of traversability in the environment. This refined set of perception fields can then be used to compute the affordance fields.

3.2 Affordance Phase – Affordance Fields

The concept of affordance was introduced by Gibson in 1954 [15]. In our context, affordances describe the various ways that an agent understands that it can use or interact with its environment. Specifically, we define an affordance as a component of a possible steering action that an agent could perform at a given point in time. For example, speed and orientation are two affordances. A set of such affordances is the control output of our system.

The role of this second phase is to compute the "strength" or "fitness" of all possible affordances. An *affordance field*, $A(q)$, quantifies the relative strength of all affordances of a particular type, based on the desired goals of the agent. Affordance fields are defined over their respective spaces, such as the space of possible speeds, or space of possible directions. Intuitively, an affordance value, $A(q_i)$ for a particular action q_i indicates how much this action would help advance the agent towards accomplishing its goals. It is computed as a function of perception fields:

$$A(q) = f(P'_1, P'_2, P'_3, ...P'_m) \tag{2}$$

where the function $f(\cdot)$ is defined so that $A(q)$ provides a numeric value indicating the strength of a particular affordance. The specific functions that we use for our implementation of steering are described in Section 5.

3.3 Selection Phase

The final output of our method is the particular affordance q_i associated with the optimal value $A(q_i)$. Optimality is defined by maximizing or minimizing

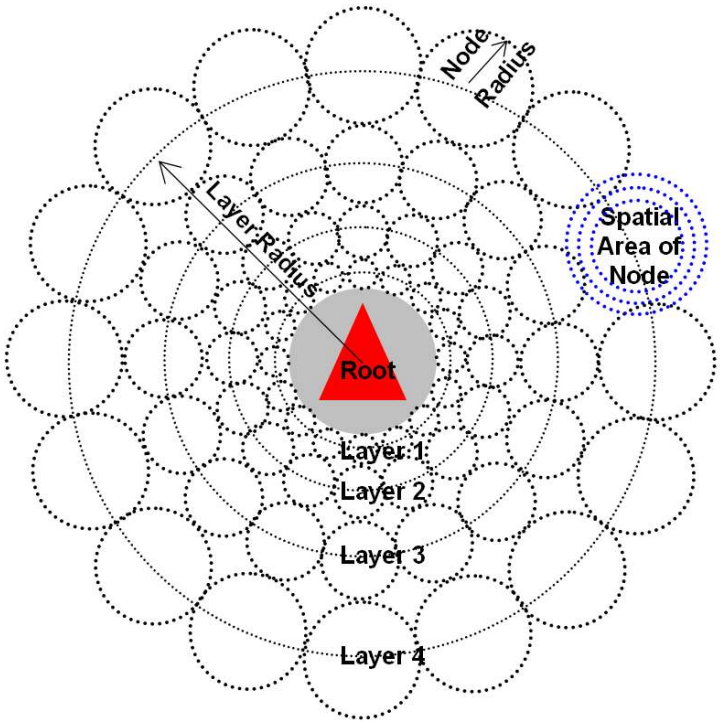

Fig. 2 Visualization of our variable-resolution discretization and the relevant parameters.

$f(\cdot)$. For example, output decisions would be a target speed and desired direction of an agent. In a discrete setting, the affordance field is a set of sample affordance values, and so this optimization simplifies to choosing the max or min value of that set.

4 Discrete Egocentric Fields

We implement a discretization of the model developed in Section 3 as a connectionist architecture that uses nodes arranged in concentric circles to maintain egocentric spatial information. At all times, the central node represents the current position of the agent. Each node perceives information corresponding to its "spatial awareness" in the environment. The structure parameters of the discretization are described in Section 4.1. Then, Section 4.3 describes the variable resolution and dynamic scaling of the discrete fields.

4.1 Structure of Discrete Egocentric Fields

Discrete Egocentric Fields comprise the following structural components, shown in Figure 2:

- **Root:** The *root* represents the current position of the agent and is the origin of the egocentric fields.
- **Layers:** The egocentric map is segregated into layers, denoted by the layer number l, where each layer comprises a fixed number of nodes that store the information of an area of the environment. The number of nodes per layer n and the number of layers m are the two user-defined parameters.
- **Layer Radius:** The distance from the root to the l^{th} layer is known as the *layer radius*, denoted as $r_{\text{layer}}(l)$.
- **Node Radius:** The radius of the area associated with node of the l^{th} layer is termed as the *node radius*, $r_{\text{node}}(l)$. The node radius increases for layers further from the root. As a result, the spatial area covered by the node increases with increase in distance from the root, giving rise to variable resolution.
- **Inter-node weight:** The *inter-node weight*, $w(l)$, determines the area between two adjacent nodes in successive layers. It allows us to dynamically scale the coverage of the environment, for a constant memory cost.
- **Node Information:** Each node contains its location, connectivity to neighboring nodes, and values of the perception and affordance fields for its given location.

4.2 Derivation from User-defined Parameters

The accuracy of the discrete representation is dependent on the layer radius, node radius, and the inter-node weight, which in turn are determined by user-defined parameters: the number of nodes per layer, n, and the number of layers, m. Here we describe how those parameters are computed given n and m.

The layer-radius $r_{\text{layer}}(l)$, node-radius $r_{\text{node}}(l)$, and inter-node weight $w(l)$ all contribute to the variable resolution and the dynamic scaling of the egocentric fields. These structural components are dependent on the layer number l and the number of nodes per layer n. Note that n is a user-defined parameter. The node radius is proportional to the circumference of the l^{th} layer. Thus,

$$r_{\text{layer}}(l) = r_{\text{node}}(l) \times \frac{n}{\pi}. \tag{3}$$

The first layer is at an offset of $r_{\text{node}}(0)$, outside the agent. The agent is modeled as a circle, with radius r_{agent}.

$$r_{\text{layer}}(0) = r_{\text{node}}(0) + r_{\text{agent}}. \tag{4}$$

From 3 and 4, the initial condition of $r_{\text{node}}(0)$ is

$$r_{\text{node}}(0) = \frac{\pi}{n - \pi} \times r_{\text{agent}}. \tag{5}$$

The node radius of the next layer increases to ensure the coverage of adjacent nodes following the structure shown in Figure 2,

$$r_{\text{node}}(l+1) = \frac{n + \pi}{n - \pi} \times r_{\text{node}}(l). \tag{6}$$

Equations 3, 5 and 6 provide a method of estimating the node radii and the layer radii respectively. The user can thus specify the number of nodes per layer, n and the number of layers, m to control the size and resolution of the map.

4.3 Variable Resolution and Dynamic Scaling

Our method allows us to take advantage of variable resolution discretization, where the accuracy of information is higher near the root and decreases further from the origin. The information storage per unit area is dense close to the origin, and density decreases further from the root. In combination with the egocentric property, variable resolution fields allow us to avoid costly computations where data is far away, both temporally and spatially. This is appropriate in the context of agent navigation, since the immediate surroundings are often more important for making navigation decisions.

This is appropriate for agent navigation, where an agent's navigation priorities are highest in its immediate surroundings, and lower further away, where it cannot robustly predict the future situation anyway. In this way, the layer radius also determines the relative importance of the information in each node.

To dynamically scale the field at runtime, we use inter-node weights, $w(l)$, to scale the field with respect to the distance of the goal, D. If the goal is further than the radius of the m^{th} layer, $D > r_{\text{layer}}(m-1)$, then the goal lies outside of the field. In this case, we scale the resolution of the field such that the goal lies inside the field. The weight of the first layer $w(0)$ is initialized to 1 to keep the level of detail in the first layer high for collision-free steering. We successively increase the weight of the other layers as follows:

$$w(l+1) = w(l) + \beta. \tag{7}$$

where β represents the common difference between successive weights. The weight of the m^{th} layer can thus be calculated as follows:

$$w(m-1) = 1 + (m-1) \times \beta. \tag{8}$$

The scaled node radius $r'_{\text{node}}(l)$ is computed as follows:

$$r'_{\text{node}}(l) = r_{\text{node}}(l) \times w(l). \tag{9}$$

For the goal to lie within the field, the goal distance, D must be equal to the scaled layer radius, $r'_{\text{layer}}(m-1)$. From Equations 3, 6, 8, and 9, we get

$$D = \left(\frac{n+\pi}{n-\pi}\right)^{m-1} \times r(0) \times (1 + (m-1) \times \beta) \times \frac{n}{\pi} \tag{10}$$

Rearranging Equation 10, we can compute β to dynamically scale the field to accommodate the goal position.

