Available online at www.sciencedirect.com

ScienceDirect

Graphical Models 69 (2007) 246–274

Graphical Models

www.elsevier.com/locate/gmod

Autonomous pedestrians

Wei Shao [a], Demetri Terzopoulos [b,*]

[a] Google Inc., Kirkland, WA, USA
[b] Computer Science Department, University of California, Los Angeles, CA, USA

Received 12 April 2006; received in revised form 18 September 2007; accepted 24 September 2007
Available online 18 Ocotber 2007

Abstract

We address the challenging problem of emulating the rich complexity of real pedestrians in urban environments. Our artificial life approach integrates motor, perceptual, behavioral, and cognitive components within a comprehensive model of pedestrians as *individuals*. Featuring innovations in these components, as well as in their combination, our model yields results of unprecedented fidelity and complexity for fully autonomous multihuman simulation in a large urban environment. We represent the environment using hierarchical data structures, which efficiently support the perceptual queries that influence the behavioral responses of the autonomous pedestrians and sustain their ability to plan their actions on local and global scales.
© 2007 Elsevier Inc. All rights reserved.

Keywords: Artificial life; Virtual humans; Autonomous characters; Behavioral animation; Cognitive modeling

1. Introduction

"Forty years ago today at 9 a.m., in a light rain, jack-hammers began tearing at the granite walls of the soon-to-be-demolished Pennsylvania Station, an event that the editorial page of The New York Times termed a "monumental act of vandalism" that was "the shame of New York.""

(*Glenn Collins, The New York Times, 10/28/03*)

The demolition of New York City's original Pennsylvania Station (Fig. 2), which had opened to the public in 1910, in order to make way for the Penn Plaza complex and Madison Square Garden, was "a tragic loss of architectural grandeur". Although state-of-the-art computer graphics enables a virtual reconstruction of the train station with impressive geometric and photometric detail, it does not yet enable the automated animation of its human occupants with anywhere near as much fidelity. Our research addresses this difficult, long-term challenge.

In a departure from the substantial literature on so-called "crowd simulation", we develop a decentralized, comprehensive model of pedestrians as autonomous *individuals* capable of a broad variety of activities in large-scale synthetic urban spaces. Our artificial life approach to modeling humans spans the modeling of pedestrian appearance, locomotion, perception, behavior, and cognition [35]. We deploy a multitude of self-animated virtual pedestrians within a large environment model, a

* Corresponding author. Fax: +1 310 794 5057.
E-mail address: dt@cs.ucla.edu (D. Terzopoulos).
URL: http://www.cs.ucla.edu/~dt (D. Terzopoulos).

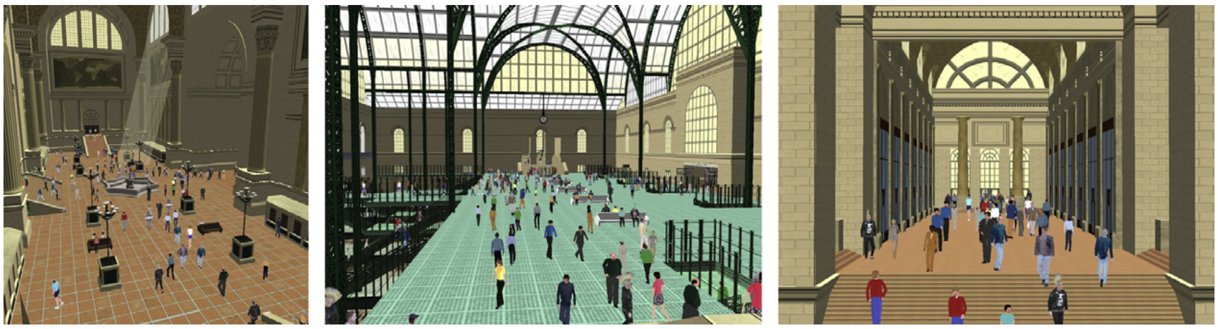
Fig. 1. A large-scale simulation of a virtual train station populated by self-animated virtual humans. From left to right are rendered images of the main waiting room, concourses, and arcade.

Fig. 2. Historical photos of the original Pennsylvania Station in New York City.

VR reconstruction of the original Penn Station (Fig. 1). The environment model includes hierarchical data structures that support the efficient interaction between numerous pedestrians and their complex virtual world through fast perceptual query algorithms that sustain pedestrian navigation on local and global scales.

We continue with a review of related work in Section 2. Section 3 briefly reviews our virtual environment model. In Section 4, we present our autonomous pedestrian model, mostly focusing on its (reactive) behavioral and (deliberative) cognitive components. Additional details regarding the autonomous pedestrian and environmental models are provided in Appendices A,B,C,D. Section 5 describes the simulation of the models. Section 6 presents results comprising long-term simulations with well over 1000 pedestrians and reports on performance. Finally, Section 7 draws conclusions and discusses future work.

2. Related work

Human animation is an important and challenging problem in computer graphics [2]. Psychologists and sociologists have been studying the behavior and activities of people for many years. Closer to home, pedestrian simulation has recently begun to capture the attention of CG researchers [1,20]. The topic has also been of some interest in the field of artificial life [4], as well as in architecture and urban planning [16,28] where graphics researchers have assisted in visualizing planned construction projects, including pedestrian animation [8,19].

In pedestrian animation, the bulk of prior research has focused on synthesizing natural locomotion (a problem that we do not consider in this paper) and on path planning (one that we do). The seminal work of Reynolds [23] on behavioral animation is certainly relevant to our effort, as is its further development in work by other researchers [37,36,17]. Behavioral animation has given impetus to an entire industry of applications for distributed (multiagent) behavioral systems that are capable of synthesizing flocking, schooling, herding, etc., behaviors for lower animals, or in the case of human characters, crowd behavior. Low-level crowd interaction models have been developed in the sciences [10,12,3,27] and by animation researchers [11,15,33,38,14] and also in the movie industry by Disney and many other production houses, most notably in recent years for horde battle scenes in feature films (see www.massivesoftware.com).

While our work is innovative in the context of behavioral animation, it is very different from so-called "crowd animation". As the aforementioned literature shows, animating large crowds, where one character algorithmically follows another in a stolid manner, is relatively easy. We are uninterested in crowds *per se*. Rather, the goal of our work is to develop a comprehensive, self-animated model of *individual* human beings that incorporates nontrivial human-like abilities suited to the purposes of animating virtual pedestrians in urban environments. Our approach is inspired most heavily by the work of [37] on artificial animals and by [9] on cognitive modeling for intelligent characters that can reason and plan their actions. We further develop their comprehensive artificial life approach and adopt it for the first time to the case of autonomous virtual humans that can populate extensive urban spaces. In particular, we pay serious attention to deliberative human activities over and above the reactive behavior level.

3. Virtual environment model

The interaction between a pedestrian and his/her environment plays a major role in the animation of autonomous virtual humans in synthetic urban spaces. This, in turn, depends heavily on the representation and (perceptual) interpretation of the environment. Recently, Lamarche and Donikian [14] proposed a suitable structuring of the geometric environment and reactive navigation algorithms for pedestrian simulation. While this part of our work is conceptually similar, our methods differ. We have devoted considerable effort to developing a large-scale (indoor) urban environment model, which is described in detail in Appendix A and elsewhere [29], and which we summarize next.

We represent the virtual environment by a hierarchical collection of maps. As illustrated in Fig. 3, the model comprises (i) a topological map which represents the topological structure between different parts of the virtual world. Linked within this map are (ii) perception maps, which provide relevant information to perceptual queries, and (iii) path maps, which enable online path-planning for navigation.

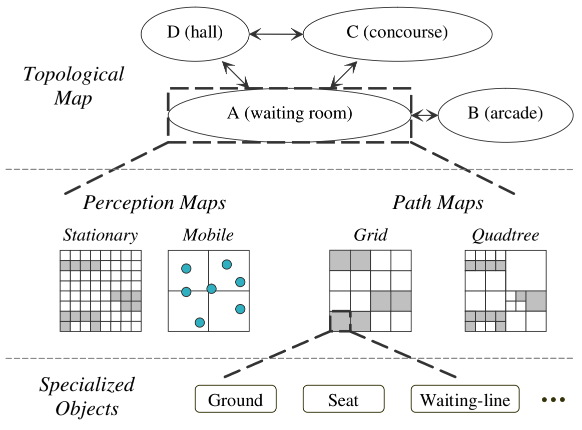

Fig. 3. Hierarchical environment model.

In the topological map, nodes correspond to environmental regions and edges represent accessibility between regions. A region is a bounded volume in 3D space (such as a room, a corridor, a flight of stairs or even an entire floor) together with all the objects inside that volume (e.g., ground, walls, benches). The representation assumes that the walkable surface in a region may be mapped onto a horizontal plane without loss of essential geometric information. Consequently, the 3D space may be adequately represented within the topological map by several 2D, planar maps, thereby enhancing the simplicity and efficiency of environmental queries.

The perception maps include grid maps that represent stationary environmental objects on a local, per region basis, as well as a global grid map that keeps track of mobile objects, usually other pedestrians. These uniform grid maps store information within each of their cells that identifies all of the objects occupying that cellular area. The typical cell size of the grid maps for stationary object perception is 0.2–0.3 m. Each cell of the mobile grid map stores and updates identifiers of all the agents currently within its cellular area. Since it serves simply to identify the nearby agents, rather than to determine their exact positions, it employs cells whose size is commensurate with the pedestrian's visual sensing range (currently set to 5 m). The perception process will be discussed in more detail in Section 4.2.

The path maps include a quadtree map which supports global, long-range path planning and a grid map which supports short-range path planning. Each node of the quadtree map stores information about its level in the quadtree, the position of the area covered by the node, the occupancy type (ground, obstacle, seat, etc.), and pointers to neighboring nodes, as well as information for use in path planning, such as a distance variable (i.e., how far the node is from a given start point) and a congestion factor (the portion of the area of the node that is occupied by pedestrians). The quadtree map supports the execution of several variants of the A^* graph search algorithm, which are employed to compute quasi-optimal paths to desired goals (cf. [5]). Our simulations with numerous pedestrians indicate that the quadtree map is used for planning about 94% of their paths. The remaining 6% of the paths are planned using the grid path map, which also supports the execution of A^* and provides detailed, short-range paths to goals in the presence of obstacles, as necessary. A typical example of its use is when a pedestrian is behind a chair or bench and must navigate around it in order to sit down.

With each of the aforementioned maps specialized to a different purpose, our environment model is efficient enough to support the real-time (30 fps) simulation of about 1400 pedestrians on a 2.8 GHz Xeon PC with 1 GB memory. For the details about the construction and update of our environment model and associated performance statistics regarding its use in perception and path planning, we refer the reader to Appendix A and [29,31].

4. Autonomous pedestrian model

Like real humans, our synthetic pedestrians are fully autonomous. They perceive their virtual environment, analyze environmental situations, make decisions, and behave naturally. Our autonomous human characters are architected as a hierarchical artificial life model. Progressing upward through the levels of abstraction, our model incorporates appearance, motor, perception, behavior, and cognition sub-models. The following sections discuss each of these components in turn.

4.1. Human appearance, movement, & motor control

As an implementation of the low-level appearance and motor levels, we employ a human animation software package called *DI-Guy*, which is commercially available from Boston Dynamics Inc. It provides textured 3D human characters with basic motor skills, such as standing, strolling, walking, running, sitting, etc. [13]. DI-Guy characters are by no means autonomous, but their actions may be scripted manually using an interactive tool called *DI-Guy Scenario*, which we do not use. DI-Guy also includes an SDK that allows external C/C++ programs to control a character's basic motor repertoire. This SDK enables us to interface DI-Guy to our extensive, high-level perceptual, behavioral, and cognitive control software, which will be described in subsequent sections, thereby achieving fully autonomous pedestrians.

Emulating the natural appearance and movement of human beings is a difficult problem and, not surprisingly, DI-Guy suffers from several limitations. The 3D character appearance models are insufficiently detailed. More importantly, DI-Guy characters cannot synthesize the full range of motions needed to cope with a highly dynamic urban environment. With the help of the *DI-Guy Motion Editor*, we have modified and supplemented the motion repertoire, enabling faster action transitions, which better enables our pedestrians to deal with busy urban environments.

Moreover, we have implemented a motor control interface between the kinematic layer of DI-Guy and our higher-level behavioral controllers. The interface accepts motor control commands from behavior modules, and it verifies and corrects them in accordance with the pedestrian's kinematic limits. It then selects an appropriate motion sequence or posture and calls upon the kinematic layer to update the state of the character. Our seamless interface hides the details of the underlying kinematic layer from our higher-level behavior routines, allowing the latter to be developed largely independently. Hence, in principle, any suitable low-level human animation API can easily replace DI-Guy in our future work.

4.2. Perception

An autonomous and highly mobile virtual human must have a perceptive regard of its environment. Our environment model (Section 3) efficiently provides accurate perceptual data in response to the queries of autonomous pedestrians.

4.2.1. Sensing ground height

In the static object perception map, each map cell contains the height functions of usually a single though sometimes multiple ground objects, such as the floor, stairs, etc. The highest object at the desired foot location of a pedestrian is returned in constant time and it is processed within the pedestrian's motor layer, which plants the foot at the appropriate height.

4.2.2. Sensing static objects

The visual sensing computation shoots out a fan of line segments, with length determining the desired perceptual range and density determining the desired perceptual acuity (Fig. 4(a)–(b)). Grid cells on the perception map along each line are interrogated for their associated object information. This perceptual query takes time that grows linearly with the length of each line times the number of lines but,

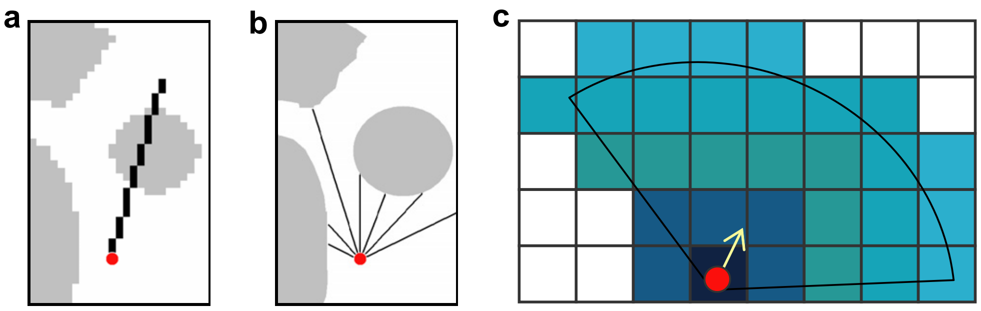

Fig. 4. Perception. (a) and (b) A pedestrian (denoted by the red circle) perceives stationary objects, (a) by examining map cells along the rasterized eye ray, while (b) perceiving the broader situation by shooting out a fan of eye rays (rasterization not shown). (c) Sensing mobile objects by examining (color-coded) tiers of the sensing fan. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

most importantly, it does not depend on the number of objects in the virtual environment.

