Modeling Synthetic Opponents in MOUT Training Simulations

Brad Best

Human-Computer Interaction Institute

Carnegie Mellon University

VIRTE: Virtual Technologies and Environments (ONR)

- Some of VIRTE's Goals:
  - Develop and demonstrate leap ahead human-immersive technology for naval training.
  - Supplement & complement live simulations using virtual & wargaming simulations and other emerging technologies.
  - Train soldiers for ever increasing complexity and chaos.
- VIRTE's Approach:
  - Expand the current knowledge base.
  - Incorporate current understanding of human behavior & learning theories into systems.
  - Leverage commercially available advanced technology.
  - Evolve current capabilities into products and transition products to the naval forces.

Project Elements

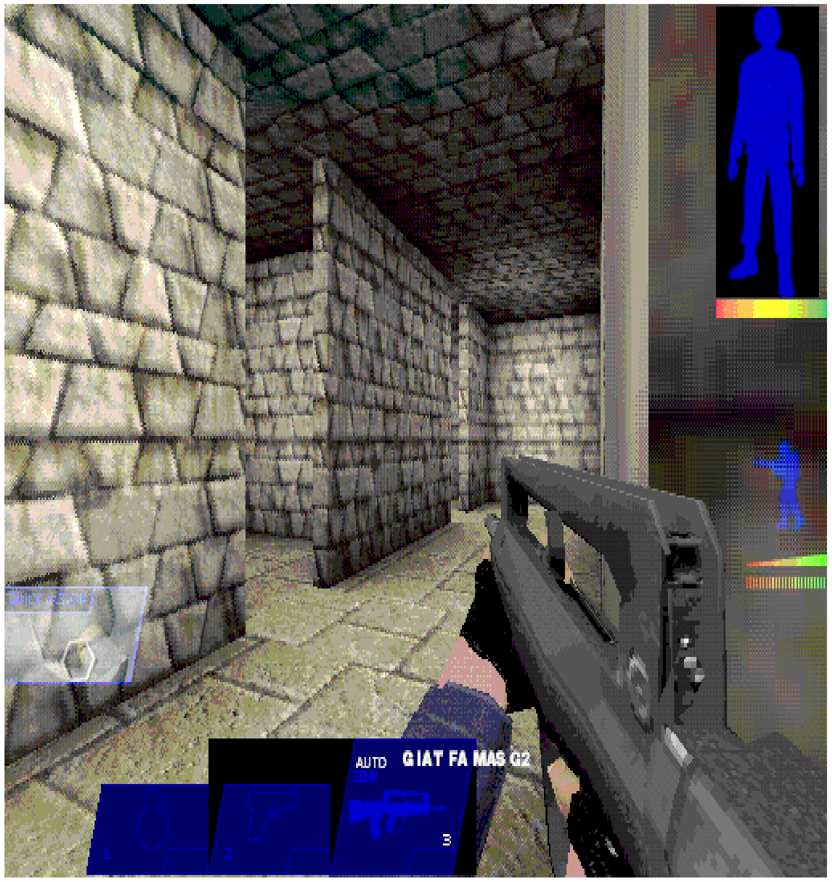
- MOUT Training Simulations
  - Urban Terrain: dominated by buildings and streets
  - Typical scenario: capture defended building
  - Platform: Unreal Tournament/Infiltration
  - Knowledge Base: SMEs and military doctrine
- Synthetic Opponents
  - Exist in a synthetic world (Unreal Tournament)
  - Must interact with each other and live soldiers
- ACT-R Cognitive Architecture
  - Used for modeling human behavior
- Putting it together: ACT-R in UT doing MOUT

Synthetic Environment: Unreal Tournament

- Off the shelf software
  - Low cost
  - Wide user base and support
- Virtual 3D First-person Game Engine
  - High performance graphics rendering
  - physics modeling
- Customizable
  - Mod authoring: Infiltration
- Support for Synthetic and Real Opponents
  - GameBots API

