AI Magazine Volume 23 Number 4 (2002) (© AAAI)

Applying Perceptually Driven Cognitive Mapping to Virtual Urban Environments

Randall W. Hill, Jr., Changhee Han, and Michael van Lent

■ This article describes a method for building a cognitive map of a virtual urban environment. Our routines enable virtual humans to map their environment using a realistic model of perception. We based our implementation on a computational framework proposed by Yeap and Jefferies (1999) for representing a local environment as a structure called an absolute space representation (ASR). Their algorithms compute and update ASRs from a 2-1/2-dimensional (2-1/2D) sketch of the local environment and then connect the ASRs together to form a raw cognitive map.[1] Our work extends the framework developed by Yeap and Jefferies in three important ways. First, we implemented the framework in a virtual training environment, the mission rehearsal exercise (Swartout et al. 2001). Second, we developed a method for acquiring a 2-1/2D sketch in a virtual world, a step omitted from their framework but that is essential for computing an ASR. Third, we extended the ASR algorithm to map regions that are partially visible through exits of the local space. Together, the implementation of the ASR algorithm, along with our extensions, will be useful in a wide variety of applications involving virtual humans and agents who need to perceive and reason about spatial concepts in urban environments.

Our goal is to develop virtual humans with believable perceptual and spatial behaviors. For a growing number of computer games, military training simulations, and immersive learning environments, the willingness of the participant to suspend disbelief hinges on the realism of the behavior of the virtual humans. Behaviors such as self-location and way finding have been investigated extensively in mobile robot applications, but there are numerous other spatial tasks more human in nature that need to be simulated in these applications. Interesting examples include communicating spatial information in natural language and social conventions such as initially blocking a doorway with your body and then stepping back to invite the visitor in. In military training simulations, these examples include coordinated tactical movements, crowd control, the avoiding of snipers and ambushes, the selecting of helicopter landing zones, and the establishing of a security perimeter, to name a few. Underlying all these behaviors is the ability to perceive and build a spatial representation of the environment.

Humans are quite good at remembering the layout of the places they inhabit or have visited and using this information to reason about everyday tasks such as finding the local grocery store and locating a parking space in spite of the traffic jam at one end of the parking lot. Becoming familiar with the configuration of a place like a town is a process that involves walking around and looking at buildings, trees, landmarks, streets, and other details of the environment that are subsequently encoded into memories that make the place recognizable and easily navigated. The process of forming these spatial memories is called *cognitive mapping* (Chown, Kaplan, and Kortenkamp 1995; Kuipers 2000, 1978; Yeap 1988; Yeap and Jefferies 1999). The ability to build a cognitive map is useful for any agent that has a need for tracking its location, navigating, and determining where places are located with respect to one another (Chown, Kaplan, and Kortenkamp 1995; Kortenkamp, Bonasso, and Murphy 1998; Kuipers 2000, 1978; Levitt and Lawton 1990).

This article describes a method for building a cognitive map of a synthetic urban setting

Copyright © 2002, American Association for Artificial Intelligence. All rights reserved. 0738-4602-2002 / $2.00

based on the realistic limits of human visual perception. Humans have a limited field of view and cannot see through solid objects such as walls, and these same limitations are imposed on our virtual agents. Only by making a series of observations from different perspectives over time can a cognitive map be built.

We based our implementation on a computational framework proposed by Yeap and Jefferies (1999) that represents a local environment as a structure called an *absolute space representation* (ASR). Building an ASR involves perceiving the local surroundings and the area immediately visible to the viewer and computing the boundaries and exits of this space. The boundaries are obstacles that prohibit movement through the space such as walls. Exits are gaps in the boundaries that permit the agents to leave one local space and enter another. For example, a room would be a single ASR with a number of boundaries (walls) and a single exit (the door). The exit would connect to another ASR (the hallway) with a number of boundaries and exits (doors) connecting to more ASRs representing other offices. By exploring a series of local spaces, representing them as ASRs, and connecting them together through their exits, a viewer builds a raw cognitive map.[2] We have taken this framework and extended it in a number of ways:

Our goal is to develop virtual humans with believable perceptual and spatial behaviors.

We applied a theoretical computational framework of cognitive mapping to a training application that includes virtual humans in a virtual environment. To date, most cognitive theories have been implemented in mobile robots, whose perceptual abilities are somewhat different than a human's and whose purpose is not to exhibit humanlike behavior. Yeap tested his theory with a simulated robot in a two-dimensional (2D) world. Our cognitive mapping is done in the urban environment of the mission rehearsal exercise (Swartout et al. 2001). Urban environments are of particular interest to game developers and the military simulation community.

We extract a 2-1/2D sketch from a scene in a graphically rendered virtual world. Yeap finesses the issue of perception by assuming that a 2-1/2D map is going to be available. Computer games and military simulations generally also avoid the perception step by using a database of 3D models.

We extended Yeap's and Jefferies's (1999) cognitive mapping algorithms. Instead of limiting the agent to only building one ASR at a time, focusing on only the immediate surroundings, we save the residue of what has been perceived through the exits in the local environment and begin the construction of the new ASRs before the areas are visited. This particular extension was made because we believe that cognitive mapping must not be limited to places that have been explored physically. Virtual humans need to build cognitive maps in anticipation of the next space they will enter.

Motivation

As previously stated, we are developing virtual humans for an immersive military training environment called the mission rehearsal exercise (MRE) system. In the MRE, the participants interact with virtual soldiers to perform missions involving tasks such as securing an area from attack, controlling an angry crowd, tending to an injured child, and securing a landing zone for a medevac. To perform these tasks, the virtual soldiers must explore their surroundings, locate a suitable clear space, identify the potential lanes of attack into that space, and position themselves to block these lanes of attack. Performing these tasks requires spatial knowledge about landing zones and lanes of attack as well as perception of the environment to locate regions and exits that match these spatial concepts.

Many current applications finesse perception and spatial reasoning as much as possible. Computer games (Liden 2001) and military simulations (Reece, Kraus, and Dumanoir 2000; Stanzione et al. 1996) often require a designer to annotate the environment with invisible spatial references to help virtual humans behave believably. Another approach is to give agents omniscient perception, giving them a complete map of the static environment and the current location of every dynamic entity. The alternative, demonstrated by the research presented here and the research of Terzopoulos and Rabie (1995), is to give virtual humans realistic perception of their environment. Perception would be realistic both in the types of information sensed (no invisible spatial cues, no map) and the limitations on this sensing (no 360-degree field of view, no sight through walls). As the virtual human moves around and views the environment from different perspectives, it constructs a cognitive map of its surroundings and uses the map for spatial reasoning.