4.2.3. Sensing mobile objects

To sense mobile objects (mostly other humans), a pedestrian must first identify nearby pedestrians within the sensing range. The range here is defined by a fan as illustrated in Fig. 4(c). On the mobile object perception map, the cells wholly or partly within the fan are divided into "tiers" based on their distance to the pedestrian. Closer tiers are examined earlier. Once a predefined number (currently set to 16) of nearby pedestrians are perceived, the sensing is terminated. This is motivated by the fact that, at any given time, people usually pay attention only to a limited number of other people, usually those that are most proximal.[1] Once the set of nearby pedestrians is sensed, additional information can be obtained by referring to finer maps, estimation, or simply querying some pedestrian of particular interest. Given the sensing fan and the upper bound on the number of sensed pedestrians, sensing is a constant-time operation.

4.2.4. Locating an object

Given a location identifier (e.g., "Track 9"), a search at the object level can find the virtual object. This is accomplished in constant time using a hash map with location names as keys. As the virtual object has an upward reference to its region (e.g., "under the lower concourse"), it can be quickly located by referring to the node in the topological graph, as can nearby objects in that region (say, "Platform 9" and "Platform 10") by referring to the perception maps linked within the node.

4.2.5. Interpreting complex situations

Abstract interpretation of the environment is indispensable for performing higher-level, motivational behaviors (Section 4.3). For example, in order to get a ticket, a pedestrian should assess (1) the length of the queues at ticketing areas and pick a short one, (2) the last pedestrian waiting in line in order to join the queue, (3) when (s)he has become the first person in line, and (4) whether a ticket booth has become available to make the purchase. These perceptual queries are efficiently answered by the specialized environmental objects that keep track of the evolving situation, in the ticketing example these include a `queue` object and several `purchase-point` objects each associated with a ticket booth.

4.3. Behavioral control

Realistic behavioral modeling, whose purpose is to link perception to appropriate actions, is a big challenge in the case of autonomous virtual humans. Even for pedestrians, the complexity of any substantive behavioral repertoire is high. Considerable literature in psychology, ethology, artificial intelligence, robotics, and artificial life is devoted to the subject. Following [37], we adopt a bottom-up strategy that uses primitive reactive behaviors as building blocks that in turn support more complex motivational behaviors, all controlled by an action selection mechanism.

4.3.1. Basic reactive behaviors

Reactive behaviors appropriately connect perceptions to immediate actions. We have developed

[1] Because of the limited processing capacity of the brain, people become consciously aware of but a small fraction of the available sensory input, as determined by their focus of attention, which is partly a subconscious and partly a voluntary selection process [6].

six key reactive behavior routines, each suitable for a specific set of situations in a densely populated and highly dynamic environment (Fig. 5). Given that a pedestrian possesses a set of motor skills, such as standing in place, moving forward, turning in different directions, speeding up and slowing down, etc., these routines are responsible for initiating, terminating, and sequencing the motor skills on a short-term basis guided by sensory stimuli and internal percepts. The details of the six routines, denoted Routines A–F, are provided in Appendix B.

Several remarks regarding the routines are in order: Obviously, the fail-safe strategy of Routine E suffices in and of itself to avoid nearly all collisions between pedestrians. However, our experiments show that in the absence of Routines C and D, Routine E makes the dynamic obstacle avoidance behavior appear very awkward—pedestrians stop and turn too frequently and they make slow progress. Enabling Routines C and D, the obstacle avoidance behavior looks increasingly more natural. Interesting multiagent behavior patterns emerge when all the routines are enabled. For example, pedestrians will queue to go through a narrow portal. In a busy area, lanes of opposing pedestrian traffic will tend to form spontaneously after a short while.

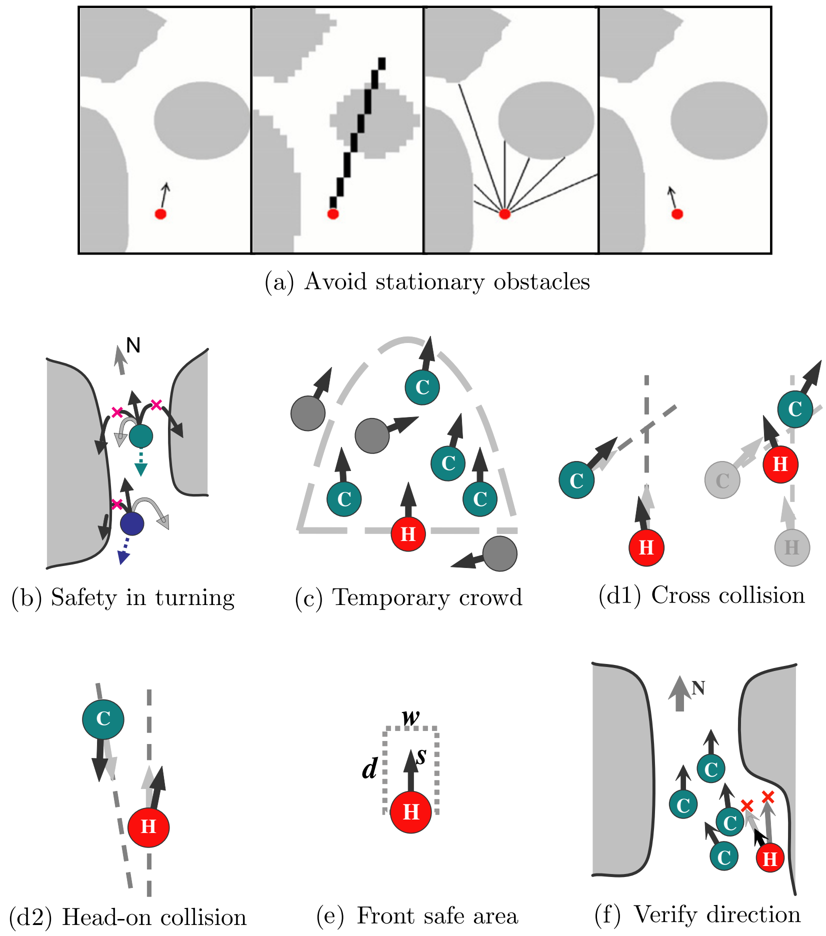

Fig. 5. Reactive behaviors. (a) (left to right) Original travel direction of pedestrian; detect stationary obstacle ahead by examining grid entries along the rasterized eye ray; perceive other nearby stationary obstacles by shooting additional eye rays; change direction and proceed. (b) Pedestrians choose best turning curves (gray) to turn southward. (c) Pedestrians within pedestrian *H*'s forward parabolic region traveling in similar directions as *H* (labeled *C*) are in *H*'s *temporary crowd*. (d1) To avoid *cross-collision*, (left) *H* slows down and turns toward *C* while *C* does the opposite until collision is cleared (right). (d2) To avoid *head-on collision*, both pedestrians turn slightly away from each other. (e) The dotted rectangle defines *H*'s *forward safe area*; *w* and *d* depend on *H*'s bounding box size and *d* is also determined by *H*'s current speed *s*. (f) Confronted by static and dynamic threats, *H* picks obstacle-free direction (light gray arrow) and slows down (black arrow) to let others pass before proceeding.

A remaining issue is how best to activate the six reactive behavior routines. Since the situation encountered by a pedestrian is always some combination of the key situations that are covered by the six routines, we have chosen to activate them in a sequential manner (Fig. 6), giving each an opportunity to alter the currently active motor control command, comprising speed, turning angle, etc. For each routine, the input is the motor command issued by its predecessor, either a higher-level behavior module (possibly goal-directed navigation) or another reactive behavior routine. The sequential flow of control affords later routines the chance to override motor commands issued by earlier routines, but this may cause the pedestrian to ignore some aspect of the situation, resulting in a collision. The problem can be mitigated by finding a ‘‘best’’ permutation ordering for activating the six routines. We have run many extensive simulations (longer than 20 min in virtual time) in the Penn Station environment with different numbers of pedestrians (333, 666, and 1000), exhaustively evaluating the performance of all 720 possible permutation orderings. The best permutation of the six routines, in the sense that it results in the fewest collisions while reasonable progress is still maintained in navigation, is C–A–B–F–E–D. Appendix C explains how we determined this best permutation.

4.3.2. Navigational behaviors

While the reactive behaviors enable pedestrians to move around freely, almost always avoiding collisions, navigational behaviors enable them to go where they desire, which is crucial for pedestrians. A pioneering effort on autonomous navigation is that by Noser et al. [21]. Metoyer and Hodgins [19] propose a model for reactive path planning in which the user can refine the motion by directing the characters with navigation primitives. We prefer to have our pedestrians navigate entirely on their own, as normal biological humans are capable of doing.

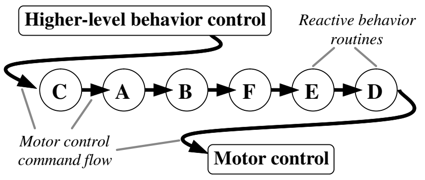
Higher-level behavior control
Reactive behavior routines
C
A
B
F
E
D
Motor control command flow
Motor control

Fig. 6. Activation of the reactive behavior routines in the best permutation order ‘‘C–A–B–F–E–D’’.

The online simulation of numerous pedestrians within large, complex environments, confronts us with many navigational issues, such as the realism of paths taken, the speed and scale of path planning, and pedestrian flow control through and around bottlenecks. We have found it necessary to develop a number of novel navigational behavior routines to address these issues. These behaviors rely in turn on a set of conventional navigational behavior routines, including moving forward, turning (in place or while moving), proceeding toward a target, and arriving at a target (see [25] for details).

In the Penn Station environment, large regions are connected by narrow portals and stairways, some of which allow only two or three people to advance comfortably side by side. These bottlenecks can easily cause extensive queueing, leading to lengthy delays. In our experience, available techniques, such as queuing in [25], self-organization in [12], and global crowd control in [20] cannot tackle the problem, as it involves highly dynamic two-way traffic and requires quick and flexible responses from pedestrians. In our solution, we employ two behavioral heuristics. First, pedestrians inside a bottleneck should move with traffic while trying not to impede oncoming pedestrians. Second, all connecting passageways between two places should be used in balance. The two behaviors that are detailed next enabled us to increase the number of pedestrians within the Penn Station model from under 400 to well over 1000 without any long-term blockage in bottlenecks.

4.3.2.1. Passageway navigation. In real life, if pedestrians are traveling in the same direction inside a narrow passageway, they will tend to spread out in order to see further ahead and maximize their pace. However, once oncoming traffic is encountered, people will tend to form opposing lanes to maximize the two-way throughput. Our virtual pedestrians incorporate a similar behavior. First, two imaginary boundaries are computed parallel to the walls with an offset of about half the pedestrian *H*’s bounding box size (Fig. 7(a)). Restricting *H*’s travel direction within a safety fan defined by the boundaries, as shown in the figure, guarantees that *H* stays clear of the walls. Second, if *H* detects that its current direction is blocked by oncoming pedestrians, it will search within the safety fan for a safe interval to get through (Fig. 7(b)). The search starts from *H*’s current direction and continues clockwise. If the search succeeds, *H* will move in

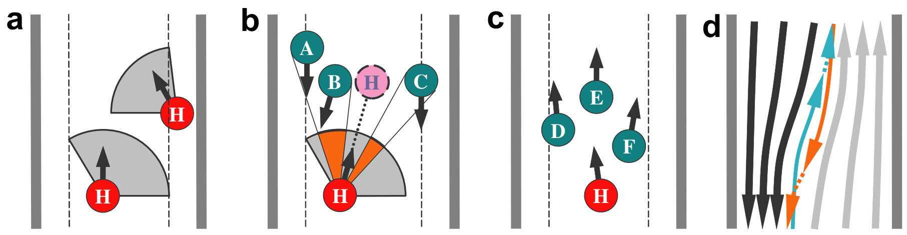

Fig. 7. Passageway navigation. (a) Two imaginary boundaries (dashed lines) and the safety fan. (b) Pedestrian H searches for a safe direction interval when confronted by oncoming traffic. (c) Spread out when no oncoming traffic is observed. (d) Typical flow of pedestrians in a passageway—big flows on the sides with small unblocking streams intermingling in the middle.

the safe direction found. Otherwise, H will slow down and proceed in the rightmost direction within the safety fan. This strategy allows non-blocking traffic to intermingle without resistance. However, in a manner that reflects the preference of real people in many countries, a virtual pedestrian will tend to squeeze to the right if it is impeding or impeded by oncoming traffic (Fig. 7(d)). Finally, Routine C (see Section B.3 in Appendix B) is used to maintain a safe separation between pedestrians travelling in the same direction. By altering their crowding factor w_i based on the observation of oncoming traffic, pedestrians can spread out or draw tightly to adapt to the situation (Fig. 7(c)).

4.3.2.2. Passageway selection. People are usually motivated enough to pick the best option from several available access routes, depending on both personal preferences and the real-time situation in and around those routes. Likewise, our pedestrians will assess the situation around stairways and portals, pick a preferred one based on proximity and density of pedestrians near it, and proceed toward it. They will persist in the choice they make, unless a significantly more favorable condition is detected elsewhere. This behavior, although executed independently by each individual, has a global effect of balancing the loads at different passageways.

Visually guided navigation among static obstacles is another important behavior for pedestrians. The following two behavioral routines accomplish this task on a local scale.

