8

Cognitive Agents Interacting in Real and Virtual Worlds

Bradley J Best and Christian Lebiere

1 INTRODUCTION

This chapter describes agents, based on the ACT-R cognitive architecture, which operate in real robotic and virtual synthetic domains The virtual and robotic task domains discussed here share nearly identical challenges from the agent modeling perspective Most importantly, these domains involve agents that interact with humans and each other in real-time in a three-dimensional space This chapter describes a unified approach to developing ACT-R agents for these environments that takes advantage of the synergies presented by these environments

In both domains, agents must be able to perceive the space they move through (i e., architecture, terrain, obstacles, objects, vehicles, etc) In some cases the information available from perception is raw sensor data, whereas in other cases it is at a much higher level of abstraction Similarly, in both domains actions can be specified and implemented at a very low level (e g , through the movement of individual actuators or simulated limbs) or at a much higher level of abstraction (e g , moving to a particular location, which depends on other low-level actions)

Controlling programs for both robots and synthetic agents must operate on some representation of the external environment that is created through the processing of sensory input Thus, the internal robotic representation of the external world is in effect a simulated virtual environment Many of the problems in robotics then hinge on being able to create a sufficiently rich and abstract internal representation of the world from sensor data that captures the essential nuances necessary to perceive properly (e g , perceiving a rock rather than a thousand individual pixels from a camera sensor bitmap) and a sufficiently abstract representation of actions to allow it to act properly

Robotic and virtual platforms must deal with the vision problem, either by bypassing it (e g , through the use of radio beacons to mark paths,

structured data describing architecture, and volumetric solids), or by solving relevant problems in vision (producing a depth map from stereo cameras, segmenting images, identifying objects, etc) Virtual synthetic domains may make bypassing some of the issues in vision straightforward but this is not a given – some virtual environments may simply present an agent with raw sensor data such as a bitmap In either case, the problem is the same: producing a representation of the environment from raw sensor data that the agent can use to reason with Although this representation will have its own problems (uncertainty, incomplete information, nonmontonic changes, etc) and should not be viewed as an idealized version of the underlying reality, it is nonetheless essential in insulating higher-level processes from the details of lower-level processes and providing a layered way for complex cognitive agents to interact with a complex world, reflecting the earlier insights of Marr (1982) in the nature of visual information processing

Actions the agent can take in the environment range from domain-general actions such as locomotion to domain-specific actions such as weapon loading, and span levels of abstraction from very low-level actions such as changing wheel actuator velocities or changing a virtual pose to higher-level actions such as movement from point to point Domain-general high-level actions such as locomotion are typically abstracted in both environments such that a simple API with high-level commands will produce equivalent movements in both a virtual environment and on a robotic platform.

In the cases of both perception and action, though the low-level implementation of an action or the processing of a sensory input will be different in the two domains, the high-level specification may remain the same These parallels between real robotic and synthetic virtual domains encouraged the development of a common platform allowing the same agents to be developed and deployed in either robotic or virtual domains This will in turn facilitate the development of increasingly large and complex teams of agents to populate both real world entities and virtual avatars

2 ACT-R

ACT-R is a unified architecture of cognition developed over the last 30 years at Carnegie Mellon University At a fine-grained scale it has accounted for hundreds of phenomena from the cognitive psychology and human factors literature The most recent version, ACT-R 5 0, is a modular architecture composed of interacting modules for declarative memory, perceptual systems such as vision and audition modules, and motor systems such as manual and speech modules, all synchronized through a central production system (see Figure 8 1) This modular view of cognition is a reflection

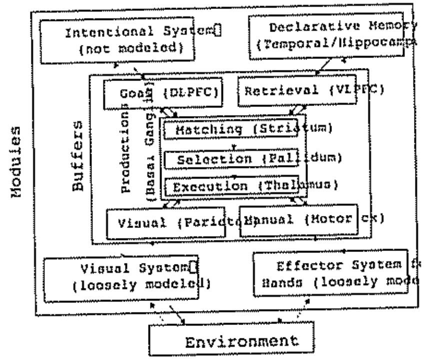

FIGURE 8.1 ACT-R architecture

both of functional constraints and of recent advances in neuroscience concerning the localization of brain functions ACT-R is also a hybrid system that combines a tractable symbolic level that enables the easy specification of complex cognitive functions, with a subsymbolic level that tunes itself to the statistical structure of the environment to provide the graded characteristics of cognition such as adaptivity, robustness, and stochasticity

The central part of the architecture is the production module A production can match the contents of any combination of buffers, including the goal, which holds the current context and intentions, the retrieval buffer, which holds the most recent chunk retrieved from declarative memory, visual and auditory buffers, which hold the current sensory information, and the manual buffer, which holds the current state of the motor module (e g walking, firing, etc) The highest-rated matching production is selected to effect a change in one or more buffers, which in turn trigger an action in the corresponding module(s) This can be an external action (e g , movement) or an internal action (e g , requesting information from memory) Retrieval from memory is initiated by a production specifying a pattern for matching in declarative memory Each chunk competes for retrieval, with the most active chunk selected and returned in the retrieval buffer. The activation of a chunk is a function of its past frequency and recency of use, the degree to which it matches the requested pattern, plus stochastic noise Those factors confer memory retrievals, and behavior in general, desirable "soft" properties such as adaptivity to changing circumstances, generalization to similar situations, and variability (Anderson & Lebiere, 1998)

The current goal is a central concept in ACT-R, which as a result 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 through flexible goal-directed behavior that gives equal weight to internal and external sources of information

There are three main distinctions in the ACT-R architecture First, there is the procedural-declarative distinction that specifies two types of knowledge structures – chunks for representing declarative knowledge and productions for representing procedural knowledge Second, there is the symbolic level, which contains the declarative and procedural knowledge, and the sub-symbolic level of neural activation processes that determine the speed and success of access to chunks and productions. 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 can be characterized as having two principal components: (1) the knowledge and procedures codified through specific training within the domain, and (2) the natural cognitive abilities 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 modeling basic human cognition and integrating it with specific domain knowledge

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 domain experts (e.g., rules specifying what to do in a given condition, a type of knowledge particularly well-suited for representation as production rules) In ACT-R, performance described by symbolic knowledge is mediated by 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, which 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 (e g , Sanner, Anderson, Lebiere, & Lovett, 2000)

3 USING A COGNITIVE ARCHITECTURE TO CREATE AGENTS FOR VIRTUAL AND ROBOTIC ENVIRONMENTS

One major goal of this work was to provide training opponents for Military Operations in Urban Terrain (MOUT) scenarios rendered in a virtual environment The state of the art in both commercial gaming packages and virtual training systems is the use of finite state machines for behavioral control Finite state machines provide simplicity of development, but at the cost of producing brittle behavior, combinatorial explosions of potential state transitions as the number of states increase, and low levels of realism and variability. Teamwork among synthetic opponents is often either lacking or completely absent Anecdotally, human players often learn to game the finite state machine and take advantage of the idiosyncrasies of the opponents

