Applying Perceptually Driven Cognitive Mapping To Virtual Urban Environments

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

USC Institute for Creative Technologies
13274 Fiji Way, Suite 600, Marina del Rey, CA 90292-7008
{hill, changhee, vanlent}@ict.usc.edu

Abstract

This paper 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 (Yeap & Jefferies 1999) for representing a local environment as a structure called an Absolute Space Representation (ASR). Their algorithms compute and update ASRs from a 2-1/2D [1] sketch of the local environment, and then connect the ASRs together to form a raw cognitive map. 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 describe a method for acquiring a 2-1/2D sketch in a virtual world, a step omitted from their framework, but which is essential for computing an ASR. Third, we extend 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.

Introduction

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 include coordinated tactical movements, crowd control, avoiding snipers and ambushes, selecting helicopter landing zones, and establishing 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, looking at buildings, 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 & Kortenkamp 1995; Kuipers 1978; 2000; Yeap 1988; Yeap & 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 & Kortenkamp 1995; Kortenkamp & Bonasso & Murphy 1998; Kuipers 1978; 2000; Levitt & Lawton 1990).

This paper describes a method for building a cognitive map of a synthetic urban setting based on the realistic limits of human visual perception. Humans have a limited field of view and cannot see through solid objects like 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 (Yeap & Jefferies 1999) that represents a local environment as a structure called an Absolute Space Representation (ASR). Building an ASR involves perceiving the local surroundings, 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

[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.

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 via their exits, a viewer builds a raw cognitive map[2]. We have taken this framework and extended it in a number of ways:

- 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 human-like behavior. Yeap tested his theory with a simulated robot in a 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 and Jefferies' cognitive mapping algorithms (Yeap & Jefferies 1999). Instead of limiting the agent to only building one ASR at a time, focusing 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 physically explored. Virtual humans need to build cognitive maps in anticipation of the next space they will enter.

Figure 1: View of a street in a virtual urban environment

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 helicopter. 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 those spatial concepts.

Many current applications finesse perception and spatial reasoning as much as possible. Computer games (Liden 2001) and military simulations (Reece & Kraus & 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 (Terzopoulos & 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 that sensing (no 360 degree field of view, no seeing 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 that 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. While the underlying ASR representation may 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 their 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 be potentially 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 since they don't require a realistic model of perception. However, for their behavior

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

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 Absolute Space Representation (ASR) algorithm developed by Yeap and Jefferies (Yeap & Jefferies 1999), our virtual human maps the local environment by continuously perceiving a scene, constructing a 2-1/2 D 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.

The basic idea behind Yeap's theory of cognitive maps (Yeap 1988) 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. But 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 the viewer is not sure of. It may 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 may actually be areas one needs to explore more fully, thus guiding the mapping process.

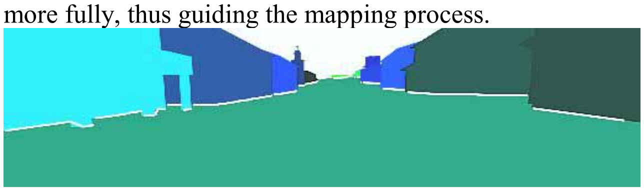

Figure 2: Detecting the edges in the urban scene

Constructing a 2-1/2D sketch

Yeap and Jefferies' cognitive mapping algorithm takes as input a 2-1/2D sketch of the scene (Marr 1982; Yeap & 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 vertices), as they appear from left to right in the field of view. But 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 recognizes objects in an effort to determine that the obstacles are indeed buildings or other real things. Much progress has been made in this area (e.g., see Kortenkamp & Bonasso & Murphy 1998 on mobile robotics), but it still remains a significant challenge. One of the contributions in this paper is an approach to building a 2-1/2D sketch in graphically rendered virtual environments.