4.3.2.3. Perception-guided navigation among static obstacles. Given a path P (the global planning of paths will be explained in the next section), a *farthest visible point* p on P—i.e., the farthest point along P such that there is no obstacle on the line between p and the pedestrian H's current position—is determined and set as an intermediate target (Fig. 8). As H progresses toward p, it may detect a new farthest visible point that is even further along the path. This enables the pedestrian to approach the final target in a natural, incremental fashion. During navigation, motor control commands for each footstep are verified sequentially by the entire set of reactive behavior routines in their aforementioned order so as to keep the pedestrian safe from collisions.

4.3.2.4. Detailed "arrival at target" navigation. Before a pedestrian arrives at a target, a detailed

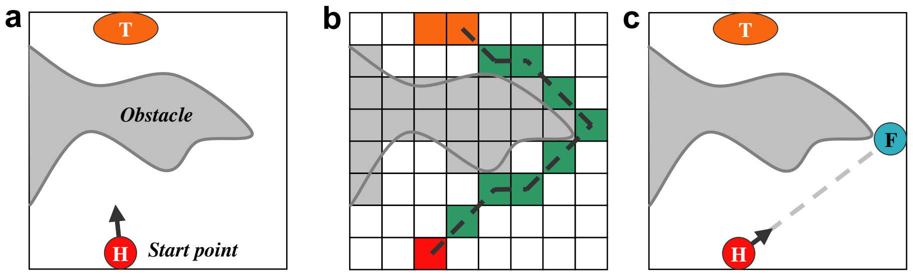

Fig. 8. Perception-guided navigation. (a) To reach target T, pedestrian H will (b) plan a jagged path on a path map (either grid or quadtree), (c) pick the *farthest visible point* (blue circle marked F) along the path, and proceed toward it. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

path will be needed if small obstacles intervene. Such paths can be computed on a fine-scale grid path map. The pedestrian will follow the detailed path strictly as it approaches the target, because navigational accuracy becomes increasingly important as the distance to the target diminishes. As some part of an obstacle may also be a part of or very close to the target, indiscriminately employing the reactive behaviors for static obstacle avoidance—Routines A and B (refer to Subsections B.1 and B.2 in Appendix B)—will cause the pedestrian to avoid the obstacle as well as the target, thereby hindering or even preventing the pedestrian from reaching the target. We deal with this by temporarily disabling the two routines and letting the pedestrian accurately follow the detailed path, which already avoids obstacles. Note that the other reactive behaviors, Routines C, D, and E, remain active, as does Routine F, which will continue to play the important role of verifying that modified motor control commands never lead the pedestrian into obstacles.

4.3.3. *Motivational behaviors*

The previously described behaviors comprise an essential aspect of the pedestrian's behavioral repertoire. To make our pedestrians more interesting, however, we have augmented the repertoire with a set of non-navigational, motivational behavior routines including, among others, the following:

- Meet with friends and chat;
- Select an unoccupied seat and sit down when tired;
- Approach and observe a performance when interested;
- Queue at a ticketing area or vending machine and make a purchase.

In the latter behavior, for example, a pedestrian joins a ticket purchase queue and stands behind its precursor pedestrian, proceeding forward until coming to the head of the queue. Then, the pedestrian will approach the first ticket counter associated with this queue that becomes available. Appendix D presents the details of several of the above behavior routines.

Note that these motivational behaviors depend on the basic reactive behaviors and navigational behaviors to enable the pedestrian to reach targets in a collision-free manner. Furthermore, these routines are representative of higher-level behaviors that involve potential competition among pedestrians for public resources. While each pedestrian generally tries to maximize their personal benefit, they also follow social conventions designed to maximize collective benefits, leading to natural, rational pedestrian behavior.

4.3.4. *Mental state and action selection*

Each pedestrian maintains a set of internal *mental state variables* that encodes the pedestrian's current physiological, psychological, or social needs. These variables include tiredness, thirst, curiosity, the propensity to be attracted by performances, the need to acquire a ticket, etc. When the value of a mental state variable exceeds a specified threshold, an action selection mechanism chooses the appropriate behavior to fulfill the need. Once a need is fulfilled, the value of the associated mental state variable begins to decrease asymptotically to zero.

We classify pedestrians in the virtual train station environment as commuters, tourists, law enforcement officers, performers, etc. Each pedestrian type has an associated action selection mechanism with appropriately set behavior-triggering thresholds associated with mental state variables. For instance, law enforcement officers on guard will never attempt to buy a train ticket and commuters will never act like performers. As a representative example, Fig. 9 illustrates the action selection mechanism of a commuter.

4.4. *Cognitive control*

At the highest level of autonomous control, a cognitive model is responsible for creating and executing plans suitable for autonomous pedestrians. Whereas the behavioral substrate described

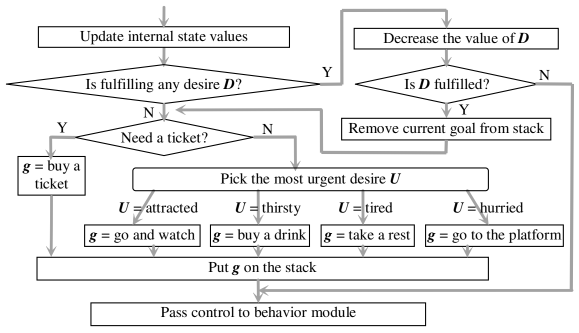

Fig. 9. Action selection in a commuter.

in the previous section is mostly *reactive*, the cognitive control layer [9] makes our pedestrian a *deliberative* autonomous agent that can exploit knowledge, reason about the world, and conceive and execute long-term plans, as real humans do.

The realism of pedestrian animation depends on the execution of rational actions at different levels of abstraction and at different spatiotemporal scales. At a high level of abstraction and over a large spatiotemporal scale, a pedestrian must be able to make reasonable global navigation plans that can enable it to travel purposefully and with suitable perseverance between widely separated regions of its environment, say, from the end of the arcade through the waiting room, through the concourses, and down the stairs to a specific train platform. Such plans should exhibit desirable properties, such as being relatively direct and saving time when appropriate. During the actual navigation, however, the pedestrian must have the freedom to decide to what extent to follow the plan, depending on the evolving situation, as we discussed when explaining the behaviors in Section 4.3.2. Priority must be given to the local (in both space and time) situation in the dynamic environment in order to keep the pedestrian safe while still permitting progress towards the long-range goal. Moreover, in a large, rich, and highly dynamic environment, an autonomous pedestrian should have the ability to fulfill not just a single goal, but possibly multiple goals. The pedestrian also needs the ability to decide whether and when a new plan is needed, which requires a bidirectional coupling between the behavioral layer and cognitive layer. With these insights, we apply three heuristics in the design of our pedestrian cognitive model:

1. Divide and conquer—decompose complex tasks into multiple simpler ones;
2. Think globally, but act locally—a plan is needed at the global level, but on the local level it will serve only as a rough guide; and
3. be flexible—always be prepared to modify local subplans in response to the real-time situation.

The subsequent sections present additional details of our cognitive model, starting with a brief description of the internal knowledge representation.

4.4.1. Knowledge of the virtual world

To make plans, especially path plans, a pedestrian must have a representation of the environment from its own egocentric point of view. For efficiency, the environment model serves as the internal knowledge base of the static portion of the virtual world for each pedestrian. Hence, except for tourists, every pedestrian knows the layout of the environment and the position of relevant static objects (such as walls, stairs, tracks, ticket booth, etc.), as well as the pedestrian's own current position within it (or *localization*, a fundamental problem in robotics [7]). However, each pedestrian also maintains, and updates at every simulation step, a representation of the dynamic aspect of the current world in the form of a list of perceived objects, including nearby pedestrians, and current situation-relevant events (e.g., "that ticket window is available", "that seat is taken", etc.).

This approach allows us to maintain only a single copy of the huge representation of the static world and still be able to support diverse cognitive and behavioral control for different pedestrians. While an over-simplification, this is reasonable for modeling pedestrians that are generally familiar with the urban space around them and is a sensible strategy for the real-time simulation of hundreds of pedestrians in extensive environments.

4.4.2. Planning

Planning is central to the cognitive model. Global path planning is an important planning task for pedestrians. Path planning directs a pedestrian to proceed through intermediate areas and reach ultimate destinations. In our pedestrian model, path planning is decomposed into subtasks at different spatiotemporal scales (Section A.4). The divide-and-conquer strategy allows the problem to be solved in a top-down manner. Each subtask can address its own portion of the overall problem in a suitable and efficient way, contributing to the final solution. Thus, the complete path planning process has global scale, yet reasonable local accuracy and economy.

As knowledge, the path planner instantiated within each individual pedestrian exploits the topological map at the top level of the environment model (Fig. 3). Given a pedestrian's current location and a target region, this map provides a set of optimal neighboring regions where the pedestrian can go. By applying path search algorithms within the path maps associated with each region known to each pedestrian, the pedestrian can plan a path from the current location to the boundary or portal between the current region and the next. The process is repeated in the next region, and so on, until it terminates at the target location. In this way, although the extent of the path is global, the processing is primarily local. Our path search algorithms (detailed in Appendix A and [31]), which are based on the well-known A^* graph search algorithm, are very efficient, but they provide rough paths—i.e., paths that are either jagged (grid path maps) or contain spikes (quadtree path maps)—as opposed to smooth, spline-like paths. Consequently, a pedestrian uses those rough navigational plans only as a guide and retains the freedom to locomote locally in as natural a manner as possible, as was described in Section 4.3.2.

In addition to the global path planner, the cognitive model incorporates other planning routines, or planners. Generally speaking, planners are like manuals that guide one through the steps needed to accomplish specific goals. Each step may also require further instructions. For instance, to "meet a friend", a pedestrian must first reach the meeting point and then wait for the friend. Accomplishing the first step requires a path planner. Once the planner finishes its work, control is handed over to the behavioral level to pursue the plan either until it is complete or until a more urgent plan arises. Continuity is maintained by the memory model.

4.4.3. Memory

A pedestrian can have many different activities inside a big train station. For instance, (s)he may have to purchase a ticket and meet with a friend before boarding a train. The route from station entrance to train track may be complicated and may require multiple strategies to navigate. On the way to the platform, (s)he may want to get a drink or to stop and watch a street artist performance, and so on. To keep important things in mind while doing others, a pedestrian needs memory. The memory model enables a pedestrian to:

- Store information—memorize intermediate and ultimate tasks;
- Retrieve information—remember pending tasks; and
- Remove information—forget accomplished tasks.

Unlike the pedestrian's static world knowledge, the memory model stores internal results directly from thinking or planning, including a list of preplanned goals (e.g., first purchase a ticket, then meet a friend, then catch the train), a sequence of subtasks decomposed from a major task (e.g., to catch a train, one must exit the arcade, cross the waiting room, pass one of the gates, cross the upper concourse, and descend the stairs to the platform), interruption/resumption (e.g., after stopping at the vending machine and performance, continue to the train platform), and so on. Therefore, the memory model is highly dynamic and usually long-term, and can vary dramatically in size.

Taking a simple approach, we use a stack as the memory. Our stack memory model precludes pedestrians from accomplishing multiple tasks in flexible (non-deterministic) ways, but real pedestrians probably do not normally exhibit such flexibility. However, the stack data structure features simple constant-time operations which permit easy and fast maintenance, regardless of size. This offers a big advantage in the real-time simulation of hundreds or thousands of pedestrians.

Algorithm 1. Skeleton routine for processing memory items

```
1: if the goal is accomplished then
2:    pop the goal from the memory stack
3: else if the goal has expired then
4:    pop the goal from memory stack
5: else if multiple subtasks are required to achieve the goal then
6:    create a plan containing these multiple subtasks based on the current situation
7:    push the first subtask on the memory stack marked with an appropriate expiration time
8: else
9:    determine the appropriate action needed to achieve the goal
```

Each item on the memory stack represents a goal and has a list of properties, including a descriptor of the goal type (e.g., catch a train, take a rest, reach a specific target point), the goal complexity (high or low), goal parameters (e.g., platform 10, position and orientation of the seat, position of the target point), preconditions (e.g., target must be within 0.5 m), and expiration time (e.g., for 3 s, until finished).

The top item on the stack is always the current goal. It will be processed by a cognitive or behavioral routine designed specifically for its type. This is done according to Algorithm 1, which decomposes complex goals into multiple simpler ones. Among these subtasks, only the first subtask is pushed onto the memory stack because task decomposition is accomplished according to the real-time situation (reflected, for instance, in the choice of goal parameters in subtasks). By the time the first subtask is accomplished, the situation may have changed and the remaining subtasks of the original decomposition may no longer be optimal or appropriate. However, when the top subtask is completed and removed from the memory stack, the original goal rises to the top and is processed again, and a new decomposition can be determined at the current time. To make pedestrians persistent to their previous choice, after the decomposition but prior to the pushing of the first subtask, the original complex goal on the top of the memory stack is updated with concise information about the decomposition. Hence, when the original goal is exposed again, this information reminds the pedestrian of the previous choice, and the pedestrian can choose to adhere to it, depending on the cognitive or behavioral routine that will process the goal.

Sometimes, the time needed to finish the first subtask can be lengthy. The expiration time attached to the memory item can cause the subtask to expire, re-exposing the original goal, thus guaranteeing frequent plan update. The expiration time also forces replanning when a task resumes after an interruption by another task. For example, if a pedestrian feels thirsty on the way to the train platform and decides to detour to a vending machine, (s)he needs to continue the trip to the platform after purchasing the drink. As the "get a drink" task is an interruption, there may be subgoals of "get to the platform" on the memory stack reflecting the original plan. Due to their expiration time, these subgoals may become invalid, which effectively forces a replanning of the "get to the platform" task. However, as it is an interruption, the "get a drink" task's expiration time is set in accordance with the spare time available before the pedestrian needs to proceed to the train platform.

4.4.4. Coupling the cognitive and behavioral layers

The goal stack of the deliberative, cognitive layer is also accessible to the underlying reactive, behavioral layer, thus coupling the two layers. If a goal is beyond the scope of the behavioral controller (for example, some task that needs path planning), it will be further decomposed into subgoals, allowing the cognitive controller to handle those subgoals within its ability (such as planning a path) and the behavioral controller to handle the others by initiating appropriate behavior modules (such as local navigation). The behavioral controller can also insert directives according to the internal mental state and environmental situation (e.g., if thirsty and a vending machine is nearby, then push "plan to get a drink"). This usually interrupts the execution of the current task and typically invalidates it. When it is time for the interrupted task to resume, a new plan is often needed. Intuitively, the goal stack remembers "what needs doing", the mental state variables dictate "why it should be done", the cognitive controller decides "how to do it" at a higher, abstract level, and the behavior controller determines "how to do it" at a lower, concrete level and ultimately attempts to "get it done".

