Autonomous Pedestrians

Wei Shao
New York University,
Courant Institute,
Media Research Lab

Demetri Terzopoulos
University of California at Los Angeles,
Samueli School of Engineering & Applied Science,
Computer Science Department
Abstract

We address the difficult open 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 model of pedestrians as individuals. Our comprehensive model features innovations in these components, as well as in their combination, yielding results of unprecedented fidelity and complexity for fully autonomous multi-human simulation in a large urban environment. We represent the environment using hierarchical data structures, which efficiently support the perceptual queries of the autonomous pedestrians that drive their behavioral responses and sustain their ability to plan their actions on local and global scales.

Keyword: Behavioral Animation, Autonomous Characters


Figure 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.

Figure 2: Original Pennsylvania Station in New York City.

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 the station’s 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. We deploy a multitude of self-animated virtual
pedestrians within a large environment model, a 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 and support 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. Section 5 presents results comprising long term simulations with well over 1000 pedestrians and reports on performance. Finally, Section 6 concludes the paper and discusses future work.

2 Related Work

Human animation is an important and challenging problem in computer graphics [Badler et al. 1993]. 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 [Ashida et al. 2001; Musse and Thalmann 2001]. The topic has also been of some interest in the field of artificial life [Blue and Adler 2000], as well as in architecture and urban planning [Lovas 1993; Schreckenberg and (Eds.) 2001] where graphics researchers have assisted in visualizing planned construction projects, including pedestrian animation [Farenc et al. 2000; Metoyer and Hodgins 2003].

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 1987] on behavioral animation is certainly relevant to our effort, as is its further development in work by other researchers [Tu and Terzopoulos 1994; Tomlinson et al. 2002; Loyall et al. 2004]. 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 [Gipps and Marksjo 1985; Helbing and Molnar 1995; Batty et al. 1998; Schadschneider 2002] and by animation researchers [Goldenstein et al. 2001; Loscos et al. 2003; Sung et al. 2004; Ulicny et al. 2004; Lamarche and Donikian 2004] 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 [Tu and Terzopoulos 1994] on artificial animals and by [Funge et al. 1999] 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 an autonomous virtual human model that can populate large-scale 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 2004] proposed a suitable structuring of the geometric environment together with 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 elsewhere [Shao and Terzopoulos 2005], 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.

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 \(0.2 \sim 0.3\) meters. 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 meters). 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. [Botea et al. 2004]). 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.

Our environment model is efficient enough to support the real-time (30fps) simulation of about 1400 pedestrians on a 2.8GHz Xeon PC with 1GB 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 [Shao and Terzopoulos 2005].

4 Autonomous Pedestrian Model

Like real humans, our synthetic pedestrians are fully autonomous. They perceive the virtual environment around them, analyze environmental situations, make decisions and behave naturally. Our autonomous human characters are architected as a hierarchical artificial life model. Progressing through 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 human characters with basic motor skills, such as standing, strolling, walking, running, sitting, etc. [Koechling et al. 1998]. 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 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 transitions, which in turn 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 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 apply the update to the character. Our seamless interface hides the details of the underlying kinematic layer from our higher-level behavior routines, enabling the latter to be developed more or less 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.

**Sensing ground height.** In the static object perception map, each map cell contains the height functions of usually a single and sometimes a few 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.

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

**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. Once the set of nearby pedestrians is sensed, further information can be obtained by referring to finer maps, estimation, or simply querying some pedestrian of particular interest. Given the sensing fan and the maximum number of sensed pedestrians, sensing is a constant time operation.
4.3 Behavioral Control

Realistic behavioral modeling, whose purpose is to link perception to appropriate actions in an autonomous virtual human, is a big challenge. 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 [Tu and Terzopoulos 1994], 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

![Diagram of reactive behaviors]

(B) Safety in turning  (C) Temporary crowd  (D1) Cross collision  (D2) Head-on coll’n  (E) Front safe area  (F) Verify direction

Figure 5: Reactive behaviors. (B) Pedestrians choose best turning curves (light gray) to turn southward. (C) Pedestrians within H’s front parabola 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 front safe area; w and d depend on H’s bounding box size and d is 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.

Reactive behaviors appropriately connect perceptions to immediate actions. We have developed six key reactive behavior routines, each suitable for a different 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 still, moving forward, turning in several 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 the Appendix.