Creating a cognitive map of the virtual environment, based on realistic perception, has a number of advantages over annotating the environment with spatial references. Different virtual humans can represent the environment with different cognitive maps based on their roles and knowledge. Although the underlying

ASR representation might be the same, the annotations placed on the spatial map would depend on the role and knowledge of the virtual human. A local resident's cognitive map of his/her home city, including street names and friend's houses, would be very different from the cognitive map of a soldier sent to defend that city, which might include lines of attack and defensive strong points. Different map representations, based on different roles, will have far-reaching implications on the behavior of the virtual humans, affecting everything from natural language understanding and generation to movement and goal selection. In addition, cognitive mapping doesn't require the environment designer to embed spatial information in the environment, which can be a time-consuming process. When spatial knowledge is encoded in the model, the designer must anticipate every behavior that could potentially be associated with a feature, leaving little for the agent to decide.

A cognitive map built from realistically limited perception also has a number of advantages over giving agents omniscient perception. At first, it might seem that omniscient agents are simpler because they don't require a realistic model of perception. However, for their behavior to be believable, omniscient agents must pretend to ignore the sensory information they wouldn't realistically perceive. Differentiating between the information they should and should not pretend to ignore requires a model of realistic perception at some level. In fact, realistically limited perception can help to guarantee that a virtual human is behaving believably by not allowing behavior to be affected by information a real human won't know. Realistic perception will lead to virtual humans that explore the environment and look around realistically to map their environment. In addition, these agents will get lost and make realistic mistakes based on their limited knowledge of the environment.

Building a Cognitive Map

Based on the ASR algorithm developed by Yeap and Jefferies (1999), our virtual human maps the local environment by continuously perceiving a scene, constructing a 2-1/2D sketch of the surfaces, building a local map, and connecting it with other local maps that it has already constructed in the process of exploring a virtual town. Our mapping algorithm takes into account major static objects (that is, buildings and trees) that exist anywhere in the urban environment. Buildings are represented in the virtual environment by polygons that form the walls and roof of each building. Features on the walls (that is, doors and windows) are texture mapped onto the polygons and are thus ignored by our system. Each tree is represented by two or three polygons arranged in an X or star shape, with the image of the tree texture mapped onto each polygon. The perception system constructs a 2-1/2D sketch from these static objects, as described here.

The basic idea behind Yeap's (1998) theory of cognitive maps is to build a representation of the open space around the viewer. As previously mentioned, this space is defined by the boundaries and exits that surround the viewer. The key to Yeap's construction of a raw cognitive map is the identification of the exits, which are defined as gaps between obstacles. This is the commonsense definition of an exit. How does one compute it? We need to start by looking for gaps in the surfaces surrounding the viewer, beginning by looking for occluded edges. An exit is a way of leaving a local space. It is also a signal to compute a new ASR. Exits serve another important purpose in that they identify places in the space that have not been uncovered yet. These are places that are occluded and that the viewer is not sure of. It might not actually be an exit, merely a place that has not been explored yet. If the goal is to build a complete raw cognitive map of an area, then the exits might actually be areas one needs to explore more fully, thus guiding the mapping process.

Constructing a 2-1/2–Dimensional Sketch

Yeap's and Jefferies's cognitive mapping algorithm takes as input a 2-1/2D sketch of the scene (Marr 1982; Yeap and Jefferies 1999). The sketch is the set of boundary surfaces, including depth information, currently perceived by the viewer. These surfaces are represented as an ordered list of edges (with vertexes), as they appear from left to right in the field of view. How is this sketch constructed? The answer depends on the domain of the application. Yeap tested the algorithm in a relatively simple 2D simulated domain but gives no details about how the sketch was derived. In a mobile robot domain, the sensors and computer vision system detect the edges and surfaces and recognize objects in an effort to determine that the obstacles are indeed buildings or other real things. Much progress has been made in this area (for example, see Kortenkamp, Bonasso, and Murphy [1998] on mobile robotics), but it still remains a significant challenge. One of the contributions in this article is an approach to building a 2-1/2D sketch in graphically rendered virtual environments.

We applied a theoretical computational framework of cognitive mapping to a training application that includes virtual humans in a virtual environment.

Figure 1. View of a Street in a Virtual Urban Environment.

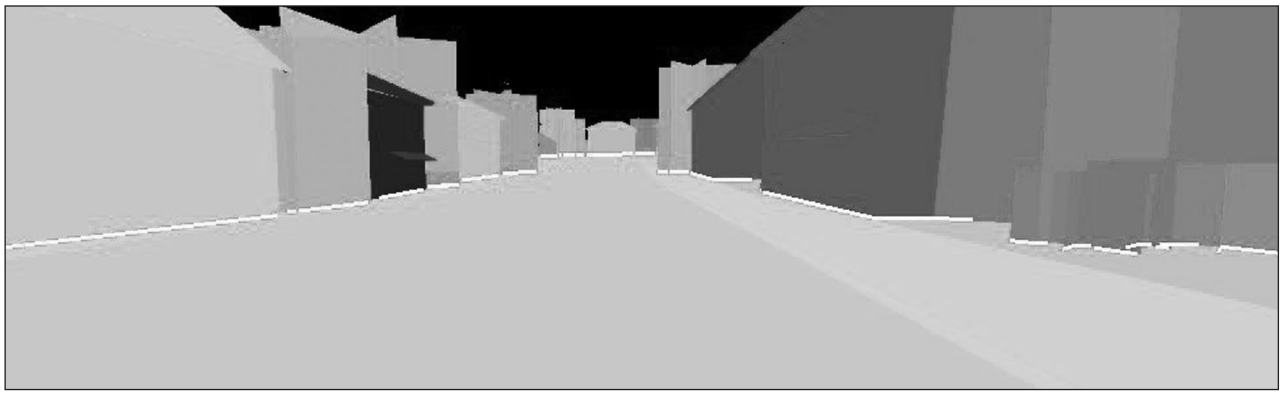

Figure 2. Detecting the Edges in the Urban Scene from Figure 1.

We took a hybrid approach to building the 2-1/2D sketch that combines the use of the graphic model (known as the scene graph), which is represented as a graph of nodes corresponding to the objects in the scene, a graphics-rendering engine, and visual routines for edge detection. Each of the buildings and other objects in figure 1 are represented as nodes in the scene graph that will be rendered in real time. Rather than relying on computer vision to recognize that these are buildings or trees, we simplify the process by using the scene graph to differentiate between individual buildings, individual trees, and the ground. However, this only takes us part of the way toward building a 2-1/2D sketch. We need to take the following steps:

First, traverse the scene graph and assign a unique number to each node corresponding to a static object (that is, building or tree), thus taking advantage of the node predraw callback function in the graphics routines. The advantage of this traversing process is that each of the static objects, which are fairly simple boxes or star-shaped "trees" underneath the texture maps, will be assigned a unique number, which will be used later for edge detection.

