Modeling Synthetic Opponents in MOUT Training Simulations Using the ACT-R Cognitive Architecture

Bradley J. Best
Christian Lebiere
K. Christine Scarpinatto
Human-Computer Interaction Institute
Carnegie Mellon University
bjbest@cmu.edu, cl@cmu.edu, scarp@cs.cmu.edu

Keywords:
ACT-R, MOUT, Cognitive Architecture, Virtual Reality, Unreal

Abstract: *The Office of Naval Research (ONR) is sponsoring a project to improve the cognitive validity of synthetic opponents in Computer Generated Forces. These synthetic soldiers will be used to improve the effectiveness of training simulations for Military Operations on Urban Terrain (MOUT). We are working towards this goal by developing human performance models using the ACT-R cognitive architecture. This paper presents the first model resulting from that effort. The current constructive simulation involves two teams of two synthetic soldiers: two attackers and two defenders. The attackers are attempting to clear a building in an urban setting of hostile forces while the defenders are attempting to prevent this action. These cognitive models engage in combat in the context of a video game, Unreal, a cost-effective development environment. The attackers employ US doctrinal strategies while the opponents use a range of strategies reflecting the broad diversity of behavior likely to be used by opposing forces. The cognitive models communicate to plan coordinated actions, act in concert, and exchange information about their status to maintain synchronous teamwork. The resulting realistic planning and interactions the models engage in demonstrate the utility of the ACT-R cognitive architecture for modeling synthetic soldiers.*

1. Using a Cognitive Architecture to Simulate Opponents in Virtual Environments

The Virtual Technologies and Environments (VIRTE) program aims to develop and demonstrate breakthrough human-immersive technology for naval training. The goal is to train warriors for increasing complexity and chaos by supplementing and complementing live simulations using virtual and wargaming simulations and other emerging technologies. The approach is to incorporate current understanding of human behavior and learning theories into systems and leverage commercially available advanced technology. In this paper, we introduce our effort to improve the cognitive validity of synthetic soldier entities in simulations such as the Dismounted Infantry Semi Automated Forces (DISAF) [9] by developing cognitive models of the opponents using the ACT-R cognitive architecture. [1]

The synthetic entities in DISAF and other CGF systems often suffer from a number of shortcomings in representing the kind of behavior that actual opponents would display. At the perceptual and motor level, their behavior is usually implemented using ad hoc models rather than models that implement principled limitations of the human perceptual and motor system. Because perceptual, motor and cognitive capacities are implemented in separate subsystems with limited communications between them, little or no integration between their operations is achieved which results in a serious lack of situation awareness. Synthetic entities are all too often purely reactive, lacking in goal-directed capabilities to guide and structure their behavior. Memory for past events is either lacking entirely or too perfect, without the gradual characteristics of human memory. Fatigue, stress, suppression effects and other behavior moderators are not taken into account in modulating the performance of the synthetic entities. Finally, complex but brittle AI mechanisms for inference, planning, reasoning and decision-making are often used that provide superhuman capabilities in some instances and catastrophic failures in others, without any of the limitations and resourcefulness of human cognition. [8]

We follow an empirically driven, principled approach to the development of cognitive models based on the ACT-R architecture of cognition. ACT-R is a hybrid cognitive architecture that consists of a symbolic production system tightly coupled to a subsymbolic neural-network-like layer. The production system can represent symbolic task knowledge and provides structured goal-directed behavior in the form of sequential steps of execution on the scale of tens to hundreds of milliseconds. The subsymbolic level controls the applicability of each piece of symbolic knowledge through a massively parallel network of parameters that are tuned optimally to the structure of the

cognitive environment by Bayesian learning mechanisms to capture the basic adaptivity, robustness and stochasticity of human cognition in dealing with a world of uncertain, noisy and partial information. Both parts of the architecture are essential to capturing the full range of human cognition and their tight integration is key to leveraging the strength of each system. ACT-R also integrates a perceptual-motor layer that consists of independent vision, motor, speech and audition modules that can operate in parallel, providing a principled interface between cognition and the environment.