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. As we enable Routines C and D, the obstacle avoidance behavior looks increasingly more natural. Interesting multi-agent behavior patterns emerge when all the routines are enabled. For example, pedestrians will queue to go through a narrow portal. In a busy
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 six key situations that are covered by the six routines, we have chosen to activate them in a sequential manner (Fig. 6), giving each the chance 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 advantage of overriding 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 the six routines. We have run many extensive simulations (longer than 20 minutes 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.

4.3.2 Navigational and Motivational Behaviors

While the reactive behaviors enable pedestrians to move around freely, almost always avoiding collisions, navigational and motivational 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. [Noser et al. 1995]. Metoyer and Hodgins [Metoyer and Hodgins 2003] 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.

As we must deal with online simulations of numerous pedestrians within large, complex environments, we are confronted 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 [Reynolds 1999] 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 queueing in [Reynolds 1999], self-organization in [Helbing and Molnar 1995], and global crowd control in [Muse and Thalmann 2001] 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 are detailed next.

**Passageway navigation.** In real life, if all 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 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 the Appendix) is used to maintain a safe separation between oncoming pedestrians. 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)).

**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.

With the above two passageway behaviors, we are able 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.
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.

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.

Detailed “arrival at target” navigation. Before a pedestrian arrives at a target, a detailed path will be needed if small obstacles intervene. Such paths can be found on a fine-scale grid path map. The pedestrian will follow the detailed path strictly as it approaches the target, because accuracy becomes increasingly important.
in the realism of the navigation as the distance to the target diminishes. As some part of an obstacle may also be a part of the target or be very close by, indiscriminately employing reactive behaviors for static obstacle avoidance—Routines A and B (refer to the Appendix)—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 Other Interesting 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 behavior routines including, among others, the following:

- Select an unoccupied seat and sit down
- Approach a performance and watch
- Meet with friends and chat
- Queue at a vending machine and make a purchase
- Queue at ticketing areas and purchase a ticket

In the last behavior, for example, a pedestrian joins the queue and stands behind its precursor pedestrian until it comes to the head of the queue. Then, the pedestrian will approach the first ticket counter associated with this queue that becomes available. Space limitations preclude the detailed specification of the above behaviors in this paper. Note, however, that these non-navigational behaviors depend on the basic reactive behaviors and navigational behaviors to enable the pedestrian to reach targets in a collision-free manner.

4.3.4 Mental State and Action Selection

Each pedestrian maintains a set of internal mental state variables, which 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 internal 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 [Funge et al. 1999] is responsible for creating and executing plans, as is necessary for a deliberative human agent. Such a model must be able to make reasonable global navigation plans in order for a pedestrian to travel purposefully and with suitable perseverance between widely separated regions of the environment. During the actual navigation, however, the pedestrian must have the freedom to decide whether or not and to what extent to follow the plan, depending on the real-time situation, as we discussed when explaining the behaviors in Section 4.3.2. On the other hand, in a highly dynamic environment such as a train station, the pedestrian also needs the ability to decide whether and when a new plan is needed. These decisions require a proper coupling between the behavioral layer and cognitive layer. Before we describe the coupling mechanism, we will explain the global path planning strategy.

**Global path planning** directs a pedestrian to proceed through intermediate regions and finally reach the ultimate destination. To do this, it exploits the topological map at the top level of the environment model.
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, 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 [Shao and Terzopoulos 2005]), 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 containing many spikes (quadtree path maps)—as opposed to smooth, spline-like paths. Consequently, a pedestrian uses those rough plans only as a navigational guide and retains the freedom to locomote locally in as natural a manner as possible, as was described in Section 4.3.2.

Coupling cognitive control to behavioral control increases the realism of our pedestrians. To this end, every pedestrian maintains a stack of goals, the top one being the current goal. The goal stack is accessible both to the deliberative, cognitive layer and to the underlying reactive, behavioral layer. 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 navigation on a local scale). The behavioral controller can also insert directives according to the internal mental state and environmental situation (e.g., if thirsty & vending machine 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 Results

Our pedestrian animation system, which comprises about 50,000 lines of C++ code, enables us to run long-term simulations of pedestrians in a large-scale urban environment—specifically the Penn Station environment—without manual intervention. The entire 3D space of the Penn Station \((200(l) \times 150(w) \times 20(h)m^3)\), which contains hundreds of architectural and non-architectural objects, is manually divided into 43 regions. At run time, our environment model requires approximately 90MB of memory to accommodate the station and all of its associated objects.

In our simulation experiments, we populate the virtual station with five different types of pedestrians: com-
muters, tourists, performers, policemen, and patrolling soldiers. With every individual guided by his/her own autonomous control, these autonomous pedestrians imbue the virtual train station with liveliness, social (dis)order, and a realistically complex dynamic.

