ROADMAP-BASED TECHNIQUES FOR MODELING GROUP BEHAVIORS IN MULTI-AGENT SYSTEMS

A Dissertation

by

SAMUEL OSCAR RODRIGUEZ

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2012

Major Subject: Computer Science

ROADMAP-BASED TECHNIQUES FOR MODELING GROUP BEHAVIORS IN MULTI-AGENT SYSTEMS

A Dissertation

by

SAMUEL OSCAR RODRIGUEZ

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Nancy M. Amato
Committee Members,	John Keyser
	Ricardo Gutierrez-Osuna
	Ergun Akleman
Head of Department,	Duncan M. Walker

May 2012

Major Subject: Computer Science

ABSTRACT

Roadmap-Based Techniques for Modeling Group Behaviors in Multi-Agent Systems. (May 2012)

Samuel Oscar Rodriguez, B.S., Texas A&M University

Chair of Advisory Committee: Dr. Nancy M. Amato

Simulating large numbers of agents, performing complex behaviors in realistic environments is a difficult problem with applications in robotics, computer graphics and animation. A multi-agent system can be a useful tool for studying a range of situations in simulation in order to plan and train for actual events. Systems supporting such simulations can be used to study and train for emergency or disaster scenarios including search and rescue, civilian crowd control, evacuation of a building, and many other training situations.

This work describes our approach to multi-agent systems which integrates a roadmap-based approach with agent-based systems for groups of agents performing a wide range of behaviors. The system that we have developed is highly customizable and allows us to study a variety of behaviors and scenarios. The system is tunable in the kinds of agents that can exist and parameters that describe the agents. The agents can have any number of behaviors which dictate how they react throughout a simulation. Aspects that are unique to our approach to multi-agent group behavior are the environmental encoding that the agents use when navigating and the extensive usage of the roadmap in our behavioral framework. Our roadmap-based approach can be utilized to encode both basic and very complex environments which include multi-level buildings, terrains and stadiums.

In this work, we develop techniques to improve the simulation of multi-agent systems. The movement strategies we have developed can be used to validate agent movement in a simulated environment and evaluate building designs by varying portions of the environment to see the effect on pedestrian flow. The strategies we develop for searching and tracking improve the ability of agents within our roadmap-based framework to clear areas and track agents in realistic environments.

The application focus of this work is on pursuit-evasion and evacuation planning. In pursuit-evasion, one group of agents, the pursuers, attempts to find and capture another set of agents, the evaders. The evaders have a goal of avoiding the pursuers. In evacuation planning, the evacuating agents attempt to find valid paths through potentially complex environments to a safe goal location determined by their environmental knowledge. Another group of agents, the directors may attempt to guide the evacuating agents. These applications require the behaviors created to be tunable to a range of scenarios so they can reflect real-world reactions by agents. They also potentially require interaction and coordination between agents in order to improve the realism of the scenario being studied. These applications illustrate the scalability of our system in terms of the number of agents that can be supported, the kinds of realistic environments that can be handled, and behaviors that can be simulated.

To dad, you taught me the value of an education.

To mom, for always telling me to do my best.

To Pat, I think you believe in me more than I deserve.

ACKNOWLEDGMENTS

This work could not have been completed without the support of many people who have been very important to me throughout this work.

I would like to thank my advisor, Dr. Nancy M. Amato, for her continual guidance and support. This can be a difficult process and your guidance is greatly appreciated. I would like to thank you for the opportunity to prove myself. Also for the times where I could go off and hopefully come back with something that looks like good work.

I would like to thank my committee members, Dr. Ergun Akleman, Dr. Ricardo Gutierrez-Osuna and Dr. John Keyser, for their cooperation, patience and suggestions throughout this work.

I would like to thank the members of the Parasol Lab, in particular the members of the Algorithms and Applications group. Dr. Marco Morales and Dr. Jyh-Ming Lien, who were my graduate student mentors when I was just starting as a graduate student, showed me just how interesting and rewarding these research projects could be. I would also to thank some of my other collaborators, in particular, Roger Pearce, Shawna Thomas, Lydia Tapia, Aimee Vargas, Xinyu Tang, Jory Denny, Aditya Mahadevan and Takis Zourntos. This work and these projects have been fun and working with you has been a big part of what makes these projects interesting.

To my family, you mean so much to me.

I consider myself lucky to have had you all in my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION

Simulating large numbers of agents, performing complex tasks that include interacting with each other in realistic environments is a difficult problem with applications in robotics [1, 2], computer graphics and animation [3, 4, 5]. A multi-agent system can be a useful tool for studying a range of situations in simulation in order to plan and train for actual events [6, 7, 8]. Systems supporting such simulations can be used to study and train for emergency or disaster scenarios including search and rescue [6], civilian crowd control [9], evacuation of a building [10, 11] or city block [12], and many other immersive training situations. Additionally, such systems can be used to improve and validate the design of a proposed building and see the effect that obstacles and building design can have on the overall flow of agents [13, 14].

Multi-agent systems vary depending on the realism of the environments that can be studied, the diversity of the agent populations, the effectiveness of the behaviors to capture real-world reactions [15, 16], and the level at which the scenario can be tuned to match the actual scenarios being modeled [11]. Behavioral based simulations allow for someone to study the result of agents performing certain behaviors, without having to see this behavior in practice. For example, someone training for an emergency evacuation of a building scenario could test the effect of placing directors at certain locations in the environment on the overall evacuation. In a rich simulation setting, the trainee could also test the effect on the evacuation planning strategy as the level of panic of the evacuating agents varies. In a search and rescue training scenario, a group of emergency responders could test search strategies in a simulated environment

---

The journal model is *IEEE Transactions on Automatic Control.*

in order to find people in need of assistance as quickly and efficiently as possible. The search of an environment is one aspect of the pursuit-evasion problem. Ideally, the simulated behaviors would relate closely to the reactions that would occur in the real-world.

There has been significant work in simulating large numbers of agents in an environment where the agents are performing basic tasks, such as moving from one location to another [4, 5, 17]. These approaches generally have the goal of studying the local motion and planning of agents moving in a crowded setting. Many approaches also make simplifying assumptions when encoding the environment. For example, for pursuit-evasion, approaches often limit the environment to being polygonal [18] or encode it with a cell-based decomposition [19]. While the scenarios in evacuation planning can include many agents, the environments are often restricted to two-dimensions [10], simplified network graphs [20] or cellular-automata (CA)-based approaches [21]. While these simplifications can allow for more complete strategies to be developed, they restrict the realism of the environments that can be studied. In general, there is less work on many agents in realistic environments, interacting and performing complex behavior.

This dissertation describes our general approach to multi-agent systems which was developed to address a number of research needs. Our approach enables us to to handle realistic, complex environments and different scenarios within the same general framework. The framework allows for a variety of pluggable behaviors to be studied under different environmental and agent conditions and is tunable to a wide range of situations. This is important as many approaches are specifically targeted to a single scenario. We also rely on an underlying grouping of agents which is beneficial for scalability, coordination and for equipping different sets of agents with their own capabilities, behaviors and roles.

A. Research Objective and Contributions

In this work, we describe a system for multi-agent simulation that integrates a roadmap-based approach with agent-based simulations. Our motion planning inspired approach to agent simulation has extensions to simulating the agent's perception and in improving agent behavior. The main contributions of this work are:

- A roadmap-based approach for agent navigation in complex environments with distinct groupings of agents.
- Techniques to measure and study agent movement and environmental impact.
- Clearing, searching and tracking strategies for multi-level structures.
- Tunable pursuit and evasion behaviors in realistic scenarios.
- Evacuation planning strategies with coordination.

The framework we have developed that integrates roadmap-based motion planning with multi-agent systems allows us to study problems in much more complex environments than are traditionally considered (Chapter III, [22, 23, 24, 25, 26, 27]). The roadmap encodes potentially complex environments and basic environments in the same way. It consists of areas that are considered valid placements for agents, called nodes, and valid transitions between nodes, called edges. Agents can then use this roadmap when navigating through the environment by querying the roadmap for a valid path from some start location to a goal location. This general framework has allowed us to extend our work to these more interesting environments since the agents only use the paths as a guide through the environment. We also abstract the idea of individual agents to groupings where agents are in fact a group and can belong to distinct groups within the simulation. An individual agent is then a group with no subgroups. This concept is useful when considering arbitrary hierarchies of groupings with distinct sets of capabilities, behaviors, roles and knowledge of the environment within each grouping. It is also important for scalability and coordination within a group. This general idea of a group also allows for behaviors to be developed for a group which is then applicable to a single agent or groups of agents with similar goals.

We have developed metrics to evaluate our agent movement model (Chapter III, Section C). This is done by relating agent density to speed and flow of agents moving through certain areas. One motivation of this is to show that our model of agent movement and navigation correlates to traditional motion principles. We also use this model to show how changes of the environment can effect the flow of pedestrians exhibiting an evacuation behavior. These motion metrics could be used when directing agents to improve flow or to evaluate evacuation.

Within our roadmap-based framework we have been able to develop a number of techniques for searching, clearing and tracking in multi-level structured environments. We have also been able to extend the study of pursuit-evasion by proposing an agent tracking model in these complex environments where agents can track other agents that have left their field-of-view. Using the roadmap, an agent can determine good potential locations that other agents of interest may have moved towards (Chapter IV, [24]). While many of our search behaviors are heuristic in nature, we also have a set of clearing behaviors which are inspired from more exact forms of the clearing problem (Chapter IV, [25]). We have used these search behaviors in studying the clearing of a building.

In pursuit-evasion, the complex scenarios that we are able to handle include complex environments with multiple levels (e.g., multi-story buildings), terrains, and crowds of agents (Chapter V, [22, 23, 24]). Our behavioral framework allows us to equip agents with different search, pursuit and evasion behaviors and study types of behaviors, agent populations and compare the effect of these parameters. This is enabled through the roadmap-based approach and the pluggable behavior framework with groupings of agents.

In evacuation planning, we are able to handle much more complex environments than are typically studied. We have taken simple evacuation behaviors that are applicable in basic environments and, using the same framework, have applied the same behaviors to a roadmap representing more complex environments (Chapter VI, [26, 27]). One aspect that is quite unique to our approach is that we allow one group of agents, the directors, to guide the agents that are attempting to evacuate. We also study how each group of agents can cooperate in order to improve the overall evacuation [27]. This cooperation can be tested between evacuating/directing agents, evacuating/evacuating agents and directing/directing agents. We also look at parameters that result in a more realistic evacuation scenario.

B. Outline

This dissertation describes our roadmap-based multi-agent system. In Chapter II, work that is related to our system and the application focuses we consider are described. This work includes various multi-agent systems that have been proposed, path planning techniques for complex environments and work in pursuit-evasion and evacuation planning. An overall description of the system is given in Chapter III, which also includes metrics for agent movement in complex environments and the use of them for system validation and for studying how the structure of the environment can impact pedestrian flow. Search techniques and an agent tracking model are described which utilize the underlying roadmap-based framework in Chapter IV. In Chapter V, we present our work in pursuit-evasion with a focus on our roadmap-based approach and the various benefits it can provide as an encoding of the environment. Chapter VI describes our work in evacuation planning which includes our initial approach in two dimensional environments and the extended work which is in complex multi-level environments with cooperation between evacuating agents and directing agents.

CHAPTER II

PRELIMINARIES AND RELATED WORK

In this chapter, we discuss relevant related work on multi-agent group behavior. Work related to randomized planning is presented because it is critical to our system. Finally, work related to the main application focus of this work, pursuit-evasion and evacuation, is described.