5. Simulation

Fig. 10 shows the layered relationship of the various components within the pedestrian model, together with the world model. We will employ the figure to explain what happens at each *human simulation step* (HSS), which comprises several motion/rendering *frames*.

At the beginning of each HSS, the pedestrian's cognitive center retrieves the top memory item as the current goal g (arrow 2). If g is a complex goal (such as "meet a friend"), it will be decomposed by a planner into subtasks (e.g., "go to the meeting point" and "wait"). The first subtask will become the current task and will be pushed onto the memory stack (arrow 2). Knowledge (usually of the static world) will be used (arrow 1) during planning as needed (say, in planning a path). If the task is simple enough (arrows 6 and 7), the action selection mechanism will choose a suitable behavioral routine to handle it. The behavior center acquires information about the current environmental situation through the sensory processes (arrows 3 and 10). The sensory data are also used to update the internal knowledge representation (arrow 4). Additional behavior-relevant information is supplied by the cognitive center (arrow 5). The behavior center issues motor control commands (e.g., go forward with speed 1.2 m/s in the next step, turn 15 degrees left with walk speed 0.5 m/s in the next step, stand still, sit down, look at the clock, look left, etc.) down to the motor control interface (arrow 12) where they will first be verified and corrected in accordance with the motion abilities and kinematic limits of the pedestrian's body. They then manifest themselves as actual motions or motion transitions that the motor center selects from the motion repertoire

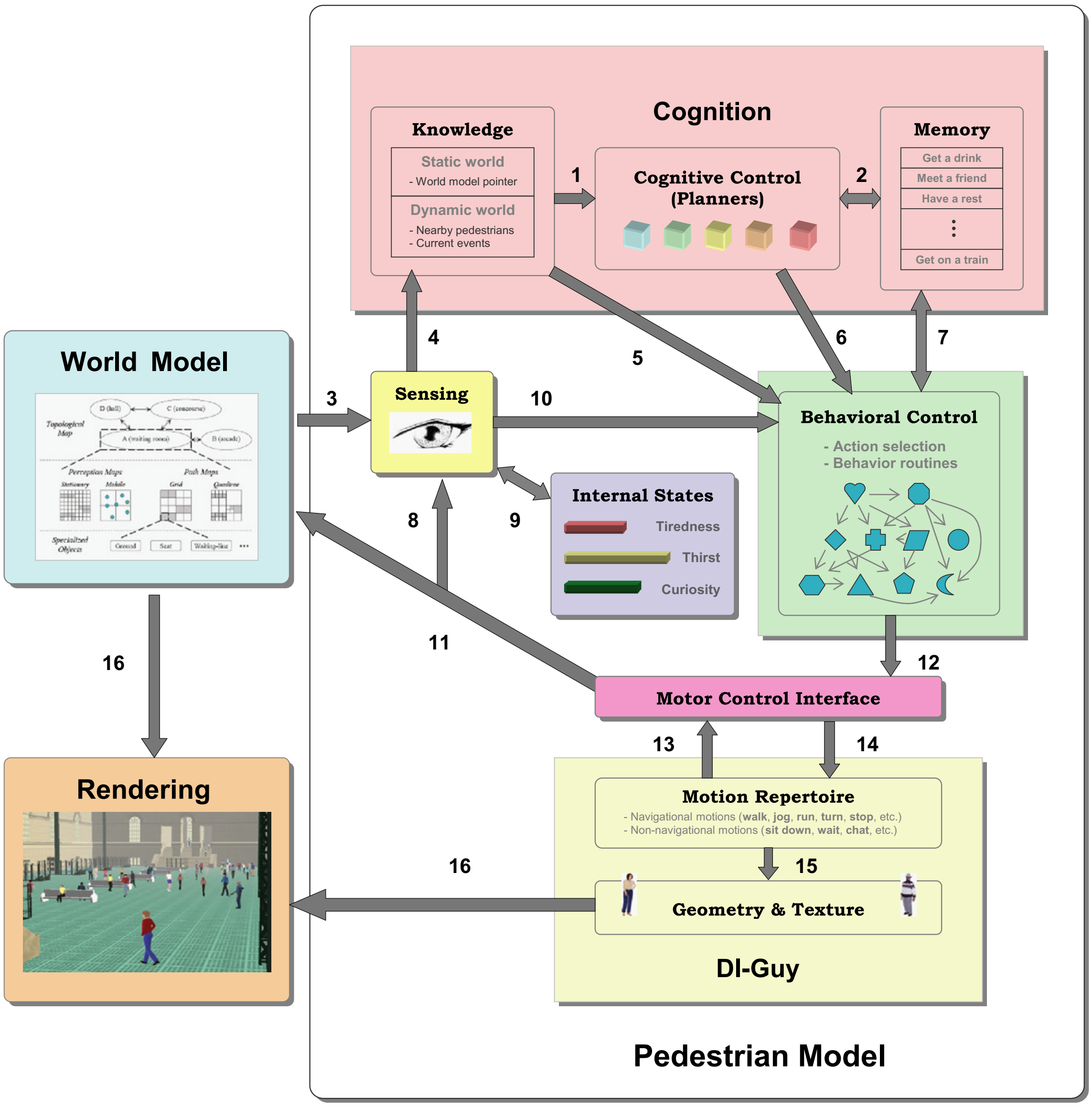

Fig. 10. Architecture of the pedestrian animation system.

(arrow 14), thus determining the pedestrian's pose sequence (arrow 15) for several subsequent *frames*. Feedback (arrow 13) from the motion level (such as a change of position/orientation, the current speed, whether in a seated state, etc.) is used to update the external world model (arrow 11) and it also becomes available proprioceptively to the perception center (arrow 8), thus resulting in a slightly different perception. This may modify the internal (mental) state variables (arrow 9), which may in turn trigger (arrow 10) the behavioral controller to initiate or terminate certain behaviors. The textured geometric human model is used to render the scene in each frame (arrows 16). After several frames, a new HSS will be processed.

Note from the figure that the motor control interface effectively hides the DI-Guy software from our higher-level controllers, which makes it easy, in principle, to replace it with other human animation packages.

6. Results

Our pedestrian animation system, which comprises about 50,000 lines of C++ code, is capable, without manual intervention, of running long-term simulations of pedestrians in a large-scale urban environment—specifically, the Penn Station environment. In our simulation experiments, we populate the virtual train station with five different types of pedestrians: commuters, tourists, performers, policemen, and patrolling soldiers. With every individual guided by his/her own autonomous control, these autonomous pedestrians imbue their virtual world with liveliness, social (dis)order, and a realistically complex dynamic.

In preprocessing the Penn Station environment model, the entire 3D space of the train station $(200(l) \times 150(w) \times 20(h)\ \mathrm{m}^3)$, which contains hundreds of architectural and non-architectural objects, was manually divided into 43 regions. At run time, the environment model requires approximately 90 MB of memory to accommodate the station and all of its associated objects.

6.1. Performance

We have run various simulation tests on a 2.8 GHz Intel Xeon system with 1 GB of main memory. The total length of each test is 20 min in virtual world time. Fig. 11 plots the computational load as the number of pedestrians in the simulation increases. The simulation times reported exclude rendering times and include only the requirements of our algorithms—environment model update and motor control, perceptual query, behavioral control, and cognitive control for each pedestrian—updating at 30 frames per virtual second. The figure shows that real-time simulation can be achieved for as many as 1400 autonomous pedestrians (i.e., 20 virtual world minutes takes 20 min to simulate). Although the relation is best fit by a quadratic function, the linear term dominates by a factor of 2200. The small quadratic term is likely due to the fact that the number of proximal pedestrians increases as the total number of pedestrians increases, but at a much lower rate. Fig. 12 breaks

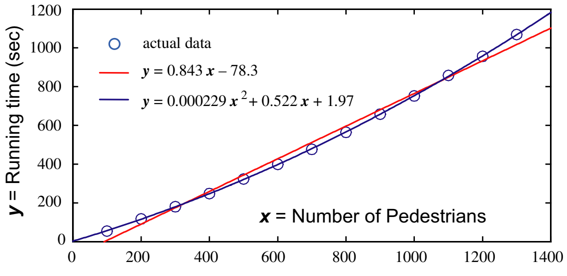

Fig. 11. Pure simulation time vs. number of pedestrians.

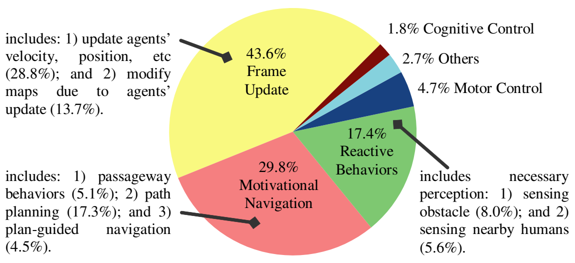

Fig. 12. Computational loads of the system's parts.

down the computational load for various parts of the simulation based on experiments with different numbers of pedestrians ranging from 100 to 1000 on the aforementioned PC. Fig. 13 tabulates the frame rates that our system achieves on the aforementioned PC with an NVIDIA GeForce 6800 GT AGP8X 256 MB graphics system. Due to the geometric complexity of the Penn Station model and its numerous pedestrians, rendering times dominate pedestrian simulation times.

6.2. Animation examples

We will now describe several representative simulations that demonstrate specific functionalities. To help place the animation scenarios in context, Fig. 14 shows a plan view of the Penn station model.

# of Pedestrians	**0**	**100**	**200**	**300**	**400**	**500**
Simulation only	n/a	64.4	32.2	23.0	16.9	12.3
Rendering only	21.0	12.5	9.2	7.6	6.0	5.4
Simulation+Rendering	21.0	10.5	7.2	5.7	4.4	3.8

Fig. 13. Frame rate (in frames/s) for pedestrian simulation only (including DI-Guy), rendering only (i.e., static pedestrians), and both simulation and rendering, with different numbers of pedestrians.

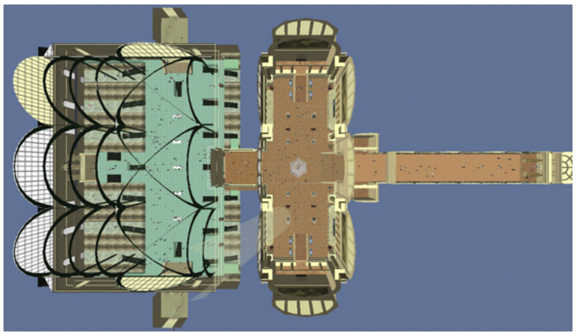

Fig. 14. Plan view of the Penn Station model with the roof not rendered, revealing the two-level concourses and the train tracks (left), the main waiting room (center), and the long shopping arcade (right).

6.2.1. Following an individual commuter

As we claimed in the introduction, an important distinction between our system and existing crowd simulation systems is that we have implemented a comprehensive human model, which makes every pedestrian a complete *individual* with a richly broad behavioral and cognitive repertoire. Figs. 15(a)–(f) and (g)–(l) show selected frames from a typical animation in which we choose a commuter and follow our subject as he enters the station (a), proceeds to the ticket booths in the main waiting room (b), and waits in a queue to purchase a ticket at the first open booth (c). Having obtained a ticket, he then (d) proceeds to the concourses through a congested portal, avoiding collisions. Next, our subject feels thirsty (e) and spots a vending machine in the concourse (f). He walks toward it and waits his turn to get a drink (g). Feeling a bit tired (h), our subject finds a bench with an available seat, proceeds towards it, and sits down (i). Later, the clock chimes the hour (j) and it is time for our subject to rise from his seat and proceed to the correct train platform. He makes his way through a somewhat congested area by following, turning, and stopping as necessary in order to avoid colliding with other pedestrians. He passes by some dancers that are attracting interest from many other pedestrians (k), but in this particular instance our subject has

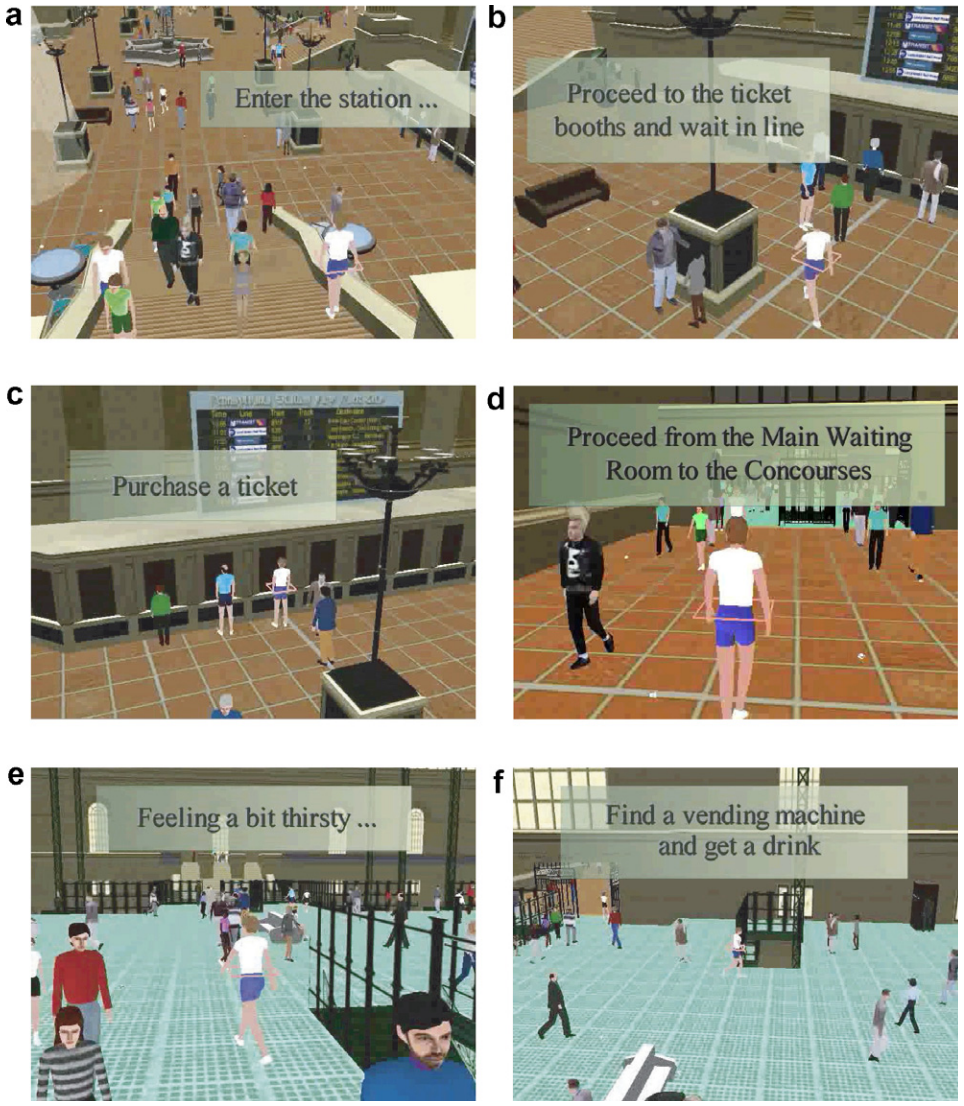
a
Enter the station ...
b
Proceed to the ticket booths and wait in line
c
Purchase a ticket
d
Proceed from the Main Waiting Room to the Concourses
e
Feeling a bit thirsty ...
f
Find a vending machine and get a drink

Fig. 15. Following an individual commuter.