The Infiltration Mod for UT

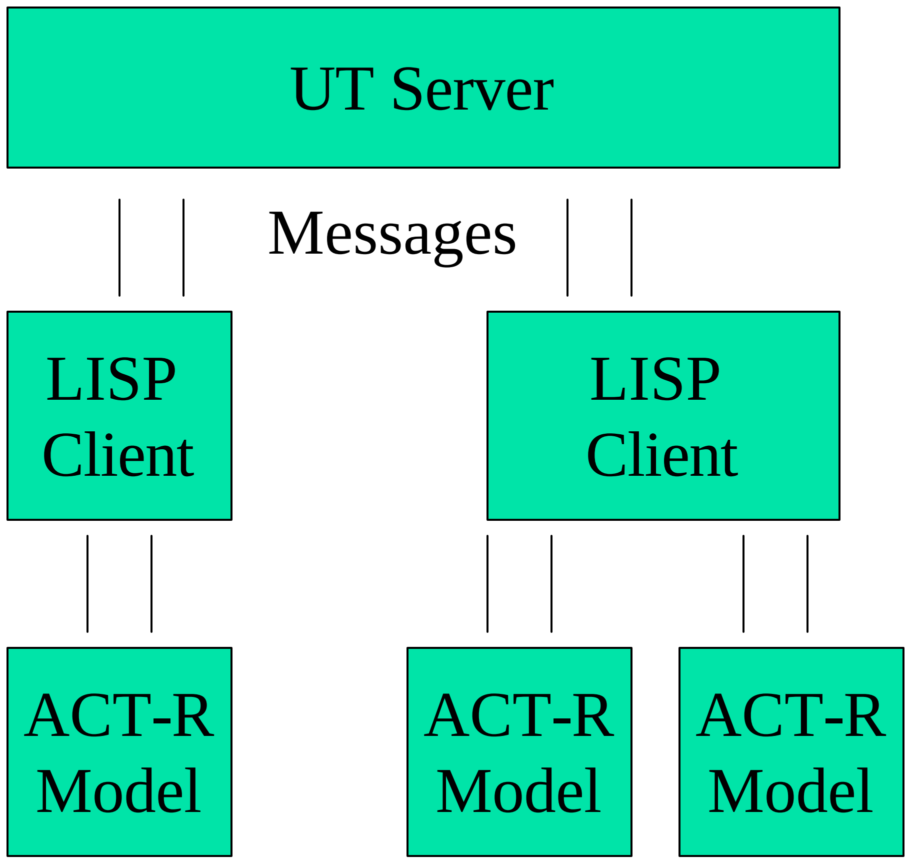
- Realistic weapons
  - Grenades
  - Machine guns that run out of ammunition and need to be reloaded, kick when fired, deadly
- Realistic actions
  - Stand, crouch, kneel, lie prone, lean
  - Carefully or loosely aimed weapons
  - Run, walk, creep



AI Development in UT

- GameBots mod allows synthetic characters in the game to be controlled via network sockets connected to other programs.
- Sensory Messages
  - Messages containing sensory information are passed from the UT server to the client
- Command Messages
  - Commands may be issued from the client to the bot on the UT server

System Architecture



- Bots interact via a Lisp client that is connected to the UT server, receiving sensory messages and sending command messages.
- Bots may be controlled with ACT-R models or Lisp code.
- Each bot runs in a thread and may use an independent ACT-R model.

Sensory Messages

- Received by the Lisp client from UT over a socket
- Messages contain a tag followed by a set of attribute/value pairs
- Synchronous messages:
  - appear in batches at a configurable interval
  - begin - begins a batch
  - slf - an update of the bot's state
  - plr - another player has come into the bot's field of view
  - end - ends a batch
- Asynchronous messages:
  - appear in response to events in the game
  - bmp - the bot just bumped into another player
  - dam - the bot took damage (from being shot, etc.)