A. Multi-Agent Systems

A number of systems to simulate agents in virtual environments have been proposed. In fact, the number of systems have been proposed is so extensive, only a few of them that are most relevant to this research are described here.

In [28], steering behaviors are described for autonomous characters. This work is similar to [29] where basic flocking behaviors are proposed but it integrates the ability to follow paths using local path information. These flocking techniques have been shown to add interesting and complex movements to a group of agents utilizing only those basic behaviors.

In [30], the problem of agents moving as a group while avoiding collision is studied. This problem is made more difficult because the agents may have complex dynamics. Examples of complex dynamics are one legged agents or bicyclists. Agents performing the group behavior try to pick their next position by predicting the positions of the agents they can perceive. Results are shown for agents performing certain maneuvers including steady-state movement, turning and obstacle avoidance. The complexity of the dynamics is what could cause the maneuvers performed to be difficult to execute. Results show how under varying circumstances, such as changing the visibility range, the agents perform the maneuvers with varying effectiveness.

Several following behaviors have been proposed which are often useful in crowd simulation. In [31] virtual crowds consist of leaders and many followers. The leader is responsible for generating its own motion while the followers use flocking principles for realistic motion while trying to follow the leader. A similar approach is presented in [32] where reactive path following is accomplished by using user data. The user data provided could then be used in similar situations.

An interesting approach to agent behavior simulation is presented in [33]. The cognitive modeling described in this approach controls what a character in the scene knows, how the knowledge was acquired, and how this information can be used to plan the agent's actions. One of the goals of this work was to allow the animator to specify an outline of a behavior which would allow the character to be directed to do something. The cognitive modeling language proposed would allow an animator to sketch the behavior that is desired. The goal was to be able to specify a behavior which would require complex actions in order to adequately perform the behavior.

A virtual environment is presented in [34] which uses a hierarchal representation of the environment for simulating pedestrian movement. The hierarchical representation is included for the topology of the environment, the perceptual model and for representing path maps for the agents. The topological map includes connections between different areas in the environment which could be useful for complex environments. The perception map supports queries for stationary and mobile agents. The path maps used are implemented with a grid or quad-tree which allows the agents to find paths through the environment. The agents are equipped with basic reactive behaviors while navigating. An interesting aspect of this work is that the pedestrians consider passage way navigation and passage way selection. Other interesting behaviors that can be simulated with this approach include selecting and sitting in an unoccupied seat, noticing a performance, a group of pedestrians talking, and forming lines at vending machines or ticket booths. Results are shown for a virtual Penn Station, a train station that has since been demolished.

In [7], a decision network is used to model various behaviors. Using this decision network, they want to enable an agent to make rational decisions about what the agent wants and what it believes. Probability distributions are used to describe different variables that are of interest to an agent performing a given behavior. By adjusting the values associated with different variables of interest, it is possible to change how the agent makes a decision. An emergency response behavior is described in which agents may respond to a situation in a variety of ways including watching the scene, calling for help, or leaving the scene after a certain amount of time. They also describe some interesting behaviors that they can handle which include characters encountering one another and being acquaintances or a partnering behavior where an agent may try to catch up to another agent and try to partner with that agent.

Crowds are used in [3] to simulate the ancient Greek agora. Some of the behaviors that are simulated are small groups of agents discussing or negotiating in the town square. Another group of behaviors looked at are flow and jamming of interacting moving crowds. Some simple and intuitive optimizations are proposed such as modeling obstacles as disks and simplified interactions between nearby agents.

A physically-based crowd simulation technique is proposed in [4] where optimal paths for characters are computed using potential functions. The potential functions are generated from an underlying grid which also uses information about crowd density. Results are shown for two groups of agents with up to ten thousand agents running at interactive speeds.

Motivation for search and rescue applications in large-scale disaster situations is presented in [6]. As well as giving examples and situations where search and rescue can be applied, they describe different disasters that can be simulated. They also describe types of planning challenges that exist, such as real-time constraints, heterogeneous agents, scenario-based and resource-bounded planning with data collection.

Crowd simulation is done in [9] where a crowd can be controlled by allowing police agents to block roads for some period of time. This paper shows a large number of civilian agents being simulated in a rescue environment. The work assumes that the agents in the crowd that are in one location will exhibit some common psychological influences.

In [8], a training scenario at a military checkpoint is the setting of the virtual environment. This kind of virtual environment allows a trainee to be immersed in a world where other agents may be performing a range of behaviors that the trainee must be able to respond to. This kind of virtual reality environment also motivates the need for other simulation-based training techniques. They also discussed many of the implementation details of their project as far as the equipment they are using.

The need to accurately model and consider the crowd in military operations is described in [35]. The project described was broken up into two phases. The first phase focused on the preparatory work researching crowd modeling which included identifying the requirements for modeling a crowd, researching the existing psychological research that might work with the kind of crowd they are interested in, and looking at things they should be aware of when designing a crowd simulation. Other details they described about their approach include how their crowd is organized, the cognitive model they may use and some target scenarios they are interested in studying. In [36], some progress that has been made in their approach was described where members of a noncombatant crowd are modeled in their simulation framework. Some interesting group based techniques include using flocking techniques for members of this kind of crowd. The goal in this technique was to be able to model a crowd for use in their real-time tactical training applications.

A trait-based personality approach to decision making is studied in [37]. In this paper, the goal is to simulate an automated commander that makes decisions that are somewhat based on the agents personality traits. Some of the traits considered include stability, anxiety, and independence. They wanted to be able to vary these traits in agents to see how the effect on how they make decisions in a stressful situation. Some interesting scenarios are considered where a commander must route supply trucks through potentially hazardous areas in order to quickly get relief vehicles to a destination. The also described how their automated commander was given the ability to make decisions in terms of making routing decisions and the routing algorithms that would be used. The personality model they use is essential in being able to model a commander that is making these routing decisions under stress.

The validity of a given human behavioral model is studied in [38]. The kind of crowd that they were interested in studying in this paper consisted of many groups of agents with each group having the following behaviors: protesters, agitators, casual members and bystanders. They wanted to study things such as the group aggression levels and the overall crowd aggression levels in order to determine if appropriate behaviors are given to the crowd. They also looked at some visualization techniques of the crowd levels to see if they made sense in the simulations they were interested in achieving.

In [39] the traditional tools that were used when studying urban simulation models and some problems they had were described. Cellular automata and multi-agent systems are considered the new direction when simulating these urban scenes in planning support systems. The multi-agent systems are considered a good approach because of their applicability to mobile entities such as vehicles or agents.

The study done in [40], explores behaviors behind groups of children that are playing. They look at the relationships that exist between children at play and how geo-spatial characteristics affect their movement through the environment. This is interesting in that it studies the interactions between smaller groups of agents.

A hybrid approach to crowd simulations is given in [41] where the system can represent very dense crowds while maintaining some individualities of agents. They utilize a Lagrangian representation of individuals along with a much coarser model for the crowd. This approach is specifically suited for very dense crowds because individual agents have very constrained movement due to other nearby agents. Instead of representing the crowd as individual agents, it is represented as a continuum fluid which has an associated density and flow.

Another approach to studying dense crowds is described at the virtual Tawaf site in [42]. The main components of this work include their simulation of very dense crowds with heterogeneous population characteristics, varying velocities and varying objectives for each agent. An overall finite state machine of the agent's objectives is presented which includes moving to important areas, in certain motions, performing actions or praying. They are able to show some interesting analysis of the agent speed and density at each area in the environment for very large size populations.

An approach to plan local paths that avoid neighboring agents is presented in [5]. While planning, each agent takes into account the neighboring agents and uses their velocity to create a velocity obstacle which the agent has to try to avoid. The approach is decentralized in that at each planning step the agent uses its local information to determine how to plan locally. This work is able to handle very large numbers of agents moving near one another.

The work on reciprocal velocity obstacles [5] was formally extended in [43] to show how agents can independently select their optimal velocities to allow them to remain collision free for a given amount of time. This assumes that all agents use the same collision avoidance scheme and can sense other nearby agents. They are able to show agents choosing an optimal velocity using a linear programming approach. Results show that their agents can navigate near one another and they show results of parallel computation as each agent acts independently.

Velocity obstacles are also used in the computation of valid trajectories for agents in [44]. In this work, additional constraints are imposed in order to guarantee collision avoidance within some pre-defined time interval by truncating the traditional velocity obstacle cone. This approach is a quadratic optimization problem and highly parallel, as shown in the results with ten thousand agents simulated on multiple cores at interactive speeds.

In [17], an approach to improving pedestrian movement in constrained spaces is proposed. This approach attempts to compute energy-efficient trajectories for agents. These trajectories are both short paths through the environment but also avoid congestion when needed. The work is based on an optimization function that all agents share and use when generating trajectories with an attempt to minimize the amount of energy used by the agents. They also use the idea of a dynamic energy roadmap which is a shared structure that allows agents to share information about congested areas.

Another effort that looks at avoiding neighboring agents when moving near one another is described in [45]. This work also uses the idea of agents working with one another implicitly while trying to avoid collision. They described new parameters to model human characteristics such as reaction time and biomechanical limitations. A good model of the agent's physical space and personal space is described, where most models only plan for an agent's physical body. They validated their model with data from actual humans walking in an enclosed area.

In [46], an approach to crowd simulation is presented. The approach claims to be scalable in the behavioral sense in that more complex crowd behaviors can be generated without increasing the complexity of a given agent. A framework for authorable behaviors is also described in such a way that the action an agent performs is dependent on the situation it encounters and available actions it has.

Shepherding behaviors are a type of flocking behavior in which outside agents guide or control members of a flock. Shepherding behaviors can be found in various forms in nature. In [47, 48], an advanced techniques for the shepherds to effectively control the flock and simulate these various types of behaviors including herding, covering, patrolling and collecting. The agents in the flock were given basic flocking techniques and were afraid of the shepherds, and so would react to the shepherds' movement and try to avoid the shepherds.

Another work that attempts to control a group of agents is presented in [49]. In this work, controller agents guide a flock from some start location in the environment to a goal location. Different motion planning strategies are presented to plan the paths for agents. Results are presented in 2D environments with different planning strategies.

Shepherding using deformable shapes is presented in [50]. This work attempts to place shepherds strategically around a shape that represents the flock that is being steered. While this work is similar to that of [47, 48, 49], the flock representation makes for selecting steering points and motion strategies more effective.

B. Roadmap-Based Planning for Autonomous Agents

In [51, 52, 53], the benefits of integrating roadmap-based path planning techniques with flocking techniques were explored. The work extends ideas from cognitive modeling [33], and embeds behavior rules in individual flock members and in the nodes and edges of the roadmap. The global information provided by the rule–based roadmaps improved the behavior of autonomous characters, and in particular, enabled more sophisticated group behavior than are possible using traditional (local) flocking methods [29].

Some key features of integrating roadmaps with basic group behavior include:

- The roadmap provides a convenient abstract representation of global information in complex environments.
- Adaptive roadmaps (e.g., modifying node and edge weights) enable communication between agents.
- Associating rules with roadmap nodes and edges enables local customization of behaviors.