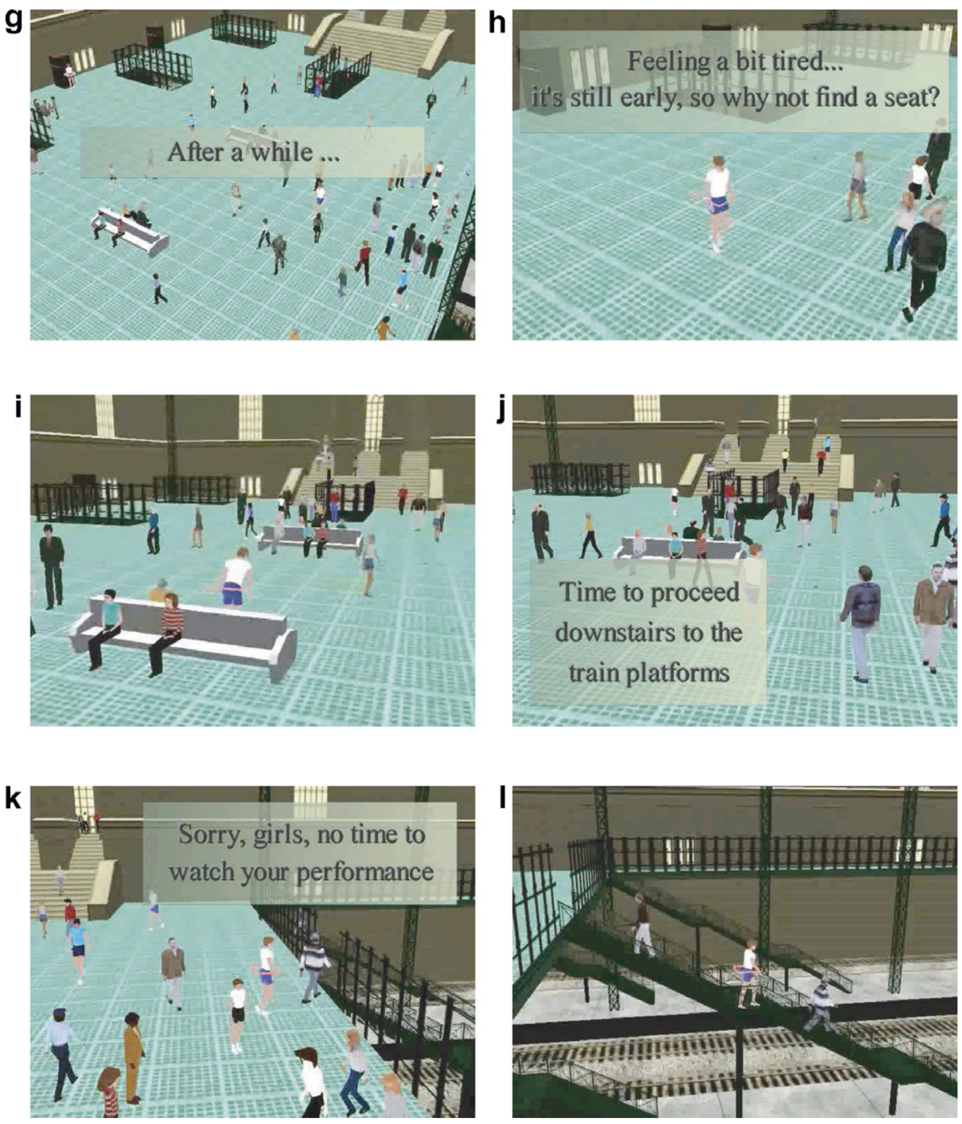

Fig. 15 (*continued*)

no time to watch the performance and descends the stairs to his train platform (l).

6.2.2. *Pedestrian activity in the train station*

Fig. 16(a)–(f) shows selected stills of a routine simulation, which includes over 600 autonomous pedestrians, demonstrating a variety of pedestrian activities that are typical for a train station. We can interactively vary our viewpoint through the station, directing the virtual camera on the main waiting room, concourse, and arcade areas in order to observe the rich variety of pedestrian activities that are simultaneously taking place in different parts of the station (see also Fig. 1). Some additional activities that were not mentioned above include pedestrians choosing portals and navigating through them (a), chatting in pairs (b) or small groups (e), congregating in the busy upper concourse (c) to watch a dance performance for amusement (d), and proceeding to the train platforms by navigating the rather narrow and oftentimes congested staircases (f).

7. Conclusions

We have developed a sophisticated human animation system whose major contribution is a comprehensive artificial life model of pedestrians as

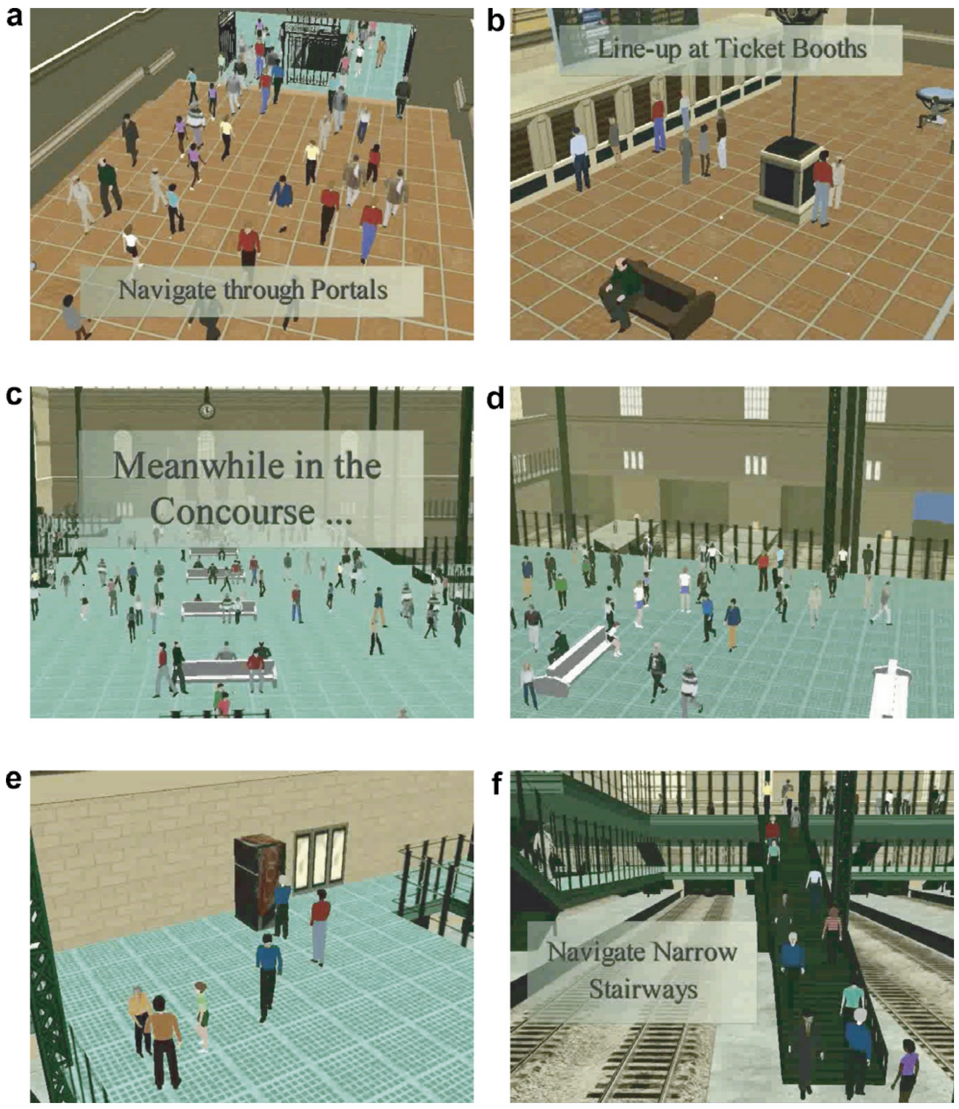

Fig. 16. Pedestrian activity in a train station.

highly capable individuals that combines perceptual, behavioral, and cognitive control components. Incorporating a hierarchical environmental modeling framework that efficiently supports perceptual sensing, situational awareness, and multiscale planning, our novel system efficiently synthesizes numerous self-animated pedestrians performing a rich variety of activities in a large-scale indoor urban environment.

Our results speak to the robustness of our human simulation system and its ability to produce prodigious quantities of intricate animation of numerous pedestrians carrying out various individual and group activities. Like real humans, our autonomous virtual pedestrians demonstrate rational behaviors reflective of their internal goals and compatible with their external world. Although motion artifacts are occasionally conspicuous in our animation results due to the limitations of the low-level DI-Guy software, our design facilitates the potential replacement of this software by a better character rendering and motion synthesis package should one become available.

7.1. Future work

With additional work, at least in principle there seems to be no reason why the appearance and performance of virtual humans cannot eventually become indistinguishable from those of real humans, though clearly not for many years to come. In future work, we plan to expand the behavioral and cognitive repertoires of our autonomous virtual pedestrians, as well as to include learning mechanisms, in a systematic effort to narrow the gap between their abilities and those of real people. Despite the satisfactory performance of our reactive behavior level for the time being, it may be beneficial to activate the behavior routines in some parallel fashion instead of sequentially. We also intend to develop a satisfactory set of reactive and deliberative head/eye movement behaviors for our virtual pedestrian model. Our autonomous pedestrians are currently too myopic for the sake of computational efficiency. Longer-range, biomimetic visual perception (e.g., [34,35]) in conjunction with appropriate head/eye behaviors would enable realistic "visual scanning" in which a mobile individual, assesses the intentions of other nearby pedestrians, especially those to the front. It also remains to imbue our pedestrians with useful manipulation skills, including the upper body motions necessary for making purchases at ticket booths or vending machines. Also, our train station simulations would be more realistic if some virtual pedestrians toted luggage and if we could simulate "families" of pedestrians that move together in small groups. The simulation of interpersonal behavior in autonomous pedestrians is a challenging problem and it has recently received attention through a probabilistic approach [39]. Finally, we are pursuing novel applications of our simulator to archaeology [30], computer vision [22], and other fields.

Acknowledgments

The research reported herein was carried out at the Media Research Laboratory of New York University's Courant Institute of Mathematical Sciences and was supported in part by grants from the Defense Advanced Research Projects Agency (DARPA) of the Department of Defense and from the National Science Foundation (NSF). We thank Dr. Tom Strat, formerly of DARPA, for his generous support and encouragement. We also thank Mauricio Plaza-Villegas for his invaluable contributions to the implementation and visualization of the Penn Station model (which was distributed to us by Boston Dynamics Inc.) and its integration with the DI-Guy API.

Appendix A. Environmental modeling

We represent the virtual environment by a hierarchical collection of data structures, including a topological map, two types of maps for perception, two types of maps for path planning and a set of specialized environmental objects (Fig. 3). With each of these data structures specialized to a different purpose, the combination is able to support accurate and efficient environmental information storage and retrieval.

A.1. Topological map

At the highest level of abstraction, a graph represents the topological relations between different parts of the virtual world. In this graph, nodes correspond to environmental regions, and edges between nodes represent accessibility between regions.

A region is a bounded volume in 3D space (such as a room, a corridor, a flight of stairs, or even an entire floor) together with all the objects inside that volume (for example, ground, walls, ticket booths, benches, vending machines, etc.). We assume that the walkable surface in a region may be mapped onto a horizontal plane without loss of essential geometric information, particularly the distance between two locations. Consequently, a 3D space may be adequately represented by several planar maps, thereby enhancing the simplicity and efficiency of environmental queries, as will be described shortly.

Another type of connectivity information stored at each node in the graph is *path-to-via* information. Suppose that $L(A,T)$ is the length in the number of edges of the shortest path from a region A to a different target region T, and $P(A,T)$ is the set of paths from A to T of length $L(A,T)$ and $L(A,T)+1$. Then the *path-to-via* of A associated with T is a set of pairs defined as $V(A,T) = \{(B, C_B)\}$, where C_B is the length of a path $p \in P(A,T)$ along which region B is next to A. As the name suggests, if (B, C_B) is in $V(A,T)$, then there exists a path of length C_B from A to T via B. Informally, $V(A,T)$ answers the question "If I am currently in A and want to go to T, to which region shall I go and what will be the expected cost?"