Command Messages

- Sent from the Lisp client/bot over a socket to the UT server
- Messages contain a tag followed by a set of attribute/value pairs
- Example Message Types:
  - changeweapon - change the weapon the bot is using
  - rotate - turn a specified amount
  - runto - turn and move directly towards a destination
  - shoot - start firing a weapon

Opponent Task Decomposition

- Situational Awareness
  - What does a soldier know?
- Unit task level
  - High level goals and key reactive behaviors
- Functional task level
  - Breakdown of unit tasks
- Perceptual/Motor task level
  - Implementation of functional task level for a platform
  - Seeing walls, openings, and doors is critical

Situational Awareness

- Global
  - Location of self, friendly units, hostile units
  - Mission
  - Navigation and weather information, cleared areas
- Local
  - Target information (location, identity, weaponry)
  - Self information (ammunition, health, location)
  - Terrain information (cover, concealment, walls, doors)
  - Avenues of entrance/escape
  - Weapon Fire (type, source, target)

Unit Task Level

- Defend Area
  - Defend L-shaped Hallway
  - Defend Room
- React to Contact
- React to Communication
- React to Fire
- Give Order/Communicate
- Retreat

Functional Task Level

- This task level is platform independent
- Example Breakdown for ‘Defend Area’ Goal
  - Locate entry point(s)
  - Identify dominant position
  - Assume dominant position
  - Locate exits (retreat path)
  - Scan area for targets
  - Respond to contact

Perceptual/Motor Task Level

- Behaviors that can be implemented
  - Platform dependent implementation (UT)
- Examples:
  - Fire weapon from shoulder
  - Fire weapon from hip
  - Move to location
  - Identify L-shaped Hallway
- The challenge at this level:
  - Perceiving walls, doors, spatial layout

Interactions of Tasks and Interruptions

- Complications
  - Goal stack: avoiding too much structure
    - Use of sub-types allows flexible matching and squashes the stack
    - Needs to be interruptible, but only by certain conditions
  - Multiple goals
    - Should a bot pursue a goal to give an order while being shot at?
    - Should a bot abandon a goal to return fire while being shot at?

Buffers Simplify the Situation

- Solution: ACT-R 5.0 Buffers
  - Goal Buffer
    - Structured behavior
  - Visual Buffer
    - See another player
  - Auditory Buffer
    - Hear shots fired
  - Allows matching multiple goal conditions simultaneously
  - Allows reactivity and more flexibility

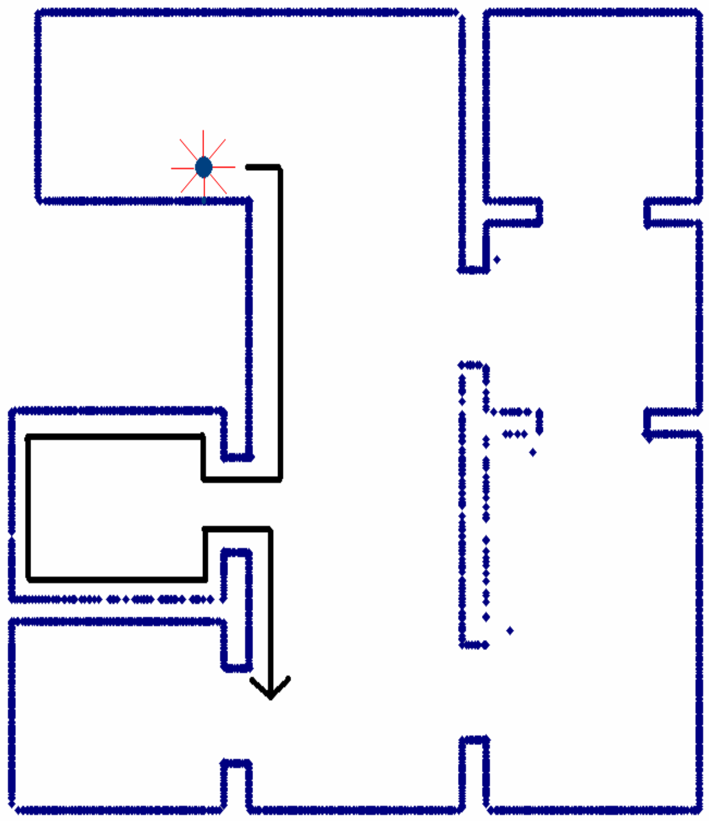
A Key Prerequisite: Perceiving Spatial Layout

- This was much harder than we expected
- UT maps do not provide boundaries easily
  - If they did, it would not be portable to another platform
  - Create UT map by carving overlapping volumes from space
- Since UT doesn’t provide the map, the challenge is very similar to robotic mapping
  - Range Sensing
    - Very little sensor noise
  - No positional uncertainty