The approach we use also utilizes a roadmap encoding representative feasible paths in the environment. While noting that our techniques could use any roadmap, our current implementation is based on the probabilistic roadmap (PRM) approach to motion planning [54]. PRMs work by sampling points 'randomly' from the robot's configuration space (C-space), and retaining those that satisfy certain feasibility requirements (e.g., they must correspond to collision-free configurations of the robot). These points are then connected to form a graph, or roadmap, using some simple planning method to connect 'nearby' points. During query processing, the start and goal are connected to the roadmap and a path connecting their connection points is extracted from the roadmap using standard graph search techniques. Enhancements have been proposed to the PRM approach which improve where and how sampling is done – for example near obstacle boundaries [55] or near the medial axis of the environment [56]. Roadmaps constructed using these enhancements can also help autonomous agents in improving their navigation abilities.

Similar to PRMs for path-planning, the corridor map method (CMM) has been proposed which allows agents to generate short, smooth paths through the environment. This method consists of an off-line map building phase after which the map can be queried as needed, similar to PRMs. The main difference between PRMs and CMM is that the edges represent corridors through a static environment and have an associated clearance value. An agent can then use the corridor path generated when applying forces to guide itself to the goal location. This method was extended in [57] to generate paths that also reduce their exposure in an attempt to remain hidden from observers. Visibility values are associated with grid-cells which are used when computing the paths. The corridor map method was further extended in [58] to generate paths with an explicitly defined amount of clearance. As in the previous method, the paths generated are smooth and short. These are interesting path planning methods which allow agents to generate more interesting paths given their environment.

A randomized tree-based path planner was developed in [59, 60] that is useful for exploring C-space. This path planning technique is known as the Rapidly-Exploring Random Tree (RRT). A similar tree-based planner, Expansive Space Tree (EST), was developed in [61]. These tree-based methods work well for single query problems by only exploring the relevant portions of the configuration space needed to solve a given query. These could lead to interesting alternatives to our purely roadmap–based approach.

There has been a great deal of work that deals with planning around other agents and in dynamic environments. While the majority of our work focuses on complex 3D environments, more dynamic planning will be essential in the near future in our work.

When planning a path for an agent with restrictions on its movement and potential restrictions on allowable motion inputs at a given time step, kinodynamic planning is often done to find a path. In [62, 63], a randomized kinodynamic approach is presented to planning a path in a static environment. The state space is explored by applying a set of allowable control inputs in order to grow a tree. The exploring of the state space is complete when the goal configuration can be reached and a path in the tree can be extracted.

In [64], an RRT-based planner is proposed which performs better over time, improving paths, over a number of runs, in a given environment. In this work, an RRT is integrated with a “way-point” cache for improved performance in static environments. It is not described how these way-points are generated or how the way-points could be updated in a dynamic environment.

In [65], a modified expansive space tree approach to planning a path for a robot with motion constraints is proposed to search the space. The goal is to find a path that has a low cost and has a relatively straight path.

An early approach to kinodynamic planning among moving obstacles was proposed in [66]. An EST approach is used to plan a path for a circular robot through an environment consisting of circular obstacles with restrictions on the velocity and shape of the obstacles. In this approach, when uncertainty in the environment is found, a path is completely replanned.

A PRM-based approach is proposed for planning paths among moving objects in [67]. A roadmap is built and updated according to the movements of the obstacles. In this work, they attempt to update only the necessary portions of the roadmap, that are relevant to the moving obstacles. If simple repairs cannot be made in the roadmap, then an RRT is used to repair the roadmap. Complete paths for the robot are obtained from the roadmap and a path to the goal is always required.

Another PRM-based approach to planning a path in a dynamic environment is proposed in [68]. The roadmap used during the planning phase is one that is constructed as a preprocessing step and does not change as the environment changes. This alone could cause many problems in environments that are constantly changing. Local trajectories are then planned along valid portions of the roadmap from the start to goal configuration. This approach also assumes the dynamics of the moving obstacles are known a priori. Although this may work in some situations, in highly dynamic environments, this could result in unnecessary computation when constantly replanning from the start to goal configuration and when the dynamics of the obstacles are not known beforehand.

In [69], the problem of planning a safe path for a robot disk among unpredictably moving obstacles is proposed. The obstacles are modeled as disks growing over time given their known maximum velocities, modeling unpredictable motion. Although this would model unpredictable motion, given the restrictions on the shape and velocities of the robot and obstacles, many situations could occur where planning a path to the goal is unfeasible. Some of these situations include having a large number of moving obstacles or having some obstacles with large maximum velocities.

A planner for coordinating between multiple vehicles is presented in [2]. While considering communication, the planner also avoids collision between neighboring vehicles and obstacles. The planning and replanning is done online and considering processing constraints. The planning attempts to avoid Inevitable Collision States, proposed in [70] and further developed in [71]. A similar greedy and safe kinodynamic planning technique is presented in [72] for a single robot. The algorithm is biased toward unreached frontiers to improve exploration and is probabilistically complete.

An algorithm for moving a large number of agents in dynamic environments is described in [73, 74]. In this framework, the roadmap used is adaptively updated to deform as the environment changes. This is done using a physically-based agent dynamics simulator. They discuss how the roadmap is updated in this framework and how navigation is done using the local dynamics model described.

Similar to the roadmap we use in our multi-level structures, a navigation mesh is proposed for multi-layered environments in [75]. Their work proposes generating a map for each layer along the medial axis and then combine the map for each layer to result in a navigation mesh which agents can use when moving through the environment. While we use a similar underlying graph for agents, we do not currently sample along the medial axis on our surfaces.

Path planning for groups of agents in proposed in [76]. The method is based on network flow within the graph data structure representing the environment. Integer linear programming is used to formulate the problem and the set of variables considered is restricted to make the problem tractable. This work allows agents navigating through the environment to separate from each other in order to reach a goal location.

Our primary utilization of the roadmap is to generate paths between an agent and its destination. We then generate forces between the agent and the successive nodes in this path until it reaches its destination. The roadmap is also used to coordinate the actions of multiple agents. We also use the roadmap to store information about the environment, and to facilitate communication between agents. Agents that pass a node in the roadmap can store information in that node. This information can be anything from observations about the local environment to instructions for other agents. Subsequent agents that pass this node can access this information and take the appropriate action based on this information. This method of communication is analogous to the way animals mark regions with their scent or the way insects leave behind pheromones to communicate information to other insects. Agents can also perform searches on the roadmap in order to determine what actions they should take. The roadmap can also be used to associate agents with regions in the roadmap.

Agents can be assigned to a region of the environment by being assigned to nodes within that region. One of the main contributions of this work is exploring the various usages of the underlying roadmap.

C. Evacuation Planning

Studying evacuation is a key aspect of this work. There has been much work addressing different aspects of this problem. A survey of the main approaches that have been taken to studying the evacuation problem is presented in [77]. Four main approaches were described: flow-based, cellular automata, agent-based and activity-based models. The focus of this work and the majority of the related work presented here is on agent-based models since this allows us to have varying agent capabilities and populations.

An early work in agent-based evacuation is described in [78]. The goal of this system was to be able to handle thousands of individuals escaping in large, geometrically complex environments. The system allowed for automatic input of building plans and discretized the area into grid cells, assigned weights based on if a grid cell is a solid object, open space or a final destination. The underlying grid was used to compute a maximum travel distance. This work presented some interesting results but focused only on individual agent evacuation.

Some grid-based approaches have been proposed to model large scale evacuation in multi-level environments. A fluid-dynamics approach is presented in [79] to describe the trajectories of agents while moving through a building in the presence of hazards. It supports the analysis of the environmental usage of stairwells. Cellular automata approaches are often limited by having a single agent occupy a grid cell and often force oversimplification of the environments.

A multi-agent based approach, described in [13], handles complex multi-level buildings which includes stairwells with agents having the ability to have their own characteristics. This tool is feature rich and has been used in the design process of buildings and in ensuring safety codes are met. While this is a commercial tool, it is the type of use we envision for with our work.

Actual crowds are studied under different conditions in [16] to see how a crowd reacts under those conditions to study human movement. Human studies are done to see phenomenon such as counter-flows, bottlenecks or intersection flows. As well as setting up situations to study these phenomenon, the authors are able to observe simulation results of crowd interactions like stripe formation in congested areas. An earlier work modeled different levels of panic seen in evacuation [15] making it possible to study evacuation from a room or area with only one small exit.

The way individual characteristics impact evacuation efficiency is studied in [80, 81]. This work is based on the social forces model developed in [15, 16] but includes other agent characteristics which include group structure, dependence level of agent, altruism level and the desired speed of agents. The impact of the individualities in the resulting average flow of people is done when varying the agent properties. They claim that some settings of these parameters can simulate trained people. This is one of the few approaches that describes the importance of grouping in these simulations, although their approach to using grouping is not actually described.

Another survey, focused on virtual crowds, is presented in [82]. The work describes many approaches that have been proposed for crowd simulation such as the types of models that have been developed with an in-depth look at a few different models. The idea of different levels of agent knowledge and planning ability is considered in [10]. This is in part due to psychology studies which show that building occupants usually decide to use familiar exits, such as where they entered the building. In some cases, emergency exits or exits not normally used for circulation are often ignored. Different agent types are considered including trained leaders, untrained leaders and followers. Their system is given a 2D maze-like environments with a defined number of exits, a number of hazards, and the parameters necessary for the simulation such as number of agents, percentage of trained agents, and the percentage of leaders. In this work, communication is considered to share locations of hazards and portions of the building that have already been explored. One interesting result they are able to find is the optimal number of leader agents. They are also able to see differences in evacuation when changing the population type.

In [11], a system is developed for simulating the local motion and global way finding behaviors of crowds moving in a natural manner within dynamically changing virtual environments. They are able to simulate patient and impatient agents and pushing between agents. Improvements on previous work, [15], were made by considering factors that reduce shaking and vibration caused by applying social forces in densely crowded areas. Other interesting factors of this work include avoiding fallen agents, wall avoidance, focusing on regions ahead of the agent, an organizing behavior and pushing between agents. They also consider the problem of avoiding bottlenecks to pick better routes. They do not consider groupings between agents or direction that can be provided to assist in evacuation.

In [12], an approach is described on how to adapt a multi-agent, transportation simulation framework to a large-scale pedestrian evacuation simulation. This work uses an underlying flow model which considers free speed and flow capacities of edges in an attempt to scale to hundreds of thousands of agents. This is also one work that focuses on one area, in this case a dam break that could affect Zurich. This approach also considers the idea of replanning and learning better plans based on an iterative technique.

Phases and behaviors in evacuation are considered in [83] with some interesting ideas presented. The total evacuation time, which is generally considered is broken down into pre-evacuation time, moving time and waiting time with pre-evacuation time often taking a majority of the time. Human factors are also considered which include panic agents might exhibit, wayfinding that needs to occur and some agents that ignore the need to exit immediately. Results are described in a multi-story building with evacuation times broken up as described. Wayfinding is done with and without obstacles present although the actual simulation is only done on a graph.

An approach to find the optimal evacuation time in simple 2D environments is described in [84] where k occupants with n possible exits use an evacuation function to select routes. This approach makes many assumptions about the environment, such as assuming a uniform distribution of agents, the existence of a route to each exit and not considering the fact that valid paths may change as the environment changes. While this is an interesting approach they made many assumptions that limit the scope where this can be applied. The idea of analyzing the flow at each exit is something we use in our evacuation profiles. As in this work, we consider the idea that minimum evacuation time may not always lead to the best usage of the environment.

In [85], multi-attribute utility theory is described to deal with evacuation scenarios where objectives may, at times, conflict and there exists high-levels of uncertainty. Benefits of multi-attribute utility theory include being able to explicitly list objectives for decision makers, the ability to appraise evacuation policies and building scenarios for training purposed. The evacuation considered is on a larger scale, for example of entire cities, and may occur over many hours.