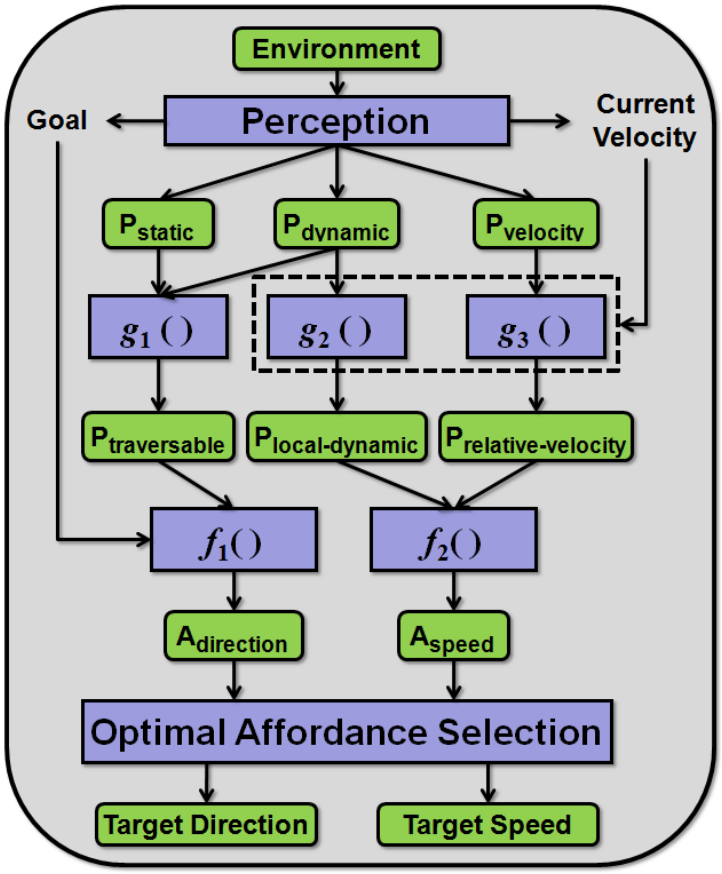

Fig. 3 Data flow diagram for steering using egocentric fields.

$$\beta = \frac{\frac{D}{\left(\frac{n+\pi}{n-\pi}\right)^{m-1} \times r(0) \times \frac{n}{\pi}} - 1}{m-1} \tag{11}$$

5 Applying Discrete Egocentric Fields to Steering

This section describes the specific egocentric fields used for steering. Sensory information, such as traversability, dynamic threats and velocity of neighboring agents, is represented using egocentric perception fields (Section 5.1). Affordance fields are computed as a function of these perception fields which provide the relative strength of all possible steering decisions, based on the goal(s) of the agent (Section 5.2). The final output decisions, in the form of target speed, s_{target} and target direction, $\mathbf{D}_{\text{target}}$ are used for locomotion of agent (Section 5.3).

5.1 Sensory Phase – Perception Fields

Figure 3(a) illustrates the specific perceptions fields used for steering, which are described below:

- **Static Field**: The static field, $P_{\text{static}}(\mathbf{X})$, represents the configuration of the obstacles in the environment surrounding the agent. A minimum value of 0 for a given location in the field indicates that the location is free from static obstacles, while a maximum value of 1 indicates that it cannot be traversed.
- **Dynamic Field**: The dynamic field, $P_{\text{dynamic}}(\mathbf{X})$ represents the configuration of all dynamic objects, by providing the predicted positions of neighboring agents at various points of time. A minimum value of 0 indicates that the likelihood of a dynamic threat at the given location is minimal, while a maximum value of 1 indicates a high probability of a dynamic threat.
- **Velocity Field**: The velocity field, $\mathbf{P}_{\text{velocity}}(\mathbf{X})$ is a vector field that provides the direction and speed magnitude of neighboring agents.
- **Traversability Field**: $P_{\text{traversability}}(\mathbf{X})$, the traversability field, is a combination of P_{static} and P_{dynamic}:

$$P_{\text{traversability}}(\mathbf{X}) = P_{\text{static}}(\mathbf{X}) + P_{\text{dynamic}}(\mathbf{X}) \tag{12}$$

- **Local Dynamic Field**: The dynamic field, $P_{\text{dynamic}}(\mathbf{X})$ is subject to a kernel function that considers the regions in which dynamic threats are most imminent. The resulting fields are known as local dynamic fields, denoted by $P_{\text{local-dynamic}}$.

$$P_{\text{local-dynamic}}(\mathbf{X}) = K(\mathbf{X}) \cdot P_{\text{dynamic}}(\mathbf{X}) \tag{13}$$

  Our current implementation uses a simple step function which only considers the information in the first $m/2$ layers of the dynamic field, where m is the total number of layers :

$$K(\mathbf{X}) = \begin{cases} 1 \text{ if } |\mathbf{X}| < r_{\text{layer}}(m/2) \\ 0 \text{ otherwise} \end{cases} \tag{14}$$

- **Relative Velocity Field**: The relative velocity field, $\mathbf{P}_{\text{relative-velocity}}(\mathbf{X})$ provides the relative velocity of neighboring agents with respect to the agent's velocity.

$$\mathbf{P}_{\text{relative-velocity}}(\mathbf{X}) = \mathbf{P}_{\text{velocity}}(\mathbf{X}) - (s_{\text{current}} \times \mathbf{D}_{\text{current}}) \tag{15}$$

  where s_{current} is the speed with which the agent is traveling and $\mathbf{D}_{\text{current}}$ is its current direction of motion.

Implicit Space-Time Planning. Previous approaches implement space-time planning by representing space in two or three spatial dimensions with time as an additional dimension. This incurs a considerable processing overhead which becomes intractable in large crowd simulations. Our approach naturally

supports efficient space-time planning by using egocentric fields. Due to the egocentric nature of our data representation, we can represent time implicitly as the distance from the origin to any point of interest, effectively reducing the dimensions in space-time planning by one.

There exists a mapping between time and a particular layer of an agents egocentric field, as both time and the layer number are proportional to the distance from the origin. Let $t[l]$ be the time taken by the agent to travel a distance $r_{\text{layer}}(l)$, for a particular layer, l. P_{dynamic} and $\mathbf{P}_{\text{velocity}}$ are computed by considering this *time-level associativity* and the neighbors, N, surrounding an agent . If the difference in the time taken by the neighbor $N(i)$ in traveling a distance $r_{\text{layer}}(j)$ and the time taken by the agent to travel a distance $r_{\text{layer}}(k)$ is below a certain threshold, ϵ, then a dynamic threat is predicted at that instance of time and space. P_{dynamic} of the agent reflects a potential threat at the predicted position of $N(i)$, at time $t[k]$. $\mathbf{P}_{\text{velocity}}$ stores the current velocity of the potential threats at that point in space. Once these fields are computed, $P_{\text{local-dynamic}}$ and $\mathbf{P}_{\text{relative-velocity}}$ are determined using equations 13 and 15.

5.2 Affordance Phase – Affordance Fields

Once the perception fields are populated, the affordance fields can then be computed. The affordance fields that we use for steering provide a fitness value for each possible speed and direction, and are defined as follows:

Speed Affordance Fields. The speed affordance field $A_{\text{speed}}(s)$ provides the relative fitness for each speed affordance. The fitness of a particular speed s is the distance of the most imminent threat for that value of s:

$$A_{\text{speed}}(s) = \arg\min_{\mathbf{X}}(\mathbf{X}_{\text{i}} + t \times P_{\text{relative-velocity}}(\mathbf{X})) \tag{16}$$

where $\mathbf{X}_{\text{i}} \in \{\mathbf{X} : P_{\text{local-dynamic}} > 0\}$. Note that the relative velocity field, $\mathbf{P}_{\text{relative-velocity}}(\mathbf{X})$ is recomputed for each s.