Algorithm 2. Computing *path-to-via* information

```
Require: $G(N, E)$, a graph with $N$ nodes and $E$ edges
  1: Initialization:
    for each node $A$ do
      for each target node $T$ do
        if $A == T$ then
          $V(A, T) \leftarrow \{(A,0)\}$
        else
          $V(A, T) \leftarrow \{\}$
  2: Collect information associated with paths of length L based on the information associated with paths of length L − 1:
    for $L = 1$ to $N - 1$ do
      for each node $A$ do
        for each target node $T$ do
          for every neighbor node $B$ of $A$ and any node $X$ in $G$ do
            if $(X, L - 1) \in V(B,T)$ then
              add $(B,L)$ to $V(A, T)$
  3: Keep only minimal cost entries:
    for each node $A$ do
      for each target node $T$ and any node $Y$ in $G$ do
        let $C_{\min}$ be the minimal cost in $V(A, T)$
        for each entry $E(Y,C)$ in $V(A, T)$ do
          if $(C > C_{\min} + 1)$ then
            remove $E$ from $V(A, T)$
```

Given a graph, the *path-to-via* information is computed offline, in advance, using the incremental Algorithm 2. Note that after Step 3 of the algorithm, only those entries are stored whose cost is $C_{\min}$ or $C_{\min} + 1$. Thus, we can avoid paths with cycles. To understand this, consider $V(A,C)$ for the graph in Fig. 3. Region C is a direct neighbor of A; so $(C,1)$ is clearly an entry of $V(A,C)$. As A-B-A-C is also a possible path from A to C, then $(B,3)$ is another entry. Obviously, A-B-A-C is not desirable as it contains a cycle. Such paths will automatically be removed from the *path-to-via* set after Step 3.

Linked within each node of the topological map are perception maps and path maps together with a list of objects inside that region. The next three sections describe each in turn.

A.2. Perception maps

Mobile objects and stationary objects are stored in two separate perception maps, which form a composite grid map. Hence, objects that never need updating persist after the initialization step and more freedom is afforded to the mobile object (usually virtual pedestrian) update process during simulation steps. Table 1 compares the perception maps, and the next two subsections present the details.

A.2.1. Stationary objects

Our definition of a region assumes that we can effectively map its 3D space onto a horizontal plane. By overlaying a uniform grid on that plane, we make each cell correspond to a small area of the region and store in that cell identifiers of all the objects that occupy that small area. The gridded "floor plan" simplifies visual sensing. The sensing query shoots out a fan of line segments whose length reflects the desired perceptual range and whose density reflects the desired perceptual acuity (cf. [37,18]). Each segment is rasterized onto the grid map (see the left and center panels of Fig. 4). Grid cells along each line are interrogated for their associated object information. This perceptual query takes time that grows linearly with the number of line segments times the number of cells on each line segment. More importantly, however, it does not depend on the number of objects in the virtual environment. Without grid maps, the necessary line-object intersection tests would be time consuming in a large, complex virtual environment populated by numerous pedestrians. For high sensing accuracy, small sized-cells are used. In our simulations, the typical cell size of grid maps for stationary object perception is 0.2–0.3 m.

A.2.2. Mobile objects

Similarly, a 2D grid map is used for sensing mobile objects (typically other pedestrians). In this map, each cell stores and also updates a list of identifiers of all the pedestrians currently within its area. To update the map, for each pedestrian H (with C_{old} and C_{new} denoting the cells in which H was and is, respectively), if $(C_{\text{old}} == C_{\text{new}})$ then we do nothing; otherwise, we remove H from C_{old} and add it to C_{new}. As the update for each pedestrian takes negligible, constant time, the update time cost for the entire map is linear in the total number of pedestrians, with a small coefficient.

The main purpose of this perception map is to enable the efficient perceptual query by a given pedestrian of nearby pedestrians that are within its sensing range. The sensing range here is defined by a fan as illustrated in the right part of Fig. 4. In

Table 1
Comparison of perception maps, including the cost to update the entire world and the query cost per pedestrian

Type	Cell size	Update cost	Query cost
Stationary	Small ($\sim 10^{-1}$ m)	0	Constant, given the sensing range and acuity
Mobile	Large ($\sim 10^{1}$ m)	Linear in the number of pedestrians	Constant, given the sensing fan and max number of sensed pedestrians

the mobile object perception map, the set of cells wholly or partly within the fan are divided into subsets, called "tiers", based on their distance to the pedestrian. Closer tiers are examined earlier. Once a maximum number (currently set to 16) of nearby pedestrians are perceived, the sensing is terminated. This strategy is intuitively motivated by the fact that usually people can simultaneously pay attention only to a limited number of other people, preferably proximal individuals. Once the set of nearby pedestrians is sensed, further information can be obtained by referring to finer maps, by estimation, or simply by querying a particular pedestrian of interest. Given the sensing fan and the upper bound on the number of sensed pedestrians, perception is a constant-time operation.

A.3. Path maps

Goal-directed navigation is one of the most important abilities of a pedestrian, and path planning enables a pedestrian to navigate a complex environment in a sensible manner. To facilitate fast and accurate online path planning, we use two types of maps with different data structures–grid maps and quadtree maps. These will be briefly presented in turn, before we discuss path planning.

A.3.1. Grid path map

Grid maps, which are useful in visual sensing, are also useful for path planning. We can find a shortest path on a grid map, if one exists, using the well-known A^* graph search algorithm [32].

In our system, grid path maps are used whenever a detailed path is needed. Suppose D is the direct distance between pedestrian H and its target T. Then, a detailed path is needed for H if D is smaller than a user-defined constant $D_{\max}$ and there are obstacles between H and T. This occurs, for instance, when one wants to move from behind a chair to the front and sit on it. Clearly, the accuracy of the path in this instance depends on the size of the cells in the grid path maps. A small cell size results in a large search space and, likely, low performance. However, detailed paths are usually not needed unless the target is close to the starting point. Therefore, chances are that paths can be found quickly, while the search has covered only a small portion of the entire search space. Roughly speaking, in most cases the space that must be searched is bounded by $4(D_{\max}/c)^2$, where c is the cell size. Typical values for these constants in our current system are $1 \leqslant D_{\max} \leqslant 10$ m and $10^{-1} \leqslant c \leqslant 1$ m.

When creating grid maps, special care must be taken to facilitate efficient updates and queries. Polygonal bounding boxes of obstacle objects represented on grid maps are enlarged by half the size of a pedestrian's bounding circle. If the center of a pedestrian never enters this "buffer" area, collisions will be avoided. This enables us to simplify the representation of a virtual pedestrian to a single point, which makes most queries simpler and more efficient.

A.3.2. Quadtree path map

Every region has a quadtree map, which supports fast online path planning [5]. Each quadtree map comprises

1. A list of nodes N_i ($i = 0, 1, 2, \ldots, m$), which together cover the entire area of the region (see Step (3) in Fig. 17);
2. C, the number of levels; i.e., the number of different node cell sizes appearing in the map (which is *3* for the quadtree map in Fig. 17); and
3. A pointer to an associated grid map with small cell size (see Step (1) in Fig. 17).

Each node N_i of the quadtree [26] stores the following variables:

1. L_i, where $0 \leqslant L_i < C$, the level of the node in the quadtree (which also indicates the cell size of N_i);
2. The center position of the area covered by this node;
3. The occupancy type (ground, obstacle, etc.);
4. A list of pointers to neighboring nodes;

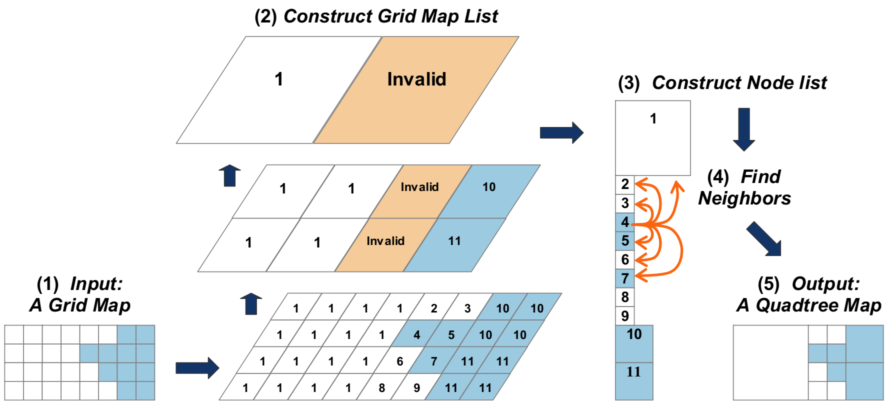

Fig. 17. Constructing a quadtree map.

5. A congestion factor g_i, which is updated at every simulation step and indicates the portion of the area covered by this node that is occupied by pedestrians; and
6. A distance variable, which indicates how far the area represented by the node is from a given start point, and will be used at the time of path-planning, especially during back-tracking as a gradient reference to find the shortest way back to the start point.

As Fig. 17 illustrates, given a grid map with small cells, the algorithm for constructing the quadtree map first builds the list of map levels containing nodes representing increasing cell sizes, where the cell size of an upper level node is twice as large as that of lower level nodes. Higher level nodes, which aggregate lower level nodes, are created so long as the associated lower level cells are of the same occupancy type, until a level is reached where no more cells can be aggregated. Quadtree maps typically contain a large number of lower level nodes (usually over 85% of all nodes) that cover only a small portion (usually under 20%) of the entire region. Such nodes significantly increase the search space for path planning. Thus, in the final stage of construction, these nodes are excluded from the set of nodes that will participate in online path planning. As the area that they cover is small, their exclusion does not cause significant accuracy loss.

A.4. Path planning

We now overview the three phases of autonomous pedestrian path planning, which from global to local include planning global paths between regions, planning intermediate length paths within a region, and planning detailed local paths. For the full details of the path planning algorithms, we refer the reader to [31].

A.4.1. Planning global paths between regions

In global path planning, the *path-to-via* information is useful in identifying intermediate regions that lead to the target region. Any intermediate region can be picked as the next region and, by applying the path-searching schemes described below, a path can be planned from the current location to the boundary between the current region and that next region. The process is repeated in the next region, and so on, until it can take place in the target region to terminate at the target location. Although the extent of the path is global, the processing is local.

A.4.2. Planning intermediate length paths within a region

To plan a path between two widely separated locations within a region, we first employ one of several variants of A^* algorithm to search for a path on the quadtree path map of the region, taking pedestrian congestion into consideration [31]. Fig. 18 compares paths computed by the four search variants of the path planning algorithm. Sometimes, however, quadtree maps may fail to find a path even though one exists. This is because low level nodes on a quadtree map are ignored during path search in order to decrease the search space (see Section A.3.2). Therefore, paths that must go through some low level nodes cannot be found on a quadtree map. To resolve this, we turn to grid path maps whenever search fails on a quadtree map. The standard A^* algorithm is used to find a path on grid maps. For

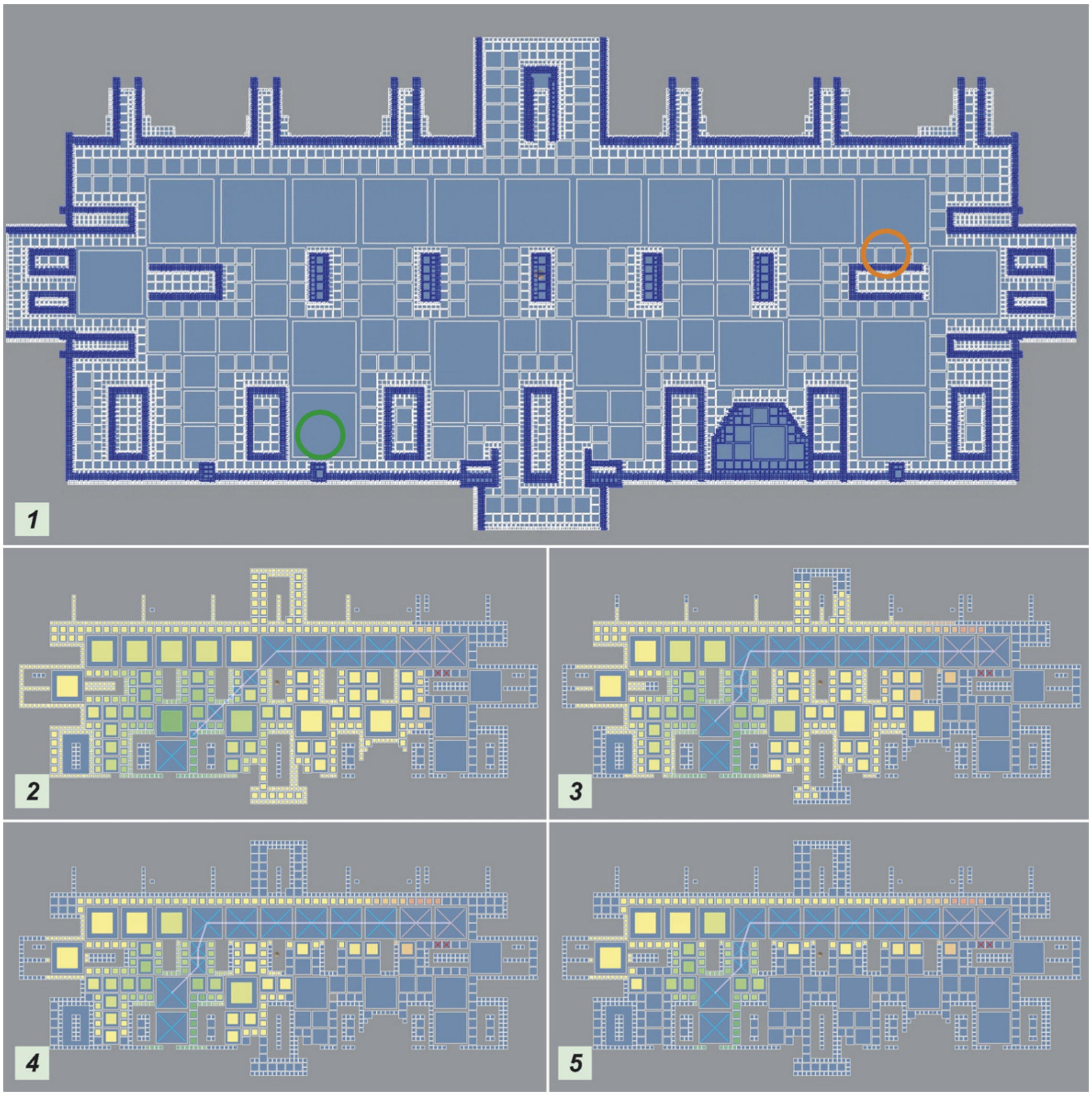

Fig. 18. Comparison of path planning algorithms on quadtree maps. (1) Visualization of the quadtree map of the upper concourse in the Penn Station environment model. The white quads denote ground nodes and the blue ones denote obstacles. The green circle (left) is the start location and orange circle (right) is the destination. With obstacle quads suppressed for clarity, (2)–(5) show results of the four path planning schemes: (2) SortedQ, (3) SingleQ, (4) MultiQ, and (5) PmultiQ. The search space is color coded with the distance variable values increasing from green to orange. Note that, although the four paths are similar, the sizes of search space differ. Refer to [31]. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

efficiency, a list of grid maps with decreasing cell sizes is kept for each region. Grid map path search starts at the coarsest level and progresses down the list so long as the search fails on coarser levels. Although grid maps with large cells are likely to merge separate objects due to aliasing, and thus cause the search to fail, the multiresolution search ascertains that, as long as the path exists, it will always be found eventually on maps with smaller cells.