Rather than basing behavior on finite state machines, we have chosen to use the ACT-R architecture as the basis for cognitive agents with the intent of maximizing realism, adaptivity, unpredictability, and teamwork. These properties are a natural aspect of human performance in many task environments, and as such are also an inherent aspect of the ACT-R architecture, making it a good match for creating agents to play the role of opponents in the MOUT domain in particular, and for creating agents that simulate human behavior in general

ACT-R also provides a platform for simulating the way humans represent space and navigate about it (e g., Schunn & Harrison, 2001) Many of the pitfalls of robotic performance in the field involve behavior that would never be conceived of by a human in the same situation Recognition of this has inspired the creation of robotic agents that simulate a human in the same situation as the robot with a goal of producing robust robot behaviors Selecting a representation of the environment that is psychologically plausible enables portability by leveraging the flexibility of the human cognitive and perceptual systems: people can effortlessly switch from navigating their own bodies in space to controlling virtual entities in a computer simulation to remotely teleoperating robotic platforms in real-world environments An agent endowed with a reasonable facsimile of the spatial and cognitive abilities of humans ought to be able to as well, requiring changes only in the low-level layers that provide information to and act upon the orders of that agent

4 SIMULATION PLATFORMS

A major trend in modeling and simulation is the use of gaming platforms for use in research Using a gaming platform to provide a virtual environment, however, provides many of the same opportunities and challenges

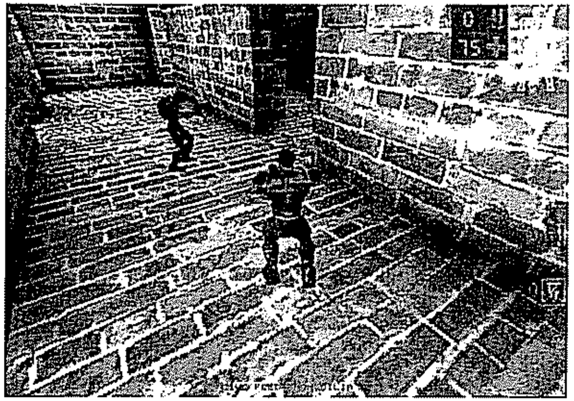

FIGURE 8.2 Two agents prepare to enter room

as working on a robotic platform The parallels and differences between these two types of platforms are discussed below

4.1 Unreal Tournament (UT) as a Platform for MOUT

The UT environment (see Figure 8 2) can easily be used to construct a basic urban battlefield – buildings, the areas around them, and the areas beneath them UT supports a range of built-in weapons including those meant to simulate military weapons (such as rifles, grenades, rocket launchers, handguns, and machine guns), as well as others that are non-violent (such as bubble wands, nets, etc.), either directly as part of the game or as part of freely available "mods"

UT allows for a wide range of player motion and action Weapons can be picked up, thrown down, and often used in different modes (e g , shooting from the hip with the handgun is fast but inaccurate) Players may crouch, jump, pivot, sidestep, run, swim, look up or down, and even feign death 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 simulation of radio communication within a team or spoken communication between players

Unreal Tournament handles multi-user games by means of a client-server architecture, allowing multiple agents running on separate

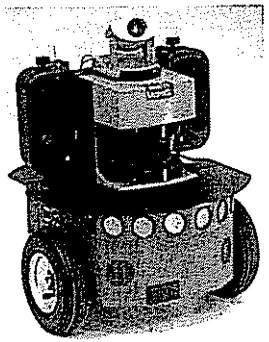

FIGURE 8.3 ActivMedia Pioneer P3-DX robot

machines to interact via the virtual environment The client provides static information about the map in which the agent is playing (i e . architecture) whereas the server sends messages to the client regarding dynamic elements, such as objects that can be picked up and other players that are in the immediate environment of the agent.

Creating a synthetic agent involves opening a TCP/IP socket to the UT server and creating a process that catches and handles the messages that the server sends Any messages received or sent on a socket affect only the agent for which it was created. This interface allows for the isolation of agents from each other. forcing interaction to take place through the game itself (e g . through text messaging between agents). providing an equal footing for both humans and agents

4 2 ActivMedia Robotics as a Platform

The ActivMedia robotics platform is a platform for mobile robotics that consists of controlling software and a range of physical robots including all-terrain robots, high-payload robots, human-interaction robots, team robots, and the robot used in this project: the Pioneer P3-DX. a general-purpose robot (see Figure 8 3)

Available perceptual inputs (sensors) for the P3-DX include sonar. ladar, contact bumpers, video cameras, range-finding infrared. stereo cameras, and compasses and gyros (as well as various actuator sensors) Available

action outputs (actuators) include the wheel motors, a speaker. pan-tilt-zoom camera controls, a gripper, and a robotic arm

The ActivMedia software provides APIs that allows for access to low-level raw sensor data and high-level processed sensor data. Similarly, actions can be specified through the APIs as either high-level actions, such as move to an (x. y) position. or low-level actions such as changes in wheel motor velocity The high-level action APIs permit a straightforward mapping of the synthetic agent API for ACT-R UT bots directly onto the ActivMedia APIs

The controlling software includes a simulation environment for the robot to allow faster testing and prototyping of control code without the need to run the real robot. or to be in the actual location being tested The simulated ActivMedia platform provides reasonably high fidelity and includes aspects of the real platform such as sensor noise and wheel slippage

4.3 Time Synchronization Details for UT and ActivMedia

Unlike many production systems, ACT-R exactly specifies the real-world timing of production matching and execution to the millisecond (a result of its role as a high-fidelity theory of human cognition) The explicit timing of ACT-R events allows for a straightforward integration of an ACT-R agent with a real-time simulation environment At the beginning of each cycle. ACT-R is provided with an up-to-date representation of the simulation environment, and ACT-R is allowed to perform whatever action it chooses The ACT-R clock is then advanced by the amount of time consumed by the action. and ACT-R will not be called for another recognize-act cycle until the simulated time has moved beyond the current ACT-R time

The effect of this scheme is that the cognitive agent always lags slightly behind the real world If updates are passed from the simulation to ACT-R at a reasonably high frequency (e.g . 10 Hz). the effect of this lag is negligible (and. in fact, roughly matches the latency of the human visual system) Thus, the ACT-R system acts on information that is. on average. slightly out of date but is never perceived before it could exist (updates are never early)

In the time synchronization scheme used. ACT-R is allowed to produce an immediate action. subject to the communication lag between ACT-R and the network infrastructure across which the agent is communicating with the simulation In this case the network latency combined with the real time required to run an ACT-R cycle approximates the actual time required for an action For the real-time systems described here. there is no obvious need for a more complicated mechanism (the agents can. for example. successfully track targets in real-time)

5 THE MOUT DOMAIN: REQUIREMENTS FOR INTELLIGENT AGENTS IN MILITARY OPERATIONS ON URBAN TERRAIN (MOUT) AND CLOSE QUARTER BATTLE (CQB) DOMAINS