Direction Affordance Fields. Direction Affordance Fields, $A_{\text{direction}}(\theta)$, quantify the relative strengths of all possible directions θ in which an agent can steer. A pedestrian in a crowd bases its direction of travel on the presence of static objects in the environment as well as other pedestrians (dynamic objects). For instance, a slow moving pedestrian in front would mandate a direction change in order to perform an overtaking maneuver. In order to compute the direction affordance field, we first compute an intermediate affordance field, $A_{\text{spatial}}(\mathbf{X})$, to provide a fitness value for all points in the spatial domain:

$$A_{\text{spatial}}(\mathbf{X}) = f_1(A_{\text{spatial}}, P_{\text{traversability}}, \mathbf{X}_{\text{goal}}) \tag{17}$$

where f_1 is an iterative process on A_{spatial}. The process starts by adding a strong fitness value at the goal position and then propagating this value in all directions. The propagation at each point in space is affected by the traversability perception field. For example, if there is an untraversable object between

the agent and its goal, the fitness value will not propagate through the object. Instead the high fitness will eventually reach the agent by propagating around the object.

The implementation of f_1 is described as follows. Given a 3D location $\mathbf{X_1}$, which is initially set to $\mathbf{X}_{\text{goal}}$, a set of points at an infinitely small displacement of Δr in all directions around $\mathbf{X_1}$ can be represented by the following function, $a(\mathbf{X})$:

$$a(\mathbf{X}) = \{\mathbf{X_2} : |\mathbf{X} - \mathbf{X_1}|^2 = \Delta r^2, \mathbf{X} \in R^3\} \tag{18}$$

The fitness value propagates from point $\mathbf{X_1}$ to point $\mathbf{X_2}$ according to the following recurrence:

$$A_{\text{spatial}}(\mathbf{X_2}) = (A_{\text{spatial}}(\mathbf{X_1}) - P_{\text{traversability}}(\mathbf{X_2})) \times \alpha \tag{19}$$

where $\alpha \in (0, 1)$ is the rate of decay. The end result of this process is a path of high fitness from $\mathbf{X}_{\text{origin}}$ to $\mathbf{X}_{\text{goal}}$ that represents the path that must be traversed to reach the goal.

The spatial affordance field, $A_{\text{spatial}}(\mathbf{X})$ provides fitness values for all points in space. However, we require the fitness for all possible directions which serve as our steering choices. We choose the fitness values of points immediately surrounding the agent as the values for direction affordance, $A_{\text{direction}}(\theta)$.

5.3 Selection Phase – Optimal Affordance Selection

Once the fitness is computed for all speed and direction affordances, the final step is to select the speed and direction having optimal fitness. The target speed, s_{target} maximizes $A_{\text{speed}}(s)$, *i.e.* it maxmimizes the distance from all imminent threats:

$$s_{\text{target}} = \arg\max_{\text{s}} A_{\text{speed}}(s) \tag{20}$$

The target direction, $\mathbf{D}_{\text{target}}$ is estimated by rotating the current direction by an angle, θ_{target} which is computed as follows:

$$\theta_{\text{target}} = \arg\max_{\theta} A_{\text{direction}}(\theta) \tag{21}$$

In the discretized implementation of our model, selecting the direction having maximum fitness produces vibrations because the fitness of adjacent directions may oscillate over key frames. We offset this undesirable effect by performing quadratic interpolation for fitness over a window of directions, and maximizing the interpolated fitness. Let $Y(\theta) = A\theta^2 + B\theta + C$ be a quadratic equation that maps angular displacements to fitness. The value of θ for which $Y(\theta)$ is maximized is simply given by $\theta_{\text{target}} = -B/2A$.

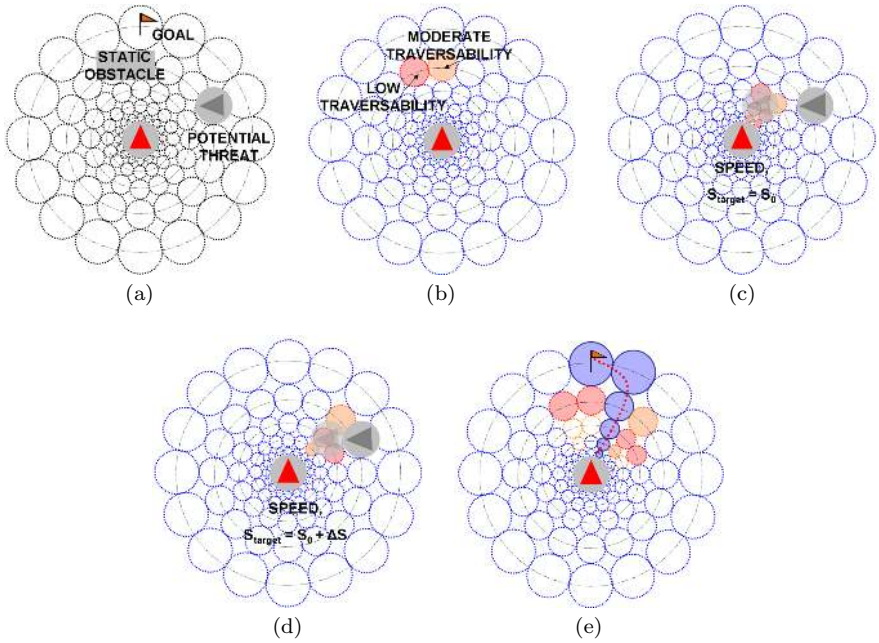

Fig. 4 The Steering Algorithm: (a) The current state of the environment. (b) Static Perception Field indicating low traversability at position of obstacle. (c) Dynamic Perception Field for speed $s_{\text{target}} = s_0$. (d) Dynamic Perception Field for speed $s_{\text{target}} = s_0 + \Delta V$, which avoids the dynamic obstacle. (e) Resulting affordances indicating a path of high fitness to goal.

6 Virtual Human Simulation Framework

This section presents the integration of our steering technique into a complete framework. Our pedestrian simulation framework comprises three modules: (1) *Pathfinding*: the process of determining a series of waypoints from the start position to a target location (Section 6.1), (2) *Steering*: the use of discrete egocentric fields to steer the agents along the planned path (Section 6.2) and, (3) *Animation*: the process of animating virtual humans that follow the position and orientation trajectories (Section 6.3).

6.1 Pathfinding

Recall that agents dynamically scale their egocentric fields to include the local goal. The resolution could become too coarse if we allowed this goal to be too far away. To avoid this problem, we first compute a series of waypoints $\{\mathbf{w}_i\}$ (a long-term path) using A* search [16] that leads an agent to its goal position which may lie outside the boundaries of the field. The next waypoint $\mathbf{w_i}$ is given as input to the steering algorithm to initialize the local goal, $\mathbf{X}_{\text{goal}}$.

6.2 Steering

Given the local goal, we employ our egocentric affordance fields approach to find an effective control decision. Sections 3, 4 and 5 presented the foundations of our approach, and the following algorithm describes how these pieces are implemented together:

1. Determine goal position of an agent, $\mathbf{X}_{\text{goal}}$.
2. Initialize node weights, $w(l) = 1, \forall\ l$. If the goal falls outside the field, perform dynamic scaling using Equations 7 and 11.
3. Estimate time-layer associativity, $t[i]$, for all m layers, where i iterates from 1 to m. $t[i]$ is the predicted time taken by the agent to travel a distance $r_{\text{layer}}[i]$, corresponding to layer i, at the current speed.
4. Populate the static perception field, P_{static}.
5. Populate dynamic threat perception field, P_{dynamic}, at the current speed.
6. Populate local dynamic threat perception field, $P_{\text{local-dynamic}}$ using Equation 13.
7. Populate velocity perception field, $\mathbf{P}_{\text{velocity}}$.
8. Populate relative velocity perception field, $\mathbf{P}_{\text{relative-velocity}}$ using Equation 15.
9. Generate speed affordance fields, $A_{\text{speed}}(s)$, using Equation 16
10. Compute the new target speed s_{target}, which maximizes the fitness of the speed affordance, using Equation 20.
11. Re-estimate time-layer associativity, dynamic fields and velocity fields at new target speed.
12. Generate direction affordance fields, $A_{\text{direction}}(\theta)$ (Equations 17-19).
13. Compute target direction, $\mathbf{D}_{\text{target}}$ (Section 5.3).

The above mentioned steps are executed at every time step for each agent. s_{target} and $\mathbf{D}_{\text{target}}$ are the control decisions made by our system. The steering simulation results in position and orientation trajectories of each agent which are input to the animation system (Section 6.3) for providing high-quality results of animated humans.