We took a hybrid approach to building the 2-1/2D sketch that combines the use of the graphical 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, we simplify the process by using this aspect of the model to tell us that these are buildings. But this only takes us part of the way toward building a 2-1/2D sketch. To do this, we take the following steps:

1. Traverse the scene graph and assign a unique number to each node corresponding to a building. This is done by taking advantage of the node pre-draw callback function in the graphics routines. The advantage of this is that each of the buildings, which are fairly simple boxes underneath the texture maps, will be assigned a unique number, which will be used later for edge detection.
2. 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 be able to create a full 3D model.
3. Draw each node with its assigned number (color). The result of this step can be seen in Figure 2, where the buildings appear as different colors, corresponding to the unique numbers that were assigned.
4. Find the edges between the ground and the buildings 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 / # 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 buildings and the ground. Pixelation may 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 vertices in a format that can be used for Yeap's ASR algorithm, which we will describe in the next section[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 & 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 (Yeap & 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 backmost surface. An exit is the shortest span between the occluding vertex and another surface. It is the gap that must be crossed in order to reach the occluded edge.

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 vertices. Since the exit is the gap that must be crossed in order to reach an occluded edge, identifying the exit starts with the occluded edge. The ordered list of visible surfaces is divided into two groups around the occluded edge, which is done by placing one of the occluded edge's vertices in one group and the other into the second group. One of the occluded edge's vertices is unoccluded. Choose this vertex and look for the closest point on a surface contained in the opposite group. The exit is the edge formed by the unoccluded 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 vertices. The surfaces beyond the exit are trimmed from the ASR. They are no longer taken into consideration for mapping local space since 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 of one's local space. This will be discussed in more detail in the section on mapping outside the local space.

[3] For the details of the algorithm see Yeap and Jeffires.

Updating the local map

Since 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.

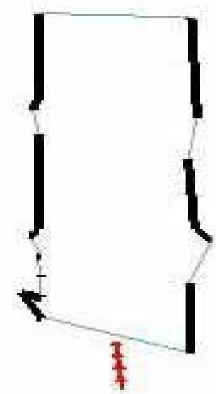

Figure 3: This is a top-down perspective of an ASR of the street scene from Figure 1. The building boundaries are shown as dark lines and the doubtless exits as thin lines. The viewer's path is shown with arrows. There are no doubtful exits in this ASR.

Yeap and Jefferies (Yeap & Jefferies 1999) distinguish between two kinds of exits: *doubtful* and *doubtless*. A *doubtless* exit is one that takes the viewer out of the local space. It consists of two unoccluded vertices—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 not longer needs to be updated.

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

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

1. Sense the environment and construct a 2-1/2D sketch. Call this perspective CURRENT-VIEW.

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

2. Check whether the viewer is still inside the current ASR. This 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.
3. If an exit has NOT been crossed, update the doubtful exits based on the CURRENT-VIEW. If the change in perspective uncovers an occlusion, this will cause the size of the corresponding doubtless exit to decrease.
   For each doubtful exit:
   a. Label the two surfaces that contribute vertices to the doubtful exit as S1 and S2.
   b. If CURRENT-VIEW includes S1 and S2, then replace the doubtful exit with the surfaces that lie between S1 and S2. Note: We found that we had to relax this condition somewhat because there are cases where the vertices of the doubtful exit are outside of the field of view of the agent.
4. Else, if an exit has been crossed, this means that 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 comprise the raw cognitive map. Whenever the viewer crosses an exit and enters a previously unexplored area, a new ASR is computed. Figure 4 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 4. Note that the third ASR contains doubtful exits on the left and right sides of the viewer. This indicates that the area has not yet been completely mapped. Once the viewer's perspective has been rotated, these areas will be filled in with surfaces and doubtless exits. Figure 5 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 re-mapped every time they were visited. The recognition routine is triggered when the viewer crosses an exit.

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

1. The newly entered space was previously mapped and the exit is a known connector between the two ASRs. When this 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.
2. 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.
3. 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.