5.1 Performance

We have run various simulation tests on a 2.8GHz Intel Xeon system with 1GB of main memory. The total length of each test is 20 minutes in virtual world time. Fig. 10 indicates the computational load increase with the number of pedestrians in the simulation. The simulation times reported include only the requirements of our algorithms—environment model update and motor control, perceptual query, behavioral control, and cognitive control for each pedestrian. 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 minutes to simulate at 30fps). 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 with a much smaller factor. Fig. 11 breaks 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. 12 tabulates the frame rates that our system achieves on the aforementioned PC with an NVIDIA GeForce 6800 GT AGP8X 256MB graphics system. Due to the geometric complexity of the Penn Station model and numerous pedestrians, rendering times dominate pedestrian simulation times.
includes: 1) passageway behaviors (5.1%); 2) path planning (17.3%); and 3) plan-guided navigation (4.5%).

includes: 1) update agents’ velocity, position, etc (28.8%); and 2) modify maps due to agents’ update (13.7%).

includes necessary perception: 1) sensing obstacle (8.0%); and 2) sensing nearby humans (5.6%).

includes: 1) sensing obstacle (8.0%); and 2) sensing nearby humans (5.6%).

Figure 11: Computational loads of the system’s parts.

Figure 12: Frame rate (in frames/sec) for pedestrian simulation only (including DI-Guy), rendering only (i.e., static pedestrians), and both simulation and rendering, with different numbers of pedestrians.

5.2 Animation Examples

We will now describe several representative simulations that demonstrate specific functionalities. To help place the animation scenarios in context, Fig. 13 shows a plan view of the Penn station model.

**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. We can therefore choose a commuter and, in a typical animation, follow our subject as he enters the station, proceeds to the ticket booths in the main waiting room, and waits in a queue to purchase a ticket at the first open booth. Having obtained a ticket, he then proceeds to the concourses through a congested portal. Next, our subject feels thirsty and spots a vending machine in the concourse. He walks toward it and waits his turn to get a drink. Feeling a bit tired, our subject finds a bench with an available seat, proceeds towards it, and sits down. Later, the clock chimes the hour and it is time for our subject to get up and proceed to his train platform. He makes his way through a somewhat congested area by following, turning, and stopping as necessary in order to avoid bumping into other pedestrians. He passes by some dancers that are attracting interest from many other pedestrians, but our subject has no time to watch the performance and descends the stairs to his train platform.

**Pedestrian Activity in the Train Station.** A routine simulation, which includes over 600 autonomous
pedestrians, demonstrates 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 (Fig. 1). Some additional activities that were not mentioned above include pedestrians choosing portals and navigating through them, congregating in the upper concourse to watch a dance performance for amusement, and proceeding to the train platforms using the rather narrow staircases.

6 Conclusion and Future Work

We have developed a sophisticated human animation system whose major contribution is a comprehensive model of pedestrians as highly-capable individuals that combines perceptual, behavioral, and cognitive control components. Incorporating a hierarchical environmental modeling framework, 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 system and its ability to produce prodigious quantities of intricate animation of pedestrians carrying out various individual and group activities. Although motion artifacts are at times 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.

In future work, we plan systematically to expand the behavioral and cognitive repertoires of our autonomous virtual pedestrians to further narrow the gap between their abilities and those of real people. It is also our intention to develop a satisfactory set of reactive and deliberative head motion behaviors for our virtual pedestrian model and to model “families” of pedestrians that move together in small groups. We will also pursue new applications of our simulator to archaeology, computer vision, and other fields.

A The Basic Reactive Behavior Routines

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.

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.

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. [Helbing and Molnar 1995]) 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 ≤ \( w_i \) ≤ 5.0).
**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.

**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 front 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.

**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.

### B Ordering the Reactive Behavior Routines

We have presented the six key reactive behavior routines and briefly described how they are activated sequentially in a best permutation ordering that we found via an exhaustive search (cf. [Reynolds 1993]), evaluating the performance of all 720 possibilities. Here in this appendix, we will present more details of this exhaustive search, explaining the criteria and discussing the result.
### Environment Setting

<table>
<thead>
<tr>
<th>Environment Setting</th>
<th>Number of Agents</th>
<th>Simulation Length (in Virtual Time)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A Simple Synthetic Environment *</td>
<td>100</td>
<td>three 20-min simulations with different initial configurations</td>
</tr>
<tr>
<td>Penn Station</td>
<td>210</td>
<td>three 20-min simulations with different initial configurations</td>
</tr>
<tr>
<td>Penn Station</td>
<td>500</td>
<td>one 20-min simulation</td>
</tr>
</tbody>
</table>

Table 1: The set of simulations for permutation search.