Using an Exploration Bot to Map a UT Level

- UT Primitives Available: Range Sensing
  - Is a point in x,y,z space reachable from my current location?
- Sampling the map
  - Using this sensor data to crawl the walls
- Extracting cognitive primitives from the data
  - Creating a map consisting of high-level concepts from this data: walls and openings

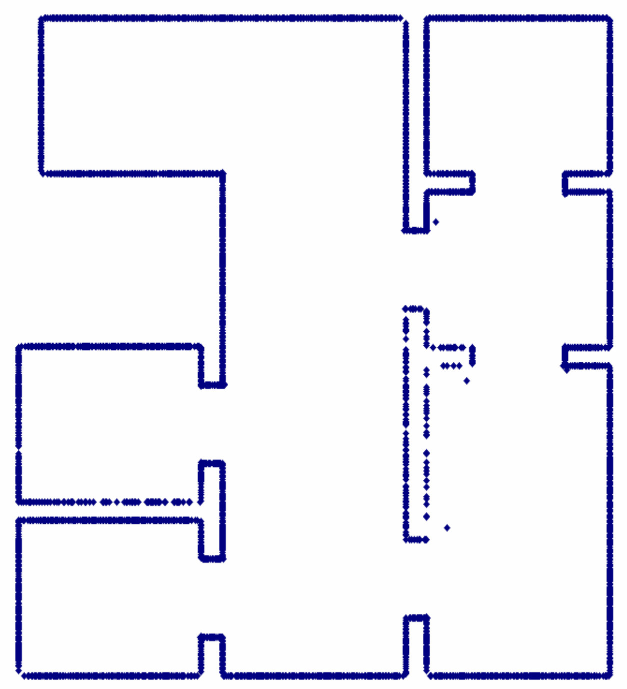
Sampling the Map

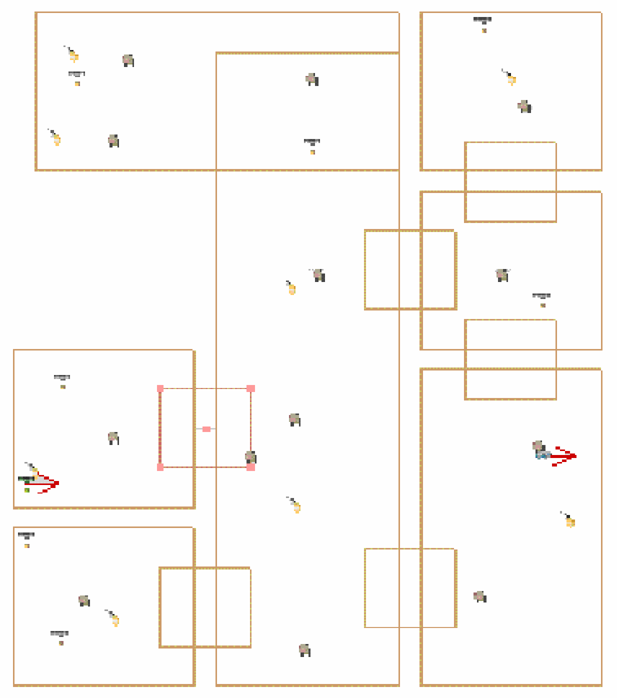


The exploration bot finds the nearest obstacle and traces its edge. It calculates grid points in the area around it, and queries the server as to whether the points are visible (i.e., is there an obstruction between the bot and the sample point).