6.3 Animation

The animation system is given the position and orientation trajectories of the agents. It produces a set of blended animations such that the virtual human moves along the specified trajectories. We use a simple finite-state machine of animations to simulate the virtual humans that are seen in the results (Figure 5). Our state machine has the following animation states: (1) `Walk`, (2) `Stop`, (3) `WalkToStop` and, (4) `StopToWalk`. In addition, we generate a `SlowWalk` animation by reducing the speed of the `Walk` animation. The state machine transitions are based on speed thresholds, and the animations are also played at different speeds to match the simulated speed of the agents. To seamlessly patch between animations, we linearly blend from one animation to the next. Each animation is composed of three pieces, a beginning overlap,

(a) (b) (c) (d)

Fig. 5 (a): Agents walking through a hallway. (b): Queue formation as agents enter narrow passageway. (c): A simulation of a forest-like scenario with a large number of agents and obstacles. (d): 5000 agent simulation with random initial positions and goals.

a non-overlapping section, and an ending overlapping section. This method animates smoothly and without foot-skate, while following the control outputs of our egocentric affordance fields method.

7 Evaluation

In this section, we describe the methodology (Section 7.1) and results (Section 7.2) of our approach, followed by a discussion (Section 7.3) of the results.

7.1 Methodology

Testing, evaluating and analyzing steering simulations is known to be a difficult task, primarily due to the lack of automated tools. SteerBench [46, 47] proposes a way of objectively evaluating and comparing the quality of steering solutions. It provides: (1) a set of test cases that can be used to exercise the steering algorithm over a wide variety of scenarios and, (2) a scoring method that can be used to gain insight into the simulation and serve as the basis for comparison between different approaches. As part of our work, we demonstrate our algorithm using the test cases and scoring methods from SteerBench, to serve as the basis for comparison with other approaches.

The test cases we used for testing represented a diverse set of navigation tasks, described as follows:
Similar direction: Agents traveling in similar directions, with slightly differing goals.
Crossing threats: Agents crossing paths, at various angles, in the presence of obstacles.
Oncoming threats: Agents traveling in opposite directions, with a potential for head-on collisions, with obstacles in the way.
Curves: Agents having to travel along a curved path to avoid obstacles.
Group-interactions: Agents traveling in groups, with other agents cutting across.
Squeeze: 2-4 agents, passing through a narrow hallway, with same or opposite directions (Figure 8).
Doorway: Agents having to pass through a narrow doorway.

Overtake: An agent, encountering a slower moving agent in front, while traveling through a narrow passageway. (Figure 9)
Confusion: Agents traveling in opposite directions, arriving at the same place, at approximately the same time. (Figure 6)
Hallway: A large number of agents, passing through a hallway, in either direction. (Figure 5 (a))
Bottleneck squeeze: Agents enter through a narrow passageway (Figure 5 (b))
Forest: A large number of agents, with random goals, in an obstacle laden environment. (Figure 5 (c))
Random: A large number of agents, with random initial positions and goals. (Figure 5 (d))
Urban: A large number of agents, with random goals, in an environment with large obstacles, resembling large buildings.

Using these test cases and some additional group behavior tests, we evaluated the qualitative behaviors of our framework, discussed below. We also computed SteerBench benchmark scores, using the `composite01` benchmark technique. The three metrics used in the `composite01` benchmark technique are: (1) average number of collisions per agent, (2) average time in seconds that an agent spends to reach its goal, and (3) average kinetic energy. A weighted sum of these metrics is used to provide a single score for a steering algorithm on a particular test case. Collisions are given a high weight (50.0) while the other two metrics are given a weight of 1.0. For more information on how the SteerBench score is computed, please refer [47].

7.2 Results

In this section, we evaluate our approach by comparing it with two state of the art steering techniques and one baseline reactive approach:

- **PPR.** The work in [48,46] presents a hybrid framework that combines reaction, prediction and planning into one single framework.
- **RVO.** The work in [3] proposes the use of reciprocal velocity obstacles to serve as a linear model of prediction for collision avoidance in crowds.
- **Reactive.** This steering technique employs the use of a simple finite state machine of rules to govern the behavior of an autonomous agent in a crowd. This technique is purely reactive in nature and does not employ the use of any form of predictive collision avoidance. The implementation of this technique is similar to the collision avoidance strategy described in [42].

These three approaches represent a sampling of the wide spectrum of techniques that have been used for simulating crowds, ranging from predictive models, rule-based techniques, and hybrid frameworks. Table 1 lists the SteerBench scores for our framework as well as these three techniques to serve as a basis of comparison.

Our steering framework is able to successfully simulate 40 out of 42 test cases that are present in SteerBench (we do not consider the large-scale test cases that are useful for stress testing the steering framework). We observe similar scores in all steering techniques for the simple, crossing and, oncoming scenarios. The PPR and reactive technique have collisions in the `Cut-across-1` scenario where an agent has to cut through a group of other agents. In `Surprise-1` and `Surprise-2`, agents are unable to detect the presence of other agents that are not in their line of sight (blocked by obstacles) and hence are unable to predictively avoid collisions; this is the expected result for these test cases. The `Overtake` and `Overtake-obstacle` scenarios are successfully solved by RVO and our framework. The confusion scenarios show unexpected results where the reactive technique performs almost as well as the other algorithms which use predictions to avoid collisions. The squeeze scenarios are much more challenging where agents need to steer in narrow passageways with oncoming threats. Our framework successfully solves 4 of the 6 scenarios without collisions. However, the agents reach a deadlock in the `Wall-squeeze` and `Doorway-two-way` scenarios as the agents arrive at the narrow entrance together and are unable to back away to let one agent through. The results of the SteerBench analysis show that our algorithm can efficiently handle a wide variety of scenarios and is competitive with the current state of the art in steering.

Group Behaviors. Introducing a high-level layer into our framework, which assigns intermediate goals to agents, provides a simple and intuitive way for implementing common group behaviors. The intermediate goals can be dynamic (e.g. other agents) or static (e.g. location in space). To implement the group behaviors demonstrated in the video, agents automatically choose a dynamic goal to be the closest agent in front of itself. The following examples of group behaviors are demonstrated:

Lane Formation and Queueing: When several agents are given the same goal, agents with no-one in front simply steer towards the goal. Agents with others in front begin to follow the agents immediately in front. The strictness of the lane is defined by a 'comfort distance' between agents. As the agents near the goal, they 'queue' up politely. (Figure 5 (b))

Snake Motion: We demonstrate snake-like motion by having a leader weaving around a set of obstacles, and each previous agent follows the next one.

Group persistence: Persistence of groups is demonstrated in the Oncoming-Groups scenario (Figure 10). Agents perceive the oncoming group as a single entity, because of the variable resolution fields. As a result, the two groups steer around each other. *Leader Following and Group Reformation*: Figure 7 shows a group of agents following a leader as they enter through passageways. The group breaks up as they steer around the obstacles and reform as they continue to follow the leader.

Additional behaviors can be implemented by varying parameters (e.g., desired speed) or adding fields to the framework. For example, an additional field could be defined based on social constraints, such as "prefer to stay on the sidewalk" or "avoid a scary group of people". Aggressive and polite behaviors,