*: This environment is shown in Figure 14.

### B.1 Fitness – The Performance Measure

For each permutation ordering of the six routines, an identical set of simulations given in Table 1 is run and the results are summarized to a single value called “fitness”. As a measure of the performance of a permutation, fitness is defined in Table 2.

<table>
<thead>
<tr>
<th>System fitness</th>
<th>Agent fitness</th>
<th>Safety factor</th>
<th>Collision frames</th>
<th>Liveness factor</th>
<th>Speed ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F = \frac{1}{N} \times \sum_{i=1}^{N} F_i$ (N is the number of agents)</td>
<td>$F_i = (S_i \times L_i)^4$</td>
<td>$S_i = \begin{cases} (1 - \frac{C_i}{50})^2 &amp; \text{if } C_i &lt; 50 \ 0 &amp; \text{otherwise} \end{cases}$</td>
<td>$C_i = \text{the average number of frames in every 10000 frames that pedestrian } i \text{ involves in collision with either stationary obstacles or other agents}$</td>
<td>$L_i = \begin{cases} 1 &amp; \text{if } R_i &gt; 0.5 \ (2 \times R_i)^8 &amp; \text{otherwise} \end{cases}$</td>
<td>$R_i = \frac{\text{average speed of pedestrian } i \text{ in simulation}}{\text{his preferred speed}}$</td>
</tr>
</tbody>
</table>

Table 2: Definition of fitness.

We put two factors, liveness and safety, in the definition of agent fitness, because we believe both of them are crucial to the realism of a pedestrian model. However, these two conflict with each other. Although we can always guarantee one of them by sacrificing the other, the most desirable is to have both of them in harmony—collisions shall surely be avoided and meanwhile pedestrians shall moving in their pleasant pace. The fitness value is such a measure that reveals the tradeoff between the two factors. It has the range from 0 to 1 and the closer to 1 the better.
Figure 14: A simple synthetic environment. The environment can be bounded by a $110 \times 110 m^2$ box. In the picture, dark blue objects are obstacles, thin long green objects are walls and red dots are agents.

B.2 Result Analysis

In Figure 15, we show the plot of fitness values of all the 720 possible permutations. The $y$-axis in this plot shows the fitness score of each permutation ordering along the $x$-axis. The plot seems to be a chaos at the first glance. But when examining it carefully, we notice that in the middle of the diagram near $x = 240$, there is a sharp boost in fitness value. In fact, $x = 240$ corresponds to the last permutation ordering starting with Routine $B$ ($B - F - E - D - C - A$) and $x = 241$ is the first starting with Routine $C$ ($C - A - B - D - E - F$). The sudden increase happens exactly from $x = 240$ to $x = 241$ and fitness values stay high for a while after $x = 241$. Now let us look at $F_b$ and $F_c$, the two sets of fitness values of permutations starting with Routine $B$ and $C$, respectively, shown by two red circle in the figure: $F_b = \{y(x)|x \in [121, 240]\}$ and $F_c = \{y(x)|x \in [241, 360]\}$.

It is obvious from the plot that most values in $F_b$ are much smaller than those in $F_c$. (Actually the mean
of $F_b$ is smaller than the minimal value of $F_c$.) Therefore, we can conclude that permutations $C - \ast\ast\ast\ast\ast$ are generally better than $B - \ast\ast\ast\ast\ast$. By similar observation, together with further data analysis such as sorting permutations by partial ordering and comparing shape of different parts of the plot, etc., we found the following fact with a permutation $P$:

- Those starting with $A$ and $B$ usually give poor performance.
- The later $D$ appears in $P$, the better the performance will be.
- It’s better for $C$ to appear before $A$, but no need for them to be adjacent.
- It’s better for $A$ to appear before $B$, but no need for them to be adjacent.
- If $F$ appears earlier than $C$, $D$ and $E$, it has the exact same performance as if $F$ is omitted. (This is obviously true as routine $F$ is designed to correct directions picked by routines $C$, $D$ or $E$. If they are not executed, $F$ will have no effect.)

Actually, almost all of the high performance permutations have $A$ before $B$ and after $C$ and have $D$ at the end. It is difficult to fully explain the result of the permutation search. We humbly provide our speculative explanation in the following and hope it will be inspiring to readers.