MOUT environments are distinguished from the terrain of rural battlefields by the dominant features of densely packed manmade structures and multiple avenues of approach MOUT tactics have been developed through the analysis of historical urban conflict and extrapolation to the capabilities of modern soldiers These tactics prescribe methods for clearing blocks of buildings, individual buildings, floors in buildings, and individual rooms and hallways. 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 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, and the use of clear and explicit procedures implemented by small teams The work described here involves the implementation of collaborative doctrinal tactics at the level of the individual infantry soldier by intelligent agents

Doctrinal MOUT tactics are extremely well-defined Movement techniques taught in MOUT training specify how to move 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 with supporting personnel leapfrogging each other, alternating covering and moving. Clearing techniques specify which teammates will direct fire where, and how to arrange units prior to room entry

As an example of the specificity involved in this training, in a doctrinal room entrance, a pair of soldiers assumes a "stacked" position along the wall outside the doorway The lead soldier directs his weapon towards the far corner whereas 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 supressing any threats A second two-person team provides 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

Though there are many variations, it is worth noting the explicit nature of the teamwork involved

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 whereas the soldier on the near wall parallels this movement The soldiers then, on a signal, move together into the hallway, one crouching and the other standing, clearing all targets

Modeling the continuum of behavior from structured doctrinal behavior to unstructured reactive behavior allows testing a range of opposing force behaviors against the expected doctrinal strategy Unlike friendly force behaviors, opposing force 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 Thus, opposing forces should be capable of using doctrinal techniques, but also should be free to diverge substantially from them

5.1 Doctrinal Approaches to Building Clearing – Case Study: Clearing an L-Shaped Hallway

The vignette described here involves 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 earlier. The cleared hallway presents the soldiers with several doorways The soldiers then stack themselves at the doorways, enter the room (also described earlier), 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 Their planning is limited to the hiding spots and defensive positions they initially assumed – their goal is to defend the building they are in As they spot the entering soldiers, they hastily fire a shot or two while falling back When cornered, they dig in and fight (one of many possible scenarios)

5.2 Sample ACT-R Models

An overall ACT-R model for building clearing involves components that handle route planning (e.g., clear the first floor, then the second, etc), specify what to do on hostile contact, and include doctrinal approaches to many subtasks within the domain Space limitations preclude detailing a complete building clearing agent, so instead agents involved in clearing an L-shaped hallway will be focused on clearing it The possible actions encoded by agents that are faced with an L-shaped corner in a hallway will be detailed for one set of attacking agents and one set of defending agents. Below are English abstractions of some of the relevant 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 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 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 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 The conflict resolution process will decide which of these productions will fire in any given situation The basis for that decision will be each production's utility Those utilities, even if the system learns them to be different. have a stochastic component that will make the choice probabilistic, though not random because it is sensitive to the quality of each choice

6 GETTING AROUND THE VISION PROBLEM

Much of the previous discussion presupposes a working real-time perceptual system that provides a useful description of where enemies and friendly forces are. and how the surrounding architecture is arranged Although substantial progress has been made in the field of computer vision in the last decade. real-time algorithms for space perception and object identification are not yet realities This necessitates bypassing the vision problem In many synthetic environments, object identity and location are passed to agents in the domain as structured symbolic data rather than as image-based data This allows these agents to perform as if they had the results of high-level vision In robotic domains it is more common to develop a special-purpose sensor and provide a distinct cue for that sensor in the location where the object of interest is For example. a researcher could hang a large blue square on a piano and identify all large blue squares as pianos Alternatively, a researcher could place a radio beacon on a location and identify that location as the piece of cheese in a maze (or the intercontinental ballistic missile to destroy) In both of these examples, the robot does not perceive the actual target of the identification. but rather an easier-to-identify stand-in. This section will elaborate on the methods used for the ACT-R MOUT agents and the ACT-R ActivMedia agents for getting from the basic sensor data provided by the simulation to a high-level representation usable in a cognitive agent

6 1 Extracting Cognitive Primitives in Unreal Tournament

For an agent to navigate and act within a space it must have a representation of the environment that supports these actions, with more complex planning and teamwork requiring 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: (1) a low-level implementation-dependent feature extraction process. and (2) a method for translating this to a model-level representation usable by the agent Although the extraction process will vary for each environment 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. being limited primarily to a range-finding mechanism The challenge was to use this mechanism to automatically build up a cognitively plausible representation of space that could be used across platforms

6.2 Sampling the Space

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 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, traveling out on a ray from the current location. because 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 defines a boundary between open space and some solid object (e g., a wall) (see Figure 8.4)

From a particular location. an agent can perform this range sensing in any direction (this is analogous to laser range sensing as provided on the ActivMedia platform) 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 sample all of the

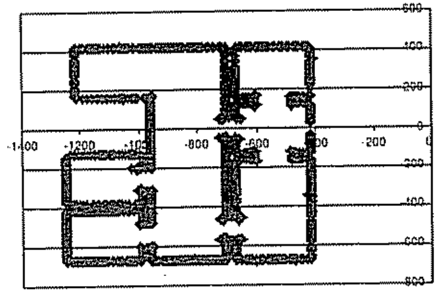

FIGURE 8.4 Sampled range sensing data

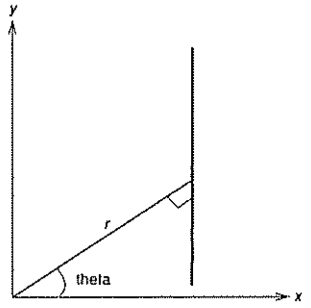

FIGURE 8.5 Parametric form of line equation

available spaces in a UT level The model reported here uses three-dimensional sampling to allow for the detection of upward sloping ramps. stairs, etc

6.3 Converting Sampled Points to a Line Segment Representation

The ActivMedia platform provides utility programs to produce a line segment representation from sampled sensor data. reducing this process to a single step Unfortunately, the Unreal Tournament environment does not provide this facility, making the derivation of a line segment representation from sensor data a more laborious process, described in detail here

Given a set of points that fall along walls on a map, determining which point falls on which wall and how to group them can be solved using a method known as the Hough transform (e g . Illingworth & Kittler, 1988) 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 (θ) represents the angle between the normal to the line and the x axis

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. each individual point will provide several values of r as theta (θ) 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

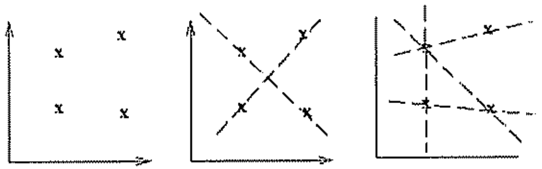

FIGURE 8.6 Candidate lines voted for by points

crossing through that point 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

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, resulting in a set of cells that correspond to lines that intersect large numbers of the input points

6 4 An Autonomous Mapping Agent for UT

Based on the sensor to line-segment process described in the sections above, we developed an autonomous mapping agent for UT that navigates the environment and gradually builds a representation of the space (see Figure 8.7).