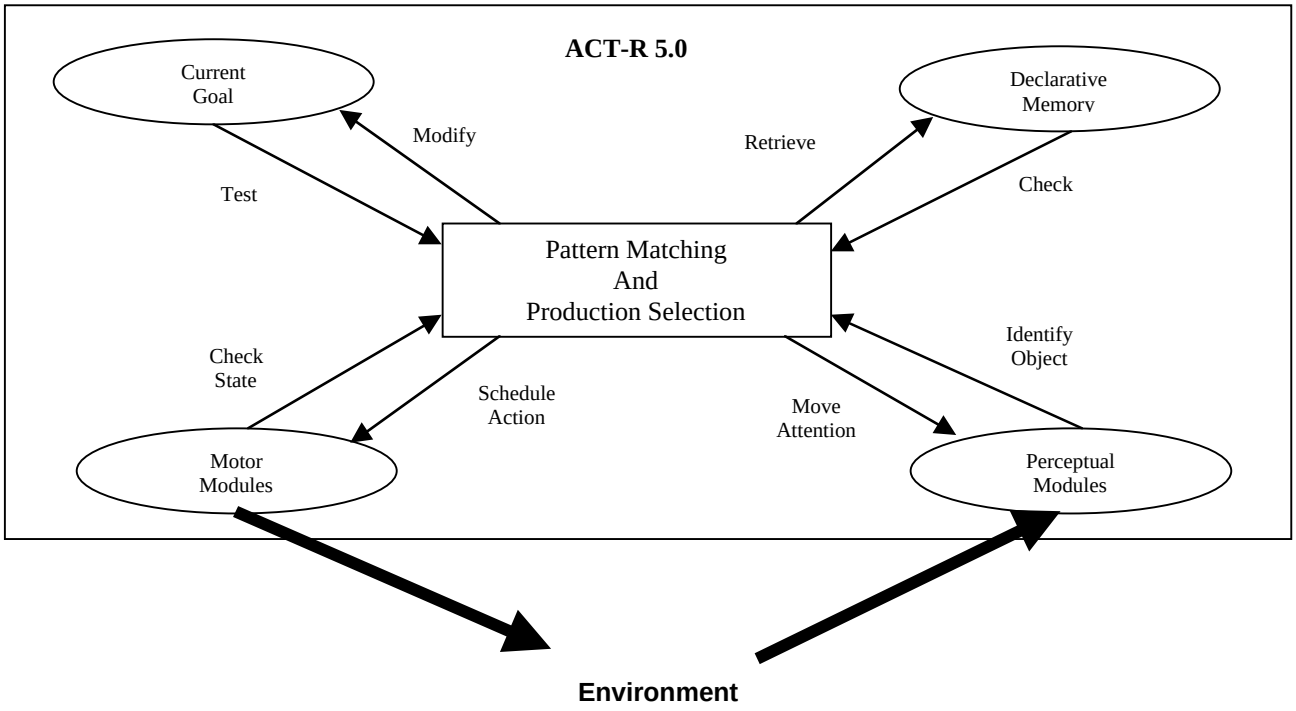

Figure 1.1: ACT-R 5.0 Cognitive Architecture

The design of the ACT-R architecture is based on a synthesis of the current state of experimental knowledge in cognitive psychology and human factors. ACT-R is a high-fidelity architecture that makes detailed predictions about every observable aspect of human behavior, including latency, errors and eye movements. It can account for the whole range of human cognition from 10msec latency effects in PRP experiments to high-level decision-making, control of complex systems, and learning spanning a scale of years, allowing for the integration of fine-grained experimental knowledge of human cognition into large-scale computational models of real-world applications.

Dozens of ACT-R models have been developed and empirically validated by an active user community in academic, industrial and military research laboratories. ACT-R has been applied to model a number of tasks of direct relevance to this project. These tasks include complex military-type simulations such as air traffic control, anti air warfare coordination and submarine command and control. The ACT-R models of these tasks provide a detailed fit to the behavior of human subjects interacting with a complex simulation environment at a precise time scale down to the level of eye movements. Those models demonstrate that ACT-R possesses the expressive and computational power to provide a detailed reproduction of human behavior in complex, challenging environments such as military operations. In turn, those domains provide a challenging test bed likely to lead to new developments in the architecture. [5]

There are specific ways in which the ACT-R architecture can contribute to the design of better synthetic entities for MOUT training simulations. Because of its inclusion of constrained perceptual and motor modules, ACT-R provides a principled interface between the agent and its environment. While the current ACT-R modules are mostly designed for experimental settings, their basic concepts such as focus of attention generalize to other settings. While the various perception and action capabilities are located in separated modules, ACT-R provides an integrated framework to tie them together with cognition in a comprehensive model of situation awareness. A production can match to the current state of the perceptual modules as well as the goal and the content of declarative memory. An integrated picture of the situation can then be elaborated in the current goal that

ties together the various perceptual modalities together with an abstract representation of the currently observed objects. This capacity of the ACT-R architecture to build an integrated representation of the situation provides a natural solution to the difficulty encountered in existing systems to integrate together information from different sensors that are handled by separate models.

The current goal is a central concept in ACT-R, which naturally provides strong support for goal-directed behavior. However, the most recent version of the architecture (ACT-R 5.0) is less goal-focused than its predecessors by allowing productions to match to any source of information, including the current goal, information retrieved from declarative memory, objects in the focus of attention of the perceptual modules and the state of the action modules. This emphasis on asynchronous pattern matching of a wide variety of information sources better enables ACT-R to operate and react efficiently in a dynamic fast-changing world. Thus ACT-R is capable of goal-directed behavior but in a flexible manner which gives equal weight to internal and external sources of information.

There are three principal distinctions in the ACT-R architecture. First, there are the two types of knowledge structures – chunks for representing declarative knowledge and productions for representing procedural knowledge. Second, there is the symbolic level, which contains this information, and the sub-symbolic level of neural activation processes that determine the speed and success of access to chunks in declarative memory. Finally, there is a distinction between the performance processes by which the symbolic and sub-symbolic layers map onto behavior and the learning processes by which these layers change with experience.

Human cognition in a domain like MOUT has two principal components. The first is the knowledge and procedures codified in military doctrine and instilled in military forces through extensive training [6]. The second is the natural cognitive abilities of soldiers that manifest themselves in tasks as diverse as memory, reasoning, planning and learning. The fundamental advantage of an integrated architecture like ACT-R is that it provides a framework for both types of knowledge to complement and leverage each other.

The advantage of a symbolic system like ACT-R's production system is that, unlike connectionist systems for example, it can readily represent and apply symbolic knowledge of the type specified by military doctrine. Moreover, the specific type of knowledge encoded in that doctrine are rules specifying what to do in a given condition, a type of knowledge particularly well-suited for representation as production rules. Symbolic knowledge however does not define performance. If it did, humans would behave more like robots, acting rigidly, deterministically, with neither errors nor inspiration. In ACT-R, performance is defined by the parameters at the sub-symbolic level that determine the availability and applicability of symbolic knowledge. Those parameters underlie ACT-R's theory of memory, providing effects such as decay, priming and strengthening. But they also play a broader and more general role: they make cognition adaptive, stochastic and approximate, capable of generalization to new situations and robustness in the face of uncertainty. Those qualities provide ACT-R models with capacities of inference, planning, reasoning, learning and decision-making that are both powerful and general without the computational complexity and specialization of standard AI techniques. Finally, because of the continuous nature of the sub-symbolic layer, architectural parameters can be varied to incorporate the effect of behavior moderators to obtain a gradual degradation of behavior in the face of fatigue, stress, injury and other conditions.

2. Requirements for Military Operations on Urban Terrain (MOUT) and Close Quarter Battle (CQB)

Expanding military roles to encompass missions like peacekeeping and the continuing trend towards population concentration in cities has made Military Operations on Urban Terrain (MOUT) more common. Compared to rural battlefields of the past, MOUT places a different set of demands on the soldier (For an in-depth reference to this topic, see [6]).