1. **The order of $C - A$.** Imagine a crowd, among which there is pedestrian $H$, moves at a certain direction. Chances are that $H$, if not on the edge of the crowd, will not bump into a stationary obstacle as long as he stays within the crowd, since the people on the boundary of the crowd may have already dealt with the obstacle, if any, and have steered the crowd to a safe direction (see Figure 16). So the only thing $H$ need to do is to stay within the crowd by using routine $C$. The order of $C - A$ allows a pedestrian to take advantage of efforts made by others—“let the others do the obstacle avoidance job and by following the crowd I probably do not have to do it at all”.

![Figure 15: Plot of fitness values of all permutations.](image)
Figure 16: Explanation of the order of $C - A$: pedestrian $H$ can avoid obstacles by simply following the crowd and letting others (those labeled $P$) deal with them. So can pedestrian $R$.

2. The order of $A - B$. Put $B$ after $A$ is sensible as whenever we want to check a turn, the turn itself should better be already determined. Otherwise, the check is a waste of effort. In the six routines, only $A$ will change the turning angle so big that the turn may need more than one step to finish. Therefore, it is better for $A$ to appear before $B$.

3. The later appearance of $D$. $D$ is to avoid the pedestrians coming toward me (either from side or front) that may cause potential collision on my future trajectory. Generally speaking, it considers pedestrians a bit far away and therefore its situations are usually not as urgent as those in other routines. So all of the avoiding options in $D$ are small changes in either moving speed or turning angle or both, which means motor control command issued by the previous routine is likely to be preserved. If $D$ appears early in a permutation, other routines may likely overwrite $D$’s motor control command with their bigger changes. However, if it gets executed at the end, its result will surely remain untouched while motor control command issued by the previous routine is preserved as well (or only slightly altered).

4. “Flexibility” of $E$ and $F$. If routines $A$, $B$, $C$, and $D$ are perfectly designed, almost all danger situations will be dealt with at their early stages and it is most likely that situations described in $E$ and $F$ will hardly happen. So ideally, $E$ and $F$ do not provide as much help as the other four. On the other hand, even if $E$ and $F$ might help a lot, both of the avoiding options for them involve “slowing down to a stop” should a threat be detected and subsequent reactive behavior routines will hardly have any effect on a pedestrian if he has already stopped. Therefore, given that a threatening situation described by $E$ or $F$ confronts the pedestrian, he will almost always have the same reaction—slow down to a stop—regardless of the order of the reactive behavior routines, which makes
and $F$ fit anywhere. However, in some cases early routines may redirect the pedestrian such that the original threat does not block the way any more but new threats appear. Due to the existence of such occasions, the positions of $E$ and $F$ do affect the performance a bit, but the effect is not strong enough for us to figure out the rule. So they appear “flexible” to us.

<table>
<thead>
<tr>
<th>Permutation</th>
<th>333 Agents</th>
<th>666 Agents</th>
<th>1000 Agents</th>
</tr>
</thead>
<tbody>
<tr>
<td>CABFED</td>
<td>4</td>
<td>22</td>
<td>84</td>
</tr>
<tr>
<td>FCABED</td>
<td>3</td>
<td>25</td>
<td>85</td>
</tr>
<tr>
<td>CEABFD</td>
<td>3</td>
<td>23</td>
<td>94</td>
</tr>
<tr>
<td>CAFBED</td>
<td>4</td>
<td>24</td>
<td>99</td>
</tr>
<tr>
<td>ECABFD</td>
<td>1</td>
<td>31</td>
<td>102</td>
</tr>
</tbody>
</table>

Table 3: Result performance. Average number of collisions happened in simulations with different number of pedestrians using various best permutations of reactive behaviors.

Before this exhaustive search, we had originally designed, out of our intuition, the order of reactive behavior routines to be simply $A - B - C - D - E - F$, which corresponds to $x = 1$ in the plot in Figure 15. This choice turned out to have the performance no better than the average. Table 3 lists some of the best permutations we found together with the number of collisions that happened in several Penn Station simulation experiments (with each being 20 minutes long in virtual time, or 36000 frames) with different number of pedestrians. (In our implementation, we do not impose hard constraint to prevent collisions.) Most of the collisions are human-human and less than 3% are human-obstacle. Collisions usually last no more than 1 second and for the human-obstacle type collisions, obstacle-crossing (e.g., moving “across” a solid wall) never happens.