Second, cull the nodes, leaving only the visible ones. This step creates the occlusions that the viewer would experience in the real world. Without this step, the model would be transparent to the viewer, enabling the virtual human to see through solid walls. This step is essential for creating a 2-1/2D sketch. Without the occlusions, the viewer would have been able to create a full 3D model.

Third, draw each node with its assigned number (shade). The result of this step can be seen in figure 2, where the static objects appear as different shades of gray corresponding to the unique numbers that were assigned.

Fourth, find the edges between the ground and the static objects using standard edge-detection techniques. Use the graphics *z*-buffer to get the depth into the picture—we need the (*x, y, z*) positions of the points. Assume you know the color/number of the ground. Scan from the sky downward to find the ground edge. Do this across the image.

The result is a set of line segments along the boundaries between the static objects and the ground. Pixelation can result in short line segments that have to be joined together to form longer lines. These longer lines are smoothed out using standard edge-detection techniques.

The output from this step is a 2-1/2D

sketch, which is a set of edges and vertexes in a format that can be used for Yeap's ASR algorithm, which we describe in the next section (figure 3).[3]

Mapping the Local Space

Once a 2-1/2D sketch has been built, the key to computing an ASR is detecting where the boundaries and exits are located in the local space. Exits serve not only the obvious functional role of providing egress from a local space, but passing through an exit also triggers the construction of a new local map, which is represented as an ASR (Yeap and Jefferies 1999). Exits serve as the connections between the maps of local spaces (ASRs), and the raw cognitive map ends up being a network of exit nodes connecting local maps. Finding the boundaries of the local space is important for defining the extent of the area. Locating the exits is essential, both as a way of indicating how to leave a local space and as a way of connecting pieces of the cognitive map together into a whole.

Exits are detected by looking for places in the scene where one surface partially occludes another. The gap between the two surfaces is what Yeap and Jefferies (1999) call an *occluded edge*. An occluded edge has a visible vertex, which is also called the *occluding vertex* and is closest to the viewer, and an *occluded vertex*, which is where the occluded edge intersects with the hindmost surface. Let's assume we want to calculate an exit in figure 4. An occluded edge, *CD*, divides the surfaces in the current field of view. Thus, we split the surfaces into two groups: (1) one containing all the surfaces left of vertex *C* and the other containing all the surfaces right of vertex *C*. An exit is the shortest span between the occluding vertex (that is, *C*) and a point in the second group of surfaces. In this example, *CJ* is selected as the shortest span. Other candidates that were rejected as longer spans include *CD, CE, CF, CP, CG, CH, CI, CJ, CK,* and *CL*. Point *P* is identified because *CP* is a normal line to *FG*. In this case, *CJ* is a doubtless exit because *J*, the selected vertex, is not the occluded vertex. The *CJ* exit is the gap that must be crossed to reach the occluded edge. If *CD* were the shortest span, this exit would have been a doubtful exit.

To identify an exit, the surfaces from the current 2-1/2D sketch are scanned in order, from left to right, in a search for occluding vertexes. Because the exit is the gap that must be crossed to reach an occluded edge, identifying the exit starts with the occluded edge. Each unoccluded vertex is chosen, and the closest point on a surface contained in the opposite group is found. The exit is the edge formed by the un-

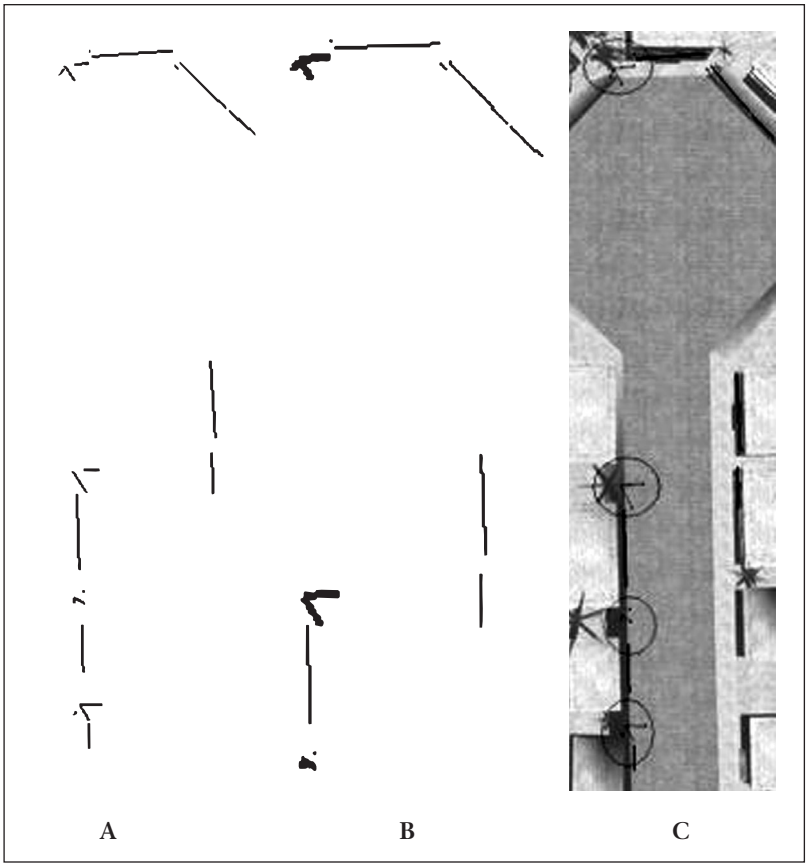

Figure 3. Three Top-Down Views of Part of an Urban Environment with the Detected Graund Edges.

A. A result of a 2-1/2D sketch. B. Boundary segments representing trees identified with a bold line. C. The boundary segments overlaid onto the urban environment with trees indicated by circles.

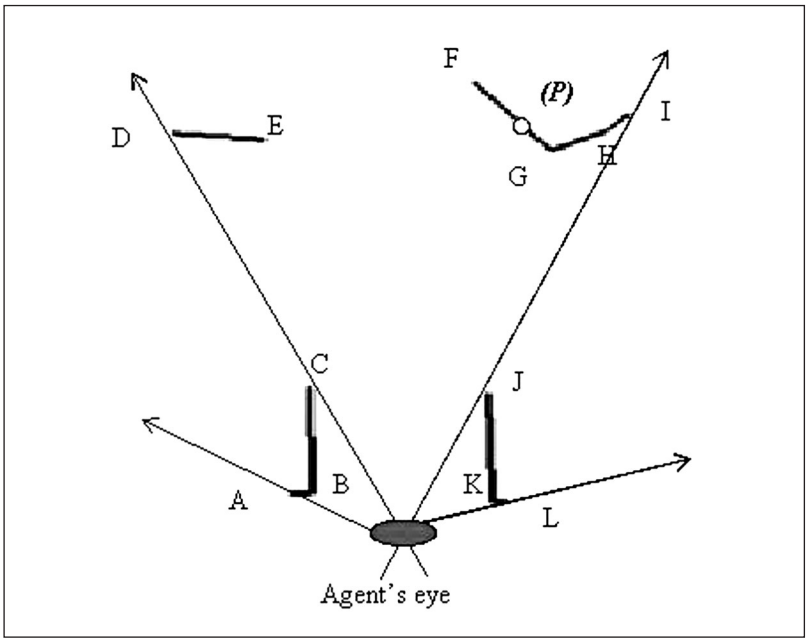

Figure 4. Calculating an Exit in a 2-1/2D Sketch.