A.4.3. Planning detailed local paths

If two locations are nearby but obstacles intervene, the finest grid path map of this region is used for finding a detailed path. The path will be slightly smoothed, and the pedestrian will follow the processed path strictly when approaching the target as we describe in Section 4.3.2.

A.5. Specialized environment objects

At the lowest level of our environmental hierarchy are environmental objects. Cells of grid maps and quadtree maps maintain pointers to a set of objects that are partly or wholly within their covered area. Every object has a list of properties, such as name, type, geometry, color/texture, functionality, etc. Many of the objects are specialized to support quick perceptual queries. For instance, every ground object contains an altitude function which responds to ground height sensing queries. A bench object keeps track of how many people are sitting on it and where they are sitting. By querying nearby bench objects, weary pedestrians are able to determine the available seat positions and decide where to sit without further reference to the perceptual maps. Other types of specialized objects include queues (where pedestrians wait in line), purchase points (where pedestrians can make a purchase), entrances/exits, etc. In short, these objects provide a higher level interpretation of the world that would be awkward to implement with perception maps alone, and this simplifies the situational analysis for pedestrians when they perform autonomous behaviors.

Appendix B. The basic reactive behavior routines

This appendix describes the six basic reactive behavior routines.

B.1. Routine A: Static obstacle avoidance

If there is a nearby obstacle in the direction of locomotion, lateral directions to the left and right are tested until a less cluttered direction is found (Fig. 4(b)). If a large angle (currently set to 90°) must be swept before a good direction is found, then the pedestrian will start to slow down, which mimics the behavior of a real person upon encountering a tough array of obstacles; i.e., slow down while turning the head to look around, then proceed.

B.2. Routine B: Static obstacle avoidance in a complex turn

When a pedestrian needs to make a turn that cannot be finished in one step, it will consider turns with increasing curvatures in both directions, starting with the side that permits the smaller turning angle, until a collision-free turn is found (Fig. 5(b)). If the surrounding space is too cluttered, the curve is likely to degenerate, causing the pedestrian to stop and turn on the spot. The turn test is implemented by checking sample points along a curve with interval equal to the distance of one step of the pedestrian moving with the anticipated turn speed.

B.3. Routine C: Maintain separation in a moving crowd

For a pedestrian H, other pedestrians are considered to be in H's *temporary crowd* if they are moving in a similar direction to H and are situated within a parabolic region in front of H defined by $y = -(4/R)x^2 + R$ where R is the sensing range, y is oriented in H's forward direction and x is oriented laterally (Fig. 5(c)). To maintain a comfortable distance from each individual C_i in this temporary crowd, a directed repulsive force (cf. [12]) given by $f_i = r_i(d_i/|d_i|)/(|d_i| - d_{\min})$ is exerted on H, where d_i is the vector separation of C_i from H, and $d_{\min}$ is the predefined minimum distance allowed between H and other pedestrians (usually 2.5 times H's bounding box size). The constant r_i is C_i's perceived "repulsiveness" to H (currently set to -0.025 for all pedestrians). The repulsive acceleration due to H's *temporary crowd* is given by $a = \sum_i f_i/m$ where m is the "inertia" of H. The acceleration vector is decomposed into a forward component a_f and a lateral component a_l. The components $a_f\Delta t$ and $a_l w_i \Delta t$ are added to H's current desired velocity. The crowding factor w_i determines H's willingness to "follow the crowd", with a smaller value of w_i giving H a greater tendency to do so (currently $1.0 \leqslant w_i \leqslant 5.0$).

B.4. Routine D: Avoid oncoming pedestrians

To avoid pedestrians not in one's *temporary crowd*, a pedestrian H estimates its own velocity v and the velocities v_i of nearby pedestrians C_i. Two types of threats are considered here. By intersecting its own linearly extrapolated trajectory T with the trajectories T_i of each of the C_i, pedestrian H identifies potential collision threats of the first type: *cross-collision* (Fig. 5(d1)). In the case where the trajectories of H and C_i are almost parallel and will not intersect imminently, a *head-on collision* (Fig. 5(d2)) may still occur if their lateral separation is too small; hence, H measures its lateral separation from oncoming pedestrians.

Among all collision threats, H will pick the most imminent one C^*. If C^* poses a *head-on collision* threat, H will turn slightly away from C^*. If C^* poses a *cross-collision* threat, H will estimate who will arrive first at the anticipated intersection point p. If H determines that it will arrive sooner, it will increase its speed and turn slightly away from C^*; otherwise, it will decrease its speed and turn slightly towards C^* (Fig. 5(d1)). This behavior will continue for several footsteps, until the potential collision has been averted.

B.5. Routine E: Avoid dangerously close pedestrians

This is the fail-safe behavior routine, reserved for emergencies due to the occasional failure of Routines C and D, since in highly dynamic situations predictions have a nonzero probability of being incorrect. Once a pedestrian perceives another pedestrian within its forward safe area (Fig. 5(e)), it will resort to a simple but effective behavior—brake as soon as possible to a full stop, then try to turn to face away from the intruder, and proceed when the way ahead clears.

B.6. Routine F: verify new directions relative to obstacles

Since the reactive behavior routines are executed sequentially (see Section 4.3.1), motor control commands issued by Routines C, D or E to avoid pedestrians may counteract those issued by Routines A or B to avoid obstacles, thus steering the pedestrian towards obstacles again. To avoid this, the pedestrian checks the new direction against surrounding obstacles once more. If the way is clear, it proceeds. Otherwise, the original direction issued by either the higher-level path planning modules or by Routine A, whichever was executed most recently prior to the execution of Routine F, will be used instead. However, occasionally this could lead the pedestrian toward future collisions with other pedestrians (Fig. 5(f)) and, if so, it will simply slow down to a stop, let those threatening pedestrians pass, and proceed.

Appendix C. Optimally ordering the reactive behavior routines

This appendix presents the details of the exhaustive search procedure (cf. [24]) that we employed to find the best permutation ordering for activating the six reactive behavior routines listed in Appendix B, which is C–A–B–F–E–D.

C.1. Fitness

We evaluate the performance of the different activation permutations of the reactive behavior routines through a set of long-term simulations with different numbers of pedestrians. The performance measure is the overall *fitness* $F = (1/N)\sum_{i=1}^{N} F_i$ of a population of N pedestrians over the course of a simulation. Here, $F_i = (S_i\, V_i)^4$ is the individual *pedestrian fitness*, where the *safety factor* $S_i = (1 - C_i/50)^2$ if $C_i < 50$ and $S_i = 0$ otherwise, with C_i denoting the average number of frames in every 10,000 frames that pedestrian i is involved in a collision either with stationary obstacles or with other pedestrians, and where the *vitality factor* $V_i = 1$ if $R_i > 0.5$ and $V_i = (2R_i)^8$ otherwise, with the *speed ratio* R_i defined as the average speed of pedestrian i in the simulation divided by the pedestrian's preferred speed. Note that the safety and vitality factors, both of which take values in the range [0,1], conflict in the sense that one can be guaranteed by sacrificing the other. Their product in F_i rewards the best tradeoff between safety and vitality.

C.2. Analysis of the results

Fig. 19 plots the fitness (y-axis) of all 720 possible activation permutations of the six reactive behavior routines (x-axis). Before carrying out this exhaustive evaluation, we had originally guessed that a sensible activation order of the reactive behavior routines is

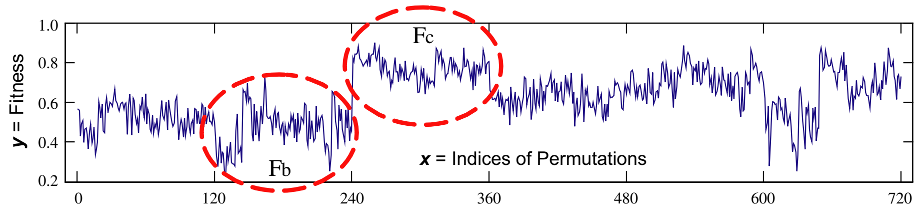

Fig. 19. Plot of the fitness values of all permutations. The best activation permutation C–A–B–F–E–D is at $x = 246$.

simply A–B–C–D–E–F, which occurs at $x = 1$ in the plot. Clearly, this choice yields a performance no better than average. Although the plot at first appears chaotic, we see a significant increase in the fitness function after $x = 240$, which marks the transition from permutations that begin with reactive behavior Routine B (in the range $121 \leqslant x \leqslant 240$) to those that begin with Routine C (in the range $241 \leqslant x \leqslant 360$). Comparing the fitness values F_b in the former range to those F_c in the latter, we see that the mean fitness value of activation permutations that begin with Routine C is higher than the maximum fitness value of those that begin with Routine B. Sorting permutations by partial ordering and comparing the shape of different parts of the plot, etc., we made the following observations:

- Permutations P starting with A and B usually give poor performance.
- The later that D appears in P, the better the performance.
- It is better for C to appear before A, but they need not be adjacent.
- It is better for A to appear before B, but they need not be adjacent.
- If F appears earlier than C, D, and E, it results in the same performance as when F is omitted.[2]
- Almost all of the high-performance permutations have A before B and after C, and end with D.

It is difficult to fully explain the above results, but we offer the following four plausible heuristics:

1. *Routine C should be activated before Routine A.* Consider a crowd in which pedestrian H moves in a certain direction. As long as H stays in the interior of the crowd, chances are that H will not bump into a stationary obstacle, since pedestrians on the periphery of the crowd should have already avoided any obstacle, thus steering the crowd to a safe direction (see Fig. 20). Hence, H need only stay within the crowd by using Routine C. Consequently, the order C–A allows a pedestrian to take advantage of obstacle avoidance efforts made by other pedestrians.
2. *Routine A should be activated before Routine B.* Ordering Routine B after Routine A is sensible as whenever a pedestrian wants to check a turn, the turn itself had better already be determined; otherwise, the check will be wasted effort. Among the six routines, however, only Routine A can change the turning angle so much that the turn may need more than one step to finish. Consequently, it is better for Routine A to be activated before Routine B.
3. *Routine D should be activated last.* Routine D avoids oncoming pedestrians (either from the front or side) that may cause potential collision on a pedestrian's trajectory. Since it generally considers pedestrians at some distance, the situations with which it deals are usually not as urgent as those dealt with by the other routines. Hence, all of the avoiding options in D are small changes in speed or turning angle or both. This means that the motor control command issued by the preceding routine is likely to be more or less preserved. If Routine D appears early in a permuta-

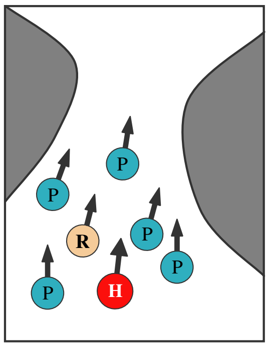

Fig. 20. Explanation of the order of C–A: pedestrian H can avoid obstacles by simply following the crowd and letting other pedestrians (labeled P) deal with them. So can pedestrian R.

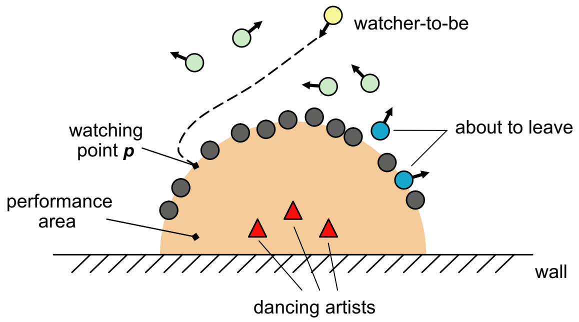

Fig. 21. Approach and watch a performance. A pedestrian (yellow), who is interested in the performance, finds an available observation spot and approaches it. Among the current observers are two (blue) who are about to leave. Outside the observation area, pedestrians (green) are passing by. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

[2] This is obviously true as Routine F is designed to correct directions picked by Routines C, D, or E. If the latter routines are not executed, F will have no effect.

tion, other routines may likely overwrite D's motor control command with their more significant changes. Consequently, it makes sense for Routine D to appear at the end of the permutation.

4. *The activation order of Routines E and F appears flexible.* If Routines A, B, C, and D are perfectly designed, almost all dangerous situations will be dealt with in their early stages and it is likely that the situations handled by Routines E and F will be rare. Ideally then, E and F will not provide as much assistance as the other routines. Even for situations when E and F can help substantially, their strategies for dealing with collision threats involve slowing to a stop, and subsequent routines will have little effect on a nearly stopped pedestrian. Hence, E and F can take any position, in principle, but their actual positions may have a small affect on performance, since earlier routines may occasionally redirect the pedestrian such that the original threat is avoided but new threats appear. Consequently, the most we can say is that the activation order of Routines E and F appears to be flexible.

Table 2 lists some of the best permutations found after an exhaustive evaluation of the performances of all possible activation permutations of the six behavior routines in 20-min (36,000-frame) Penn Station simulation trials with different numbers of pedestrians, along with the average number of collisions that occurred over several simulations involving the indicated number of pedestrians. Analogous with the real world, we do not impose any hard constraints to prevent collisions. Most of the collisions occur between pedestrians and fewer than 3% of them occur between a pedestrian and an immobile obstacle. Collisions usually last no more than 1 s, and for the pedestrian-obstacle collisions, pedestrians never move completely through the obstacle.