Despite the satisfactory performance, our approach at the reactive behavior level may not be the only good answer. Alternatively, it might be possible to integrate these reactive behavior routines somehow “in parallel” instead of “sequentially” or one can even come up with another set of key routines.

C Additional Motivational Behavior Routines

In this appendix, we describe several representative higher level behavior routines in detail. As mentioned in Section 4.3, these routines depend greatly on other routines that are beneath them in our bottom-up 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 at the spot, standing, sitting, etc. In addition, they also rely on specialized environment objects for abstract level interpretation of situation in order to make decisions. Next we detail the routines.
Figure 17: Approach and watch performance. The yellow pedestrian is interested in the performance. He finds an available spot and approaches it. Among the current watchers, there are two (in blue) on the right who are about to leave. And outside the area, pedestrians (in green) are constantly passing by.

1. If $A$ is still far away, use navigation behaviors to approach $A$.
2. Once $A$ is close enough, find an available watching point $p$ around $A$.
3. Use detailed arrival behavior to approach and reach $p$.
4. Turn to face the performance.
5. Watch the performance for a while.
6. Turn to face the outside.
7. Leave the performance.

Table 4: Surround artists and watch performance

C.1 Surround Artists and Watch Performance

Suppose a pedestrian is attracted by a performance nearby, he will use the routine shown in Table 4 to approach the performance and watch it until he leaves. In the routine, $A$ is the performance area defined as a (part of) circular area surrounding the performing artists, illustrated in Figure 17.

In Step 2, a pedestrian will use detailed path-planning to find an available watching point and plan a path to it simultaneously. He first puts the performance area $A$ as a target on a grid path map and then add every watcher surrounding the area as a static circular obstacle, which can effectively prevent path search to access the target area through its standing point. This can lead path search toward the nearest available interval space around the area, if one exists. In cases when no space is available, the pedestrian can either give up or enlarge the target performance area by the size of pedestrian bounding box and try finding a path again. This probabilistic
strategy leads to a sparse second layer of watchers.

In order to quickly identify all the current watchers, a pedestrian needs the help from a specialized object—the performance area object. This object keeps track of the watchers that surrounding the area. Whenever a new watcher joins, it will be registered into the watcher list. Once it leaves, it will be removed. These operations are triggered by the watchers themselves. If a watcher-to-be is approaching and is close to its watching position (say one meter away), he will request a registration. This also in a way resolves conflict of two watchers-to-be competing a spot big enough for only one. The first one that issues the registration request will get the spot. It is fair enough as the first issuer is usually also the closer one. When he leaves the performance area, a watcher will issue a remove request once he is far away enough (say more than one meter away). Specifically in the routine, the registration request is usually issued late in Step 3 and the removing request late in Step 7.

### C.2 Make a Purchase

When a pedestrian need to get something $T$, say a ticket or a drink, through a purchase, the routine “make a purchase” shown in Table 5 will be employed.

In Step 2, a routine similar to passageway selection (see Table ??) will be used for the pedestrian to make a choice among several available purchasing places. In Step 7.2, one of several waiting motions of different styles in the motion repertoire will be picked in a probabilistic way to express the (im)patience of the pedestrian.

Like the routine in last section, “make a purchase” requires two types of specialized objects to help it analyze the situation. The wait-line object which keeps track of the waiting pedestrians on the line will tell a pedestrian how many people are on the line, who is the first and who is the last. The purchase-point object can point out
1. Find out all places within the current region that sell $T$.
2. Pick the best one $B$ among them in terms of proximity and expected wait time.
3. If $B$ is far
4. use navigation behavior to approach it.
5. else if there is a spot available in $B$ for transaction
6. approach and take it.
7. else
7.1 Go and stay behind the last person on the waiting line.
7.2 Wait either patiently or impatiently.
7.3 If the line moves forward, follow it to move forward.
7.4 If I become the first on the line and a spot for transaction is available
7.5 then approach and take the spot.
8. Make a purchase at the transaction spot.
9. Leave the transaction spot.

Table 5: Make a purchase

1. Find all seats around within the current region.
2. Pick the best available one $B$ among them in terms of proximity and expected resting comfort.
3. If $B$ is far, use navigation behavior to approach it.
4. Once $B$ is close, 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. Sit for a while.
7. Stand up and leave the seat.

Table 6: Pick a comfortable seat and take a rest

whether a transaction spot is available for taking. And similarly, pedestrians need to issue requests to register themselves into and remove themselves from those specialized objects. Given that there are other pedestrians constantly passing by (as shown in Figure 18), the help of these specialized objects are crucial to situation analysis and decision making.