Test Case	Time	Energy	Ego	RVO	PPR	Reactive
Simple-1	11.1	240.5	251.6	266.3	254.8	254.8
Simple-2	7.3	137.4	144.7	141.3	145.6	145.6
Simple-3	5.7	112.0	117.8	114.4	118.8	117.9
Simple-obstacle-1	6.7	126.4	133.1	130.5	133.9	133.9
Simple-obstacle-2	14.2	253.1	267.3	265.7	268.2	268.2
Similar-direction	37.3	633.9	671.2	672.3	672.2	672.2
Simple-wall	23.9	415.4	439.3	450.1	440.1	474.5
Curves	21.5	363.9	385.4	431.2	385.5	385.5
Crossing-1	14.6	246.7	261.3	259.7	261.4	259.5
Crossing-2	14.2	246.9	261.0	257.5	261.6	259.4
Crossing-3	16.3	282.9	299.2	295.7	299.4	297.3
Crossing-4	16.3	277.9	294.1	291.2	294.7	292.5
Crossing-5	20.6	279.7	300.3	296.8	297.2	297.15
Crossing-6	22.2	277.8	298.3	298.9	295.3	295.3
Crossing-obstacle	13.9	227.9	241.8	268.0(0.5)	244.2	247.1
Crossing-trick	5.6	95.0	114.2	111.1	115.2	115.2
Oncoming-1	14.4	254.8	269.2	265.4	270.0	268.6
Oncoming-2	14.3	253.6	267.9	266.4	268.7	267.5
Oncoming-3	15.0	253.6	268.5	265.4	268.7	267.4
Oncoming-4	14.4	254.8	269.2	265.4	270.1	268.6
Oncoming-obstacle	16.8	267.9	284.7	289.6(0.5)	284.1	276.8
Oncoming-trick	6.6	120.9	127.5	124.8	178.9	621.9
Oncoming-groups	41.7	598.9	640.5	643.8	637.5	638.8
Fan-in	17.3	236.5	254.1	267.3	253.9	255.8
Fan-out	32.3	519.9	552.2	549.5	551.3	551.3
Cut-across-1	32.6	505.9	538.6	545.8	571.9(0.6)	551.2(0.3)
Cut-across-2	33.6	505.9	539.5	546.8	537.1	536.9
Surprise-1	24.3	350.5	424.8(1)	408.8(0.5)	484.9(2)	403.9
Surprise-2	25.7	353.8	429.5(1)	401.6	407.2	406.6
Overtake	17.6	273.4	290.9	306.5	Fail	Fail
Overtake-obstacle	16.9	279.3	296.2	300.3	Fail	Fail
3-way-conf-1	17.5	276.2	293.7	293.4	293.4	292.7
3-way-conf-2	15.6	254.7	270.3	267.0	270.0	263.6
4-way-conf-obs	15.8	253.6	269.4	261.7	271.7	293.5(0.5)
4-way-conf	15.3	253.4	268.7	267.0	269.1	265.6
Frogger	14.2	227.6	241.8	240.4	241.6	241.4
Squeeze	19.45	315.0	334.5	332.9	335.2	382.8(1)
3-squeeze	20.77	312.9	333.7	360.3(0.6)	366.6(0.6)	396.9(1.3)
Double-squeeze	22.63	308.4	331.1	371.3(1)	354.6(0.5)	379.5(1)
Doorway-one-way	20.35	313.4	333.5	334.9	333.2	332.7
Doorway-two-way	–	–	Fail	331.4	Fail	Fail
Wall-squeeze	–	–	Fail	Fail	434.4(2)	Fail

Table 1 Evaluation Results using SteerBench. Lower score is better. (1) Time: Average time per agent in reaching goal (seconds). (2): Energy: Total energy spent per agent ($kg \cdot m^2/s^2$). (3) Ego: The cumulative score of our method, computed as a weighted sum of average number of collisions, time, and energy. Collisions are given a weight of 50.0 while time and energy are given a weight of 1.0. (4) RVO: Score for Reciprocal Velocity Obstacles [3]. (5) PPR: Score for hybrid framework that uses planning, prediction and, reaction for steering [48, 46]. (6) Reactive: Score for a reactive technique that uses a set of simple rules for steering. For 3,4,5,6 the number in () is the average number of collisions per agent. For more information on how the SteerBench score is computed, please refer [47].

Fig. 6 Four agents (rendered as circular discs), traveling in opposite directions form a vortex as they maneuver around an obstacle.

such as agents being pushy or patient, can be modeled by affecting the computation of traversability fields of agents. For example, a pushy agent's position as well as future position would have low traversability in the computation of other agents fields, while a timid person would perceive other agents as dynamic obstacles with greater obstacles.

7.3 Discussion

In this section, we discuss the results of our framework on the scenarios described in 7.1. Agents exhibit a wide variety of behaviors including local interactions between agents as well as group behaviors, some of which are described below:

Local Agent Interactions. Agents steer naturally around each other, with and without obstacles. This is shown in all scenarios, particularly the Crossing, Oncoming, Confusion, and Curves scenarios.
Human-like Behaviors. Natural reactions are also captured by our framework. For example in the Surprise scenarios (with sharp turns), agents do not see each other until the collision is imminent. In such cases, behavior is affected by each individual's visual field. Macro-scale crowd simulations with global knowledge cannot model this. Our framework models this individuality successfully.
Implicit Space-time Planning. The importance of space-time planning is demonstrated in doorway, overtake, confusion, and squeeze (narrow passage) scenarios. Comparisons of behaviors with and without implicit space-time for the 3-way confusion and overtaking scenarios show that the natural, anticipatory behaviors are a result of our implicit space- time planning. In general, we observe that space-time planning is essential for complicated interactions involving 3 or more agents. In contrast, we observe that purely reactive approaches suffer from agents colliding with one another or fail to arrive at a solution, resulting in deadlocks.
Crowd Behaviors. We also demonstrate several bottleneck and densely crowded scenarios where agents cooperatively wait or steer around each other. Note that many previous approaches steer unnaturally into each other and into obstacles, relying on collision resolution and "greedy" reactive steering decisions to progress the agents. We do not explicitly prevent agents from colliding and overlapping; collision avoidance is purely a result of our steering algorithm.

Fig. 7 Agents exhibiting leader following and group re-formation.

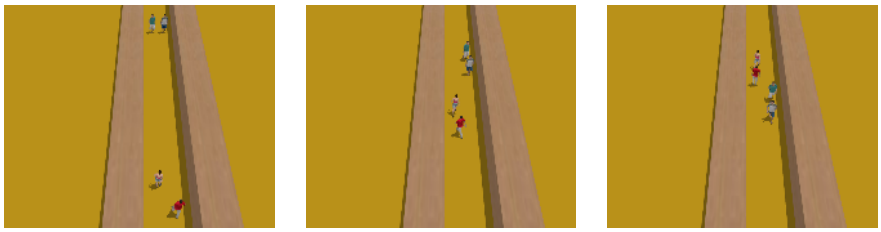

Fig. 8 Two agents traveling in the same direction encounter two oncoming agents in a narrow passageway. Results demonstrated with animated humans.

Fig. 9 An agent overtakes another agent while traveling through a narrow passageway with an obstacle.

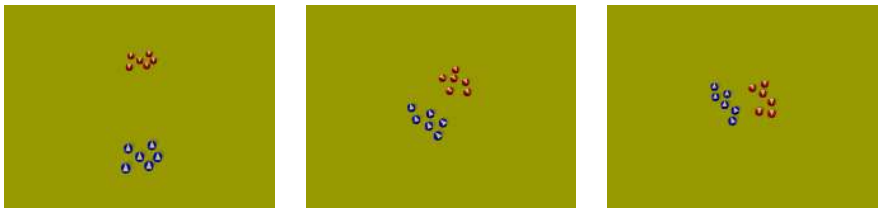

Fig. 10 Group persistence is observed as two groups of oncoming agents efficiently steer around each other.

8 Parallelization

Most agent-based crowd simulation techniques, including ours, require synchronization at an agent level. This is because agents read each others' data to react and predict around each other. In this section we show that our approach can be parallelized effectively. We investigate parallelization strategies to achieve near-linear speedup with increase in processors by leveraging the egocentric nature of our fields. Section 8.1 presents the performance results

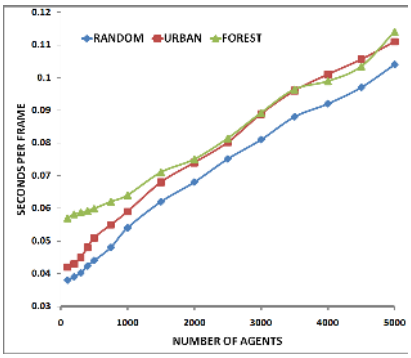

Fig. 11 Time of update for stress cases. Number of nodes per layer, n = 16. Number of layers, m = 8.

and memory requirements of our algorithm on a single thread. Section 8.2 discusses the rationale behind our choice of decomposition strategies, while Section 8.3 describes the different scheduling and synchronization methods. Finally, Section 8.4 presents the performance results of the parallel version of our algorithm for different benchmarks and crowd densities.

8.1 Performance Analysis on Single Thread

Each agent has associated with it a set of fields that serve as its memory repository. We observe that a field with 8 layers and 16 nodes per layer is sufficient to perform effective steering in a virtual environment. It takes $2.5 - 3$ KB of memory per agent to store the information of these fields. This is about 3 MB of memory per 1000 agents, which we believe to be a manageable overhead that increases linearly with increase in crowd size. We tested the runtime performance of our algorithm using three stress test scenarios (`Random`, `Forest`, and `Urban`) on a 2.66 GHz Core 2 processor with a single thread. The time of update per agent for each of these scenarios is outlined in Figure 11. We observe a frame rate of approximately 25 frames per second for up to 500 agents which decreases to 10 frames per second for 5000 agents. The computation time linearly increases with the number of agents, and is independent of the complexity of the environment. In the following sections, we present the parallelized version of our algorithm which is capable of handling much larger numbers of agents in real-time.