Using the range-sensing data as the only spatial primitive, and a Hough transform to detect straight lines within the data, a cognitive-level description that consists of walls, corners, rooms, and openings is

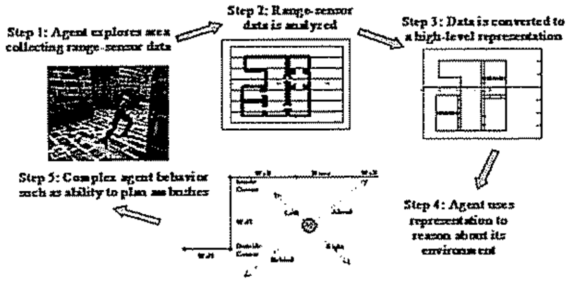
Step 1: Agent explores area collecting range-sensor data
Step 2: Range-sensor data is analyzed
Step 3: Data is converted to a high-level representation
Step 4: Agent uses representation to reason about its environment
Step 5: Complex agent behavior such as ability to plan ambushes

FIGURE 8.7 Autonomous mapping process

constructed From this static representation, the dynamic perceptual presence of architectural primitives relative to the agents' current location can be determined in real-time (Best et al . 2002)

6 5 Mapping for ActivMedia Robotics Platforms

The robotic platform presents a nearly identical challenge to UT in sampling the space Unlike Unreal Tournament, however, the ActivMedia platform provides direct support for obtaining a line-segment representation of a particular environment, making the development of an automated map-building agent unnecessary Mapping may be accomplished in several ways on the ActivMedia platform The robots are packaged with a utility that allows them to randomly wander, sampling the space as they go Due to the lack of guidance in this wandering, maps derived in this way are often very incomplete The alternative, and likely the most common approach, is for a human operator to teleoperate the robot during this process In this case, the operator essentially pilots the robot around the spaces to be mapped, ensuring the robot enters all of the corners, openings, and dead ends to provide complete coverage of the space

6.6 Data Structures: Line Segments, Bounding Volumes, and Binary Space Partitioning Trees

At this point in building the spatial representation, there is a set of wall segments defined by endpoints and openings (doors) aligned with those wall segments 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 aligned with a wall segment 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 whereas the other child subdivides the back half-space of the plane This recursive subdivision continues until all of the segments have been used as splitters The resulting data structure provides a means for computationally efficient determination of visibility determination that can be used to quickly determine the visible surfaces of the space surrounding an agent

6.7 Algorithms for Calculating Analytic Visibility

Although many algorithms for calculating analytic visibility exist, many of them are too computationally expensive to be used in real-time. One way around this difficulty, and the approach we have taken here, is the

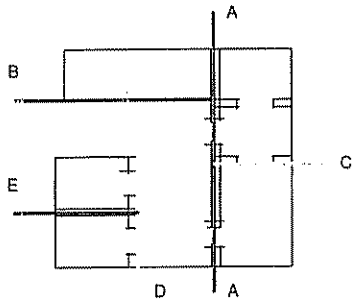

FIGURE 8.8 BSP Tree Splits map along wall segments.

use of a BSP tree for computing visibility. The primary benefit of the BSP tree representation is that it guarantees that an in-order tree traversal will draw edges in the tree in either front to back or back to front visibility order, so that no edge will ever be drawn before a potentially occluding edge is drawn This allows the fast calculation of walls and obstructions visible from a particular vantage point by traversing the tree in-order, which results in drawing walls from front to back (i e . closest walls first), and short-circuiting the process when all of the space around the agent is enclosed by an occluding wall This technique is an extension of a z-buffer technique where the tree traversal is done when all of the pixels in the buffer have been drawn once

7 COGNITIVE REPRESENTATION OF SPACE AND PERCEPTION: EGOCENTRIC AND ALLOCENTRIC REPRESENTATIONS OF DISTANCE AND BEARING

To perceive, react, navigate, and plan, it is necessary for the agents to have a robust spatial representation Like people, agents can represent things in two fundamental ways: where something is relative to the agent's location, or egocentrically (e g . something is to my left); or where something is in absolute terms relative to a world coordinate system, or allocentrically (e g., something is at a particular latitude/longitude)

The egocentric representation of an item used includes both the distance to the item and relative bearing, in both quantitative and qualitative terms (see Figure 8 9) A qualitative distance is how distant something is relative to the current visual horizon, and ranges across a set of logarithmically

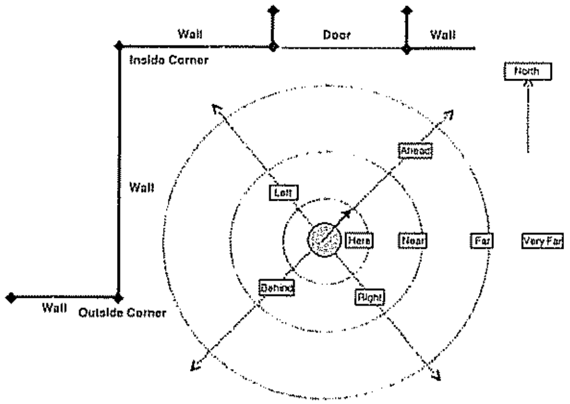

FIGURE 8.9 Cognitive representation of space used by agents

spaced rings denoted "here," "near," "far," and "very far," whereas a quantitative distance is the distance to the object in numerical units (e g . 7 meters away) Bearing is represented quantitatively as absolute compass bearing to target relative to current orientation (e g . 30 degrees to the left. 5 degrees up), and qualitatively as "right," "left," "ahead," "behind," or any of the four intermediate bearings "ahead right," "ahead left," "behind right," or "behind left "

The allocentric representation of an item includes the location of an item in the world coordinate system (in this case, x, y, and z) and potentially its orientation relative to that coordinate system (pitch, yaw, and roll – the angles relative to the axes) An allocentric representation is particularly important in reference to maps (which are typically defined relative to some world coordinate system), and correspondingly to navigation tasks

Many of the doctrinal rules for MOUT can be spelled out clearly using an egocentric representation For example, the following describes how to employ the cross method of clearing a room:

When employing the cross method, two Marines position themselves on either side of the entryway. Each Marine faces into the room covering the corner of the room opposite his position On a prearranged signal, each Marine alternately enters the room Each Marine crosses quickly to the opposite corner while covering the half of the room toward which he is moving Once in the near corner, he assumes an outboard kneeling position to reduce his silhouette and continues to maintain coverage of his half of the room (MWCP, p A-34)

However, for other purposes, the use of an allocentric representation is preferable to an egocentric representation For instance, tasks such as navigation need a representation of the features in a global frame of reference that allows for computations (such as the path between a given room and the exit of the building) independently of the current location and its immediate environment The challenge is to integrate the two representations, using each where it is best while maintaining consistency between them That integration takes the form of an accumulation of egocentric information provided at each instant that is particularly useful for reactive behavior into a global, persistent map of the environment in an allocentric representation suitable for navigation and other global planning behaviors

8 NAVIGATION

Navigation is one of the most essential tasks an agent must undertake in a spatial environment Navigation is accomplished through combining basic locomotive behavior with the immediate results of perception – that which is perceived at that moment – and interaction with a memory-based cognitive map of the environment

8.1 Navigational Primitives

Locomotion is simply moving from one location to another This is a fundamental behavior that need not be attended to once it is initiated, and thus may occur in parallel with other activities. The basic behavior of locomotion involves commencing movement to a location, the continuation of that movement while not at the destination, the abandonment of that movement if an obstacle, or threat is encountered or a change in plans is initiated, and the cessation of movement upon arrival at the destination Locomotion can be performed in several modes: walking, running, stalking, and sidestepping while facing another direction

8.2 Higher-Order Navigational Behavior

Higher-order navigational behavior involves an interaction of the cognitive map of the environment (the allocentric reference frame) with the current visual scene (egocentric cues) and memory for goals and past events (paths followed and destinations) As such, it represents a significant theoretical challenge in both cognitive psychology (Klatzky, 1998) and robotics (Beetz, 2002; Frank, 2000).