A MOUT environment is distinguished from the terrain of rural battlefields by the dominant features of densely packed manmade structures and multiple avenues of approach. The urban battlefield can be divided into four basic levels: 1) building level, 2) street level, 3) subterranean level, and 4) air level.

A typical MOUT mission involves initial strategic planning in relation to gathered intelligence. The plan is then executed in relation to doctrinal MOUT tactics. These tactics have been developed through the analysis of historical urban conflict and extrapolation to the capabilities of modern soldiers. MOUT combat tactics are focused around small infantry units and prescribe methods for clearing, at increasing levels of detail, blocks of buildings, individual buildings, floors in buildings, and individual rooms and hallways. MOUT operations and MOUT doctrine also typically require close coordination of armor units with infantry.

Important aspects of terrain in MOUT environments include fields of view, the closely related fields of fire (which depend on the available weapons and the field of view), available cover and concealment, obstacles to navigation, available lighting, and avenues of approach and escape. Weather is also significant in MOUT terrains and may impact visibility (especially when darkness is included), combat force effectiveness, vehicle operation, citizen and opponent behavior, and weapon effectiveness.

Close quarter fighting in and around buildings makes command and control extremely difficult. The main approach to this problem is to systematically clear zones in the battlefield, sector by sector, with certain units assigned to particular zones. The focus for this document is the implementation of doctrinal tactics at the level of the individual infantry soldier.

The individual soldier receives extensive training in MOUT doctrine. This training includes weapons handling and firing techniques, movement techniques, building entry techniques, and clearing techniques.
Weapons handling and firing techniques are intended to give an advantage over less rigorously trained opponents. By consistently shouldering a weapon and by carrying it in a way that makes it easier to shoulder, a soldier can substantially increase his weapon accuracy. Movement techniques specify how to move in an urban setting while reducing the exposure to enemy fire. Open areas between buildings are crossed along the shortest possible path. Movement inside building hallways is done along the walls instead of down the center of the hallway.

Although not always possible, the preferred doctrinal method of entering and clearing a building is from the upper levels. Gravity assists with the placement of grenades while a floor plan typically favors the combatant above (some subject matter experts disagree with the doctrinal approach because it exposes the lower body of soldiers as they descend stairwells). If the only possible entry is from the ground floor, the preferred method is to breach a wall instead of entering through a doorway or window that can be booby-trapped or effectively covered by defensive fire. If a breach cannot be made, an existing entrance is used after supporting fire and/or grenades are directed to that location. The use of large quantities of grenades is common in close quarter battles.

Clearing techniques specify which teammates will direct fire where, and how to arrange units prior to room entry. In a doctrinal room entrance, a pair of soldiers first assumes a “stacked” position along the wall outside the doorway. The first soldier directs their weapon towards the far corner while the second soldier steps around and behind them and tosses a grenade into the room. The use of a grenade is signaled to other assault team members nonverbally if possible, but otherwise verbally. After grenade detonation, the first shooter steps through the doorway (one step away from the wall, two steps in) and clears their immediate area using weapon fire if necessary. The second shooter (who was stacked behind) steps through the doorway, buttonhooks, and clears their section of the room. Both shooters start from the outside corners and rotate towards the center wall, eventually converging after eliminating any threats.

Normally a second two-person team will provide covering fire and security in the hallway behind the first team. The clearing team and covering team also communicate with a series of doctrinal statements, such as “Clear”, “Coming out”, etc. There are many variations on these methods for clearing a room. If a door is closed, soldiers may assume split positions with one clearing soldier initially stationed on each side of the door. With larger teams and open doors, the stacking method is used, but instead of immediately buttonhooking, the soldiers spread out along the inside wall.

Clearing hallways is similarly well specified. To clear an L-shaped hallway, a team of two soldiers will each take one wall of the initial portion of the hall. The soldier on the far wall will advance to just before the intersection while the soldier on the near wall parallels their movement. The soldiers then, on a signal, move together into the hallway, one crouching and the other standing, clearing all targets.

The eventual goal of this project is realistic behavior from synthetic opposing forces. Clearly specifying the doctrinal strategies for friendly forces allows testing a range of opposing force (OPFOR) behaviors against the expected doctrinal strategy. Unlike friendly force behaviors, OPFOR behavior is not well specified and ranges from coordinated, planned attacks by well-trained forces who carefully aim their weapons to disorganized sporadic attacks from enemies using the “pray and spray” weapon discharge technique.

3. Simulation Platforms

A major trend in modeling and simulation is the developing synergy between the entertainment and defense industries as shown in [12]. While at first glance it appears that these industries have fundamentally different goals (selling successful entertainment products and improving troop performance, respectively), they have in fact been solving similar problems for years because fundamentally both share the same constraint of developing a product that provides the best possible immersiveness, ease of use, realism and challenging situations. Because time to market is a crucial determinant of success in the entertainment world, the

commercial game industry is characterized by a breakneck pace of technological development. The defense industry, with its emphasis on scientific research, cannot match this pace. For example, the visionary Land Warrior system researched and developed for military use was available on the shelf to the general public as a commercial game product before it became available to troops in the field.

Thus using a commercial game product in the original development phase then later developing the final model for actual use in MOUT training simulations might be the optimal strategy. Commercial game products are cheap to acquire, typically costing as little as $50, or sometimes even free. They run on standard PC hardware, sometimes requiring a special plug-in card for speed and graphics enhancement, and do not require expensive graphics workstations often used in defense simulations. They are relatively free of bugs, having been used by a large number of users. They have a well-documented Applications Programmer Interface (API), since the manufacturer wants to encourage further development by the user community. Moreover, most games now come in networked as well as individual versions, and computer-generated characters are becoming an increasingly popular feature to populate multi-player games, so the infrastructure is already present for connecting synthetic players (bots) to the game and have them interact with human players. Finally, they provide a fairly realistic immersive environment, since it is the primary dimension of popular success in the commercial marketplace.