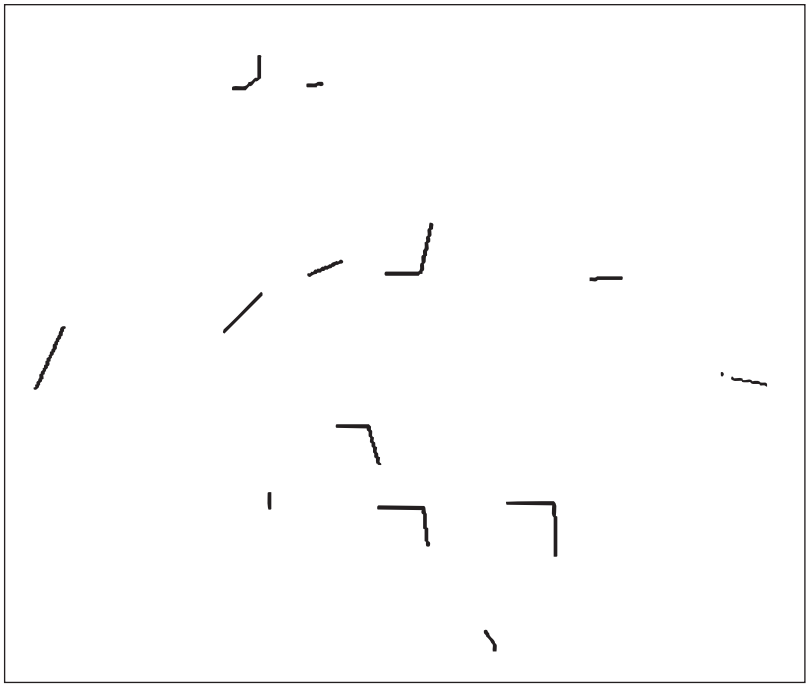

Figure 5. A Top-Down Perspective of a 2-1/2D Sketch of the Virtual Environment, as Perceived from the Position Marked by the End of the Line on the Right.

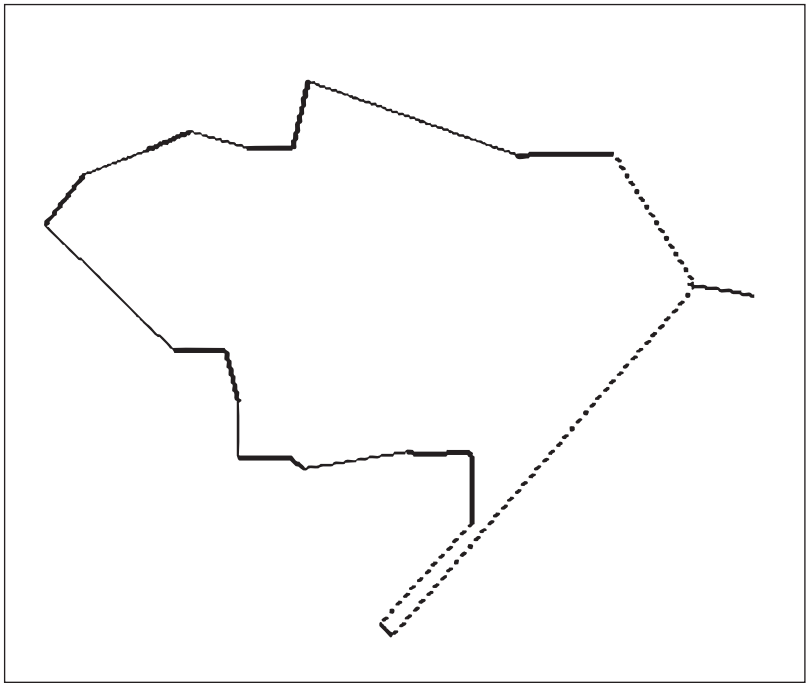

Figure 6. The 2-1/2D Sketch from Figure 5 after Exit Identification.

The building boundaries are shown as dark lines, the doubtless exits as thin lines, and the doubtful exits as dotted lines. The viewer's position is shown by the rightmost line.

occluded vertex and the closest point. Once identified, it is then inserted into the list of surfaces in its logical place adjacent to the surfaces contributing the vertexes. The surfaces beyond the exit are trimmed from the ASR. They are no longer taken into consideration for mapping local space because they have been determined to be outside the exit. Yeap discards the trimmings, but in our implementation, this residue is saved and used to map spaces outside one's local space. This subject is discussed in more detail in the section on mapping outside the local space. Figure 5 shows a 2-1/2D map of a virtual urban environment before the exits are identified. Figure 6 shows the same 2-1/2D map with the doubtless and doubtful exits identified and displayed. The slight differences between the boundaries in the two images are the result of slight variations in the edge detection between runs.

Updating the Local Map

Because the viewer's perspective changes over time, the ASR must continually be updated. Even a simple shift in gaze will uncover more details about the environment. Moving through the environment will cause some occlusions to be uncovered and others to be formed, so the question is how to incorporate this information into the raw cognitive map.

Yeap and Jefferies (1999) distinguish between two kinds of exits: (1) doubtful and (2) doubtless. A *doubtless exit* is one that takes the viewer out of the local space. It consists of two unoccluded vertexes; they must both have been visible sometime during the mapping process. In determining a doubtless exit, it is the shortest possible span between two surfaces. Once Yeap's algorithm has determined that an exit is doubtless, it no longer needs to be updated.

When one of an exit's vertexes is occluded, it is a *doubtful exit*. As the viewer moves through the environment, this type of exit must be updated. Thus, as one's perspective changes, more of the occluded edge can be uncovered, and the location of the occluded vertex will also change to be the shortest distance spanning the gap. This process goes on until one of two things happens: (1) the exit is identified as doubtless (that is, both vertexes are unoccluded) or (2) the occluded surface is completely uncovered, and it is discovered that there is no exit.

The ASR is updated once each frame, where the frame rate can be as high as 20 to 30 frames a second. This might prove to be excessive in the long run, but it works for now. Each update involves taking the following steps:[4]

First, sense the environment and construct a 2-1/2D sketch. Call this sketch perspective current-view.

Second, check whether the viewer is still inside the current ASR, which can be achieved with a simple intersection test: Draw a line from the viewer's current location to the initial position in the ASR, and check whether this line intersects with the surface of the ASR.