Agents in this simulation use a node-link representation for rooms and pathways between them Attacking agents build up a representation of rooms visited and episodic traces of items and other agents seen there When moving from the current room through a doorway to a new room, the agent creates a chunk in declarative memory corresponding to that path allowing an overall map to be built Defending agents, who are assumed to have intimate knowledge of the area to be defended, are given a complete representation of the rooms and pathways connecting them allowing them to fluidly and quickly choose paths for attack and escape that real defenders would have knowledge of (but attackers would not)

8.3 The Interaction of Memory and Perception in Navigation

Although memory for paths exists in a complete form in the defenders' declarative memories, the attackers may be forced to rely on other methods In addition to remembering the path followed, attackers may also encode individual moves at particular situations This is similar to the heuristic applied by some people who "retrace their footsteps" when trying to find their way. These previous moves include actions relative to landmarks (e g , turn left at the L-shaped hall), actions relative to an allocentric frame (e g , proceed at a compass bearing of 90 degrees), or actions relative to an egocentric frame (e g., turn left 45 degrees) These representations are complementary, and are typically used by people as the context allows Landmarks are often preferred, but in a situation where landmarks are impoverished, people quickly adopt the other strategies If going to a house in a subdivision where all of the houses look alike, people commonly depend on memory for the moves such as "turn left at the second street in the subdivision and go to the fourth house on the right " In a navigation context, an allocentric frame such as that encoded in a map is often used This is particularly useful in military situations for exchanging information about threats, destinations, and movements, because allocentric coordinates such as GPS coordinates are unambiguous, whereas egocentric coordinates depend on knowing the egocentric orientation of the perceiver and are therefore often less useful

9 COMMUNICATION

Planning, as presented above, requires at the least the ability for agents to signal each other We have provided a grammar that the agents use to communicate that includes signaling, acknowledgment, sending and receiving orders, communication of intention, and specification of the type and location of a contact (e g., friendly fire, from location (x, y, z)).

The most fundamental of these, simple communication, involves the passing of signals and the acknowledgment of their receipt For example, saying "On the count of three, go" requires the receipt of the signal "three" while ignoring other signals In the UT environment, this is implemented by passing text messages between the agents Although the agents could have passed tokens in a coded language, the agents use actual English phrases for readability and extensibility to interactions with human players

The passing of orders initiates execution of schematic plans These plans include actions such as clearing an L-shaped hallway, supplying covering fire, moving to a particular location, standing guard, providing assistance in storming a particular room, or retreating from overwhelming fire These schematic plans depend on doctrinally defined simple communications For example, when storming a room, attackers typically "stack" outside the doorway Attackers in front are signaled by the attackers behind that they are in position to enter the room, obviating the need to turn away from a potentially hazardous entrance at a critical moment Although it is possible for real combatants to signal each other non-verbally (e g , with a touch on the back), agents in this environment are limited to the passing of text messages

In addition to orders, agents can also share information, such as a spot report of enemy activity A spot report includes a brief description of the enemy forces spotted including their numbers and armament if known, their location, and their movement (if any) Other agents may use this information to provide coordinated ambushes and attacks

10 IMPLEMENTING PLANNING AND TEAMWORK IN ACT-R FOR MOUT

Within the framework developed for this project, a set of productions interprets the schema within the current context, leading to a literal interpretation of the schema for that context. In this way, an abstract plan plus a context results in a set of concrete actions This allows the abstract plan to be fairly brief and vague until the agent actually selects it to be put into action. At that point, the plan will be instantiated in a manner consistent with the details of the current situation

The current modeling effort includes plans for a team of two for clearing: rooms with and without doors, halls, L-corners, T-intersections, and stairs In addition, plans are included for advancing and retreating in a leapfrog style, and for firing a defensive shot in an attempt to cause casualties immediately prior to escaping (cut and run) A sample chart of the interactions of two agents clearing an L-shaped hallway is presented in Figure 8 10

At each step of the plan, agents perform an action, communicate, or wait for a predetermined signal or length of time before moving on to

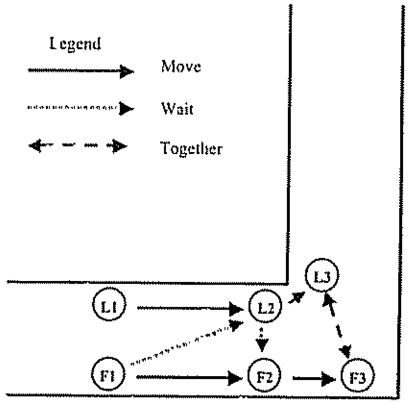

FIGURE 8.10 Schematic representation of plan to clear L-shaped hallway

their next action In this way, the agents synchronize their actions taking turns appropriately. The plans the agents adhere to are not ad-hoc but instead come directly from doctrinal MOUT documents Doctrine typically not only specifies how one agent should be positioned relative to another for activities such as clearing L-shaped halls but even specifies the exact language to be used in this situation This knowledge is typically a set of steps spelled out in declarative form with the particular actions, triggers, and synchronization of action all clearly defined Given the cookbook nature of some of these doctrinal maneuvers, we noticed an opportunity to create an authoring tool to aid in the conversion of doctrine to cognitive models

The model for the agents described here was authored like a typical ACT-R model However, the knowledge structures, especially declarative chunks of information and production rules, were written using an abstract notation rather typical of production systems Table 8 1 presents a typical example of a production rule and related chunks:

The production implements the sequential retrieval of a piece of an action plan, and the declarative chunks represent some of those action pieces The situation involves two agents, L for Leader and F for Follower, moving in coordinated fashion through a sequence of positions and actions, as shown in Figure 8 10 Their various positions are indicated by

TABLE 8.1. *Production Rule Used in Planning and Relevant Chunks Representing a Plan*

```
(p get-next-action               (action11
  =goal>                         isa action
    isa action                   plan 1
    plan =plan                   index 1
    index =index                 type move
    type nil                     argument 11)
    argument nil                 (action 12
  =action>                       isa action
    isa action                   plan 1
    plan =plan                   index 2
    index =index                 type wait
    type =type                   argument go)
    argument =argument           (action13
==>                              isa action
  =goal>                         plan 1
    index (1+ =index)            index 3
    type =type                   type end
    argument =argument)          argument none)
```

an index Solid arrows between successive positions indicate movement Dotted arrows indicate when an agent waits for the other to have performed an action (such as reached a position) to proceed with the next step of its plan Dashed arrows indicate synchronized actions between the two agents Other codes specific to the domain can be added to the graphical interface in a modular fashion