3.1 Unreal Tournament (UT) as a Platform

One instance of a game that displays all these advantages is Unreal Tournament (UT). UT is a (mostly) open source game with a thriving online community that emphasizes networked play and synthetic participants. We have connected a multi-model version of ACT-R to UT and have obtained very good responsiveness between human players and synthetic agents. A single dedicated server can handle at least 16 players or agents, making it more than suitable for the interaction between a MOUT squad and a small group of opponents. UT provides high-level abstractions for actions, allowing us to bypass that problem until additional accuracy of movement is required. For perception, UT provides fast regular updates of the environment (typically every 100 milliseconds but that can be tuned for performance) as well as asynchronous updates of special events. The simulation provides updates of the exact global coordinates of the various objects or artifacts in the current field of view (FOV), which are transformed into a quantitative representation. [14]

After the initial development phase, the model can then be transferred to another platform for integration into MOUT training. Although this requires a transition between simulation platforms, this two-phase development has a number of advantages. It requires a development in terms of progressively more accurate layers of abstractions, a beneficial practice that is not always followed when the developer can rely on the idiosyncrasies of a single platform. It requires a re-examination of the model after the initial development phase, again a beneficial practice that is not always followed in the rush of development that tends to cement in place early but poor design decisions. Finally, it allows efficient development of a suitable interface between the model and the platform after those requirements have been determined from the prototyping phase (rather than as those requirements are evolving).

One concern is that learning acquired through human interaction with the model in the desktop game environment be preserved when moving to a virtual reality training environment. Because of the immersive nature of the game environment and the relatively high level of skills targeted by the training simulation, there is reason to be optimistic about that transfer.

3.2 Unreal Tournament (UT) Platform Details

A UT server communicates with clients through a message passing protocol. Messages from UT are of two types---synchronous and asynchronous. Synchronous updates occur roughly every 100 milliseconds. They are sent as a batch, and no game time elapses between the first and last message of a batch. Synchronous messages provide information about the state of the agent, the state of the game (such as scores), and nearby objects such as weapons that can be picked up, among other things. Asynchronous messages provide information about such things as the collision of an agent with a wall, the amount of damage an agent receives as a result of being shot, etc.

UT provides direct information about the location of items of interest and other players that are visible in the game environment. The messages from UT unfortunately do not give information about specific topographical features of a map, such as walls and doors. To overcome this lack of information available to synthetic players during game play, the creators of UT have compensated by providing “navigation points.” Navigation points are an invisible (to human players) set of interconnected nodes that synthetic agents may use in order to move around a map. These points may also have information about how they can be used. For example “ambush point” is a subclass of “navigation point.” This allows simple synthetic UT opponents to appear to move around in the game in a realistic way, but only if the level

designer has included enough of the appropriate type of navigation points. Using this approach, changing the behavior of synthetic agents is accomplished by changing the location, quantity, and type of navigation points spread around the map. [16]

The use of navigation points gives agents valuable information, but it has the disadvantage of making their behavior too predictable. Human players, after playing a map repeatedly, can deduce the location and purpose of navigation points and use this information to their advantage. Reliance on navigation points requires significant human intervention to tune a map for appropriate seeming synthetic agent behavior. Even moving between navigation points can sometimes be problematic. It is possible to query the server for a path between two navigation points, but if the points are too far apart, or if the map compiler failed to recognize a possible path, a query may come up empty even though a path exists. Map builders often find themselves manipulating the network of paths by hand in order to improve performance of synthetic agents.

Given that we need to move agents around a map in accordance with high-level goals, such as locating and clearing a room of enemy fighters, reliance on this system of navigation points is unacceptable. To address this we have designed and implemented an "exploration agent." The purpose of an exploration agent is to build up a representation of a particular map that does not depend on navigation points, but instead on the actual topological properties of the map (walls, rooms, halls, doors). The exploration agent uses the navigation points to move around the map and survey what can be seen. From each point, it calculates coordinates for nearby points and queries the server as to whether the points are reachable or not. Once a sufficiently large number of points have been gathered, we use the unreachable points to infer the existence of walls and other obstructions. Exploration cannot currently be done in real time. The exploration is done in advance; the data gathered is analyzed, and then stored for later use.

3.3 UT Support for MOUT

The UT environment maps nicely onto the MOUT environment. UT provides direct support for the first three levels of an urban battlefield – buildings, the areas around them, and the areas beneath them. The use of the air level, and in particular aircraft for troop deployment, reconnaissance, and ground strikes is not directly supported. In addition, support for armor units is not provided in the UT environment and therefore is outside the scope of this report.

UT provides a range of built-in weapons such as rifles, grenades, rocket launchers, handguns, and machine guns that have different effects when used on opponents and different accuracies, ranges, and fields of effect. In addition, custom weapons can be created to closely simulate the effects of known weapon types. Each weapon uses a specific type of ammunition that the player must keep track of (and potentially find more of to prevent running out). Weapons can be picked up, thrown down, and sometimes even used in different modes (e.g., shooting from the hip with the handgun is fast but inaccurate). In addition, UT allows for a wide range of player motion and action. Players may crouch, jump, pivot, sidestep, run, and look up or down, and even feign death. A standing player can see and shoot over a crouching player. Players can also communicate via text messages that can be sent to a specific player, all players on the same team, or all players.

UT can also handicap a player appropriately to simulate a player's skill level with a weapon. By assigning accuracy to a particular agent, UT neatly abstracts the effect of repeated weapons training (or lack thereof). Thus, well-trained opponents who properly shoulder their weapons and aim them are simply more accurate. In UT, this can be expressed by giving the entire friendly force a relative accuracy advantage, or by specifying a secondary weapons technique that is less accurate (e.g., shooting from the hip).