One of the main benefits of agent-based systems is that heterogeneous agent populations can be created to study evacuation. This was done varying the dependence and altruism level of agents [80], varying agent knowledge and training information [10], including patient and impatient agents [11]. This is one reason we have chosen to develop an agent-based system.

Other physical factors are considered in [78], but the ability to include social factors is also described. There are also known evacuation scenarios where agents have vastly different traveling speeds which includes people with disabilities [86] who may require evacuation in groups. Another work that describes the need to consider grouping in evacuation is [87], where depending on the population type, agents may be either individuals or be considered familial groups which may contain small children.

Pedestrians evacuating a large stadium are shown in [20]. These agents operate on a simplified network graph of the stadium and generally follow the first-in first-out principle where agents navigating in a constrained environment are not likely to be able to overtake agents ahead of them. They describe their usage of the link usage when agents are deciding on the route to take. Metrics are used to describe the evacuation which include total time to evacuate, length traveled, and waiting time. They show space-time paths for evacuating agents which are the two dimensional paths the agents take through the simple network of links with the time component showing the evacuation time. While this work is very interesting given the complexity of the environment, the abstraction of the problem to the simple graph may lose too much information about the environment.

Looking at the crowd-mind as a thing that changes when a large group gathers is described in [88]. In this work, the author extensively describes aspects of a crowd that made individuals react differently than they might normally react. Some interesting aspects of the crowd behavior include de-individuation, contagion and suggestibility. The cognitive behavior described in this work is based on emotions, beliefs and actions which are aspects of multi-agent systems that are usually more difficult to simulate.

While the simulation results presented are not necessarily impressive, the ideas that can be taken from simulating the crowd as a whole with some of the characteristics described can enrich a simulation.

A cellular automaton-based model is proposed for evacuation where obstacles may be present [21]. In their model they include a panic parameter, which causes an agent to not move, and include some random movement to reduce the likelihood of a conflict during the update process. They also consider two types of doors, where either one agent can pass at a time or where multiple agents can pass. The evacuation is done in very simplified environments due to the representation and the behavior is also simple, where agents move to the nearest exit.

Risk assessment in the presence of fire is considered in [89]. Whereas probabilistic methods are sometimes used in this type of assessment, this work uses the idea of fuzzy risk assessment. Event trees are built and used to analyze potential fire scenarios and the consequences of what can happen when failures occur with the calculation of scenario occurrences and expected casualty numbers in fire scenarios.

There is a need for a flexible framework to be able to handle a number of situations. Many different approaches have been proposed to handle specialized environments including pedestrians moving through Penn Station [34], underground malls [90] and a passenger ship [87]. Evacuation caused by a dam breaking near Zurich were considered in [12]. Evacuation simulations have been used to help in the design and permit application process at Linz Central Station [14].

Evacuation in high-rise buildings is shown in [91] using a cellular automata approach. The environment is discretized into grid-cells of free space or blocked space and agents select evacuation routes based on finding a path through unoccupied cells. Limitations of this approach are described as being factors such as grid size, ignoring fatigue factors, route selection through the underlying grid, uneven usage of stairwells, and their difficulty in simulating stairwells using this approach.

An overview of pedestrian movement and measures that can be taken for pedestrian movement is described in [92]. This report describes terms that are important when considering pedestrian motion, relationships between speed-density, flow-density speed-flow and speed-space of a moving crowd. Parameters associated with different levels of service is also given which is dependent on the density of the crowd.

This all points to the need for a flexible system that can be used in many situations with the ability to simulate a number of different conditions. The system required will also need the ability to easily vary the agent population types, agent capabilities and cooperation level.

D. Pursuit-Evasion

In this section we describe work that is relevant to the pursuit evasion problem that we consider. A classical, purely graph-based approach [93] looks at the problem of pursuit-evasion on a graph in which agents move on edges between nodes in the graph and where the pursuer has the goal of occupying the same node as the evader. The problem has been looked at in polygonal environments [18], with much interest in finding exact bounds on the number of pursuers needed [94]. An approach based on exploring undiscovered portions of the frontier is described in [95].

An overview of the art gallery problem is presented in [96] which includes solutions, bounds and variants. In the most basic form of this problem, the goal is to cover a simple polygon with as few guards as possible. The polygon is considered covered if every point in the polygon lies within the visibility range of one of the guards. A discussion of the 3D version of the same problem (guarding a polyhedron) is included. An experimental approach to finding guard locations is presented in [97] where the goal is to heuristically find guard locations. A candidate set of potential guard locations is generated in a number of ways. The next step is the selection of guard locations from the set of all guard locations. In our approach, we use a similar greedy strategy for selecting guard locations from the selections of nodes at the current level being searched. Our approach also has to account for pathways between levels to guarantee clearing.

Roadmap-based pursuit-evasion, where the pursuer and evader share a roadmap and play different versions of the pursuit-evasion game is described in [1]. These games include determining: 1) if the pursuer can eventually collide with the evader, 2) if the pursuer can collide with the evader before the evader reaches a goal location, and 3) if the pursuer can collide with the evader before the evader collides with the pursuer in a dog fight scenario. In this work, a wider range of capture conditions are supported than in most previous approaches.

A number of other forms of the problem have been considered. A graph-based approach to the pursuit-evasion problem is given in [98] where agents use either blocking or sweeping actions by teams of robots to detect all intruders in the environment. Large teams of robots have been considered to detect intruders with communication allowed between other agents within some range and with no map of the environment [99]. One form of the problem [100] involves a team of agents attempting to maximize the average number of targets that are observed by at least one of the team members. Agents deal with limited sensing information by building gap navigation trees with gap sensors in [101]. The idea of using probabilistic shadow information spaces for targets which move out of a pursuing agent's field-of-view has also been considered [102]. Other forms of collaboration between searching agents has been considered where agents utilize frontier information [103] with communication to switch between roles [19].

The level of visibility between agents plays a role in what the agents detect in the environment and what kinds of environments can be handled. The problem of restricting or limiting the amount of visibility is considered in [104] with the field of view being variable in [105]. In [106] a team of agents with limited sensory abilities attempts to capture a single evading agent using sweep formations. Teams of robots with different kinds of vehicles and sensing information is looked at in [107].

The problem of pursuit-evasion on height maps, which restrict agent visibility, is described in [108]. An underlying visibility graph is used and a large-scale terrain environment was then able to be searched by a group of agents.

Crowds of agents moving in an environment [29, 4, 5] present many interesting challenges for pursuit-evasion, including large numbers of moving obstacles and visibility calculations which include the crowd. Sophisticated, path planning techniques have been created for planning the motion of agents in crowded areas [5], some with specific applications of pursuit-evasion scenarios [109].

This pursuit-evasion problem is similar to the camera tracking problem where an external camera has the goal of tracking target agents through virtual environments. In [110], an approach is described which extracts paths to goal locations through a roadmap with the paths weighted on a visibility criteria and distance. In [111], an approach to directing camera motion is described where visibility volumes are computed and used when planning a path in the scene. While tracking approaches are similar, they assume the target's location is always known, that the target is not taking evasive actions, and visibility and navigation limitations of the camera are not considered.

Another work that focuses on camera following is described in [112]. In this work, the objective is to follow and maximize visibility of multiple moving targets in an environment with obstacles. This problem is more difficult than having a camera track a single agent because the goal is to track a group of agents where it cannot be guaranteed that all agents remain within view. Multiple strategies are presented for 2D environments.

Our representation of the environment and the tracking probability model we present does not suffer from most of the restrictions described. For instance, complex environments are abstracted and represented in our roadmap, which also guides transitions and movements, however we compute visibility throughout the 3D environment considering agent capabilities and limitations whereas most approaches are limited to 2D.

CHAPTER III

INTEGRATING MULTI-AGENT SYSTEMS WITH ROADMAP-BASED PLANNING

In this chapter, an overview of the simulation system we have developed is described. This includes descriptions of agents, behaviors, the simulation cycle, and the role of visibility in our simulation. We also describe our use of the roadmap for representing potentially complex environments and the components that affect agent navigation.

A. Problem Parameters

We utilize an agent-based, distributed planning approach. This allows us to easily study different sets of agents and allows us to easily vary an agent's behaviors and capabilities. In this section, properties that can greatly affect the simulation are described which includes agent properties, behaviors, usage of roadmaps, forces exerted on agents and the simulation cycle.

1. Agent and Grouping Overview

Agents are defined by a range of the capabilities which include a view radius, view angle, maximum velocity, acceleration and behavior being executed. These can vary between individual agents or types of agents. The roadmap that an agent is equipped with can also be considered a capability since the connectivity and mapping of the environment can play a role in how the agent performs. The parameters that define each agent population describe the scenario that is being simulated. Communication between agents is often used which is another capability that can affect a simulation. The communication is either explicit in that one agent passes information to another agent directly or implicit in that agents may modify a shared data structure, often the roadmap.

An important distinction that should be mentioned at this point is that in our framework we use a general group structure to implement both groups of agents and single agents such that a single agent is simply a group that has no subgroups. This makes creating our framework more general, especially when creating behaviors. Although we will still refer to agents and groupings of agents, it is important to note that we use the idea of a group in a general way.

In many situations it is important to have agents moving through the environment with grouping restrictions, for example having to remain within some predefined distance of other agents. We use this idea of grouping to have the agents move through the environment together. This is also important in that grouping can help with coordination. As an example, a group that is performing an evacuation behavior need only have the main grouping of agents perform that behavior. Agents at lower levels in the group hierarchy then only need to follow along the group path. This grouping can also be useful when coordinating between directing agents or agents performing a pursuit behavior. In fact, it is a natural way to conceptualize different sets of agents performing different behaviors with distinct sets of capabilities, behaviors, roles and knowledge of the environment within each grouping.

2. Behaviors

The behavior that the agent is equipped with will determine how the agent reacts. These behaviors determine the actions that the agent or groups of agents take. The behaviors developed need to be dynamic enough so they can be applied to a group at any point and the group will then start performing that behavior. At each time step, all groups update their state based on the last plan that was generated by the behavior rule. The state is then resolved with other agents and the environment (i.e.

preventing collision). Behaviors can range from individual behaviors to group-based behaviors which require cooperation with other agents. More detailed descriptions on each behavior being executed will be described in each specific application section.

3. Forces on Agents

Force rules defined for agents dictate how they will move through the simulation given the paths that have been computed and perceived information in the environment. In our system, we allow multiple force rules to be given to an agent throughout the simulation. Each force rule can also be defined as having a maximum allowable force which will prevent any one force rule from overwhelming the agent's state at any step of the simulation.

A agent can have a *goal-based* force rule in which the agent attempts to follow a path that has been computed. In this force rule, the agent will attempt to push itself toward the next subgoal along a path. The force that is generated is simply in the direction given by the agent's current location to the subgoal.

A classic force rule that *flocks of agents* exhibit was described in [29]. This force rule attempts to have agents maintain a certain amount of cohesion, separation and alignment. Parameters can be set that define the weight of each component. We include this force rule in our system as it allows for interesting movement to be exhibited between agents that are moving together.

We also employ a simple *nearest neighbor avoid* force rule. This force rule attempts to generate a force away from the agent that has been observed to be the nearest. This allows for a very fast, computationally inexpensive way to have agents avoid one another when they are in close proximity. It also mimics the body forces that are described in the social forces model from [15].

The agents have the composed force rule applied during the update state component of the general simulation loop in Algorithm A.1. Shortly after the forces have been applied to the agents their state needs to be resolved in the environment to ensure that they have not moved into a state that is in collision with the environment.