Third, if an exit has not been crossed, update the doubtful exits based on the current-view. If the change in perspective uncovers an occlusion, the size of the corresponding doubtless exit will decrease.

For each doubtful exit, (1) label the two surfaces that contribute vertexes to the doubtful exit as *S*1 and *S*2 and (2) if current-view includes *S*1 and *S*2, then replace the doubtful exit with the surfaces that lie between *S*1 and *S*2. **Note:** We found that we had to relax this condition somewhat because there are cases where the vertexes of the doubtful exit are outside the field of view of the agent.

Fourth, else, if an exit has been crossed, the viewer is no longer in the local space represented by the current ASR. The next section deals with this situation, which involves extending the raw cognitive map with the current ASR and either starting a new ASR or using a previously computed one.

Extending the Cognitive Map

As new areas are mapped, they are added to a network of ASRs that make up the raw cognitive map. Whenever the viewer crosses an exit and enters a previously unexplored area, a new ASR is computed. Figure 7 shows a raw cognitive map with three ASRs. In this example, the viewer starts where the arrows begin and proceeds up the street, turns left at an alley, goes between two buildings, and enters an open area surrounded by some buildings. The first ASR maps the street and ends when the street enters an intersection with another street; the second ASR represents the alleyway between the buildings; and the third ASR is still being formed for the open area, as shown on the left side of figure 7. Note that the third ASR contains doubtful exits on the left and right sides of the viewer, indicating that the area has not yet completely been mapped. Once the viewer's perspective has been rotated, these areas will be filled in with surfaces and doubtless exits. Figure 8 shows a more complete map of the third ASR overlaid onto the image of the town.

Extending a raw cognitive map requires the ability to recognize that an area that has previously been visited; otherwise, areas would be remapped every time they were visited. The recognition routine is triggered when the viewer crosses an exit.

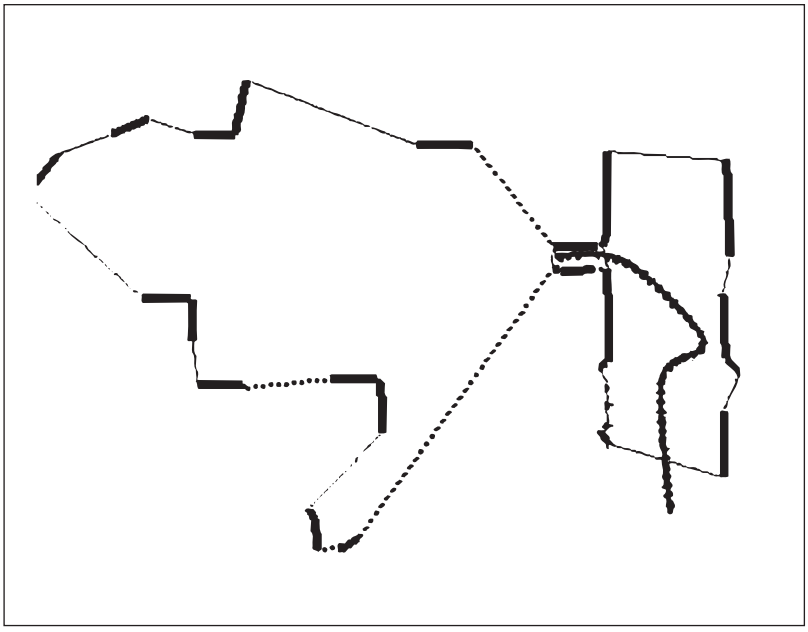

Figure 7. Three Absolute Space Representations Are Shown Connected Together.

The third ASR contains both doubtless (thin lines) and doubtful (dotted lines) exits.

When the viewer crosses an exit, there are three possible cases:

First, the newly entered space was previously mapped, and the exit is a known connector between the two ASRs. When such is the case, no updates to the raw cognitive map are required. Use the ASR from the raw cognitive map as a map of the local space.

Second, the newly entered space was previously mapped, but it was not known that this exit connected these two ASRs. In this case, update the raw cognitive map to reflect the fact that this exit is a connector, and use the ASR from the raw cognitive map.

Third, the newly entered space is unexplored, so the viewer must begin mapping it. The steps in mapping this space are (1) place the just-exited ASR into the raw cognitive map, (2) create a new ASR, and (3) connect the ASR the viewer just departed with the new ASR at the exit point.

Mapping outside the Local Space

We developed an extension to Yeap's and Jefferies' algorithm that enables the viewer to map spaces outside the current ASR. In their version, the ASR algorithm maps the local space by iteratively identifying exits and trimming off the surfaces beyond the exit. The only thing that is mapped is what is in the current