All those codes transform readily into a piece of the plan for each agent as encoded in declarative chunks in Table 8 1 Each chunk contains a number of slots. The index of the plan, plan. and the index of each action. index. can easily be supplied automatically by the interface The nature of the action, type. depends on the code used in the graphical interface. e g. a solid line would translate into a move action. etc A list of interfaces codes and associated actions can simply be encoded into the interface for each domain The last slot. argument. is an action qualifier, such as where to move. e g. to position L2 This argument represents the most difficult part of the mapping. because obviously one does not want to encode a specific location but instead one that will generalize to similar situations (in this case. the position nearest the corner of the L-shaped hallway) Humans, even non-experts, usually understand readily a set of spatial relationships between the various positions and landmarks to generalize them across situations, e g . to symmetrical situations The challenge before us is to provide in the model a sufficient knowledge base to supply those spatial relationships automatically

TABLE 8.2 *Schematic Plan for Clearing an L-Corner*

```
(p get-next-action               (action-L1
  =goal>                         isa action
    isa action                   plan take-L-corner
    plan =plan                   index 1
    index =index                 type move
    type nil                     argument inside-corner)
    argument nil                 (action-L2
  =action>                       isa action
    isa action                   plan take-L-corner
    plan =plan                   index 2
    index =index                 type wait
    type =type                   argument go)
    argument =argument           (action-L3
==>                              isa action
  =goal>                         plan take-L-corner
    index (1+ =index)            index 3
    type =type                   type move
    argument =argument)          argument around-corner)
```

10.1 Schematic Plans

A large part of the teamwork exhibited by these agents hinges on sharing common knowledge about how to approach certain tasks The details on how to do this come directly from military doctrinal manuals (e g , see MCWP in the references) and are routinely taught to trainees as part of their fundamental training Each agent knows, as a trained human combatant does, what actions to perform when playing any of the roles in different scripts This knowledge is stored as a set of chunks in declarative memory of the agent These chunks, analogous to a schema, are a somewhat general description of what to do in a certain situation. The details are then filled in by productions that interpret the declarative script given the currently perceived environmental context. Table 8 2 gives an example of a production that selects the next step in an action plan as well as three steps of the plan.

Plans in which an abstract script is filled with details later are sometimes referred to as "skeletal plans," or sometimes simply as "scripts" (Stefik. 1995) We have chosen to use "schematic plans." because the plans we are dealing with here have a spatial component. and are most easily visualized using a schematic diagram

For example. when clearing an L-shaped hallway, the procedure for clearing the hallway is well-defined (see Figure 8.10) A pair of attackers will split up and take position along the front wall (agent L in the diagram) and back wall (agent F) respectively Agent L then moves forward close

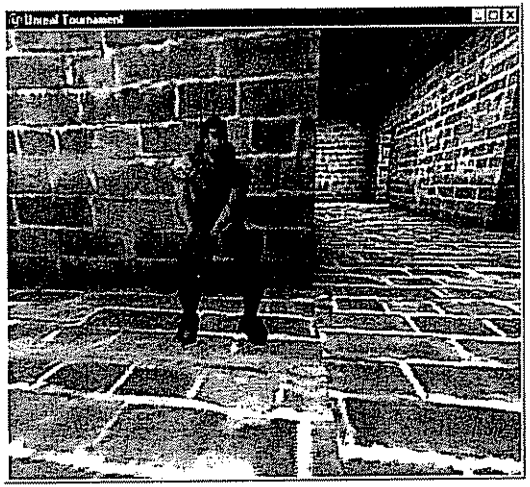

FIGURE 8.11 Agent viewing teammate preparing to clear L-shaped corner.

to the corner as agent F waits for a signal Once in position, agent L signals agent F to move Agent F then advances to a position almost directly across the hall from agent L At this point, agent L waits for F to signal the next move Upon agent F's signal, L and F simultaneously move into the hallway, L staying close to the corner and dropping to a crouch whereas F sidesteps along the back wall This brings both of their weapons to bear on the hallway simultaneously, although allowing both of them an unobstructed field of fire including the whole hallway

These schematic plans, then, are scripts with a spatial component that describe how multiple agents are expected to work together in a particular situation For both human combatants and agents, this bypasses the potentially deadly inefficiency of trying to decide what to do at each step Each agent knows what to do, and what to expect from a partner Signals are predefined and the potential for confusion is relatively low The MOUT environment provides a clear example of a domain where teamwork is explicitly taught at a fine level of detail A visual snapshot of two agents performing this script in UT is presented in Figure 8 11 (viewed from the perspective of one of the agents)

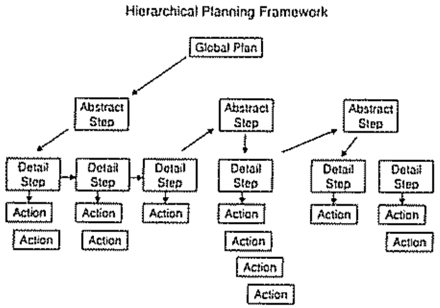

FIGURE 8.12 Hierarchical plan levels and components.

10 2 Hierarchical and Partial Planning

Due to the dynamic nature of the task environment, it is not possible to fully develop a plan prior to performing the first actions Instead, a rough plan consisting of abstract steps is developed The abstract steps themselves can be implemented by the schematic plans described earlier In turn, individual actions to accomplish the schematic plans are combined with elements from declarative memory and perception to form an action plan on an as-needed basis (see Figure 8 12 for a diagram of the hierarchical planning framework) This provides flexibility and robustness in the actual actions taken because they are planned with the immediately perceived context in mind.

This method of planning has roots in means-ends analysis and has much in common with skeletal planning and hierarchical match algorithms (see Stefik, 1995, for a discussion of this) Because the plan can be modified at several abstract levels, it may be better described as hierarchical planning. However, the individual action steps themselves are highly constrained whereas the planning at the more abstract levels is less constrained This significantly reduces planning complexity as the sequence of action nodes is most often predefined by a schematic plan The interesting implication is that human combatants have developed schematic plans to deal with exactly those situations that present many options In any case, this type of hierarchical planning, modified by on-the-fly circumstances, provides planned, goal-directed behavior that is sensitive to context. The abstract plan of clearing the two floors will not change under most circumstances, but the details of carrying out these steps often cannot be known in advance (Schank & Abelson, 1977) This provides an efficient compromise between

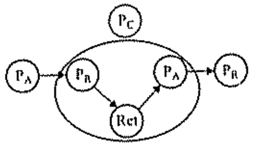

FIGURE 8 13 Proceduralization of retrievals.

the need for flexibility in robustly adapting one's behavior to unforeseen (and unforeseeable) circumstances with the need for efficiency in executing any actions in dealing with immediate threats This tradeoff is representative of many everyday though less dramatic human environments, e g driving

11 PROCEDURALIZATION, GENERALIZATION. AND FLEXIBILITY

Plans of action are represented in the form of a list of declarative chunks (see Table 8 2 for an instance) each representing a particular step of action such as moving to a location. waiting for a partner, firing at a target. etc The execution of those plans takes the form illustrated in Figure 8 13 Each cycle consists of a production firing requesting the retrieval of the next step (P_R). the retrieval itself (Ret), then one or more production firings implementing that course of action (P_A).