Messages can be sent between agents in UT allowing team coordination. Messages in the form of text may be freely transferred from one player to another, from a player to all players on the same team, or from a player to all players in the game. This allows easy simulation of radio communication within a team or spoken communication between players. In addition, messages can be broadcast as audio messages in which case the human players will hear the message sent.

Unreal Tournament handles multi-user games by means of a client-server architecture. The client is not a dumb terminal that simply follows the instructions of the server, but contains the same code as the server. The map of a level in UT contains all of the static data about that level. Both client and server have the complete set of information about the map in which the agent is playing. The server sends messages to the client regarding things that change over time, such as objects that can be picked up and other players. The server only sends information about players and objects in the immediate environment of the agent. The client has the ability to project the behavior of entities in the game slightly into the future so that it does not always have to wait for responses from the server in order to act appropriately. The purpose of this architecture is to simultaneously accommodate users who

may have very different network connection speeds and hardware. [16]

Creating a synthetic agent is relatively straightforward. One need only open a socket to the UT server and create a process that catches and handles the messages that the server sends. An individual socket is created for each agent. Any messages received or sent on a socket affect only the agent for which it was created. A dedicated server can handle up to 16 agents.

UT includes a level editor for constructing custom levels. A UT level consists of a set of intersecting volumes with intersecting surfaces. Surfaces may be given different textures and there is extensive control over lighting and appearance. As a level is created, volumes can be added and subtracted to produce the exact kind of space desired. This makes it possible to model, for instance, the inside of a building. It is straightforward to also model a larger space consisting of a set of buildings enclosed in an open volumetric space. Within spaces, furniture such as tables, chairs, and desks can be freely added to areas. Elevators, doors, and other moving objects are supported as well. This enables the creation of a realistic MOUT environment that can include textures such as sky textures, water that the player can be submerged in, etc.

Figure 3.1: View of Agents Storming a Room in UT

4. Extracting Cognitive Primitives

For an agent to navigate and act within a space it must have a representation of the environment that supports these actions. Although simply reacting to other nearby agents can simulate rudimentary behavior, any complex planning and teamwork requires a more complete representation of space. This representation can be constructed from basic elements available within the particular virtual environment, in this case UT.

The representation we have used is generated from a process that can be divided into two parts: a low-level implementation-dependent feature extraction process, and a method for translating this to a model-level representation usable by the agent. Note that the abstract representation is implementation-independent. Implementations on other platforms would focus on extracting low-level primitives available in that environment and mapping them onto the model-level representation.

The low-level spatial primitives available in UT are fairly sparse. Our challenge was to use information available within the game to automatically build up a cognitively plausible representation of space that could be used across platforms. An agent in UT can request information about whether a particular point is reachable from its current location, which means that there is a clear navigable straight path between where the agent is standing and the target location. The next section describes how, using this as the only spatial primitive, the exploration agent builds up a usable model-level representation of the space.

4.1 Sampling the space

As described above, an agent in UT can send messages back and forth to the UT engine. One of the messages that can be sent from an agent is a request for information on whether a particular point in UT space (using a three-dimensional x,y,z coordinate system) is reachable in a straight line from the current location of the agent. Locations inside walls are, by definition, not reachable. Locations around corners are also not reachable though they may be at some later point in time. Reachability is relative to the current location.

This mechanism can be used to determine the boundaries of walls. Given a current location, it is possible to extend a ray out from this point and at various points along the ray, query the UT engine. Eventually as we travel out on a ray from the current location, since a UT level is a finite space that is bounded by unreachable borders, a point will be far enough away that it is unreachable. The transition from reachable to unreachable along that ray defines a boundary between open space and some solid object (e.g., a wall).

Once two points along the ray are determined to have different reachability values, the fidelity of this estimate can be improved by sampling the midpoint between these two points. This allows a binary search across this interval and gives more precision in the estimate of the distance to the wall, though this process still has some noise in it. Determining if a ray intersects a surface is a hard problem in computer graphics and computer vision.

Rays projecting from the exploration agent that pass near corners of doorways give somewhat unpredictable results as can be seen from the figure below. This is a limitation of the UT engine but does not appear to be a significant limiting factor.

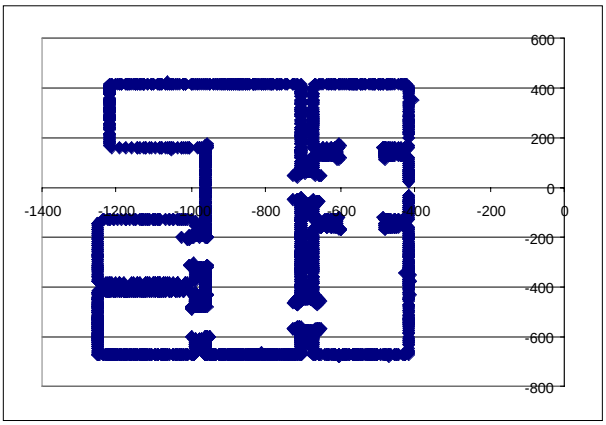

Figure 4.1: Detected Obstruction Points

For a particular location, an agent can perform this range sensing in any direction (this is analogous to laser range sensing used in robotics). By standing in one place and rotating, an agent can determine the distance to the outer edges of the space it is in. If an agent also moves to other parts of the space, it is possible to perform this range sensing and sample all of the available spaces in a UT level.

This range sensing can be done in two or three dimensions. The model reported here uses three-dimensional sampling although it is not necessary for this map. This is allows for the detection of upward sloping ramps, stairs, etc.

4.2 Hough transform to extract walls and doors

Given a set of points that fall along walls on a map, determining which point falls on which wall and how to group them is not a trivial problem. One solution to this problem is a method known as the Hough transform [15]. The equation of a line can be represented in the following parametric form:

$$R = X\cos\theta + Y\sin\theta$$

In this form, R represents the distance from the origin to the line along the normal, and theta represents the angle from the origin of the normal to the line.