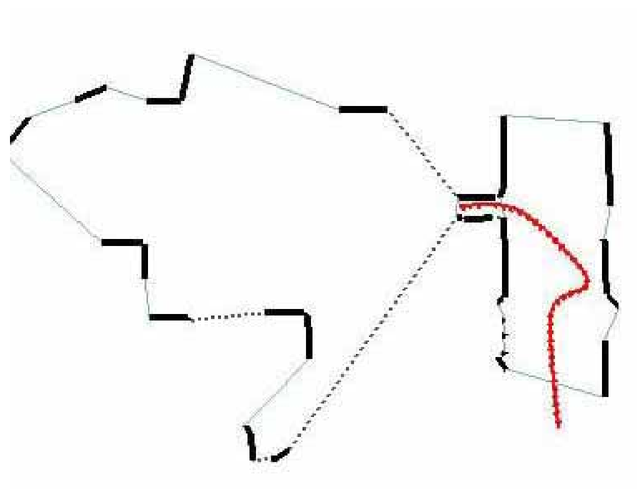

Figure 4: Three ASRs are shown connected together. The third ASR contains both doubtless (thin lines) and doubtful (dotted lines) exits.

Mapping Outside the Local Space

We developed an extension to Yeap 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 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, may provide important information that will impact the behavior of the viewer even before that 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 helicopter to land. A quick glance down an alley or street may reveal that there is no open space in the immediately adjacent spaces, but further down the street there is a major intersection where it may be possible for a helicopter 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 physically visited, 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 5 would be able to see into the intersection that is covered by the gray shading. But according to Yeap & Jefferies 1999 this 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 of the current local space, we modified the ASR algorithm so that the areas outside the exits are not discarded. These are saved to form partial ASRs of the adjacent local spaces.

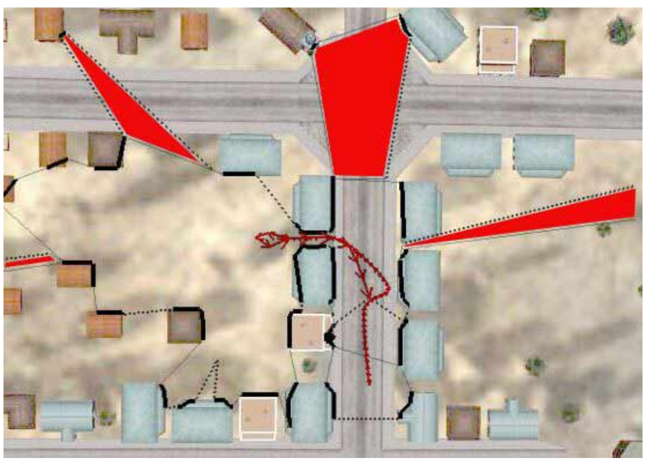

Figure 5: A raw cognitive map, including residual-ASRs (shaded regions) constructed from the residue of local computations.

The basic idea is to not only compute an ASR of the current local space, but at the same time also map the perceivable surroundings outside the local space. We call this set of surroundings outside the local space *residual*-ASRs since they are built by trimming the residue off of 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:

1. Each perception cycle create 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.
2. Subtract the current ASR from the CURRENT-VIEW. Call the remainder the *residue*. This computation involves two steps:
   a. For each currently visible exit in the ASR, identify the surfaces and gaps in the CURRENT-VIEW that appear through that exit. Designate these surfaces and spaces as the *residue* for that exit.
   b. Once the residue for an exit has been identified, use it to compute an ASR, i.e., identify the exits (doubtless and doubtful) and the surfaces using the same algorithm described previously. The result is the *current-residual-ASR* for that exit.
3. 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 may 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 of the local space while retaining the spatial interpretation afforded by the ASR. But what happens to these residual ASRs as the viewer travels from one local space to another? There are three cases we have considered:

1. As the viewer moves from one local space (ASR) to another, all of the residual-ASRs are saved and indexed by the location of the exit through which the residue was collected. An ASR may have multiple residual-ASRs, one for each exit. When the viewer re-enters an ASR, the residual-ASRs become available again.
2. When a viewer goes through an exit into an area that was not previously visited, it will likely have a residual-ASR that it computed for that 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.
3. When the viewer looks through an exit into a local space that has already been visited, then 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 into the raw cognitive map, which contains all the exits and their locations.

This extension to Yeap and Jefferies' theory and algorithms provides the viewer with the ability to map areas outside of its local space. Figure 5 shows three residual-ASRs shaded in gray. For example, on the right hand side of the diagram 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 due in part to the occlusions in the environment.