C.3 Take a Rest

The last example (see Figure 19) is the routine that enables the pedestrian to take a rest as shown in Table 6. The basic structure is pretty much the same as the previous two. The selecting behavior here uses the resting comfort in addition to proximity as the criteria. And as usual, there is a specialized object—*seat-space*, which tracks all the available spaces on a resting facility—that helps the execution of this behavior routine.
C.4 Summary

To summarize, the routines described in this appendix represent a big category of meaningful behaviors suitable for pedestrians. Such behaviors involve inter-personal conflicts among pedestrians on available resources. While everyone is trying to maximize its own benefit through various selecting behaviors in resolving these conflicts, they shall at the same time obey rules that are explicitly or implicitly defined for the sake of others in the society. It is exactly the balancing of these two sides that makes their ultimate behaviors appear to be natural and rational.

D Environmental Modeling

As outlined in Section 3, 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 environment objects (see Figure 3 on page 6). 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.

D.1 Topological Map

At the highest level of abstraction, a graph serves to represent the topological relations between different parts of a 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, such as 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 momentarily.

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 \( V(A, T) \), the path-to-via of \( A \) associated with \( T \), is a set of pairs defined as follows:

\[
V(A, T) = \{ (B, C_B) | \ B \text{ is a region} \quad \& \quad \exists p \in P(A, T) \quad \& \quad C_B = \text{length of } p \quad \& \quad B \text{ is next to } A \text{ on } p \}. \tag{1}
\]

As the name suggests, if \( (B, C_B) \) is in \( V(A, T) \), then a path of length \( C_B \) from \( A \) to \( T \) via \( B \) exists. In other words, \( V(A, T) \) answers the question “To which region shall I go, and what cost shall I expect if I am currently in \( A \) and want to reach \( T \)?”

Given a graph, the path-to-via information is computed offline in advance using the incremental algorithm shown in Table 7. Note that after Step 3 in the algorithm, only those entries are stored whose cost is minimal or \((\text{minimal} + 1)\). In this way we can avoid paths with cycles. To understand this, consider \( V(A, C) \) for the graph in Figure 3. \( C \) is a direct neighbor of \( A \); so \((C, 1)\) is clearly an entry of \( V(A, C) \). \((B, 3)\) is another entry as \( A-B-A-C \) is also a possible path from \( A \) to \( C \). 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 graph, are perception maps and path maps together with a list of objects inside that region. Next we will describe them in turn.

### D.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 8 summarizes their
Given $G(N, E)$, a graph with $N$ nodes and $E$ edges:

1. Initialization:
   for each node $A$
   for each target node $T$
   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 length $L$ = 1 to $N - 1$
   for each node $A$
   for each target node $T$
   for every neighbor node $B$ of $A$ and any node $X$ in $G$
   if ($X, L - 1) \in V(B, T)$
       then add $(B, L)$ to $V(A, T)$

3. Keep only low cost entries:
   for each node $A$
   for each target node $T$ and any node $Y$ in $G$
   let $C_{min}$ be the minimal cost in $V(A, T)$
   for each entry $E(Y, C)$ in $V(A, T)$
   if ($C > C_{min} + 1$)
       then remove $E$ from $V(A, T)$

Table 7: Algorithm for computing path-to-via information.

similarities and differences, and the next two subsections present the details.

D.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. Thus, the grid defines a rasterization of the region. This rasterized “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. [Tu and Terzopoulos 1994; Maes et al. 1995]). Each segment is rasterized onto the grid map (see the left and center panels of Figure 20). 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. Most importantly, however, it does not depend on the number of objects in the virtual environment. Without the help of grid maps, the necessary line-object intersection tests would be time consuming given 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 $0.2 \sim 0.3$ meters.

### D.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):

1. If ($C_{old} == C_{new}$) then do nothing.

2. Otherwise, remove $H$ from $C_{old}$ and add him to $C_{new}$.

A hash map is used to store the list of pedestrians within each cell. 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 Figure 20. In 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 will be examined earlier. Once a maximum number (currently set to 16) of nearby pedestrians are perceived, the sensing is terminated. This is intuitively inspired by the fact that usually people can simultaneously pay attention only to a limited number of other people, especially those in close proximity. 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.