Although production firings are quite rapid (usually taking about 50 milliseconds). retrieval of a chunk of information from declarative memory typically takes several hundreds of milliseconds This corresponds to a poorly trained opponent who consistently thinks about his actions rather than simply executing them To represent a better trained opponent able to execute plans of action much more quickly and efficiently, one can take advantage of a feature of the ACT-R architecture that compiles consecutive productions, together with an intervening information request such as retrieval from memory. into a single production (P_C). specialized to the task at hand. which can then fire much more quickly than the series of interpretive steps that it replaced However. one feature of declarative retrievals is the ability to generalize to related situations based on similarities between components such as distances. angles, appearances, etc This is quite useful in applying plans of action flexibly to situations that do not quite match the original design Therefore. to allow proceduralized plans to retain the flexibility of interpreted plans, we need to provide production rules with the same kind of flexible matching primitives as declarative memory

Perhaps the most significant difficulty in authoring production system models (e g expert systems) is specifying the conditions under which

productions can apply Because of the lack of a conventional control structure, it is often difficult for the author to forecast exactly the full range of symbolic conditions under which an action is applicable Moreover, in dynamic. approximate and uncertain domains (such as a MOUT simulation). the all-or-none symbolic conditions (i e either specify a specific value required or else no restriction on that value) that determine production rules' applicability have significant limitations in capturing the loose range of conditions under which a behavior might be applicable What is desired is the ability to specify a canonical case for which a production is applicable. then have the production system generalize it to related situations

A similar need for flexibility on matching chunks of information in declarative memory has long been recognized and addressed with the addition of a partial matching mechanism to memory retrieval. allowing chunks that only partially match the desired pattern specified by a production retrieval request to qualify for matching A chunk's activation. which represents in ACT-R the likelihood of a chunk being relevant to a particular situation. is decreased by the amount of mismatch. thereby reducing the probability of retrieving that chunk but not eliminating it altogether The similarity values used in specifying partial matches between chunk values can be viewed as a high-level equivalent to distributed representations (specifically. to the dot product between representation vectors) in PDP networks It seems logical to implement the same mechanism for production rule matching, thereby emphasizing the symmetry between the declarative and procedural parts of architecture by unifying their matching mechanisms Practically, this allows pieces of knowledge that were specified and used as declarative instances to seamlessly transition to production rules

Currently, only production rules whose conditions match perfectly to the current state of various information buffers (goal. memory retrieval. perceptual. etc) qualify to enter the conflict set Because ACT-R specifies that only one production can fire at a time, the rule with the highest expected utility is selected from the conflict set as the one to fire The utility of a production rule is learned by a Bayesian mechanism as a function of its past history to reflect the probability and cost of achieving its goal. In a manner similar to partial matching in declarative memory, all rules (subject to types of restrictions for tractability reasons) will now be applicable but the new mechanism of production rule matching will scale the utility of a rule by the degree to which its conditions match the current state of the buffers Specifically, the scale utility (SU_p) of a rule p is specified as:

$$SU_p = U_p + \sum_{conds} MP \quad Sim_{vd} \qquad \textbf{Scaled Utility Equation}$$

where U_p is the usual utility of the rule. and the penalty term is a product of MP, a mismatch scaling constant. and Sim_{vd}. the similarity between the actual value v present in a buffer and the desired value d specified in the production condition. summed over all production conditions Similarities are 0 for a perfect match, leading to no change in production utility, and negative for less-than-perfect matches. leading to decrement in utility that lowers the probability of the rule being selected with the degree of mismatch The mismatch penalty MP can be seen as a regulating factor, with large values trending towards the usual all-or-none symbolic matching

Our experiences using this mechanism. show that it succeeds in providing the desired approximate and adaptive quality for production rule matching All things being equal. productions will generalize equally around their ideal applicability condition. However, productions with higher utility will have a broader range of applicability, up to the point where they reach their limits and failures lower their utility, thereby providing a learning mechanism for the range of applicability of production rules Moreover, the range of applicability of a production rule will be a function of the presence of production rules with similar competing conditions In the initial learning of a new domain, a few production rules will be generalized broadly as all-purpose heuristics As more knowledge of the domain is accumulated and new production rules created. the range of application of those rules will be increasingly restricted

Using this modification to the ACT-R production-matching scheme. no "hard" boundaries exist between conditions for matching productions; the boundaries are instead continuous For example. if production A is appropriate when a doorway is to the front. whereas production B is appropriate when a doorway is to the left side. both productions may match when a doorway is both ahead and to the left Although specifying directions such as "left" as a range makes it possible to match a production in a symbolic system to a range of situations, specifying "left" as a precise direction and allowing productions to match based on similarity to that condition allows both cleaner specification of the underlying representation (i e . "left" is 90 degrees to the left instead of between 45 degrees and 135 degrees to the left). and easier authoring of the productions with a reduction in unwanted interactions between pieces of procedural knowledge In this case. if the author later decided that a new production. production C. was appropriate when a doorway was ahead and to the left. adding the new production C to the system would result in that production predominating over the others without any revision of productions A and B

This feature has significant theoretical and practical importance. because it imbues the ACT-R architecture with many of the properties of a case-based reasoning system. or a fuzzy matching rule-based system (similar to the similarity-based reasoning proposed in Sun, 1995) Partial matching in procedural memory allows ACT-R to automatically select the closest (imperfectly) matching production in a case where no production is exactly appropriate This provides similarity based generalization where the

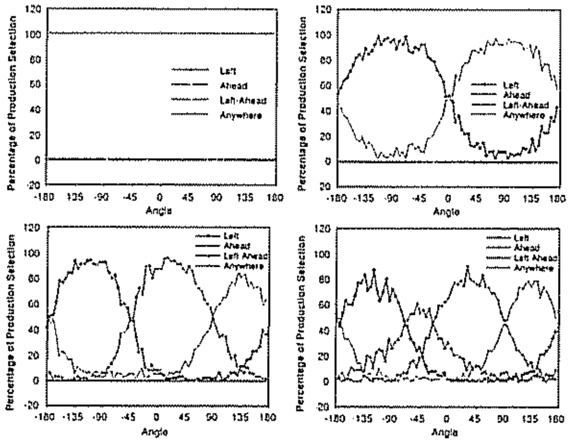

FIGURE 8 14 Production selection frequency by similarity to condition

similarity metric can be determined from a psychologically appropriate model of the stimulus domain (e g . logarithmic scales to determine similarity in the size of geometric solids) On the practical side. this feature allows the ACT-R pattern matcher to select the most appropriate production when faced with a novel situation. and can provide a substantial boost to robustness by preventing the system from falling into a behavioral black hole where. as no rule is exactly applicable. it does nothing

12 ACTION VS REACTION

The schematic plans outlined earlier indicate that the system is capable of goal-directed activity However. in a real-time system such as this. the environment may change in a way that is incompatible with the current goal As an example. an agent may have a goal to move towards the doorway of an unexplored room If an enemy enters the hallway within sight, the agent clearly should abandon exploration and deal with the threat ACT-R 5 0 provides a mechanism built into the architecture that allows for interruption by critical events – multiple buffers In this case. a buffer is used to keep track of any perceived threats Exploratory behavior continues in the absence of a threat, but once a threat is perceived. the perception of the threat interrupts the current system goal and forces the agent to deal with the threat (though the agent could then choose to ignore the threat, that choice still must be made explicitly).