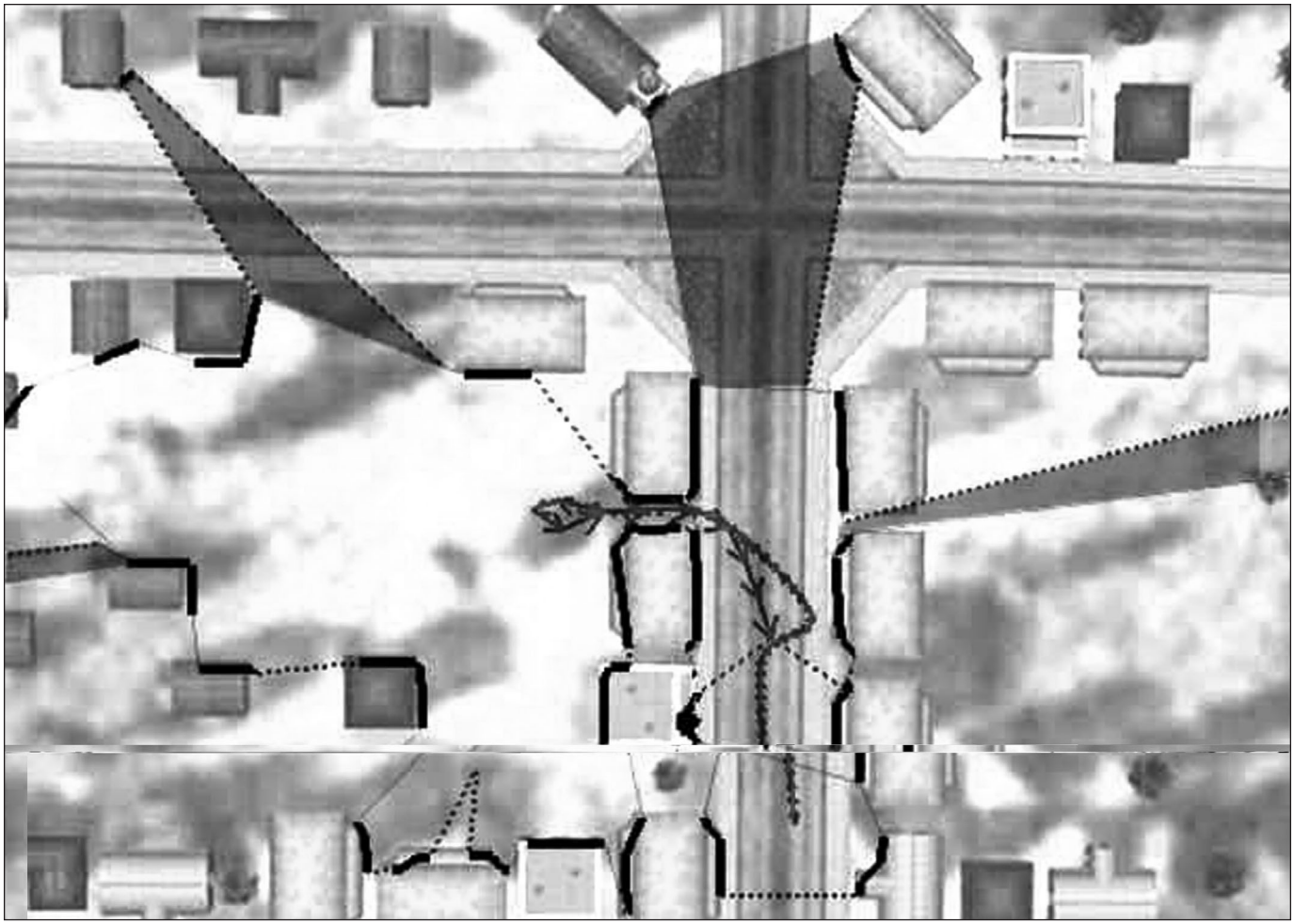

Figure 8. A Cognitive Map, Including Residual Absolute Space Representations (Shaded Regions) Constructed from the Residue of Local Computations.

local space as they define it. Our extension to Yeap's approach is to use the surfaces beyond exits to create a preliminary map of spaces that aren't local to the agent.

We do not believe that humans discard what they see on the other side of an exit. The cognitive mapping process is not confined to one's local space. A person walking around in an unfamiliar building will probably focus their attention on perceiving and mapping the local space, but it seems highly improbable that they would ignore the layout of a room that happens to be on the other side of a door or down a hallway. In fact, what is seen down the hallway (or down the street), which is a different local space, might provide important information that will impact the behavior of the viewer even before this space is entered.

An example of this arises in the context of an application that we have been working on for a military peacekeeping operation training exercise. Some virtual soldiers are looking for an open area that would be suitable for a medevac to land. A quick glance down an alley or street might reveal that there is no open space in the immediately adjacent spaces, but farther down the street, there is a major intersection where it might be possible for a medevac to land. The intersection can be observed and partially mapped without physically leaving the current local space. If we restricted the cognitive mapping to only areas that had been visited physically, then the soldiers would have to behave unrealistically to acquire knowledge that is literally right before their eyes. For example, a soldier standing on the upper end of the first ASR shown in figure 8 would be able to see into the intersection that is covered by the gray shading. However, according to Yeap and Jefferies (1999), the shaded regions would not be mapped and therefore would not be accessible unless the soldier took a step out of the current ASR toward the intersection.

To map areas outside the current local space,

we modified the ASR algorithm so that the areas outside the exits are not discarded. These areas are saved to form partial ASRs of the adjacent local spaces.

The basic idea is to not only compute an ASR of the current local space but, at the same time, to map the perceivable surroundings outside the local space. We call this set of surroundings outside the local space *residual ASRs* because they are built by trimming the residue off the current ASR. Residual ASRs are updated every perception cycle, and their composition relies completely on the successive visual perspectives of the viewer. Computing a residual ASR involves the following steps:

First, each perception cycle creates a 2-1/2D sketch of the area in the agent's field of view.[5] We refer to this sketch as the *current-view*.

Second, subtract the current ASR from the current-view. Call the remainder the residue. This computation involves two steps: First, for each currently visible exit in the ASR, identify the surfaces and gaps in the current-view that appear through this exit. Designate these surfaces and spaces as the *residue* for this exit. Once the residue for an exit has been identified, use it to compute an ASR; that is, identify the exits (doubtless and doubtful) and the surfaces using the same algorithm described previously. The result is the *current-residual ASR* for this exit.

Third, after each perception cycle, update the cumulative residual ASR for each of the exits. The current-residual ASR is only a snapshot. Its results are used to update the *cumulative-residual ASR*. The updating can involve adding new surfaces, changing exits from doubtful to doubtless, or reducing the size of doubtless exits where occlusions are uncovered.

With this extension to the basic ASR algorithm, a virtual human can map the perceivable areas outside the local space while it retains the spatial interpretation afforded by the ASR. What happens to these residual ASRs as the viewer travels from one local space to another? There are three cases we have considered:

First, as the viewer moves from one local space (ASR) to another, all the residual ASRs are saved and indexed by the location of the exit through which the residue was collected. An ASR can have multiple residual ASRs, one for each exit. When the viewer reenters an ASR, the residual ASRs become available again.

When a viewer goes through an exit into an area that was not visited previously, it will likely have a residual ASR that it computed for the space. At this point, the residual ASR is discarded, and an ASR is computed. In our future work, we will use the residual ASR as a starting point for computing a new ASR.

When the viewer looks through an exit into a local space that has already been visited, the viewer will recognize the space as having already being mapped, so it will not create a residual ASR. It recognizes the space by taking the coordinates of the exit and indexing them into the raw cognitive map, which contains all the exits and their locations.

This extension to Yeap's and Jefferies' theory and algorithms provides the viewer with the ability to map areas outside its local space. Figure 8 shows some residual ASRs shaded in gray. For example, on the right-hand side of figure 8, there is a residual ASR for the exit between the two buildings, looking out to the space beyond. In some cases, phantom edges were detected in part because of the occlusions in the environment. In figure 8, there is a slight mismatch between the lines of the cognitive map and the background urban image because of scaling and alignment differences in the two software packages used to produce the two images.

Applications of Cognitive Maps

Once a cognitive map of an area of the environment has been generated, the virtual human that generated the map can use it in a number of ways. In the mission rehearsal exercise (Swartout et al. 2001) mentioned earlier, many of the predicates used by the virtual human's planner involve spatial concepts. These predicates represent concepts such as individuals or groups occupying a specific region (medic-at-injury-site, crowd-in-landing-zone) and exits and entrances to a region being covered (landing-zone-secure, injury-site-secure). Currently, the status of these predicates is updated through the script that drives the exercise. However, we are currently updating how these predicates are calculated within the virtual human's perception and spatial reasoning. In the new approach, the virtual human will create a cognitive map that includes ASRs corresponding to regions such as the landing zone and injury site. Updating a predicate such as medic-at-injury-site will involve visually locating the medic and comparing the medic's location to the boundaries of the injury site ASR. Updating the landing-zone-secure predicate will usually involve visually inspecting each exit of the landing zone ASR to ensure that friendly soldiers are protecting the exits.