4. Simulation Cycle

The outline of our simulation loop is given in Algorithm A.1. We use a general concept of individual agent, calling them a group. In this way the our framework is extendible, especially when creating behaviors, since a behavior created for individual agents can easily be applied to a clustering of agents, with additional logic used to ensure grouping restrictions are maintained. At each time step, a subset of all the simulation groups execute a behavior rule.

Algorithm A.1 General Simulation Loop

```
Input: simulator $sim$, environment $env$
1: $groups_{all}$ = sim.getAllGroups()
2: $groups_{sel}$ = sim.getNextSetOfGroups()
3: for $g \in groups_{sel}$ do
4:     $g$ →applyBehaviorRule($env$)
5: end for
6: for $g \in groups_{all}$ do
7:     updateState($g$)
8: end for
9: ResolveStateInEnvironment($groups_{all}$,$env$)
10: Evaluation
```

5. Visibility

The agent's ability to detect other agents in the environment can be affected by a number of factors. In simple 2D problems, the agent's visibility to other agents may only be restricted by the obstacles in the environment. Agent capabilities can also determine the amount of the environment that can be sensed by setting a maximum view radius and angle. Visibility can be further restricted by considering that agents may block one another's view in the environment, i.e., the 3D representation of all agents are potential obstacles in visibility checks. When surfaces are included in environments, these surfaces may also block visibility from one agent to another. We have included each of these aspects in our visibility checks which allows us to handle complex surfaces and crowd examples.

6. Areas and Hazards

We consider a number of different area types in creating our framework. This is to allow it to be more general and handle a wide range of scenarios. A general area definition is used to represent to a number of geometric regions. For example, in two dimensions this can be above or below a certain value along an axis or one or more point–radius pairs. More advanced area types could include polygonal regions or ones defined by a roadmap, which can include a collection of nodes and the transitions between them.

a. Area Types

In our simulation, an exit is an area used when performing evacuation. Agents use this area as a subgoal to a safe location. We often define exits as a set of point–radius pairs at a specified level. Safe areas are used in our simulations as goal areas for agents evacuating to reach. These areas often define a wider range of space, for example near the boundary of the environment but can also be represented as point areas (as with exits).

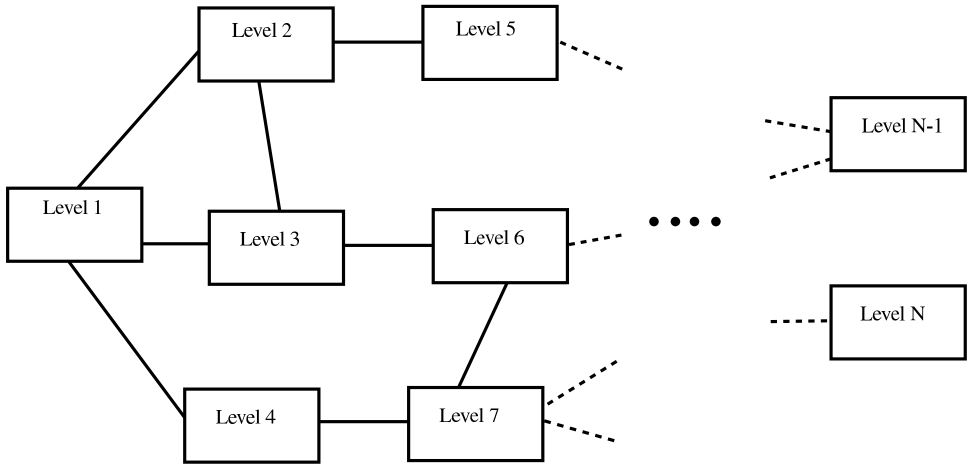
b. Hazardous Areas and Changing Areas

A hazardous area is another important thing to be able to simulate. We allow these kinds of areas to be represented by any area type. Although we only show examples with static hazardous areas, in a complete framework this kind of area could happen at any point and time in the environment. It could also spread and expand as the simulation progresses.

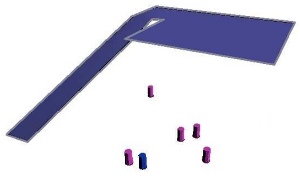
B. Multi-Level 3D Roadmaps

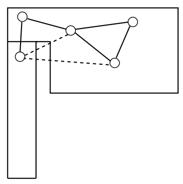
1. Environment Construction and Composition

The environments considered here are composed of obstacles and surfaces. Surfaces represent areas on which the agents can navigate. An agent's position on the surface will dictate the agent's height component. The building-like environments studied here are composed of levels, stairwells and exits. Each level represents a floor on which the agents can navigate. Adjacent levels are connected by stairwells. Input to the problem includes connections between levels and stairwells. Exits for a level are where stairwells connect to the level. A map of this type of environment has a structure like the graph shown in Figure 1(a). The subgraphs are shown, labeled SG_i, which represent valid movements at level i.



(a) Abstract Environmental Representation





(b) Simple 3D Surface (c) Projected Surface

Figure 1: (a) An abstraction of the problem is given showing a complete graph, G, with the subgraphs labeled $SG_{1..N}$. Each level in our building construction can be decomposed into subgraphs, with exits defining connections between other levels. (b) An example 3D surface with a long ramp leading up to the main level. (c) The surface projected to 2D. Also shown are sample configurations with valid and invalid connections.

2. Creating Roadmaps in Our System: Sampling and Connection in 2D and On Surfaces

Using roadmaps in evacuation planning systems and crowd motion provide many advantages over traditional navigation techniques. A roadmap is a light weight representation of the environment. It allows an agent to find routes through the environment as it approximates the connectivity of the free space in the environment and can be quickly updated as the situation changes. Paths can be selected that avoid certain areas or are biased towards other areas.

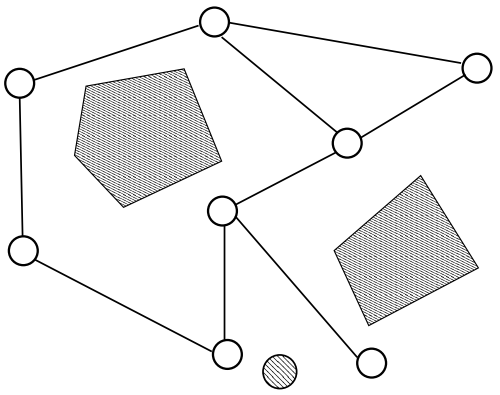
Roadmaps are graph representations of an environment that encode feasible, collision-free paths through the environment. Agents navigating in environments consisting of multi-level surfaces, such as buildings, need the ability to map these spaces. The agents encode valid movements on the surfaces by first sampling collision-free nodes on each surface. Connections are allowed between nodes on the same surface in which the straight line along that surface, projected to a 2-dimensional plane, remains completely within the projected polygon and is not in collision with objects on the surface. An example surface and its projection to a 2D surface is shown in Figure 1(b,c). The last step is to make connections between surfaces that are defined to be connected based on an input environmental configuration.

Sharing a roadmap among agents may seem to be an unrealistic capability, however, this may allow agents to select paths that more closely resemble paths that humans would select. This is because humans inherently have reasoning skills about where other agents may go and areas that may be congested. We investigate techniques for using a shared roadmap in our evacuation behavior that improves the evacuation routes selected. The shared roadmap may be adjusted by each agent resulting in the path selected being affected by other agents implicitly.

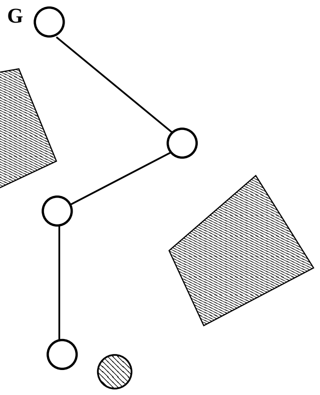
Roadmaps can also encode global information which may be useful to groups of agents, such as by storing information about an area to avoid, or by associating certain zones in the environment with certain groups of agents. Additionally, agents can then perform searches on the roadmap based on actions and observations that other agents they are cooperating with have made. These graphs/roadmaps are an essential component for our pursuit/evasion scenarios. Other benefits of the roadmap include the low computational cost of querying them once they are created and repairing them when the environment has changed. Rather than having to check for collisions during every time step (which would be very computationally-intensive), agents can query a valid roadmap quickly for paths guaranteed to be both available and collision-free.

3. Path as a Guide With Optimization at Various Levels

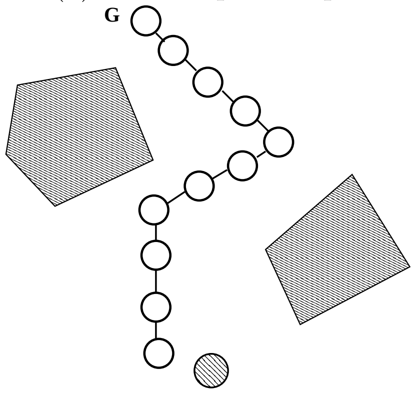
The roadmaps are used to generate paths through the free areas of the environment that will *guide* an agent from some start to goal location. These paths can be adhered to strictly or act as a guide that is optimized based on agent specific criteria. In many of the examples we study, we allow the agents to use the paths from the roadmap as merely a guide through the environment. Agent path optimization parameters allow the agent to improve the path to more quickly reach a goal location or to modify it to allow for greater clearance. An example of this process is shown in Figure 2 with varying levels of path smoothing shown in simulation in Figure 3. As the agent is navigating through the environment following a path, another parameter determines the distance required for a subgoal to be considered reached. These parameters play an important role in determining how closely an agent adheres to the roadmap and the time it takes an agent to reach a goal in the environment.