8.2 Decomposition Strategies

The current trend is for processors to deliver high performance through multithreading by exploiting multiples cores in their architectures. We investigate the following decomposition techniques for partitioning our problem for par-

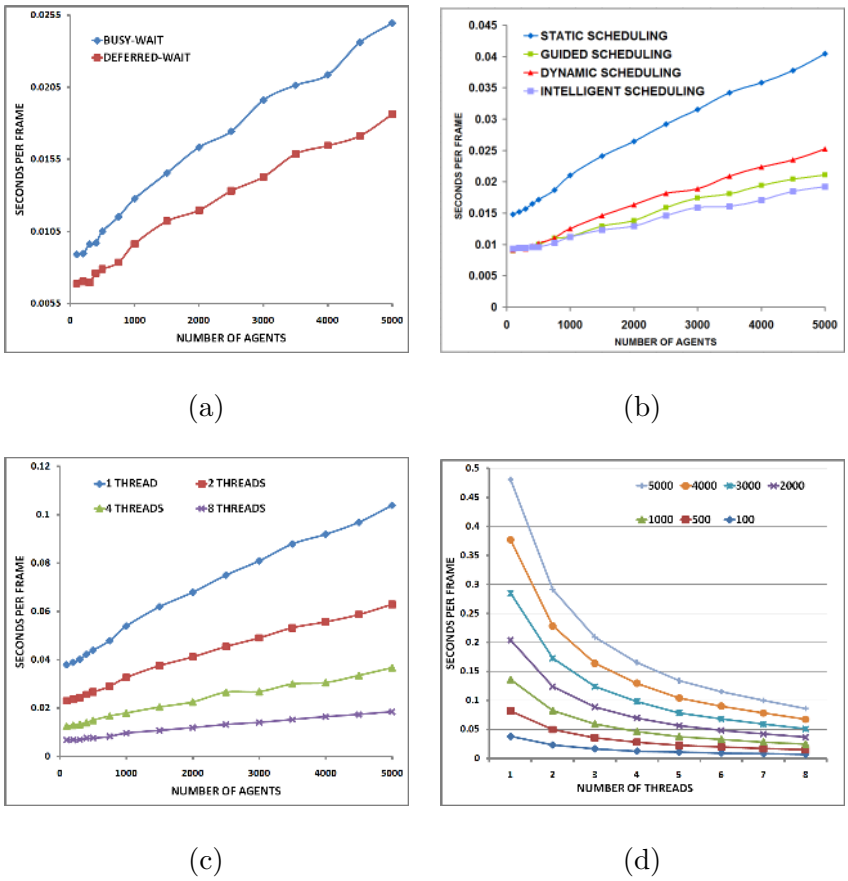

Fig. 12 Parallelization results: (a) Comparison of synchronization strategies for `Random` test case, 8 threads, guided scheduling. (b) Comparison of scheduling strategies for `Random` test case, 8 threads, deferred-wait synchronization. (c) Performance comparison with increase in number of threads for `Random` test case, intelligent guided scheduling, busy-wait synchronization. (d) Performance evaluation of 100, 500, 1000, 2000, 3000, 4000 and, 5000 agents with increase in number of threads.

allelization: (1) Task decomposition and, (2) Data decomposition.

Task Decomposition. In this approach, the task to be executed is decomposed into smaller independent sub-tasks and each sub-task is executed in parallel. Table 2 outlines the profiling results of the function blocks in a single update step for one agent. We observe that the population of direction affordance fields, $A_{\text{direction}}(\theta)$ takes up approximately 71% of the total execution. This procedure is described in Section 5.2. The recursive nature of this procedure makes it an unlikely candidate for parallelization. Hence, we turn our focus to decomposing the problem based on agents.

Task	Execution time (ms)	Execution profile	Need for Synchronization
Query Grid Database	0.0136	4%	No
Populate dynamic perception field	0.0142	7%	Yes
Compute speed affordance field	0.0211	9%	No
Compute direction affordance field	0.1681	71%	No
Total	0.2337	100%	Yes

Table 2 Task Decomposition of Steering Algorithm.

Data Decomposition. It is often possible to partition the data associated with a task into pieces that can be processed independently of each other, thus allowing multiple instances of the task to execute concurrently. For the purpose of crowd simulation, this entails farming of a set of agents on different threads. The steering algorithm described above, operates on a per-agent basis in a distributed manner which makes it amenable for data-level parallelism. However, there are two syncrhonization issues. First, all agents independently make a steering decision and execute the decision by updating their position in the environment by writing to a centralized grid database. We resolve this by seperating the computationally inexpensive position update operation from the rest of the AI and executing it serially for all agents. Second, the agents read data from the fields of neighboring agents while computing the dynamic perception field, $P_{\text{dynamic}}(\mathbf{X})$. The dynamic fields allow agents to perform implicit space-time planning and handle complex agent-agent interactions. This inter-agent communication prevents simply splitting agents and farming them off to different threads.

We implement synchronization between agents using a simple read-write locking scheme in which there are multiple readers and a single writer. Each agent has a lock associated with it. Whenever an agent wishes to write to its internal data structures, it yields for any neighboring agents that are currently reading from it. Similarly, future readers yield to agents that are currently updating their state. Section 8.3 discusses the different scheduling strategies for allocating agents to threads.

8.3 Scheduling Strategies

There are two main issues that affect scalability of performance with increase in number of threads:

– **Load Balancing.** Agents in different parts of the scenario may require different processing time based on the density of agents and obstacles in the neighboring region. Hence, an equal partitioning of agents between threads may suffer from severe load-imbalance. The main reason is that the agents assigned to a particular thread may finish their computation early,

while the remaining ones are still performing computations. For this reason, we investigate dynamic partitioning to agents for better load balancing between threads.
- **Synchronization.** Spatially co-located agents need to read each others data while making their steering decisions. If neighboring agents are dispatched onto different threads, the synchronization overhead may result in a decrease in speedup.

In this section, we investigate the effect of performance of our parallel algorithm using the following four scheduling strategies:

Static Scheduling. In static scheduling, the number of agents are divided equally among the available threads before execution. Static scheduling is suited for decomposition strategies where the units of decomposition are load balanced.

Dynamic Scheduling. In dynamic scheduling, the number of agents are not divided equally among the threads. Instead, contending threads acquire agents to be processed from a pool of pending agents. Once a thread finishes its allocated processing, it picks another agent from this pool. This form of scheduling is particularly suited for decomposition strategies where the units of decomposition require varying computational resources.

Guided Scheduling. In guided scheduling, a large chunk of agents are allocated to each thread dynamically, similar to dynamic scheduling. The chunk size decreases exponentially with each successive allocation to a minimum specified size. This strategy is similar to the method used in [37] for parallelizing the social forces model.

Intelligent Guided Scheduling. To minimize the synchronization overhead between agents, agents that are allocated to the same thread are spatially co-located. This ensures that neighboring agents that read each others data are more likely to be processed on the same thread. The size of chunks is determined in a manner similar to guided scheduling. Intelligent scheduling requires a kd-tree data structure to be maintained that partitions agents based on their spatial locations, which incurs a computational overhead.

8.4 Parallelization Results

Table 3 describes the setup used for running the parallelization experiments described below. Figure 12(a) illustrates the comparison between busy-wait and deferred-wait synchronization for the `Random` test case using 8 threads and guided scheduling. We observe that a deferred-waiting strategy performs consistently better than busy-waiting with increase in number of threads. The average time for one update step of the above scheduling strategies using

Hardware Specification	Intel® Xeon® Processor 7030 processor-based platform with2 dual-core processors, each with Intel® Hyper-Threading Technology
Operating System	Red Hat Linux 4.1.2
Programming Language	C++ and OpenMP
Software Base	SteerSuite [46]

Table 3 Experimental Setup for Parallelization Results.

deferred-wait synchronization is shown in Figure 12(b). Static scheduling suffers from load imbalance between threads due to agents taking widely different amounts of processing time based on their surrounding environment configuration. Hence, we observe a large difference in performance between static scheduling and the other strategies.

The intelligent scheduling strategy allocates spatially co-located agents on the same thread. Hence, two agents that read each others' data (because they are spatially co-located) will have fewer synchronization problems because they are likely to be updated serially. As a result, intelligent guided scheduling outperforms the other scheduling strategies. However, there is an extra computational overhead of spatially ordering agents at every frame by maintaining a kd-tree data structure which takes about $2-5$ ms of processing time per frame.