In addition to updating spatial predicates, a cognitive map can also be used to implement spatially oriented strategies. For example, a flanking maneuver might involve locating the

ASR the enemy is in and attacking through two of the ASR's exits simultaneously. Inherent in this strategy are the concepts of scouting, examining many ASRs to locate the enemy, and desirable defensive positions, ASRs that have a small number of exits. An ASR with a single exit might not be desirable because it leaves no escape route.

Cognitive maps will also be useful in communicating spatial information between agents. If both agents have similar cognitive maps, then once a common set of names for ASRs and exits has been negotiated, the agents can reference features of each other's cognitive maps. Furthermore, one agent can add to another agent's cognitive map (at an abstract level) by communicating spatial information about areas that the second agent hasn't seen. For example, a sergeant might report to his/her lieutenant that "we've located a suitable space for a landing zone. It's an open area through the west exit of this area. It has three lanes of approach which have been secured."

Related Work

Cognitive mapping research has been applied in the areas of mobile robotics, military simulations, and computer games. We briefly summarize the relationship of the research in these three areas to our own research (Hill, Han, and van Lent 2002).

Kuipers (1978) did some groundbreaking work in cognitive mapping. He recently proposed a spatial semantic hierarchy (Kuipers 2000) as a way of representing knowledge of large-scale space. The spatial semantic hierarchy is actually a set of distinct but related ways of describing space, including sensory, control, causal, topological, and metric representations. Kuipers and Remolina recently also developed a formal logic for causal and topological maps (Remolina and Kuipers 2001). Kuipers has tested his approach on simulated robots. There are numerous other researchers in mobile robotics who have also developed and implemented cognitive mapping techniques; for example, see Kortenkamp, Bonasso, and Murphy (1998) and Levitt and Lawton (1990). Chown, Kaplan, and Kortenkamp (1995) developed the PLAN system, which also uses viewer-based information to build a cognitive map. PLAN was implemented with a connectionist network with the purpose of integrating way finding with cognitive mapping. Although the research in mobile robotics has a lot in common with our domain, one of the chief differences is that many of their methods were developed to deal with noisy sensors and the difficulty of discerning one's location. Our emphasis is somewhat different in that we are trying to build agents with believable humanlike behaviors. The sensors are not noisy, but they do operate with limitations. The end use of our cognitive maps is also somewhat different in that we are not just concerned about way finding but also about spatial awareness for a wide variety of tasks that robots are not normally concerned about.

Computer game characters commonly have perceptual omniscience. Their perception is not modeled after human capabilities and limitations. To achieve humanlike behavior, the designers have to give the appearance of limited perception. Alternatively, their superhuman capabilities are either attributed to superior ability or cheating, which can be disheartening for human players. Spatial reasoning is frequently programmed into the environment rather than into the game's characters (Liden 2001). The game map consists of nodes linked together in a graph structure, which are then used as paths for the characters. For the characters to exhibit intelligent behavior, knowledge is encoded into the nodes and links about what behavior is appropriate at these locations. Thus, a node or link can have information saying that a location is good for an ambush or that the character should crawl when traversing this link to remain undercover. As we mentioned earlier in this article, the designers have to encode everything into the environment. Although this approach is efficient in terms of run-time computation, it does not address the issue of *generality*. It is a labor-intensive process that must be done for each new game environment. An alternative to real-time spatial reasoning is to automatically precompute and store information about the environment using the methods described here. This would avoid the problem of having to analyze and hand encode the spatial characteristics of the environment into the map representation. Laird (2001) is the one exception in the computer games world. He combines the use of simulated perception (to trace the walls) and mapping to support more sophisticated AI-based behaviors such as ambushing and trapping.

Military simulations generally require a lot of spatial reasoning. Research on virtual humans for military simulations has addressed the issue of spatial reasoning more broadly than in games. For example, Reece, Kraus, and Dumanoir (2000) have built on the work in path planning from AI and robotics. For areas outside buildings, they use A* search and represent the space with cell decomposition and graph planning (which is somewhat similar to what

games do.) However, the characters are not limited to the graph when moving through most environments. Forbus, Mahoney, and Dill (2001) are striving to apply qualitative spatial reasoning to both military simulations and strategy games. They are currently looking at ways to improve path planning to take into consideration trafficability, visibility, and fields of fire. They use a hybrid approach that combines the representations from cell decomposition and skeletonization. Until now, however, they have focused on analyzing terrain rather than urban environments.

Acknowledgments

This article was developed with funds from the Department of the Army under contract DAAD 19-99-D-0046. Any opinions, findings, and conclusions or recommendations expressed in this article are those of the authors and do not necessarily reflect the views of the Department of the Army.

Notes

1. Marr (1982) defines a 2-1/2D sketch to be a list of surfaces and their spatial layout. The sketch only includes the visible portions of the surfaces in the agent's field of view.

2. A raw cognitive map just contains information about the local environment without the addition of semantic interpretation (Yeap 1988; Yeap and Jefferies 1999).

3. For the details of the algorithm, see Yeap and Jefferies (1999).

4. These steps are based on the extend-ASR algorithm in Yeap and Jefferies (1999).

5. This sketch is the same 2-1/2D sketch that is used as input to the ASR-update algorithm.

References

Chown, E.; Kaplan, S.; and Kortenkamp, D. 1995. Prototypes, Location, and Associative Networks (PLAN): Toward a Unified Theory of Cognitive Maps. *Cognitive Science* 19(1): 1–51.

Forbus, K.; Mahoney, J.; and Dill, K. 2001. How Qualitative Spatial Reasoning Can Improve Strategy Game AIs. Paper presented at the 2001 AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment, 26–28 March, Stanford, California.

Hill, R.; Han, C.; and van Lent, M. 2002. Perceptually Driven Cognitive Mapping of Urban Environments. Paper presented at the First International Conference on Autonomous Agents and Multiagent Systems, 15–19 July, Bologna, Italy.

Kortenkamp, D.; Bonasso, R. P.; and Murphy, R., eds. 1998. *Artificial Intelligence and Mobile Robots*. Menlo Park, Calif.: AAAI Press.

Kuipers, B. 2000. The Spatial Semantic Hierarchy. *Artificial Intelligence* 119(1–2): 191–233.

Kuipers, B. 1978. Modeling Spatial Knowledge. *Cognitive Science* 2:129–153.

Laird, J. 2001. It Knows What You Are Going to Do: Adding Anticipation to a Quakebot. Paper presented at the Fifth International Conference on Autonomous Agents, 28 May–1 June, Montreal, Canada.

Laird, J., and van Lent, M. 2001. Human-Level AI's Killer Application: Interactive Computer Games. *AI Magazine* 22(2): 15–25.