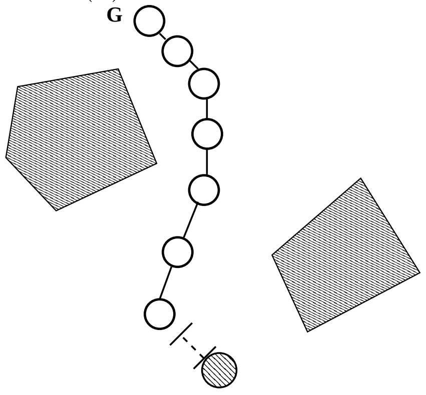
(a) Roadmap Example



(b) Path Extracted

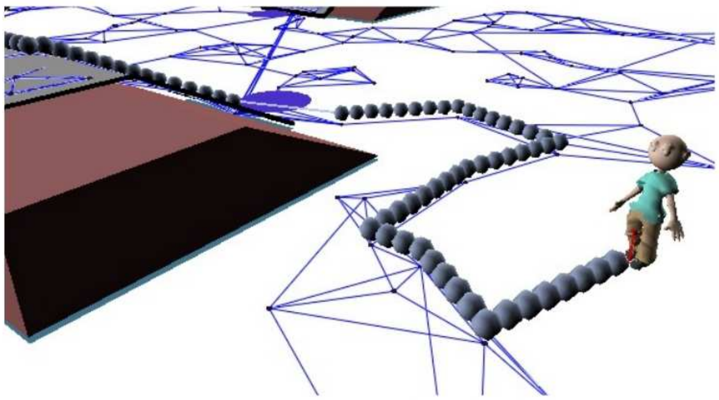


(c) Path Reduced

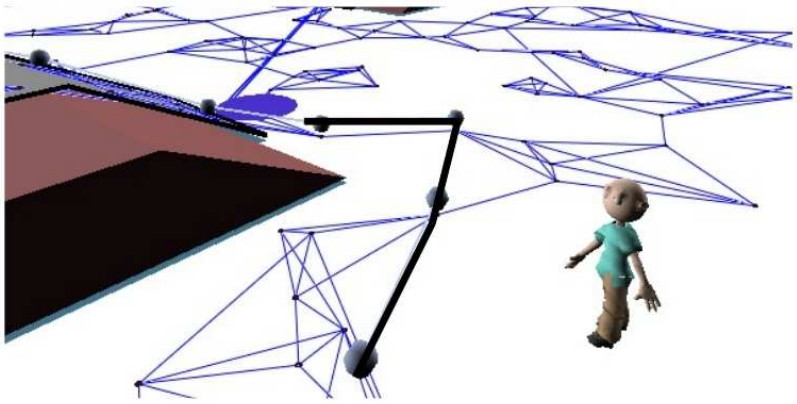


(d) Optimized Path

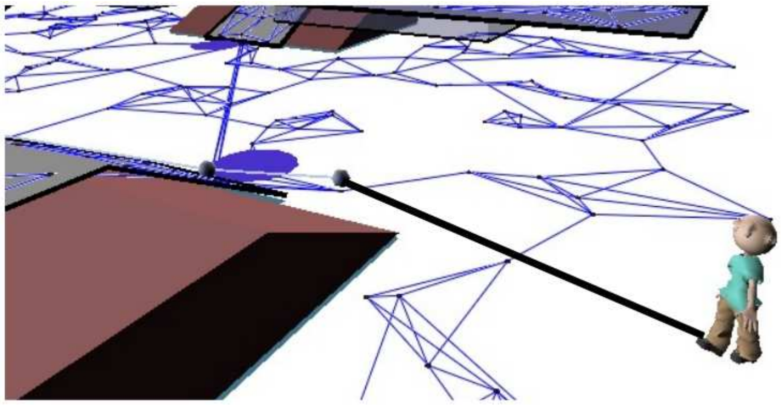
Figure 2: This figure illustrates the usage of roadmaps and paths used during navigation. (a) An example environment shown with two polygonal obstacles (shaded), representative roadmap and agent utilizing roadmap (shaded circle). (b) A path extracted from the agent's location to a goal location G. (c) A path reduced to a pre-defined resolution. (d) The reduced path is optimized to allow the agent to reach the goal faster. An agent considers a sub-goal along the path as reach when within a predefined distance. Steps (c) and (d) can be repeated in order to obtain a more optimized path.



(a) No Smoothing



(b) Some Smoothing



(c) High Smoothing

Figure 3: Path smoothing at various levels: (a) No smoothing applied to a path ; (b) Path with medium smoothing and (c) Highly smoothed path resulting in highly optimized route.

C. Agent Movement and Environmental Impact

In this section, the movement of agents in our simulation and the relation to other approaches is described. Our multi-agent system has inherent similarities to other approaches that have been proposed and is addressed here. We also give some validation on the movement of agents as they navigate in crowded scenes which is based on classical observed values [113]. Here we also show the impact that the environment can have on the movement of agents.

The social forces model [15, 16] and basic flocking forces [29, 53, 52, 51] are models that have been proposed to improve local motions of agents navigating in an environment. While we support a variety of force models, including the basic flocking force model, we emulate the social forces model by including the body force between an agent and the agent nearest to it. This allows agents to appear to avoid nearby agents without requiring complex computation between all agents in it's sensory range. As with other models, the amount of force applied away from the nearest agent is a parameter that can vastly affect the agent's movement.

An aspect of evacuation that has been observed is that people will often prefer or only have knowledge about certain exits [82]. We encode this by varying the level of environment knowledge that each agent has, for example some agents may know about all exits and safe areas in the environment while others are given a random subset.

The directors in our simulations can be considered the trained leaders as in [10]. A cooperation parameter dictates whether or not evacuating agents will accept information from a director. This allows us to achieve guidance from one set of agents to another with the potential for the evacuating agents to choose to ignore the information being given.

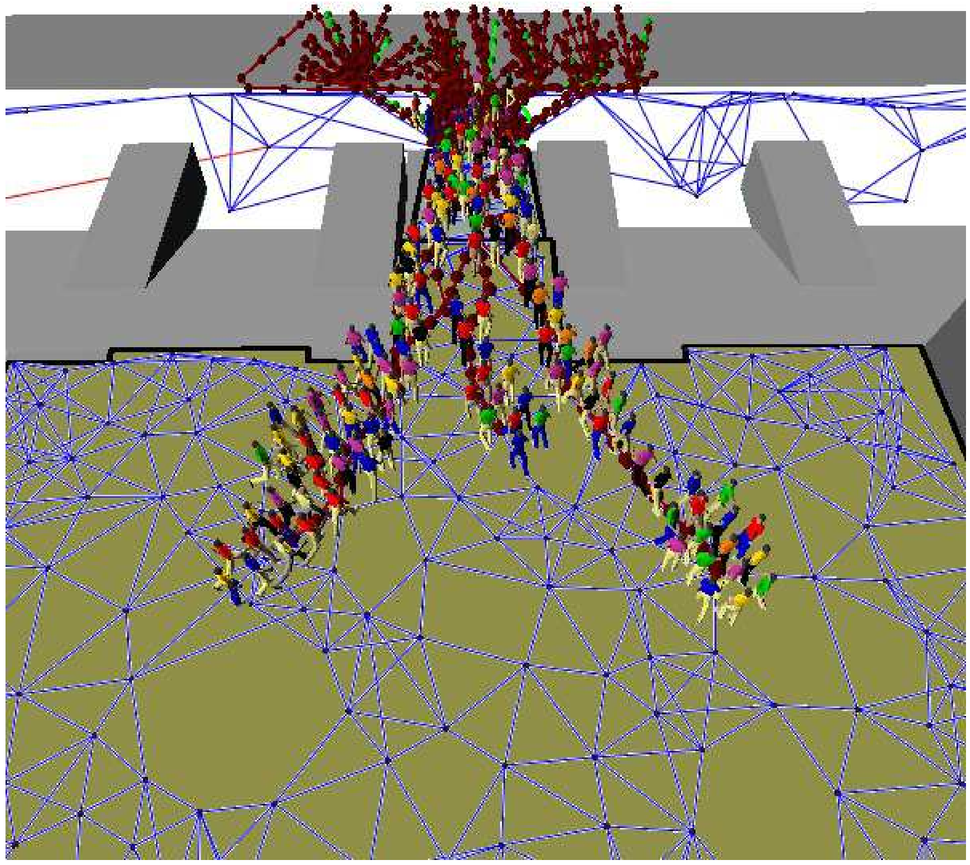


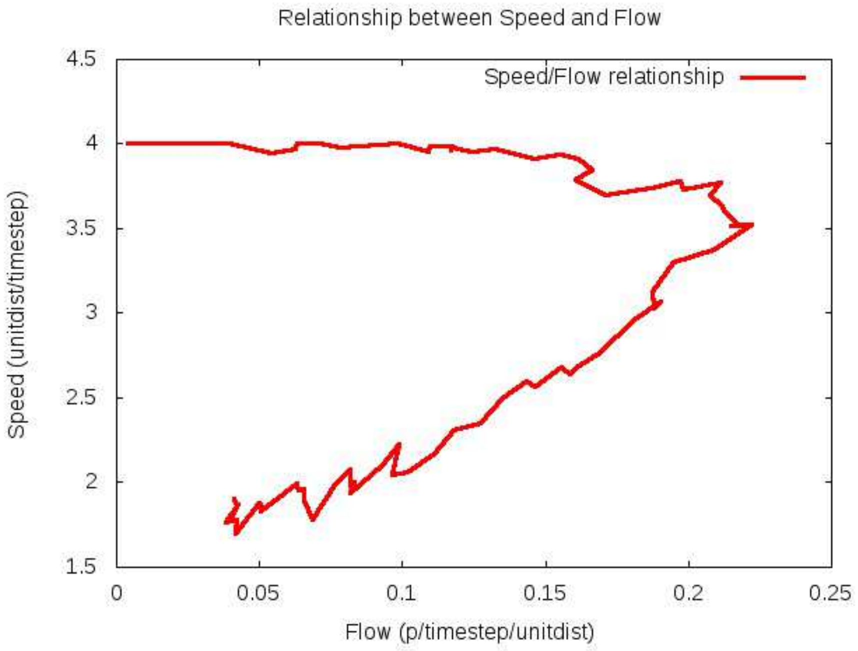
Figure 4: Environment used to compare movement of agents which consists of agents on one level that must pass through a corridor in order to reach a safe area.

1. Agent Movement Metrics

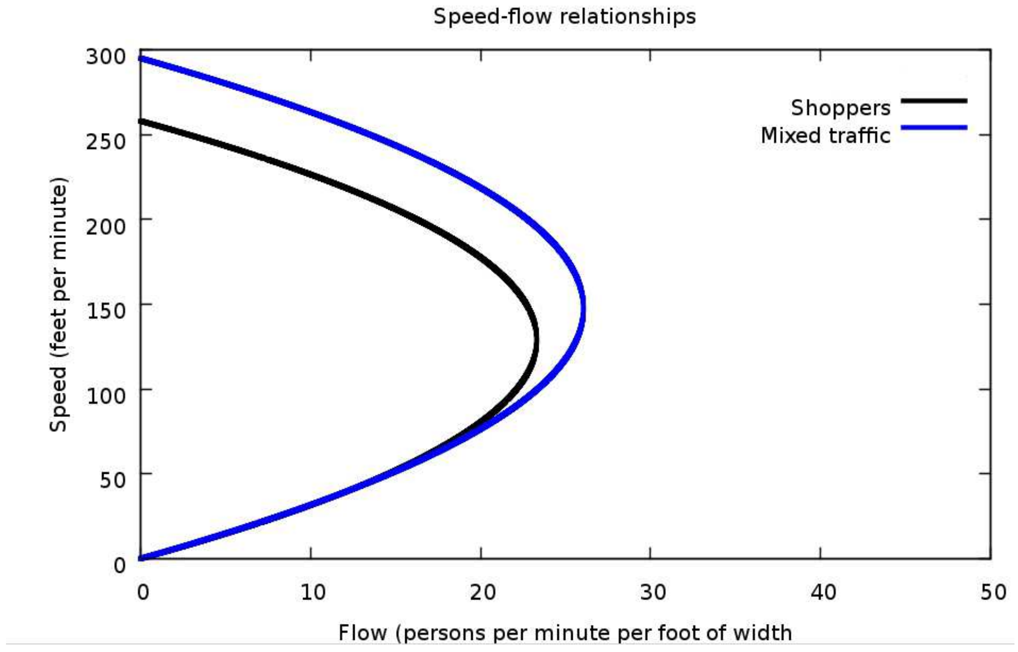
The local motion of agents navigating through a simulated environment is important in our work. This motion is one aspect looked at to determine how well a system models the local interaction occurring between agents. While these interactions and the impact that agents have on one another are best seen visually, they can also be measured quantitatively.

We have developed metrics to show that our model of agent movement can reproduce classic local motion values that have been observed [113]. These metrics can be used to evaluate an evacuation or in the future, directors can use these metrics to improve the guidance they are providing to evacuating agents. The local motion is important because we would like to capture local interactions that occur as passageways become congested. In general, we are just as interested in pathways selected and how they are adapted as interaction occurs.