Figure 12(c) illustrates the performance of intelligent guided scheduling and deferred-wait synchronization with increase in number of threads. Figure 12(d) evaluates the performance of our approach by increasing the number of agents from 100 to 5000 for $1-8$ threads. The less-than-linear speedup with increase in number of threads is due to the synchronization between agents at the boundaries of the spatial partitions. As we increase the density of agents in the environment, the number of agents querying the data structures of neighboring agents increases which increases the synchronization overhead. However, the egocentric nature of the fields ensures that agents will only request read locks of other spatially co-located agents that fall within the boundary of their field.

9 Conclusion

We have presented egocentric affordance fields that quantify the relative strength of all possible actions that an agent can take over a set of view-dependent models of the environment. The implicit dependency on time that these fields have, allows our agent navigation framework to successfully resolve complex dynamic situations, such as certain deadlocks that arise in narrow corridors and openings.

We have evaluated our framework using SteerBench which provides us with a large number of test cases that exercise the steering system against basic validation scenarios, oncoming and crossing threats, local agent-agent interactions and, large-scale stress tests. We also perform a detailed emperical comparison

between the proposed method and established prior art [3,48]. The results of our analysis show that our algorithm can efficiently handle a wide variety of scenarios and is competitive with the current state of the art in steering.

Limitations and Future Work. The system used to animate the virtual characters shown in the supplementary video is a simple finite state machine that chooses between four animations and uses linear blending to transition between animations. This results in certain artifacts such as discontinuities in character movement, unnecessary stopping and starting, and translation of the character even though there is no animation playing. For future work, we are investigating the use of parameterized motion graphs [18] and more advanced motion blending techniques [34].

Our method currently does not support lateral movements of the character (e.g. side-step), where the body of the virtual character is not tangential to the trajectory of the motion. This causes some unrealistic walking movements such as large changes in orientation where a small sidestep might have been sufficient. The use of footsteps [49] as an interface between steering and locomotion offsets these problems by allowing steering algorithms to have more fine-grained control over the locomotion of the virtual character.

The resolution detail of the field is concentrated in the region closest to the agent and decreases with increase in radial distance. One possible extension is to investigate the relationship between the foveal angle and the resolution of the field such that the portion of the environment directly ahead of the agent is considered most significant for steering. We are also investigating using a hierarchy of fields as an extension to our model, whereby each node can be equipped with a sub-egocentric field of the area it encompasses. This would increase the resolution of areas along the path that are further away, thus providing information detail where necessary.

The main focus of our work is in the realm of pedestrian simulation which is largely based in two dimensions. As illustrated in Section 3, our generic model is inherently in three dimensions, representing space with an implicit time dimension. In the future, we aim to implement a variable resolution 3-D model of the environment that may find use in a wide variety of applications not limited to steering.

Acknowledgements We wish to thank the anonymous reviewers for their comments. The work in this paper was partially supported by NSF grant No. CCF-0429983. We thank Intel Corp., Microsoft Corp., and AMD/ATI Corp. for their generous support through equipment and software grants.

References

1. Altun, K., Koku, A.: Evaluation of egocentric navigation methods. Robot and Human Interactive Communication, 2005. ROMAN 2005. IEEE International Workshop on pp. 396–401 (2005). DOI 10.1109/ROMAN.2005.1513811

2. Arkin, R.: Motor schema based navigation for a mobile robot: An approach to programming by behavior. In: Robotics and Automation. Proceedings. 1987 IEEE International Conference on, vol. 4, pp. 264 – 271 (1987). DOI 10.1109/ROBOT.1987.1088037
3. van den Berg, J., Lin, M.C., Manocha, D.: Reciprocal velocity obstacles for real-time multi-agent navigation. In: IEEE International Conference on Robotics and Automation, pp. 1928–1935. IEEE (2008)
4. van den Berg, J., Patil, S., Sewall, J., Manocha, D., Lin, M.: Interactive navigation of multiple agents in crowded environments. In: SI3D '08: Proceedings of the 2008 symposium on Interactive 3D graphics and games, pp. 139–147. ACM, New York, NY, USA (2008)
5. Boulic, R.: Relaxed steering towards oriented region goals. In: Motion in Games, First International Workshop, pp. 176–187 (2008)
6. Chao, G., Dyer, M.: Concentric spatial maps for neural network based navigation. Artificial Neural Networks, 1999. ICANN 99. Ninth International Conference on (Conf. Publ. No. 470) **1**, 144–149 vol.1 (1999)
7. Chenney, S.: Flow tiles. In: Proceedings of the ACM SIGGRAPH/EG Symposium on Computer Animation (2004). DOI http://doi.acm.org/10.1145/1028523.1028553
8. Clements, R.R., Hughes, R.L.: Mathematical modelling of a mediaeval battle: the battle of agincourt, 1415. Math. Comput. Simul. **64**(2), 259–269 (2004)
9. Dechter, R., Pearl, J.: Generalized best-first search strategies and the optimality af a*. J. ACM **32**(3), 505–536 (1985). DOI http://doi.acm.org/10.1145/3828.3830
10. Farenc, N., Schweiss, E., Kallmann, M., Aune, O., Boulic, R., Thalmann, D.: A paradigm for controlling virtual humans in urban environment simulations. Applied Artificial Intelligence **14**, 69–91 (1999)
11. Fiorini, P., Shiller, Z.: Motion Planning in Dynamic Environments Using Velocity Obstacles. The International Journal of Robotics Research **17**(7), 760–772 (1998). DOI 10.1177/027836499801700706. URL http://ijr.sagepub.com/cgi/content/abstract/17/7/760
12. Fleming, P.: Implementing a robust 3 dimensional egocentric navigation system. Master's thesis, Graduate School of Vanderbilt University (2005)
13. Gibson, J.J.: The Theory of Affordances. In Perceiving, Acting, and Knowing (1977)
14. Goldenstein, S., Karavelas, M., Metaxas, D., Guibas, L., Aaron, E., Goswami, A.: Scalable nonlinear dynamical systems for agent steering and crowd simulation (2001)
15. Greeno, J.G.: Gibson's affordances. Psychological Review pp. 336–342 (1994)
16. Hart, P., Nilsson, N., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. Systems Science and Cybernetics, IEEE Transactions on **4**(2), 100–107 (1968). DOI 10.1109/TSSC.1968.300136
17. Hart, P.E., Nilsson, N.J., Raphael, B.: Correction to "a formal basis for the heuristic determination of minimum cost paths". SIGART Bull. (37), 28–29 (1972). DOI http://doi.acm.org/10.1145/1056777.1056779
18. Heck, R., Gleicher, M.: Parametric motion graphs. In: Proceedings of the 2007 symposium on Interactive 3D graphics and games, I3D '07, pp. 129–136. ACM, New York, NY, USA (2007). DOI http://doi.acm.org/10.1145/1230100.1230123. URL http://doi.acm.org/10.1145/1230100.1230123
19. Helbing, D., Buzna, L., Johansson, A., Werner, T.: Self-organized pedestrian crowd dynamics: Experiments, simulations, and design solutions. Transportation Science **39**(1), 1–24 (2005). DOI http://dx.doi.org/10.1287/trsc.1040.0108
20. Helbing, D., Molnár, P.: Social force model for pedestrian dynamics. Phys. Rev. E **51**(5), 4282–4286 (1995). DOI 10.1103/PhysRevE.51.4282
21. Hoogendoorn, S.P.: Pedestrian travel behavior modeling. In: In 10th International Conference on Travel Behavior Research, Lucerne, pp. 507–535 (2003)
22. Kant, K., Zucker, S.W.: Planning collision-free trajectories in time-varying environments: a two-level hierarchy. The Visual Computer **3**(5), 304–313 (1988)
23. Kapadia, M., Singh, S., Hewlett, W., Faloutsos, P.: Egocentric affordance fields in pedestrian steering. In: Proceedings of the 2009 symposium on Interactive 3D graphics and games, I3D '09, pp. 215–223. ACM, New York, NY, USA (2009). DOI http://doi.acm.org/10.1145/1507149.1507185. URL http://doi.acm.org/10.1145/1507149.1507185