Resulting Sampled Data

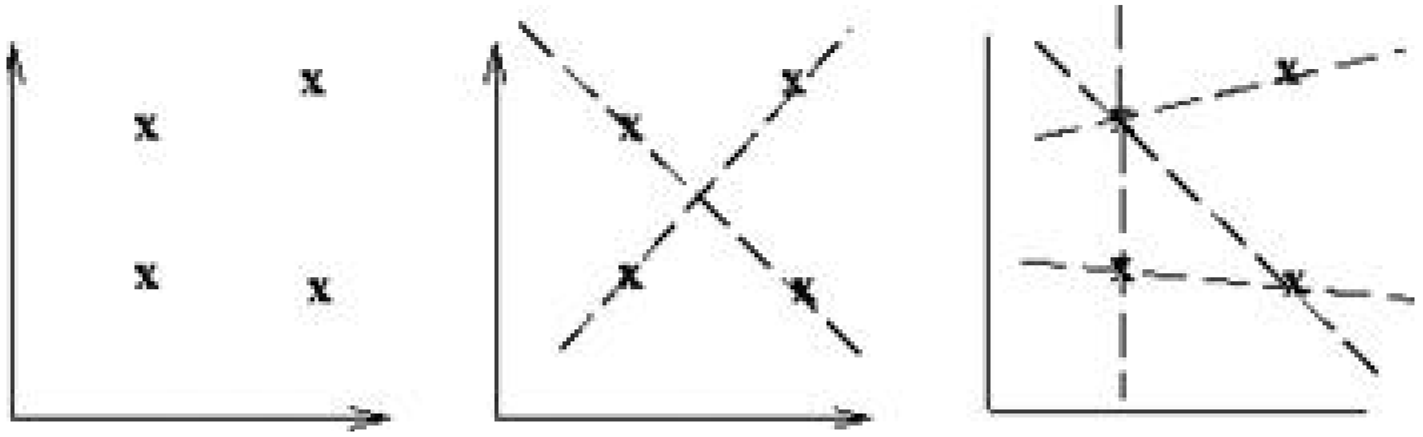
The result is a set of points representing the occupied space in a map. Note that, though we perceive lines, the computational representation is just a collection of points.





Analysis of Exploration Bot Data

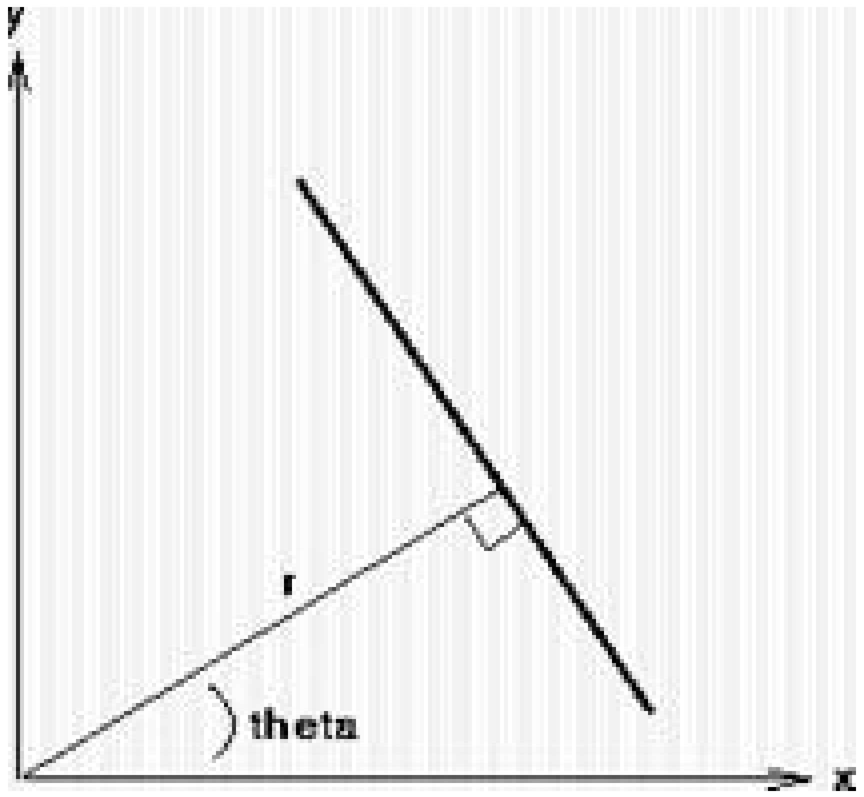
- Fits a set of lines to the sample points gathered by the exploration bot
- For any given set of points, there are a number of possible fits:



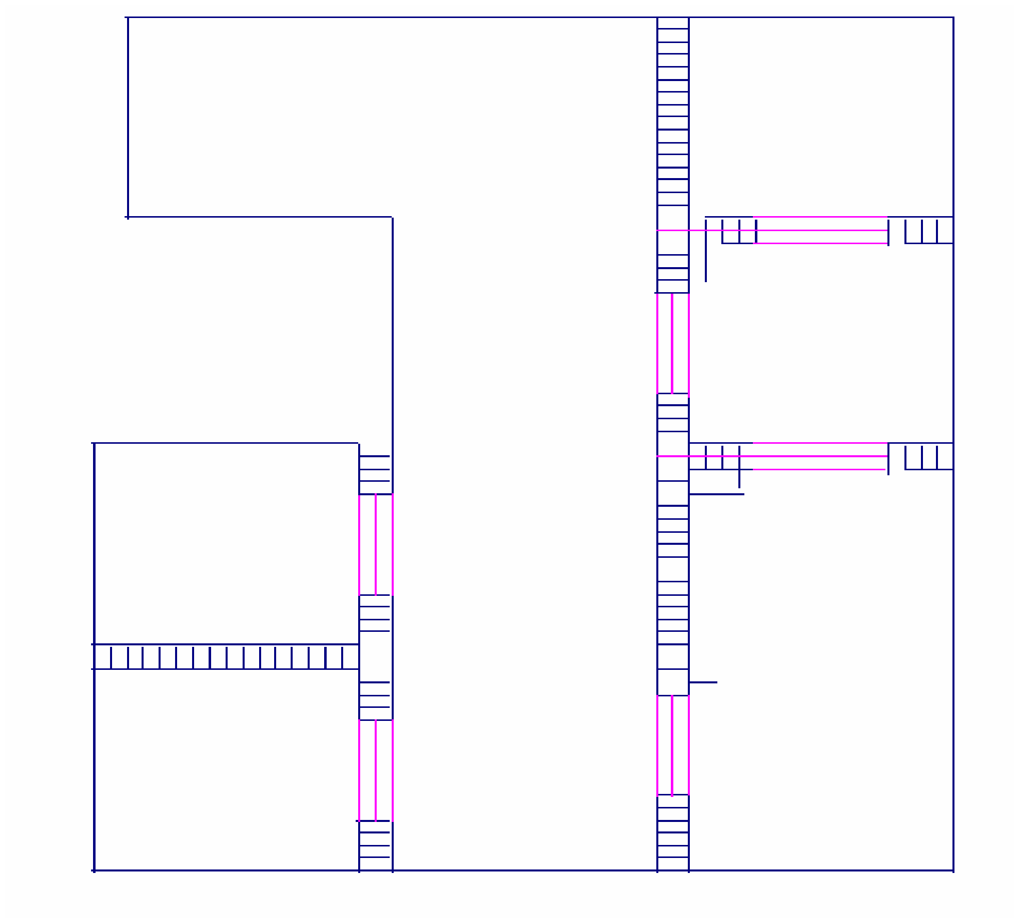
Hough Transform

A convenient equation for describing a line that fits a set of points is:

r = x (cos ?) + y (sin ?)