Similarly, occasionally it is desirable to simultaneously pursue two goals. For example, while moving from one location to another, an informational message may be received Although it would be simple to abandon the movement to handle the incoming transmission, the preferred solution is to continue the agent's movement while handling the transmission. This is also accomplished through the use of a buffer for keeping track of an initiated movement The human behavioral equivalent is driving from place to place – often once the drive is initiated, other goal-directed behavior occurs without interrupting the drive It is not that the two goals are being serviced at the same time but that the pursuit of compatible simultaneous goals can be achieved without simultaneous actions – actions can be interleaved through the use of architectural primitives such as goal retrievals and buffer switching, which do not provide all-powerful ways to multi-task but instead a knowledge-based, robust, and flexible way to reconcile goal-directed and reactive behavior

13 SUMMARY

The focus of this work has been the development of agents capable of interacting in small teams within a spatial domain and real-time environments. The social interactions within these domains are particularly well-defined and cooperation is either ensured (in the case of teammates in a military domain or a robot assistant in a robotic domain), or deliberately avoided (in the case of opponents) Teamwork, in these cases, depends on the ability to share interleaved plans, the ability to effectively communicate the intention to execute a plan and the current step involved, and the ability to share a representation for describing the surrounding space Cognitive modeling in these domains provides for straightforward implementation of human behavior that provides a demonstration of how explicit fully-committed teamwork functions The cognitive models discussed in this chapter will hopefully enable the abstraction of deeper principles of teamwork from a fully-specified domain that can then be generalized to domains where things are not spelled out quite so literally

In addition to the emphasis on teamwork, this chapter has brought out what we believe to be a significant synergy in research efforts for virtual synthetic and real robotic platforms. In this case we have demonstrated that the same framework for action and perception at the cognitive level used in the ACT-R agents discussed earlier can be used to control behavior in both virtual and robotic domains By insulating low-level perception and action from higher-level cognition through a principled, cognitively plausible spatial and motor representation, we have shown that a mature, validated cognitive architecture can be used to provide robust high-level behavior in a broad range of spatial environments, real and virtual Agents built in this manner are insulated from an overly tight dependency on the low-level details of their environment, providing the opportunity for reuse of models or parts of models across environments, thereby allowing the porting of previously validated models at a low relative effort

References

Anderson, J R., & Lebiere, C (1998) *The Atomic components of thought* Mahwah, NJ: Erlbaum.

Beetz, M. (2002). *Plan-based control of robotic agents. Lecture notes in artificial intelligence 2554* Berlin: Springer-Verlag.

Best, B. J., Scarpinatto, K C., & Lebiere, C (2002). Modeling synthetic opponents in MOUT training simulations using the ACT-R cognitive architecture *Proceedings of the 11th Conference on Computer Generated Forces and Behavior Representation* Orlando, FL: University of Central Florida

Capps, M., McDowell, P., & Zyda, M (2001) A future for entertainment – defense research collaboration, *IEEE Computer Graphics and Applications*, *21*(1), 37–43

Frank, A (2000) Spatial Communication with maps: Defining the correctness of maps using a multi-agent simulation. *Spatial Cognition II*, 80–99

Gillis, P. D (2000) *Cognitive behaviors for computer generated command entities* U S. Army Research Institute Technical Report.

Hicinbothom, J. H. (2001). Maintaining situation awareness in synthetic team members *Proceedings of the 10th Conference on Computer Generated Forces and Behavior Representation*, pp. 231–241 Norfolk, VA: SISO, Inc

Illingworth, J, & Kittler, J (1988) A Survey of the Hough Transform *Computer Vision, Graphics and Image Processing* *44*, 87–116.

Kaminka, G. A., Veloso, M., Schaffer, S., Sollitto, C., Adobbati, R., Marshall, A N, et al (2002) GameBots: A flexible testbed for Multiagent team Research. In *Communications of the ACM* 45(1), 43–45.

Klatzky, R L (1998). Allocentric and egocentric spatial representations: Definitions, distinctions, and interconnection *Spatial Cognition*, 1–18.

Lebiere, C, Anderson, J R., & Bothell, D (2001) 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.

Lyons, D., Cohn, J, & Schmorrow, D (2002) Virtual technologies and environments for expeditionary warfare training. *Proceedings of the IEEE Virtual Conference Reality 2002* p 261 Washington, DC: IEEE Computer Society.

Marine Corps Warfighting Publication (MCWP) 3–35.3, *Military operations on urbanized terrain (MOUT)* Washington DC: Headquarters, USMC, 1998

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

Newell, A, & Simon, H A (1972) *Human Problem Solving* Englewood Cliffs, NJ: Prentice-Hall

Pew, R. W . & Mavor. A S (1998). *Modeling human and organizational behavior: Application to military simulations.* Washington, DC: National Academy Press

Reece D. (2001) *Notes on cognitive limitations of DISAF* SAIC Technical Report

Reece, D., Ourston. D , Kraus, M., & Carbia, I (1998). Updating ModSAF for individual combatants: The DISAF program *Proceedings of the 7th Conference on Computer Generated Forces and Behavior Representation* Orlando. FL: University of Central Florida

Sanner, S , Anderson J R., Lebiere, C , & Lovett, M (2000) Achieving efficient and cognitively plausible learning in backgammon. In *Proceedings of the 17th International Conference on Machine Learning* (pp 823–830) San Francisco: Morgan Kaufmann.

Schank, R. C., & Abelson. R P. (1977) *Scripts, plans, goals and understanding* Hillsdale. NJ: Erlbaum

Schunn. C., & Harrison, A (2001) ACT-RS: A neuropsychologically inspired module for spatial reasoning. *Proceedings of the Fourth International Conference on Cognitive Modeling* (pp. 267–268) Mahwah, NJ: Erlbaum.

Silverman. B G . Might, R., Dubois, R . Shin, H., Johns, M . & Weaver. R (2001) Toward a human behavior models anthology for synthetic agent development *Proceedings of the 10th Conference on Computer Generated Forces and Behavior Representation* Norfolk, VA. 2001

Stefik. M (1995) *An introduction to knowledge systems* San Francisco: Morgan Kaufmann.

Sun, R. (1995). Robust reasoning: Integrating rule-based and similarity-based reasoning. *Artificial Intelligence, 75,* (2). 241–296.

Weaver, R., Silverman. B G , & Shin. H (2001) Modeling and simulating terrorist decision-making: A performance moderator function approach to generating virtual opponents. *Proceedings of the 10th Conference on Computer Generated Forces and Behavior Representation* Norfolk. VA