24. Lamarche, F., Donikian, S.: Crowd of virtual humans: a new approach for real time navigation in complex and structured environments. In: Computer Graphics Forum 23. (2004)
25. Lee, K.H., Choi, M.G., Hong, Q., Lee, J.: Group behavior from video: a data-driven approach to crowd simulation. In: Proceedings of the ACM SIGGRAPH/EG Symposium on Computer Animation, pp. 109–118 (2007)
26. Lerner, A., Chrysanthou, Y., Lischinski, D.: Crowds by example. Computer Graphics Forum **26**(3), 655–664 (2007)
27. Li, T.Y., Chen, P.F., Huang, P.Z.: Motion planning for humanoid walking in a layered environment. In: Proceedings of IEEE ICRA, vol. 3, pp. 3421–3427 (2003). DOI 10.1109/ROBOT.2003.1242119
28. Loscos, C., Marchal, D., Meyer, A.: Intuitive crowd behaviour in dense urban environments using local laws. In: TPCG '03: Proceedings of the Theory and Practice of Computer Graphics 2003, p. 122. IEEE Computer Society, Washington, DC, USA (2003)
29. Lovas, G.: Modeling and simulation of pedestrian traffic flow. In: Transportation Research Record, pp. 429–443 (1994)
30. Michael, D., Chrysanthou, Y.: Automatic high level avatar guidance based on affordance of movement. In: Eurographics 2003. Eurographics Association (2003)
31. Milazzo, J., Rouphail, N., Hummer, J., Allen, D.: The effect of pedestrians on the capacity of signalized intersections. In: Transportation Research Record, pp. 37–46 (1998)
32. Paris, S., Gerdelan, A., O'Sullivan, C.: Ca-lod: Collision avoidance level of detail for scalable, controllable crowds. In: MIG '09: Proceedings of the 2nd International Workshop on Motion in Games, pp. 13–28. Springer-Verlag, Berlin, Heidelberg (2009)
33. Paris, S., Pettré, J., Donikian, S.: Pedestrian reactive navigation for crowd simulation: a predictive approach. In: EUROGRAPHICS 2007, vol. 26, pp. 665–674 (2007)
34. Park, S.I., Shin, H.J., Shin, S.Y.: On-line locomotion generation based on motion blending. In: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation, SCA '02, pp. 105–111. ACM, New York, NY, USA (2002). DOI http://doi.acm.org/10.1145/545261.545279. URL http://doi.acm.org/10.1145/545261.545279
35. Pelechano, N., Allbeck, J., Badler, N.: Virtual Crowds: Methods, Simulation, and Control (Synthesis Lectures on Computer Graphics and Animation). Morgan and Claypool Publishers (2008)
36. Pelechano, N., Allbeck, J.M., Badler, N.I.: Controlling individual agents in high-density crowd simulation. In: Proceedings of the ACM SIGGRAPH/EG Symposium on Computer Animation, pp. 99–108 (2007)
37. Quinn, M.J., Metoyer, R.A., Hunter-zaworski, K.: Parallel implementation of the social forces model. In: in Proceedings of the Second International Conference in Pedestrian and Evacuation Dynamics, pp. 63–74 (2003)
38. Reynolds, C.: Steering Behaviors for Autonomous Characters. In: Game Developers Conference 1999 (1999)
39. Reynolds, C.W.: Flocks, herds and schools: A distributed behavioral model. In: Proceedings of ACM SIGGRAPH, pp. 25–34. ACM, New York, NY, USA (1987)
40. Rodrigues, R.A., Lima Bicho, A., Paravisi, M., Jung, C.R., Magalhães, L.P., Musse, S.R.: Tree paths: A new model for steering behaviors. In: Proceedings of the 9th International Conference on Intelligent Virtual Agents, IVA '09, pp. 358–371. Springer-Verlag, Berlin, Heidelberg (2009)
41. Rudomín, I., Millán, E., Hernández, B.: Fragment shaders for agent animation using finite state machines. Simulation Modelling Practice and Theory **13**(8), 741–751 (2005)
42. Shao, W., Terzopoulos, D.: Autonomous pedestrians. In: Proceedings of the ACM SIGGRAPH/EG Symposium on Computer Animation, pp. 19–28 (2005)
43. Shao, W., Terzopoulos, D.: Autonomous pedestrians. Graph. Models **69**, 246–274 (2007). DOI 10.1016/j.gmod.2007.09.001. URL http://portal.acm.org/citation.cfm?id=1323742.1323926
44. Shapiro, A., Kallmann, M., Faloutsos, P.: Interactive motion correction and object manipulation. In: I3D '07: Proceedings of the 2007 symposium on Interactive

3D graphics and games, pp. 137–144. ACM, New York, NY, USA (2007). DOI http://doi.acm.org/10.1145/1230100.1230124
45. Shimoda, S., Kuroda, Y., Iagnemma, K.: Potential field navigation of high speed unmanned ground vehicles on uneven terrain. Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on pp. 2828–2833 (2005)
46. Singh, S., Kapadia, M., Faloutsos, P., Reinman, G.: An open framework for developing, evaluating, and sharing steering algorithms. In: MIG '09: Proceedings of the 2nd International Workshop on Motion in Games, pp. 158–169. Springer-Verlag, Berlin, Heidelberg (2009)
47. Singh, S., Kapadia, M., Faloutsos, P., Reinman, G.: Steerbench: a benchmark suite for evaluating steering behaviors. In: Computer Animation and Virtual Worlds, pp. 533–548 (2009)
48. Singh, S., Kapadia, M., Hewlett, W., , Glenn Reinmann, P.F.: A modular framework for adaptive agent-based steering. In: Proceedings of the 2011 symposium on Interactive 3D graphics and games, I3D '11. ACM (2011)
49. Singh, S., Kapadia, M., Reinmann, G., Faloutsos, P.: On the interface between steering and animation for autonomous characters. In: In Workshop on Crowd Simulation, Computer Animation and Social Agents. Saint-Malo, France (2010)
50. Sud, A., Gayle, R., Andersen, E., Guy, S., Lin, M., Manocha, D.: Real-time navigation of independent agents using adaptive roadmaps. In: VRST '07: Proceedings of the 2007 ACM symposium on Virtual reality software and technology, pp. 99–106. ACM, New York, NY, USA (2007)
51. Surasmith, S.: Preprocessed solution for open terrain navigation. In: AI Game Programming Wisdom, pp. 161–170 (2002)
52. Takeuchi, R., Unuma, M., Amakawa, K.: Creating and animating the virtual world. chap. Path planning and its application to human animation system, pp. 163–175. Springer-Verlag New York, Inc., New York, NY, USA (1992). URL http://portal.acm.org/citation.cfm?id=141248.141259
53. Tecchia, F., Loscos, C., Conroy, R., Chrysanthou, Y.: Agent behaviour simulator (abs): A platform for urban behaviour development. In: In GTEC2001, pp. 17–21 (2001)
54. Thalmann, D., Musse, S.R.: Crowd simulation. Springer (2007)
55. Torrens, D.P.M.: Behavioral intelligence for geospatial agents in urban environments. In: IAT '07: Proceedings of the 2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, pp. 63–66. IEEE Computer Society, Washington, DC, USA (2007). DOI http://dx.doi.org/10.1109/IAT.2007.37
56. Treuille, A., Cooper, S., Popović, Z.: Continuum crowds. ACM Trans. Graph. **25**(3), 1160–1168 (2006). DOI http://doi.acm.org/10.1145/1141911.1142008
57. Trovato, K.I., Dorst, L.: Differential a*. IEEE Trans. on Knowl. and Data Eng. **14**(6), 1218–1229 (2002). DOI http://dx.doi.org/10.1109/TKDE.2002.1047763
58. Tsubouchi, T., Kuramochi, S., Arimoto, S.: Iterated forecast and planning algorithm to steer and drive a mobile robot in the presence of multiple moving objects. In: IROS '95: Proceedings of the International Conference on Intelligent Robots and Systems-Volume 2, p. 2033. IEEE Computer Society, Washington, DC, USA (1995)
59. Turner, A., Penn, A.: Encoding natural movement as an agent-based system: an investigation into human pedestrian behaviour in the built environment. Environment and Planning B: Planning and Design **29**, 473–490 (2002). URL http://eprints.ucl.ac.uk/73/
60. Warren, C.: Global path planning using artificial potential fields. In: Proceedings of IEEE ICRA, vol. 1, pp. 316–321 (1989). DOI 10.1109/ROBOT.1989.100007
61. Warren, C.: Multiple robot path coordination using artificial potential fields. In: Proceedings of IEEE ICRA, vol. 1, pp. 500–505 (1990). DOI 10.1109/ROBOT.1990.126028