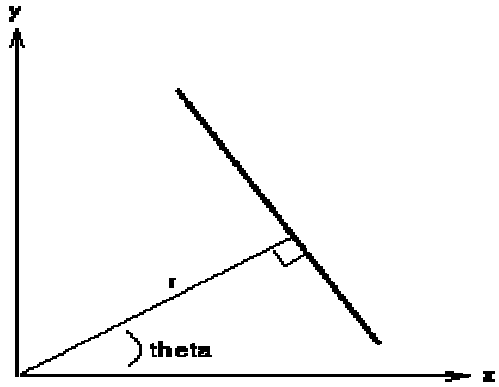

Figure 4.2: Parametric Form of a Line

With a large number of points, it is possible to search the parameter space for line equations that many points could potentially fall on. Using this technique, an accumulator array with bins along the R and theta dimensions is constructed. A particular bin corresponds to a range of values (e.g., the theta bins could be 10 degrees each, giving 36 bins spanning the 360 possible degrees). Given a point with x and y coordinates, iterating across the theta bins allows substituting these three values (x, y and theta) in the equation above and solving for R. The bins for the R parameter have to extend to large enough values to include all lines in the current space (in a finite space the range of the R parameter is bounded by the maximum distance from the origin of any point – the bins then divide that range).

Each individual point will provide several values of R as theta is iterated across. Given a particular x, y, and theta, the resulting R gives an index into the accumulator array that is incremented to indicate a solution for this point. These parameters (R and theta) represent the lines crossing through that point (the use of bins forces this to be a small, discrete number). If two points in space are processed in this way, each will individually vote for several values of R and theta, but the only value of R and theta that receives votes from both points will be the line they both fall on.

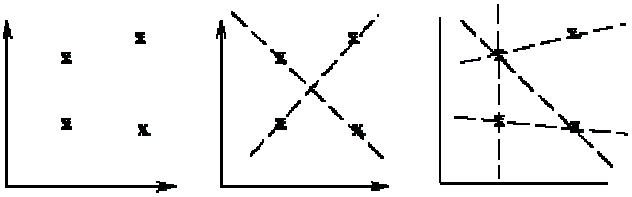

Figure 4.3a-c: Individual Points Voting for Lines

Continuing this process with each new point, the accumulator array will be incremented the most at locations corresponding to line equations that cross through many of the points. Applying a threshold to ignore accumulator array cells with only a few votes results in a set of cells that correspond to lines that

intersect large numbers of the input points. We have chosen to use this method in two dimensions though it is straightforwardly extensible to three dimensions (by parameterizing the equation for a plane instead of a line).

4.3 Data structures: line segments, bounding volume, and binary space partitioning tree

At this point in building the spatial representation, there are a set of points that fall on particular lines in the two-dimensional representation of the UT environment, and a set of equations for the likely lines given those points. Combining the points with the equations for the lines gives a set of segments bounded by endpoints. If all of the points along a line are ordered (by sorting), then a line segment is defined by the presence of points in some density. In this case, any gap greater than the size of the agent's collision cylinder is taken to be a real gap (this assumption relies on thorough sampling of the level).

The resulting data structures are a set of wall segments and openings (doors) aligned with those wall segments. A wall segment is defined by a set of endpoints and a normal to it. Searching for the walls that bound a location and determining which walls are visible from a particular location can be aided by a data structure called a binary space partitioning tree (BSP tree).

A BSP tree represents space hierarchically. In the two-dimensional case, each node in the tree subdivides the space recursively with a splitting plane. The root node in the BSP tree divides the whole plane with a line. Given the normal to that line, every point on the plane is either in front of, behind, or on the line. This node has two children: one child further subdivides the front half-space of the plane while the other child subdivides the back half-space of the plane. This recursive subdivision continues until some completion criterion is met.

Using the line segments as splitters, the space in a UT map can be divided up with a BSP tree. An algorithm for constructing a tree does this automatically from a set of splitters. At each iteration, a splitting line is chosen (from among the wall segments) that splits the fewest other wall segments in the space. Objects that are intersected by a splitter must be split into a front half and a back half.

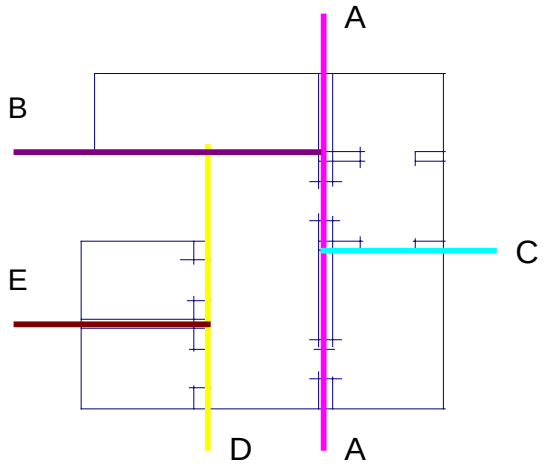

Figure 4.4: First 5 Splitting Planes in BSP Tree

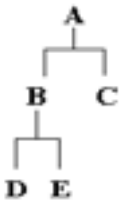

Figure 4.5: BSP Tree of First 5 Splitting Planes

4.4 ACT-R Representation

Given a particular location of an agent, the most important question is "What are the features of the space around me?" The use of a BSP tree allows rapid determination of a bounding volume for the current map. That is, the walls of a room that an agent is in can be quickly determined by searching down the tree for the node in the BSP tree corresponding to the room and maintaining a list of the walls used as splitters. The majority of the nodes in the tree will not need to be considered. This allows for a search in O(logN) time, where N is the total number of walls (for more complex operations such as ray-tracing the expected complexity is still only O(N)).

As an agent moves around within a map, a set of the walls and openings of the current bounding volume are pushed into a buffer for use by the production system. This buffer is constantly refreshed and represents the up-to-date spatial layout of the room or area the agent is in. This representation allows agents to navigate a space relative to doorways and openings in rooms as well as along walls – both requirements of MOUT tactics.