Levitt, T., and Lawton, D. 1990. Qualitative Navigation for Mobile Robots. *Artificial Intelligence* 44(3): 305–361.

Liden, L. 2001. Using Nodes to Develop Strategies for Combat with Multiple Enemies. Paper presented at the 2001 AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment, 26–28 March, Stanford, California.

Marr, D. 1982. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. New York: W. H. Freeman.

Reece, D.; Kraus, M.; and Dumanoir, P. 2000. Tactical Movement Planning for Individual Combatants. Paper presented at the Ninth Conference on Computer-Generated Forces and Behavior Representation, 16–18 May, Orlando, Florida.

Remolina, E., and Kuipers, B. 2001. A Logical Account of Causal and Topological Maps. In Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01). Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Stanzione, T.; Evans, A.; Chamberlain, F.; Buettner, C.; Mabius, L.; Fisher, J.; Sousa, M.; and Lu, H. 1996. Multiple Elevation Structures in the Improved Computer-Generated Forces Terrain Database. Paper presented at the Sixth Computer-Generated Forces and Behavioral Representation Conference, 23–25 July, Orlando, Florida.

Swartout, W.; Hill, R.; Gratch, J.; Johnson, L.; Kyriakakis, C.; LaBore, C.; Lindheim, R.; Marsella, S.; Miraglia, D.; Moore, B.; Morie, J.; Rickel, J.; Thiebaux, M.; Tuch, L.; Whitney, R.; and Douglas, J. 2001. Toward the Holodeck: Integrating Graphics, Sound, Character, and Story. Paper presented at the Fifth International Conference on Autonomous Agents, 28 May–1 June, Montreal, Canada.

Terzopoulos, D., and Rabie, T. 1995. Animat Vision: Active Vision in Artificial Animals. Paper presented at the Fifth International Conference on Computer Vision (ICCV '95), 20–23 June, Cambridge, Massachusetts.

Yeap, W. K. 1988. Toward a Computational Theory of Cognitive Maps. *Artificial Intelligence* 34(3): 297–360.

Yeap, W. K., and Jefferies, M. E. 1999. Computing a Representation of the Local Environment. *Artificial Intelligence* 107(2): 265–301.

Randall W. Hill, Jr., is the deputy director of technology at the University of Southern California Institute for Creative Technologies. His research interests include modeling perceptual attention and cognitive mapping in virtual humans. He received a B.S. from the United States Military Academy at West Point in 1978, and an M.S. and a Ph.D. in computer science from USC in 1987 and 1993, respectively. His e-mail address is hill@ict.usc.edu

Changhee Han received his master's degree in computer science from Syracuse University in New York in 1994. He is currently a graduate research assistant at the Institute for Creative Technologies at the University of Southern California. His research interests include cognitive mapping, spatial reasoning, and movement planning. His e-mail address is changhee@ict.usc.edu

Michael van Lent is a research scientist at the University of Southern California Institute for Creative Technologies focusing on the application of AI research techniques to commercial computer games. He received his Ph.D. at the University of Michigan in 2000. He is currently working on after-action review tools for FULL SPECTRUM COMMAND, a training aid built for the Army using game industry development techniques. One of these tools is an explanation facility inspired by previous AI research efforts in explainable expert systems. His e-mail address is vanlent@usc.edu.

Call for Applications

Eighth AAAI / SIGART / IJCAI Doctoral Consortium

August 10–11, Acapulco, Mexico

Collocated with IJCAI–03

Sponsored by the American Association for Artificial Intelligence, SIGART, & IJCAII

AAAI, ACM/SIGART, and IJCAI invite students to apply for the Eighth AAAI/SIGART Doctoral Consortium. The Doctoral Consortium (DC) provides an opportunity for a group of Ph.D. students to discuss and explore their research interests and career objectives with a panel of established researchers in artificial intelligence.

The consortium has the following objectives:

- To provide a setting for mutual feedback on participants' current research and guidance on future research directions
- To develop a supportive community of scholars and a spirit of collaborative research
- To support a new generation of researchers with information and advice on academic, research, industrial, and non-traditional career paths
- To contribute to the conference goals through interaction with other researchers and participation in conference events.

The Doctoral Consortium will be held as a workshop on August 10-11, 2003, immediately before the start of the main conference. Student participants in the Doctoral Consortium will receive complimentary conference registration and a fixed allowance for travel/housing.

Important Dates for Application Submission

- February 7, 2003: Application Package Submission Deadline
- March 21, 2003: Acceptance Notification
- August 10-11, 2003: Doctoral Consortium

The Application Packet

Applicants to the Doctoral Consortium must submit a packet consisting of six copies of the following items. Hardcopy submissions are required; no electronic submissions will be accepted.

1. *Thesis Summary.* A two-page thesis summary that outlines the problem being addressed, the proposed plan for research, and a description of the progress to date. Please be sure to distinguish between work that has already been accomplished and work that remains to be done. Be sure to include a title for your work.
2. *Background Information.* Information (at most two pages) on your background and relevant experience. This should include information typically found in a curriculum vita, plus additional information that may indicate your potential contribution to the DC.
3. *Letter of Recommendation.* A letter of recommendation from your thesis advisor. It must include an assessment of the current status of your thesis research, and an expected date for thesis submission. In addition, your advisor should indicate what he or she hopes you would gain from participation in the DC.
4. *Participant's Expectations.* A short (one page or less) statement of what you expect to gain from presenting and participating in the DC, as well as what you think you can contribute to the DC.

Mail your submission packet to:
AAAI/SIGART Doctoral Consortium
445 Burgess Drive
Menlo Park, CA 94025-3442
Tel: 650-328-3123

Review Process

The consortium organizing committee will select participants on the basis of their anticipated contribution to the workshop goals. We solicit applications from any topic area and methodology within artificial intelligence. Students will be selected who have settled on their thesis direction, but still have significant research to complete. The perfect stage is having just had a research proposal accepted by the thesis committee. Students will be selected based on clarity and completeness of the submission packet, stage of research, advisor's letter, and evidence of promise such as published papers or technical reports.

At the Conference

The organizers invite all students to attend and participate in the Doctoral Consortium, whether or not they apply to present their work. In previous years, many non-presenting students said they found it useful to observe their peers' presentations and to participate in the ensuing discussions.

All participants selected to present their work at the Doctoral Consortium are expected to be present throughout the consortium. Our experience has been that participants gain almost as much by interacting with their peers as by having their presentations critiqued by the faculty panel. As such, we expect a commitment from participating students to attend the entire DC.

Acknowledgements

Support for the 2003 Doctoral Consortium is provided by ACM's SIGART, AAAI, and IJCAI.

Inquiries

Additional information may be obtained by contacting the chair of the organizing committee:

Marie desJardins
University of Maryland, Baltimore County
Department of Computer Science and Engineering
mariedj@cs.umbc.edu