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

Applications Of Cognitive Maps

Once a cognitive map of an area of the environment has been generated, the virtual human who generated that map can use it in a number of ways. In the Mission Rehearsal Exercise (Swartout 2001) mentioned in Section 2, 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/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 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 that 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 may not be desirable, as 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 lieutenant "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 & van Lent 2002).

Kuipers (Kuipers 1978) did some of 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 metrical representations. He and Remolina recently also developed a formal logic for causal and topological maps (Remolina & 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, e.g., see (Kortenkamp & Bonasso & Murphy 1998; Levitt & Lawton 1990). Chown et al. (Chown & Kaplan & 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 wayfinding with cognitive mapping. While 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 human-like 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 wayfinding 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 human-like behavior the designers have to give the appearance of limited perception. Alternatively their superhuman capabilities are either attributed to superior ability or to 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 into 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 those locations. So a node or link may 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 paper, the designers have to encode everything into the environment. While this is efficient in terms of runtime 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 pre-compute 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 (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 et al. (Reece & Kraus & Dumanoir 2000) have built on the work in path planning from AI and robotics. For areas outside of buildings they use A* search and represent the space with cell decomposition, and graph planning (which is somewhat similar to what games do.) But the characters are not limited to the graph when moving through most environments. Forbus et al. (Forbus & Mahoney & Dill 2001) are striving to apply qualitative spatial reasoning to both military simulations and to 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. Up until now, however, they have focused on analyzing terrain rather than urban environments.

Acknowledgments

The work described in this paper was supported in part by the U.S. Army Research Office under contract #DAAD19-99-C-0046. The content of this article does not necessarily reflect the position or the policy of the US Government.

References

Chown, E., Kaplan, S., and Kortencamp, D. 1995. Prototypes, location, and associative networks (PLAN): Towards a unified theory of cognitive maps. *Cognitive Science* 19:1-51.

Forbus, K., Mahoney, J., and Dill, K. 2001. How qualitative spatial reasoning can improve strategy game AIs. In *Proceedings of the 2001 AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment*: AAAI Press.

Hill, R., Han, C., and van Lent, M. 2002. Perceptually driven cognitive mapping of urban environments. In *Proceedings of the First International Conference on Autonomous Agents and Multiagent Systems*.

Kortenkamp, D., Bonasso, R.P., and Murphy, R., (editors). 1998. *Artificial Intelligence and Mobile Robots*. Menlo Park, CA: AAAI Press.

Kuipers, B. 1978. Modeling spatial knowledge. *Cognitive Science* 2:129-153.

Kuipers, B. 2000. The spatial semantic hierarchy. *Artificial Intelligence* 119:191-233.

Laird, J. 2001. It knows what you are going to do: Adding anticipation to a Quakebot. In *Proceedings of the Fifth International Conference on Autonomous Agents*. Montreal, Canada, May 28-June 1.

Laird, J. and van Lent, M. 2001. Human Level AI's killer application: Interactive computer games. *AI Magazine*, Volume 22, Issue 2, Summer.

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. In *Proceedings of the 2001 AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment*: AAAI Press.

Marr, D. 1982 *Vision*: A computational investigation into the human representation and processing of visual information: W.H. Freeman and Company.

Reece, D., Kraus, M., and Dumanoir, P. 2000. Tactical movement planning for individual combatants. In *Proceedings of the 9th Conference on Computer Generated Forces and Behavior Representation*. Orlando, FL.

Remolina, E. and Kuipers, B. 2001. A logical account of causal and topological maps. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI-01).* Seattle, WA.

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. In *Proceedings of the 6th Computer Generated Forces and Behavioral Representation Conference*. University of Central 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. In *Proceedings of the Fifth International Conference on Autonomous Agents*. Montreal, Canada, May 28-June 1.

Terzopoulos, D. and Rabie, T. 1995. Animat Vision: Active Vision in Artificial Animals. In *Proceedings of the Fifth International Conference on Computer Vision (ICCV '95*). 801-808. Cambridge, MA, June.

Yeap, W.K. 1988. Towards a computational theory of cognitive maps. *Artificial Intelligence* 34:297-360.

Yeap, W.K. and Jefferies, M.E. 1999. Computing a representation of the local environment. *Artificial Intelligence* 107:265-301.