This representation is egocentric---it is relative to the viewer's current position. Allocentric representations, which are relative to some external reference, (e.g., the UT map origin, the equator and prime meridian, etc.) are an alternative way of representing space [4]. Because of the simple topology of the map used in this model, and the implied strategies of the agents, an allocentric representation of space is not needed. For scenarios involving mission planning, multiple floors, or unusual room layouts, an allocentric representation must be built out of the egocentric representation.

5. Sample Scenario

We support teams of synthetic agents using a multi-model version of ACT-R. Multi-model ACT-R allows for an arbitrary number of independent models to exist simultaneously in the same application. By means of a function call, one can switch between different models in the environment. Each synthetic agent has an identifier associated with it, allowing the appropriate model to be accessed. As messages come from the server over the sockets, our message-handling code takes the identifier of an agent, switches to the appropriate model, and then updates that model's knowledge base and cycles the production system as appropriate. Productions can contain function calls that send messages over the socket to the agent, enabling the production system to directly affect the behavior of an agent. Communications between agents and switches of control all happen within the same application, which increases efficiency.

In the interest of creating robust models, we have created models of both friendly forces and opposing forces. This allows simulated combat between both sides with no human intervention. Players are then free to enter the simulation as a member of either force. In addition, specifying doctrinal behavior for the attacking force allows us to test the behavior of the opposing force against the expected form of attack. This has the added benefit of allowing exploration of the stochasticity of behavior across multiple runs of the same scenario.

5.1 Basic Scenario: Storming a Floor in a Building

The vignette we work with here is a pair of soldiers starting at the end of an L-shaped hallway whose mission is to clear the floor of opposing forces. The friendly forces employ doctrinal tactics and first clear the hallway itself using the covering movements described above. The cleared hallway presents the soldiers with several doorways. The soldiers then stack themselves at the doorways, enter the room (also described above), and clear any inner rooms discovered.

Opposing forces return fire if they are cornered or run and escape if they can (while firing some poorly aimed shots). These forces are very reactive compared to the friendly forces. Planning is implicit in the hiding spots and defensive positions they initially assumed – their goal is to defend the building they are in. As they spot defenders, they hastily fire a shot or two while falling back. When cornered, they dig in and fight to the death (this is one of many possible scenarios – see [6] and [11] for more).

5.2 Sample ACT-R Models for Basic Scenario

Below are English abstractions of some of the productions used in ACT-R models for the attacking force and the opposing force:

Opposing Force Sample Productions:

1. If there is an enemy in sight and there is no escape route then shoot at the enemy
2. If there is an enemy in sight and there is an escape route then set a goal to escape along that route.
3. If there is a goal to escape along a route and there is an enemy in sight then shoot at the enemy and withdraw along the route

Attacking Force Productions (a space is a room or hallway):

1. If there is a goal to clear a building and there is an entrance to the current space that has not been cleared and it is closer than any other entrance to the current space that has not been cleared then set a goal to clear the adjoining space through that entrance
2. If there is a goal to clear a space and an enemy is in sight then shoot at the enemy
3. If there is a goal to clear a space and I am the lead shooter then take up position on the near side of the entrance to that space
4. If there is a goal to clear a space and I am the second shooter then get behind the lead shooter
5. If there is a goal to clear a space and I am the lead shooter and I am positioned at the entrance and the second shooter is positioned behind me then signal to the second shooter to move, step into the entrance and clear the area to a the left
6. If there is a goal to clear a space and I am the lead shooter and I am positioned at the entrance and the second shooter is positioned behind me then signal to the second shooter to move, step into the entrance and clear the area to a the right
7. If there is a goal to clear a space and the lead shooter has signaled to enter the space then step into the

entrance and clear the area to the opposite side of the lead shooter

8. If there is a goal to clear a space and I am the lead shooter and there is no enemy in sight then pan towards the opposite corner
9. If there is a goal to clear a space and I am the lead shooter and I am have panned to the second shooter and there are no enemies in sight then signal to the second shooter that the space is clear and note that the space is cleared
10. If there is a goal to clear a space and the current space is cleared and there is no other entrance to the space that has not been cleared then remove the goal to clear the space and return to the adjoining space through the entrance

Note that productions 5 and 6 differ only by which way they specify to move. This allows for variability of behavior – either of these two productions can match the conditions for entering a room.

6. Conclusion

This paper lays out the foundation for our work on synthetic agents that implement MOUT tactics in Unreal Tournament, a virtual reality game-playing environment. This work establishes the feasibility of UT as a platform for MOUT training and simulation and the ACT-R cognitive architecture as a platform for developing synthetic forces. These two pieces are independent of each other. By abstracting perceptual cognitive primitives instead of depending on some facility provided directly by UT, the behavior of the models are specified in a way that does not depend on UT in any way.

The basic vignette described above deals with combat at a tactical level within one floor on a building. Two soldiers are approaching several rooms with the intent to clear any hostile forces encountered. The next step in this work is to expand the scope of the modeling effort to include strategic decisions involving more units clearing multiple levels of buildings and multiple buildings. The addition of extra units that need to be coordinated in a comprehensive plan will require expanding the role of communication and teamwork between units as well.

ACT-R allows the construction of models whose behavior range from highly reactive to highly structured. In the set of models presented, the defending force is much more reactive than the attacking force. This demonstrates the range of reactivity ACT-R allows in modeling both forces. In addition to variable reactivity at the level of strategy, ACT-R also allows for stochastic choice of actions to pursue from the actions defined within a strategy. If two actions could both be taken, a model will tend to pick the more likely action, but it is not a certainty. This allows for flexible behavior even when running the same vignette repeatedly.