Table 2
Average number of collisions occurring in simulations with different number of pedestrians using some of the best permutations of reactive behaviors

Permutation	333 Pedestrians	666 Pedestrians	1000 Pedestrians
C–A–B–F–E–D	4	22	84
F–C–A–B–E–D	3	25	85
C–E–A–B–F–D	3	23	94
C–A–F–B–E–D	4	24	99
E–C–A–B–F–D	1	31	102

Algorithm 3. The *observe-a-performance* routine (Fig. 21)

```
Require: The performance area $A$, a semi-circular
  area surrounding the performing artists
  (Fig. 21), and the performance-area
  specialized object
  1: if $A$ is far away, use navigation behaviors to
     approach $A$
  2: once $A$ is close enough, find an available
     observation point $p$ around $A$
  3: use the detailed arrival behavior to approach
     and reach $p$
  4: turn to face the performance
  5: watch the performance for some time
  6: turn away from the performance
  7: leave the performance
```

Appendix D. Details of motivational behavior routines

This appendix presents the details of several representative motivational behavior routines. These routines depend on lower-level routines in our behavioral hierarchy, including navigational behaviors and reactive behaviors, as well as a collection of action-level motor skills, such as walking, running, turning while moving, turning on the spot, standing, sitting, etc. In addition, they rely on specialized environmental objects for abstract level interpretation of situations in order to make decisions.

D.1. Observe a performance

When attracted by a nearby dance performance, a pedestrian will use the routine described in Algorithm 3 to approach the performance and observe it until it is time to leave.

In Step 2, a pedestrian will use detailed path-planning to identify an available observation point and simultaneously plan a path to it. The pedestrian first sets the performance area A as a target on a grid path map and adds every observer as a static circular obstacle. The path search finds the nearest observation point. However, a dense arrangement of existing observers can effectively block the path from the target area. In this case, the pedestrian can either give up or perhaps enlarge the target performance area by the size of a pedestrian bounding box and attempt to plan a new path. This strategy leads to a sparse second layer of observers.

To identify all the current observers quickly, a pedestrian employs a specialized object—the `performance-area` object—which maintains a list of the observers currently surrounding the area. Incoming prospective observers are registered onto the list (late in Step 3), while outgoing observers are removed (late in Step 7). On a first-come-first-served basis, prospective observers request registration on the list when they are within a meter of A, which resolves conflicts if two pedestrians compete for a spot big enough for only one. When leaving the performance area, a former observer is removed from the list once more than a meter away.

D.2. Make a purchase

To make a purchase at a ticket booth or a vending machine, a pedestrian invokes the routine detailed in Algorithm 4, whose functionality is illustrated in Fig. 22. In Step 2, a routine similar to passageway selection (see Section 4.3.2) is used by the pedestrian to choose among several available purchasing places. In Step 13, one of several styles of waiting motions in the motion repertoire will be chosen probabilistically to express the pedestrian's (im)patience.

This routine requires two types of specialized objects to help it analyze the situation. The `queue` object keeps track of the pedestrians waiting in line and efficiently informs a pedestrian how many people are waiting, which pedestrian is first and which one is last. The `purchase-point` object indicates whether or not a transaction spot is available. Analogous to the previous routine, pedestrians issue requests to register themselves into and remove themselves from those specialized objects. Since there are other pedestrians constantly passing by (Fig. 22), these specialized objects support efficient situation analysis and decision making by pedestrians.

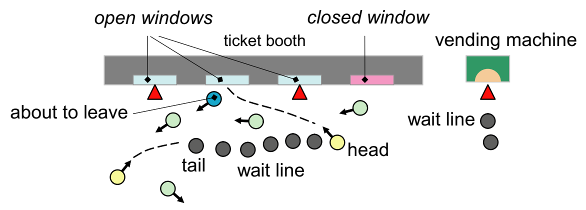

Fig. 22. Joining a queue to make a purchase. On the left, a pedestrian (yellow) joins a ticket purchasing line. The (yellow) pedestrian at the head of the line is about to take the transaction spot freed as a (blue) pedestrian leaves the second ticket window. On the right, a line forms for making purchases at a vending machine. The triangles denote pedestrians currently making purchases. With other pedestrians (green) passing by, the scene is efficiently analyzed using specialized objects. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Algorithm 4. The *make-a-purchase* routine

```
Require: queue and purchase-point specialized objects
 1: identify all places in the current region that sell the desired item
 2: pick the best one B among them in terms of proximity and expected wait time
 3: if B is far
 4:   approach B using the navigation behavior
 5: else if there is a spot available in B for the transaction and there is no queue then
 6:   approach and take the transaction spot
 7: else
 8:   join the end of the queue, behind the last person (if any)
 9:   while not (at the head of the queue and a transaction spot is available) do
10:     if waited too long and the purchase is optional then
11:       leave queue.
12:     else
13:       stand patiently or impatiently
14:       if the line advances then
15:         follow the line, moving forward
16:   approach and take the transaction spot
17: make the purchase
18: leave the transaction spot
```

Algorithm 5. The *take-a-rest* routine

```
Require: seat-space specialized object
 1: identify the available seats in the current region
 2: pick the best available seat B among them in terms of proximity and expected resting comfort
 3: if B is far, use the navigation behavior to approach it
 4: once B is proximal, plan a detailed-path and use detailed arrival behavior to approach and reach it
 5: when in front of the seat, turn to face the correct direction and sit down
 6: repeat
 7:   sit
 8: until rested or it is time to leave
 9: stand up and leave
```

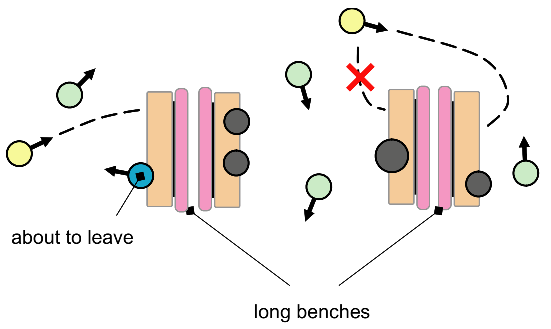

Fig. 23. Choosing a seat to take a rest. Four long benches are illustrated. Four pedestrians (gray) are currently sitting on the benches. A pedestrian (blue) is about to leave the left-most bench, on which a pedestrian (yellow) desires to sit. To the upper right, another pedestrian (yellow) also wants to take a rest. With two nearby choices, the pedestrian picks the more comfortable seat (offering the most space) and proceeds to it. Meanwhile, other pedestrians (green) are passing by. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

D.3. Take a rest

The final example is the routine that enables the pedestrian to take a rest as illustrated in Fig. 23 and detailed in Algorithm 5. Its basic structure is similar to the previous two routines. The selecting behavior here uses the resting comfort in addition to proximity as the criteria. Again, there is a specialized object—`seat-space` that tracks all the available spaces on a resting facility—which assists in the execution of this behavior routine.

References

[1] K. Ashida, S. Lee, J. Allbeck, H. Sun, N. Badler, D. Metaxas, Pedestrians: Creating agent behaviors through statistical analysis of observation data, in: Proceedings of the IEEE Conference on Computer Animation, 2001, pp. 84–92.
[2] N. Badler, C. Phillips, B. Webber, Simulating Humans: Computer Graphics, Animation, and Control, Oxford University Press, Oxford, 1993.
[3] M. Batty, B. Jiang, M. Thurstain-Goodwin, Local movement: agent-based models of pedestrian flow. Center for Advanced Spatial Analysis Working Paper Series 4, 1998.
[4] V. Blue, J. Adler, Cellular automata model of emergent collective bi-directional pedestrian dynamics, in: Proceedings of the Artificial Life VII, 2000, pp. 437–445.
[5] A. Botea, M. Muller, J. Schaeffer, Near optimal hierarchical path-finding, Journal of Game Development 1 (1) (2004) 7–28.
[6] D. Broadbent, Information processing in the nervous system, Science 150 (October) (1965) 457–462.
[7] G.N. De Souza, A.C. Kak, Vision for mobile robot navigation: A survey, IEEE Transactions on Pattern Analysis and Machine Intelligence. 24 (2) (2002) 237–267.
[8] N. Farenc, S. Musse, E. Schweiss, M. Kallmann, O. Aune, R. Boulic, D. Thalmann, A paradigm for controlling virtual humans in urban environment simulations, Applied Artificial Intelligence 14 (1) (2000) 69–91.
[9] J. Funge, X. Tu, D. Terzopoulos, Cognitive modeling: Knowledge, reasoning and planning for intelligent characters, in: Proceedings of the SIGGRAPH 99, 1999, pp. 29–38.
[10] G. Gipps, B. Marksjo, A micro-simulation model for pedestrian flows, Mathematics and Computers in Simulation 27 (1985) 95–105.
[11] S. Goldenstein, M. Karavelas, D. Metaxas, L. Guibas, E. Aaron, A. Goswami, Scalable nonlinear dynamical systems for agent steering and crowd simulation, Computers & Graphics 25 (6) (2001) 983–998.
[12] D. Helbing, P. Molnar, Social force model for pedestrian dynamics, Physical Review 51 (5) (1995) 4282–4286.
[13] J. Koechling, A. Crane, M. Raibert, Applications of realistic human entities using DI-Guy, in: Proceedings of the Spring Simulation Interoperability Workshop, 1998.
[14] F. Lamarche, S. Donikian, Crowd of virtual humans: A new approach for real time navigation in complex and structured environments, Computer Graphics Forum 23 (3) (2004) 509–518.
[15] C. Loscos, D. Marchal, A. Meyer, Intuitive crowd behaviour in dense urban environments using local laws, in: Theory and Practice of Computer Graphics, IEEE, 2003, pp. 122–129.
[16] G.G. Lovas, Modeling and simulation of pedestrian traffic flow, in: Proceedings of the European Simulation Multiconference, 1993.
[17] A.B. Loyall, W.S.N. Reilly, J. Bates, P. Weyhrauch, System for authoring highly interactive, personality-rich interactive characters, in: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, 2004, pp. 59–68.
[18] P. Maes, T. Darrell, B. Blumberg, A. Pentland, The ALIVE system: Full-body interaction with autonomous agents, in: CA'95: Proceedings of the Computer Animation, IEEE Computer Society, Washington, DC, USA, 1995, 11.
[19] R. Metoyer, J. Hodgins, Reactive pedestrian path following from examples, in: Computer Animation and Social Agents, 2003, pp. 149–156.
[20] S. Musse, D. Thalmann, Hierarchical model for real time simulation of virtual human crowds, IEEE Transactions on Visualization and Computer Graphics 7 (2) (2001) 152–164.
[21] H. Noser, O. Renault, D. Thalmann, N. Magnenat-Thalmann, Navigation for digital actors based on synthetic vision, memory and learning, Computers & Graphics 19 (1995) 1.
[22] F. Qureshi, D. Terzopoulos, Surveillance camera scheduling: A virtual vision approach, Multimedia Systems 12 (3) (2006) 269–283.
[23] C.W. Reynolds, Flocks, herds, and schools: A distributed behavioral model, Proc. SIGGRAPH 87, Computer Graphics 21 (4) (1987) 25–34.
[24] C.W. Reynolds, An evolved, vision-based behavioral model of coordinated group motion, in: Proceedings of the Second International Conference on from Animals to Animats 2:

Simulation of Adaptive Behavior, MIT Press, Cambridge, MA, USA, pp. 384–392.

[25] C.W. Reynolds, Steering behaviors for autonomous characters, in: Proceedings of the Game Developers Conference, 1999, pp. 763–782.

[26] H. Samet, Spatial Data Structures, Addison-Wesley, Reading, MA, 1989.

[27] A. Schadschneider, Traffic flow: A statistical physics point of view, Physica A313 (2002) 153–187.

[28] M. Schreckenberg, S. Sharma (Eds.), Pedestrian and Evacuation Dynamics, Springer-Verlag, Berlin, 2001.

[29] W. Shao, D. Terzopoulos, Environmental modeling for autonomous virtual pedestrians, in: SAE Symposium on Digital Human Modeling for Design and Engineering, 1–8. SAE Technical Paper 2005-01-2699, 2005.

[30] W. Shao, D. Terzopoulos, Populating reconstructed archaeological sites with autonomous virtual humans, in: Intelligent Virtual Agents, Springer-Verlag, vol. 4133 of Lecture Notes in Artificial Intelligence, in: Proceedings of the Sixth International Conference on Intelligent Virtual Agents (IVA 06), 2006, pp. 420–433.

[31] W. Shao, Animating Autonomous Pedestrians. Ph.D. thesis, New York University, Computer Science Department, New York, NY, 2006.

[32] B. Stout, The basics of A* for path planning, Game Programming Gems (2000) 254–263.

[33] M. Sung, M. Gleicher, S. Chenney, Scalable behaviors for crowd simulation, Computer Graphics Forum 23 (3) (2004) 519–528.

[34] D. Terzopoulos, T. Rabie, R. Grzeszczuk, Perception and learning in artificial animals, in Artificial Life V: Proceedings of the Fifth International Conference on the Synthesis and Simulation of Living Systems, 1996, pp. 313–320.

[35] D. Terzopoulos, Artificial life for computer graphics, Communication of the ACM 42 (8) (1999) 32–42.

[36] B. Tomlinson, M. Downie, M. Berlin, J. Gray, D. Lyons, J. Cochran, B. Blumberg, Leashing the alphawolves: Mixing user direction with autonomous emotion in a pack of semi-autonomous virtual characters, in: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, 2002, pp. 7–14.

[37] X. Tu, D. Terzopoulos. Artificial fishes: Physics, locomotion, perception, behavior, in: Proceedings of the SIGGRAPH 94, 1994, pp. 43–50.

[38] B. Ulicny, P. de Heras Ciechomski, D. Thalmann, Crowdbrush: Interactive authoring of real-time crowd scenes, in: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, 2004, pp. 243–252.

[39] Q. Yu, D. Terzopoulos, A decision network framework for the behavioral animation of virtual humans, in: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA'07), 2007, pp. 119–128.