<table>
<thead>
<tr>
<th>Type</th>
<th>Cell Size</th>
<th>Update Cost for the Entire World</th>
<th>Query Cost per Pedestrian</th>
</tr>
</thead>
<tbody>
<tr>
<td>stationary</td>
<td>small ($\sim 10^{-1} m$)</td>
<td>0</td>
<td>constant, given the sensing range and acuity</td>
</tr>
<tr>
<td>mobile</td>
<td>large ($\sim 10^1 m$)</td>
<td>linear in the number of pedestrians</td>
<td>constant, given the sensing fan and max number of sensed pedestrians</td>
</tr>
</tbody>
</table>

Table 8: Comparison of perception maps.
D.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, which will be briefly presented next. The path planning algorithms will be detailed in a subsequent chapter, after we have finished describing the environment representation hierarchy.

D.3.1 Grid Path Map

Grid maps, which are useful in visual sensing, are also useful for path planning. We can always find a shortest path on a grid map, if one exists, using the well-known $A^*$ graph search algorithm [Rabin 2000; Stout 2000].

In our system, grid path maps are used whenever a detailed path is needed. Suppose $D$ is the direct distance between pedestrian $H$ and his 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 its 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 are found quickly, after the search covers only a small portion of the entire search space. Roughly speaking, in most cases the space that must be searched is bounded by $4(D_{\text{max}}/c)^2$, where $c$ is the cell size. The typical values for these constants in our current system are $1 \leq D_{\text{max}} \leq 10$ meters and $10^{-1} \leq c \leq 1$ meters.

In addition, grid path maps are used when path search fails on a quadtree path map. This will be explained later in the section on path planning (Chapter ??).

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 of 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.

D.3.2 Quadtree Path Map

Every region has a quadtree map, which supports fast online path planning [Botea et al. 2004]. Each quadtree map comprises

1. a list of nodes $N_i$ ($i = 0, 1, 2, \cdots, m$), which together cover the whole area of the region (see Step (3) in Figure 21);

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 Figure 21); and

3. a pointer to an associated grid map with small cell size (see Step (1) in Figure 21).

Each node $N_i$ of the quadtree [Samet 1989] stores the following variables:

1. $L_i$, where $0 \leq L_i < C$, 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;

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
Figure 21: Construct a quadtree map.

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 Figure 21 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. However, in occasions when path planning fails because of this exclusion, grid maps will be used to find the path, as will be described in Chapter ??.

D.4 Specialized Environment Objects

At the lowest level of our environmental hierarchy are environment 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 environment 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 sit. By querying nearby bench objects, weary virtual 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.

D.5 Processing Queries

The environment representation, together with the algorithms designed on it, efficiently provides accurate perceptual data as well as planning results in response to the various queries that may be issued by an autonomous pedestrian. Typical queries are explained next in order of increasing abstractness.

**Sensing ground height** To ensure that a virtual pedestrian’s feet touch the ground in a natural manner, especially when climbing stairs or locomoting on uneven ground, the pedestrian must query the environment model in order to sense the local ground height so that the feet can be planted appropriately. Each grid map cell contains the height functions of sometimes a few (most often a single) ground objects. The greatest height at the desired foot location is returned in constant time.

**Visual sensing** As stated earlier in section D.2 (see also Figure 20), our data structures dramatically increase the efficiency of the sensing processes when a pedestrian must perceive static obstacles and nearby pedestrians, which is a crucial component of obstacle avoidance. On the perception map for static objects, rasterized eye rays are used to detect static obstacles. On the perception map for dynamic objects, a constant number of neighbor cells are examined to identify nearby pedestrians. Both of the algorithms are localized and do not depend on the size of the world, the number of objects or pedestrians, or anything else that increases with world size.

**Locating an object** Given a location identifier (say, “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, it can be located quickly (say, “under the lower concourse”) 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.

**Acquiring high level interpretation of specific situation** Abstract interpretation of the environment is indispensable for creating high level behaviors such as “to get a ticket” to be described in Section 4.3. Take it as
an example. In order to get a ticket, a pedestrian needs to figure out (1) the length of wait line at each booth to pick a fast one, (2) the last person on the wait line in order to go and wait on the line, (3) when he becomes the first person on the line, (4) whether a window is available for him to make the purchase. Such queries are all answered by specialized objects, in this case a queue object and several purchase point objects associated with the ticket booth, which keep track of the evolving situation.

Planning paths between regions Here, 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 one of the path-searching schemes described in Chapter ??, 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.

D.6 Summary

In this chapter, we presented a large-scale environment model, which includes a sophisticated set of hierarchical data structures that support fast and accurate perceptual queries and path search algorithms. In the next chapter, we will describe the algorithms employed for online path-planning on path maps in the environment model.

Acknowledgements

The research reported herein 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 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 software API.

References