An example of this motion validation is done in the environment shown in Figure 4. This environment consists of agents on one level that must pass through a corridor in order to reach a safe area. In Figure 5, the speed-flow relationship is shown in our system. It can be observed that low flow corresponds to either few agents passing through the corridor or too many agents passing through. The best flow is reached at a certain point and then declines as more agents pass through. The speed-density relationship is shown in Figure 6. As in the previous example, the average speed of agents passing through the corridor decreases as the density increases. It should be noted that in this example we do not compute density in the same way as [113]. Rather, we compute the average velocity of agents as they pass through the corridor when increasing the number of agents in the simulation. The relationships shown in Figure 5(b) and Figure 6(b) are recreated from values obtained from [113].

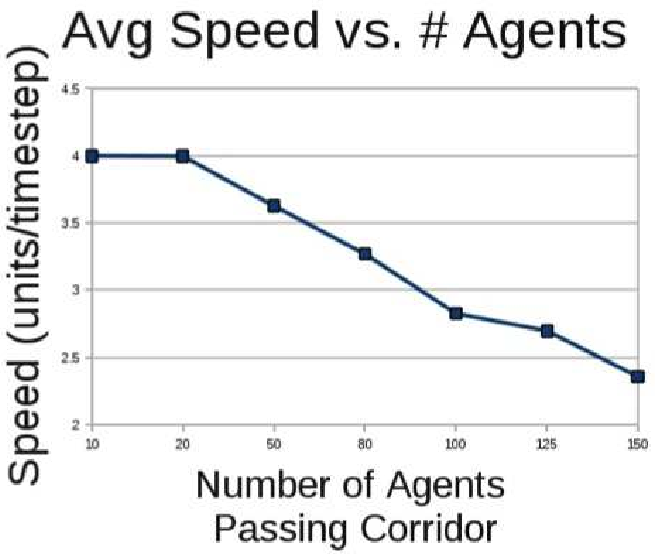


(a) Speed vs. Flow (our system)

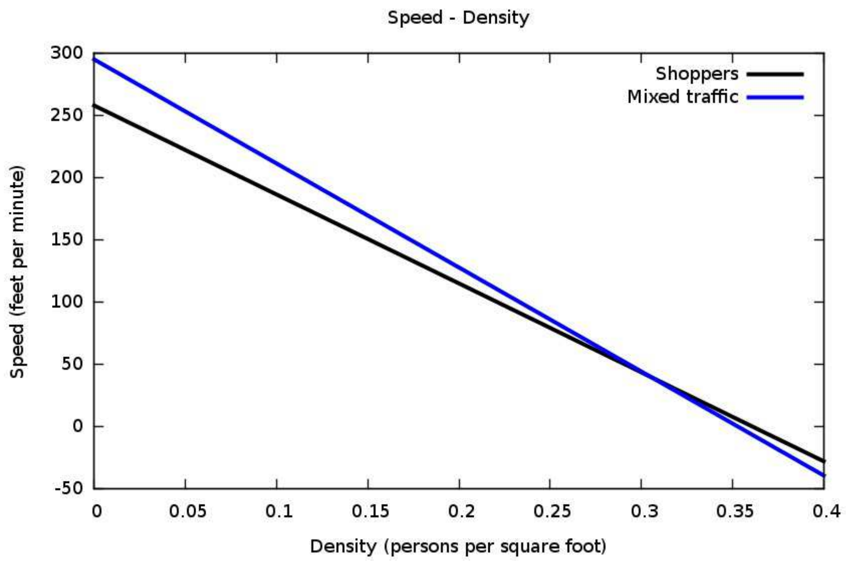


(b) Speed vs. Flow (classic values)

Figure 5: (a) Speed/flow relationship in our system compared to (b) classic values observed.



(a) Speed vs. Density (our system)



(b) Speed vs. Density (classic values)

Figure 6: (a) Speed/density relationship in our system compared to (b) classic values observed.

2. Impact of Environment on Pedestrian Flow

Our system can be used to evaluate the result of environmental changes on the evacuation process. This could be a useful tool during the design process when determining necessary widths for corridors, accesses and other portions of the environment that could impact movement through the structure. Examples of how the environment can change the overall pedestrian flow is shown in Chapter VI, Section F.7.

CHAPTER IV

SEARCH AND TRACKING TECHNIQUES*

The proposed roadmap-based approach is well suited for supporting many search techniques. In this chapter, an overview of search strategies that have been developed to handle more complex scenarios is described. Also in this chapter, a probability-based agent tracking model is described which will allow an agent to track another set of agents when they have left the agent's field-of-view.

A. Behavior: Search

In this section, we describe the search behaviors that are used in this work. Some of the behaviors were included as baselines for testing our system and for improving the behavioral development process. These include some of basic heuristic search behaviors. Other behaviors have been developed for use with roadmaps and as components of cooperative pursuit-evasion behaviors.

1. Heuristic Roadmap-Based Search Behaviors

We describe three distinct types of searching behaviors. For the first type, the decisions are made by agents based on the information available to the agent at that given time. These behaviors, *basic* and *scanning*, are more individual-based although the agents do interact with each other. The second type of behavior is a group-based local search. These agents divide into groups and search a local area cooperatively. The behavior we propose here that exhibits this property is referred to as *rendezvous*.

*Part of the data reported in this chapter is reprinted with the kind permission of Springer Science+Business Media from "Roadmap-Based Level Clearing of Buildings" in *Lecture Notes in Computer Science*, Vol. 7060, by S. Rodriguez and N. M. Amato, 2011, Springer, Berlin. Copyright 2011 by Springer. [25]

The final type of behavior we propose divides the environment into areas in which each agent is responsible for searching, a behavior we refer to here as *territorial.* In *frontier* searching, the agents explore the boundaries of the searched space.

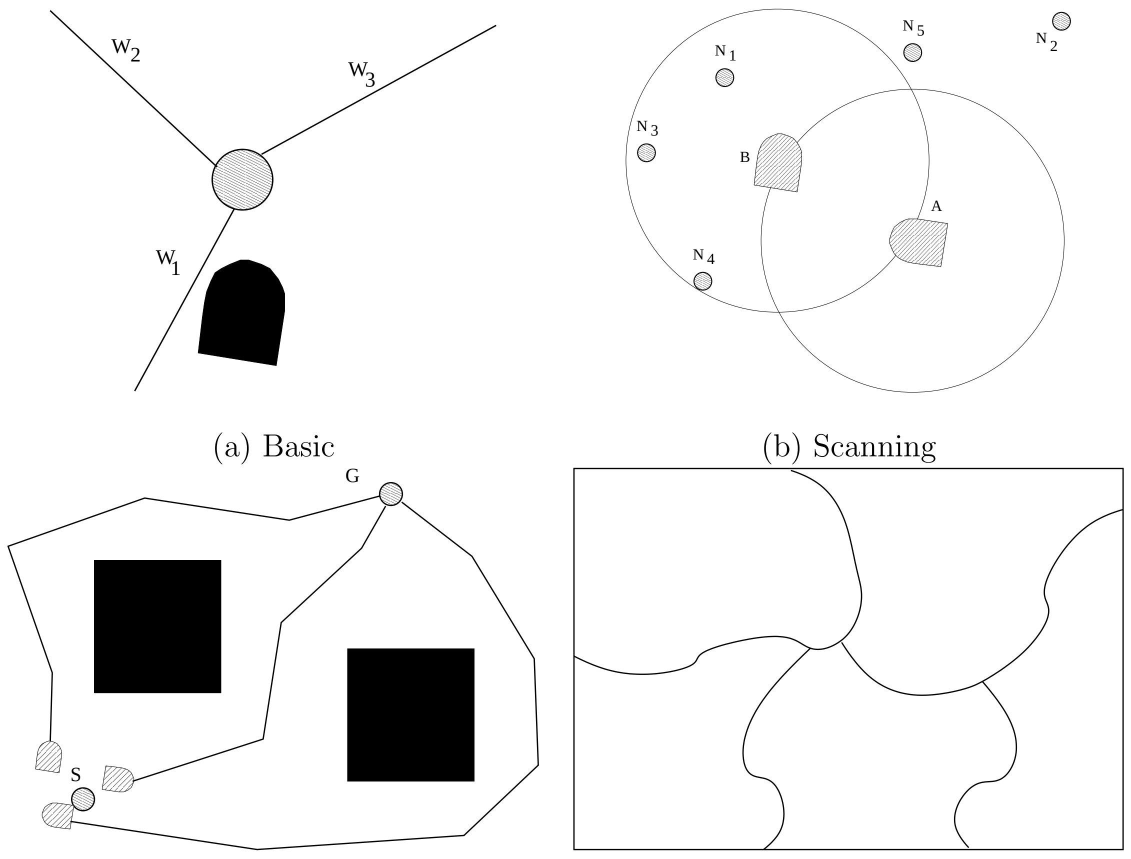
a. Basic Search

This behavior was developed to show how a group of agents can utilize a roadmap in order to effectively search an environment [52]. The goal of this behavior is for the agents in the group to visit as much of the environment as possible. For this behavior, agents use the roadmap to obtain pathways to free areas of the environment. The effectiveness of this behavior has an underlying dependence on the quality of the roadmap used by the agents.

In this behavior, the goal is to have some agent in the group visit all edges and vertices of the roadmap. As agents in the group traverse a roadmap, they select roadmap edges to follow based on the edge weights of edges connected to the nearest roadmap node. All edges are initially weighted the same. As the agents traverse a roadmap edge the weight is increased to indicate that the edge has been traversed. Since the goal is to cover the environment, the individual agents are biased toward relatively uncovered areas of the roadmap. This is achieved by having agents select lower weighted roadmap edges with a higher probability. In Figure 7(a), an example is shown of a roadmap node being reached by an agent with the potential outgoing edges and edge weights labeled.

b. Covering by Scanning Nearby Areas

The objective of this behavior is to determine locations that are currently unobserved, and move to a position where those locations are observable. The agent initially constructs a list of points that satisfy one of two conditions. The first condition is



(a) Basic

(b) Scanning

(c) Rendezvous

(d) Territorial

Figure 7: This figure illustrates the decisions made by each behavior. (a) Basic: an agent reaches a subgoal then randomly selects the next goal based on the edge weights of neighboring nodes. (b) Scanning: an agent communicates with neighboring agents in order to determine its next subgoal, ignoring goals visible to neighboring agents. (c) Rendezvous: a group of three agents agree to a local goal and generate different paths to that goal. (d) Territorial: the environment is partitioned and each agent is responsible for searching their portion of the environment.

that the point is out of the sensory range of the agent. Alternately, the point could be within sensory range of the agent, but with an obstacle between the agent and the point, thus preventing the detection of the point.

The agent utilizes the roadmap to navigate to the currently selected point in the list. Once that point is observed, it is removed from the list. Furthermore, any other points in the list that are observed on the way to the currently selected point will also be removed from the list.

The agents are able to communicate when in direct sensory range, provided that there are no obstacles between the agents. When communicating, any points that one agent can observe that the other is seeking will be marked as covered. Similarly, when obtaining a new list of locations to explore, an agent will discard points that are detectable by agents with whom it can communicate. In Figure 7(b), an example of agents scanning an area is shown with the potential for agent A to communicate with agent B.

c. Rendezvous: Team-based Covering

The objective of this behavior is for teams of agents to traverse the roadmap to a common, randomly selected goal node. Furthermore, the agents attempt to select different paths on the roadmap if possible. In order to accomplish this, the agents must have access to a shared, global roadmap with adaptive edges, preferably with a high level of connectivity to provide many alternate routes from a given node to any other node.