The current models use an egocentric representation of the space around the agents. This is sufficient for the scope of this model but larger scale operations will almost surely require a broader spatial representation to use in reference to planning and strategy. In the models presented here the strategy planning is already complete: the attacking force in inside the entry point for the building and that determines the direction to start clearing the building. An allocentric representation would be necessary to consider which side of a building to enter from. Defending forces in particular are likely to have a well-established allocentric spatial representation they can use to navigate and plan. Friendly forces, on the other hand, use intelligence information to take the place of memorized layouts. The time cost of looking information up on a map, or some other memory aid, in contrast to acting fluidly due to knowing it from memory is something ACT-R is particularly well suited to model.

Allocentric representations can be built up using the learning features of the ACT-R architecture and the existing egocentric representation. In this case an allocentric representation is composed of a set of linked egocentric representations that can be reasoned about. In addition to building a representation, it is also possible for models to learn which behaviors are more effective and then adapt by preferentially choosing those behaviors. In other words, the architecture directly supports the effect of training.

The models presented here provide a demonstration of the efficacy of the ACT-R cognitive architecture and the Unreal Tournament platform for simulating MOUT operations. The perceptual primitives developed lead to a platform independent egocentric spatial representation. This egocentric representation (viewer centered) will be used in turn to develop an allocentric representation (e.g., map centered) for use with future models in exploration of more complex attacking and defending strategies.

Acknowledgements

This work was funded by grant N00014-02-1-0020 from the Office of Naval Research. We would like to thank Scott Douglass and Dan Bothell for their code interfacing ACT-R with Unreal Tournament and Jon Fincham for insightful discussions.

References

[1] Anderson, J. R., and Lebiere, Christian: *The Atomic Components of Thought*, Erlbaum, Mahwah, NJ 1998
[2] Gillis, P. D.: *Cognitive Behaviors for Computer Generated Command Entities*. U.S. Army Research Institute Technical Report, 2000
[3] Hicinbothom, J. H.: Maintaining situation awareness in synthetic team members. In *Proceedings of the 10th Conference on Computer Generated Forces and Behavior Representation*. Norfolk, VA, 2001
[4] Klatzky, Roberta L.: Allocentric and Egocentric Spatial Representations: Definitions, Distinctions, and Interconnection, *Spatial Cognition* pp. 1-18, 1998
[5] Lebiere, C., Anderson, J. R., & Bothell, D.: Multi-tasking and cognitive workload in an ACT-R model of a simplified air traffic control task. In *Proceedings of the 10th Conference on Computer Generated Forces and Behavior Representation*. Norfolk, VA, 2001
[6] Marine Corps Warfighting Publication (MCWP) 3-35.3, *Military Operations on Urbanized Terrain (MOUT)*
[7] Pew, R. W. & Mavor, A. S.: *Modeling Human and Organizational Behavior: Application to Military Simulations*. Washington, DC: National Academy Press, 1998
[8] Reece, D. (2001). *Notes on Cognitive Limitations of DISAF*. SAIC Technical Report
[9] Reece, D., Ourston, D., Kraus, M., & Carbia, I.: Updating ModSAF for individual combatants: The DISAF program. In *Proceedings of the 7th Conference on Computer Generated Forces and Behavior Representation*. University of Central Florida, 1998
[10] Silverman, B. G., Might, R., Dubois, R., Shin, H., Johns, M., & Weaver, R.: Toward a human behavior models anthology for synthetic agent development. In *Proceedings of the 10th Conference on Computer Generated Forces and Behavior Representation*. Norfolk, VA, 2001
[11] Weaver, R., Silverman, B. G., & Shin, H.: Modeling and simulating terrorist decision-making: A performance moderator function approach to generating virtual opponents. In *Proceedings of the 10th Conference on Computer Generated Forces and Behavior Representation*. Norfolk, Va.. 2001
[12] Zida, M., Capps, M., and McDowell, P.: A future for entertainment—defense research collaboration, *Digital Media*, pp. 2-8, January/February 2001

Links

[13] Unreal Tournament Gamebots Site: http://www.planetunreal.com/gamebots/index.html
[14] Johnson, Martin: *The Hough Transform* (Lecture 11), http://cs-alb-pc3.massey.ac.nz/notes/59318/l11.html
[15] Polge, Steven: *Unreal Tournament AI and Gameplay for Level Designers*, http://unreal.epicgames.com/, 1999
[16] Sweeney, Tim: *Unreal Networking Architecture*, http://unreal.epicgames.com/, 1999

Author Biographies

BRAD BEST is completing his Ph.D. in Psychology at Carnegie Mellon University where his graduate career has focused on simulations of human problem solving in spatial tasks. He received his B.S. in Computer Science from the University of Detroit and his M.S. in Computer Science from Central Michigan University. In his graduate work there he concentrated on connectionist models of visual systems. In addition to academic pursuits, he has spent several years in commercial positions applying rule-based artificial intelligence to real-world problems. His main research interest is the development of artificial intelligence planning systems with spatial components and their application to complex tasks including navigation, game playing, and strategizing.

CHRISTIAN LEBIERE is a Research Scientist in the Human-Computer Interaction Institute at Carnegie Mellon University. He received his B.S. in Computer Science from the University of Liege (Belgium) and his M.S. and Ph.D. from the School of Computer Science at Carnegie Mellon University. During his graduate career, he worked on the development of connectionist models, including the Cascade-Correlation neural network learning algorithm that has been used in hundreds of scientific, technical and commercial applications. Since 1990, he has worked on the development of the ACT-R hybrid cognitive architecture and is co-author with John R. Anderson of the 1998 book "The Atomic Components of Thought". His main research interest is cognitive architectures and their applications to psychology, artificial intelligence, human-computer interaction, decision-making, game theory, and computer-generated forces.

CHRISTINE SCARPINATTO is a research programmer in the Human-Computer Interaction Institute at Carnegie Mellon University. She received her BSLA in Japanese from Georgetown University and her MS in Computational Linguistics from Carnegie Mellon University. Her graduate work focused on intelligent tutoring software for students of foreign languages. She spent four years developing intelligent mathematics tutors for children ages 12-16 for the PACT Center at Carnegie Mellon. She has practiced aikido, a Japanese martial art, for 9 years.