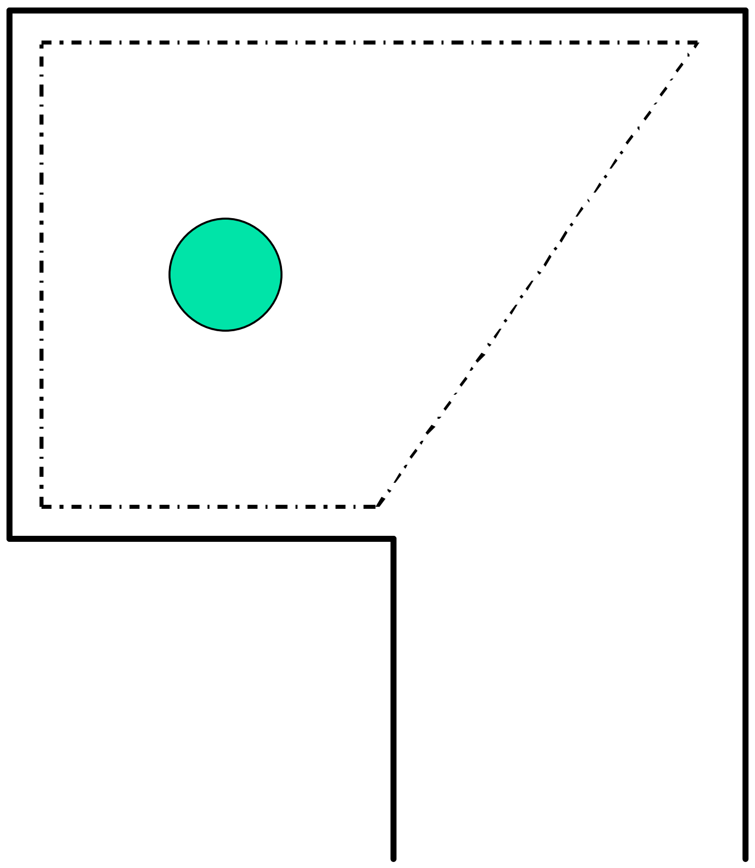
Finding the Walls



Using a Hough Transform, we can extract walls and openings from the sample point set.

This can be done for multiple floors.

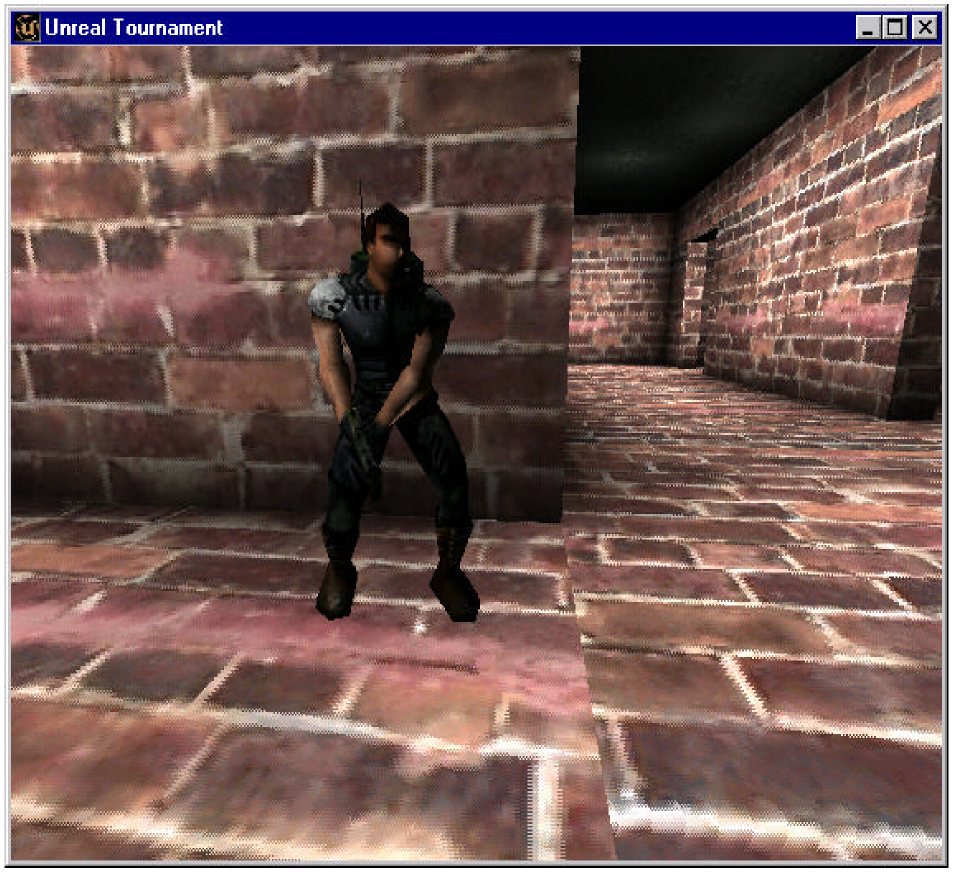
Egocentric Spatial Representation



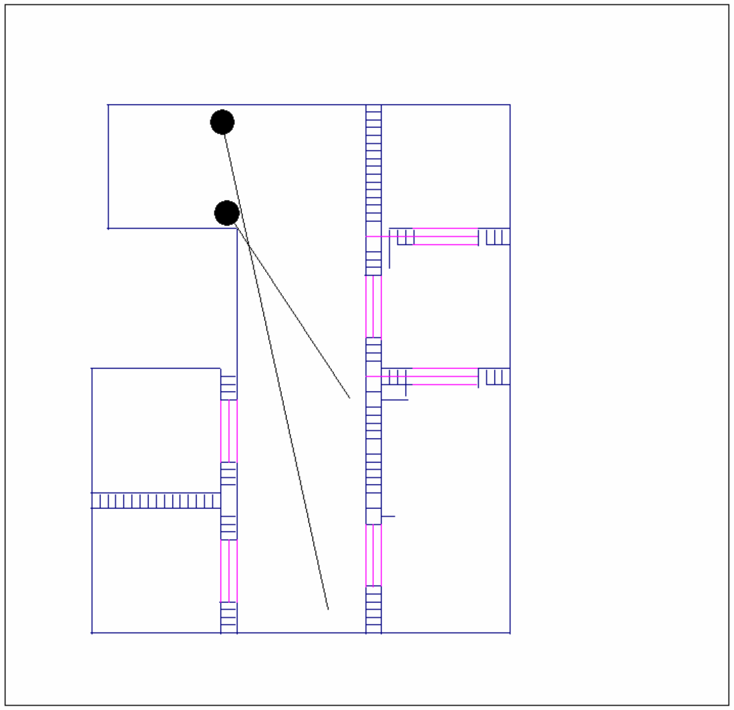
- Which of the walls and openings are visible from the current location?
- First find all walls and openings with both endpoints visible
- Connecting visible endpoints gives a bounding box representing walls and openings visible from a particular location

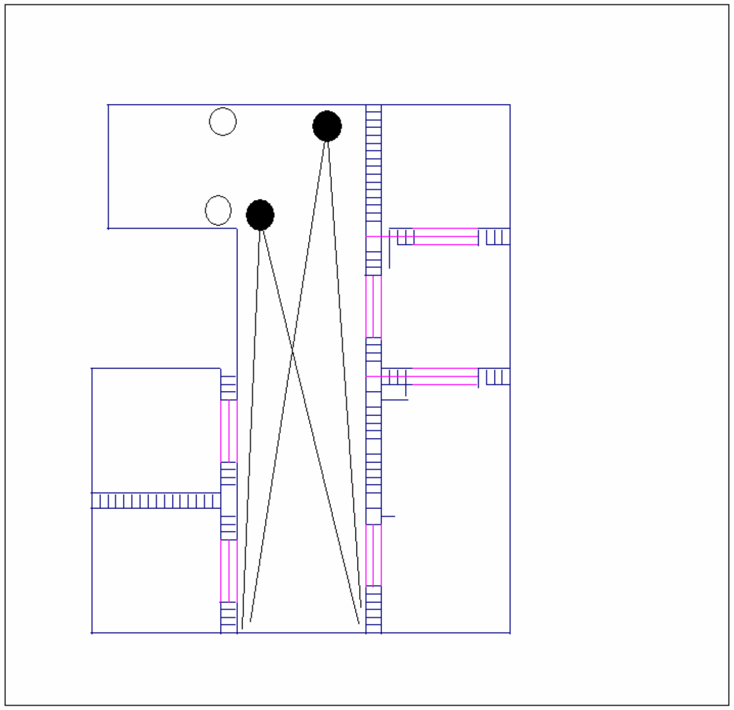
Attack Bot

- Uses MOUT tactics
  - Bots cover each other, stack at doorways, follow MOUT doctrine
- Each area is cleared and goals to clear adjoining areas are created
  - Systematically clears hallway and rooms off of it



Clearing an L-shaped Hall





Defense Bot

- Also uses spatial primitives extracted by exploration bot
- For Attackers to win, they must have superior weapons, skills, numbers, or tactics.
  - Defense bot uses less accurate shooting from hip
  - Defends when it can, runs when it can’t
  - Very reactive: simple goal structure
  - Does not effectively use tactics or teamwork

Conclusions

- UT/Infiltration provides realism and a with the Gamebots API, a convenient platform
- Separation of implementation task level from unit and functional task level allows for portability – the model is independent of the platform
- Low level perception of walls through range sensing was needed to build high level representations of architecture
- ACT-R allows for the creation of synthetic opponents that blend reactive and goal-directed behavior