As the agents determine their paths to a goal, they will reweight the edges connected to the nodes on the path. Subsequent agents will be biased away from these reweighted edges, and will seek other paths.

Upon reaching the goal node, an agent will wait at that node until the other agents have also arrived. At this point, the original weights on the roadmap will be restored, and a new common goal will be randomly chosen. An example of agents generating rendezvous routes to a common goal location is shown in Figure 7(c).

d. Territorial Covering

In a territorial covering using the roadmap, the roadmap will be first partitioned into components, one for each agent. Once the roadmap has been partitioned, an agent will be assigned to search each region of the environment. In this way, the responsibility of searching the environment is divided among the agents based on the partitioning of the roadmap. An example is shown in Figure 7(d) of an environment partitioned for five agents.

e. Frontier Covering

Exploring the *frontier* between what has been observed and what is still unobserved is a traditional approach to searching [95]. We extend this approach by marking nodes in the shared roadmap with labels of clear, frontier or unclear. In this way, the agents benefit from work done by other agents in clearing the environment. This also represents a form of communication since the roadmap is shared and marked when cleared.

In multi-level environments, the frontier behavior may be paired with a *level-by-level* search so the agents can search each level in a coordinated way. We allow the scenario designer to specify the order in which agents should search and attempt to clear the environment in order to see the effect of different strategies. This also allows agents to be assigned to search certain portions of the environment.

f. Probabilistic Exploration

Probabilistic exploration of the environment is when an agent uses its tracking probability model in order to explore the environment to search for the target being tracked. In this searching behavior the agent will plan its motion towards the node of the roadmap with the highest temperature as described in Section C.

B. Visibility and Graph Theoretical Approaches within Our Framework

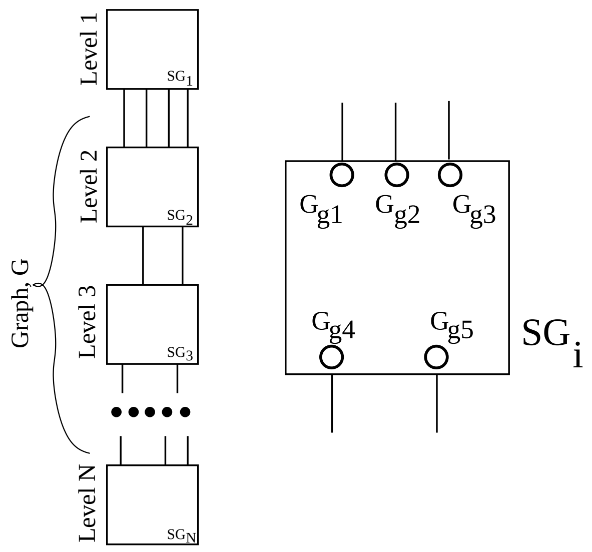
Our level clearing algorithm is an intuitive approach to handle complex three-dimensional environments in our roadmap-based approach. It can be described as a process where agents iteratively attempt to clear adjacent levels of the environment until reaching the last level. This clearing process is done by guarding the stairwells between levels, which prevents recontamination of cleared levels from portions of the environment that have not yet been cleared. An overview of the entire process can be seen in Algorithm B.1. The first step is to generate a roadmap based on the environment. The roadmap is then partitioned into sub-graphs, as in Figure 1(a), based on levels, the input configuration of the environment and the exits and stairwells that exist. The partitioning is something that could be automated with more advanced graph-based techniques; we leave this for future work. The agents then search the environment from one level to the next until each level has been cleared. We describe two strategies for clearing each level. The first strategy involves selecting coverage locations from the roadmap with the goal of covering the entire roadmap at a given level. The second strategy involves building clearing paths on a level between cleared portions of the roadmap to uncleared areas until the roadmap (at that level) is fully cleared. The agents plan their paths to goal locations dictated by each strategy and traverse their path reaching the final goal. Once all agents are at the goal locations, the environment at that level is considered covered (based on the mapping) and the process can continue at the next level.

Algorithm B.1 Overview of Level Search

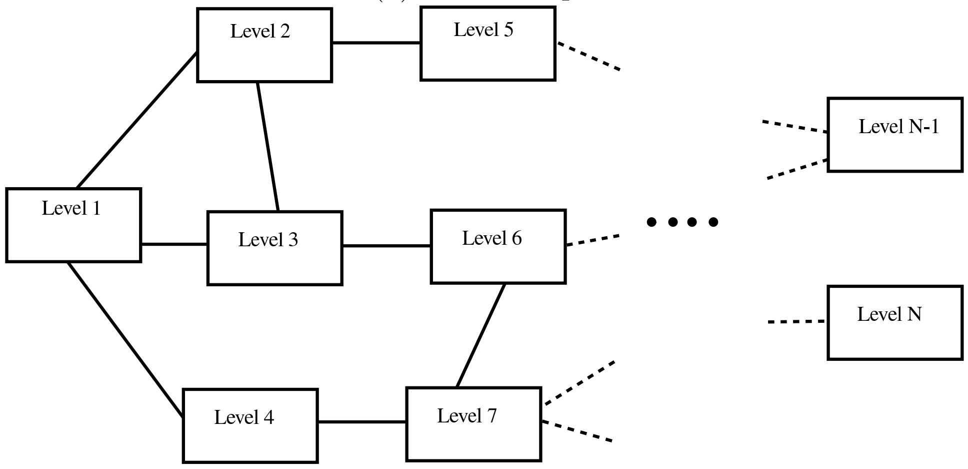
```
Input: environment $env$
 1: $rdmp \leftarrow$ generateRoadmap($env$)
 2: initialize agents
 3: Partition $rdmp$ into subgraphs based on levels/environment
 4: for $level \in env$ #bottom-up/top-down do
 5:     generateCoverageLocation (....)
 6:     assignPathWays (...)
 7:     repeat
 8:         agents traverse path to assigned goal
 9:     until all agents reach goals
10: end for
```

1. Strategy 1: Guard and Coverage Locations

The process of generating coverage locations for a given level, i, is shown in Algorithm B.2. In this process, $nodes_i$ in the input roadmap at level i are first selected. Each exit at level i is considered a guard location G_{g_i}. These exits are located near stairwells and would prevent a target agent from entering or leaving the current level without being detected. These locations are shown in Figure 8(a) and correspond to



(a) Serial Graph



(b) General Env. Graph

Figure 8: (a) The types of graph we were initially able to search which represented multi-level buildings with the additional requirement of identifying exit information at each level. (b) The general type of graph we can handle in this work where connections can exist between any levels and the underlying clearing path algorithm can be used to generate clearing paths for each level. The environments in (a) can be represented as the graphs in (b).

the entrances and exits of the underlying subgraph at level i, SG_i. In our current problem formulation, these exit locations are input configurations.

The remaining problem is to cover level i as well as possible with the given agents. We achieve this using a greedy algorithm based on the roadmap nodes that were generated at level i. The nodes, $nodes_i$, which are covered by guard locations G_{g_i} are marked as cleared. Coverage locations, G_{c_i}, for level i are generated in a greedy manner such that the next node added to G_{c_i} is the node that will maximize coverage among all the remaining uncleared nodes in $nodes_i$. This is a process similar to what was presented in [97]. The number of agents needed to guarantee a covering of level i is $G_{g_i} + G_{c_i}$.

Overall, the maximum number of agents needed to guarantee a clearing of an environment is given by: $\max_{i \in env} \{G_{g_i} + G_{c_i}\}$ where i represents each level to clear and G_{g_i} are guard locations at each level and G_{c_i} are the coverage locations selected.

An interesting aspect of this solution is that the ordering of the returned coverage locations affects the amount of coverage. Often, the total number of searching agents needed may be much more than the number of agents available. In these cases, we can report the portion of each environment that would be covered based on the coverage value returned by the number of agents available. These agents would also attempt to maximize coverage by going to coverage locations in the order that they were selected based on the greedy algorithm presented in Algorithm B.2.

2. Strategy 2: Guard Locations and Clearing Paths

The basic idea of this strategy is similiar to Strategy 1. However, when using clearing paths, agents can clear entire areas using the underlying encoding of the roadmap and cooperation between other agents. As in Algorithm B.2, guard locations are generated initially and the nodes covered are marked as clear. The main difference is that instead

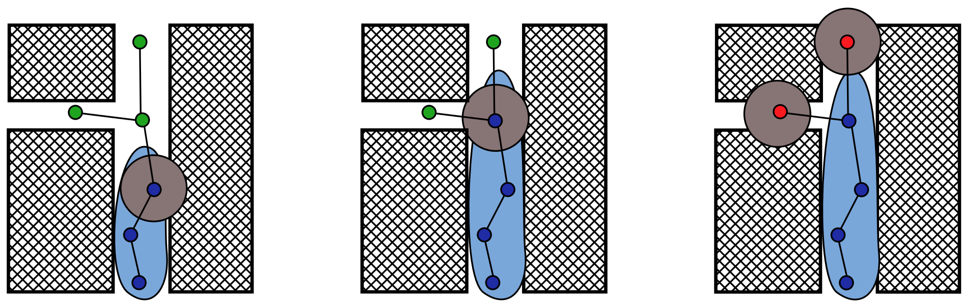
Algorithm B.2 Generate Coverage Locations

```
Input: environment $env$, roadmap $rdmp$, current level searching $i$
 1: $nodes_i \leftarrow rdmp$.nodesAtLevel($i$)
 2: $G_{g_i} \leftarrow env$.ExitsAtLevel($i$)
 3: $markNodesCleared(nodes_i, G_{g_i})$
 4: $G_{c_i} \leftarrow \emptyset$
 5: while existNodesUncleared($nodes_i$) do
 6:     $N_j \leftarrow$ uncleared node $\in nodes_i$ with max coverage
 7:     $G_{c_i} += N_j$
 8:     markNodesCleared($nodes_i, N_j$)
 9: end while
10: return { $G_{g_i} + G_{c_i}$ }
```

of selecting the node that maximizes the coverage, the clearing path selected is one that maximizes the path traveled through the level as a way of maximizing the area covered by the clearing path.

In our initial work [25], we looked at the search of multi-level building-like structures that could be represented by a layered graph as shown in Figure 8(a). We have extended the work to abstract the allowable environmental representation where arbitrary level connections are allowed which allows for much more general environment types to be considered. An example of the connections allowed between levels can be seen in Figure 8(b). In this example, levels to be searched are nodes in the graph and edges in the graph are the allowable connections between levels.

Clearing paths are started on the frontier of the area that is already covered (i.e.,



(a) Initial Clearing Path (b) Valid Move (c) Invalid Moves

Figure 9: An example of clearing paths shown. Obstacles are shown thatched, a roadmap is shown that traverses the corridor with an agent's view radius also shown (gray). (a) An initial clearing path with three nodes covered (nodes in blue). The covered area is outlined in blue. (b) A valid move is shown extending the area covered. (c) No more valid moves available.

from nodes that are not already cleared). A clearing path, CP, keeps track of the nodes that are initially covered, N_{init}, and nodes that are currently covered, N_{cur}. As a clearing path is built up we consider the set of potential moves to be the nodes that are outgoing from N_{cur} which are not in N_{cur} or N_{init}. A move is considered valid if moving to that node leaves the nodes leading to the previous outgoing nodes covered. In this way we can ensure clearing paths are built up which prevent their N_{cur} from becoming contaminated. The paths are built up from all frontier nodes and continue until no more valid moves exist. A greedy strategy is used to select the path that maximizes the area covered. We continue to build clearing paths until all nodes on a level are cleared.

The overall clearing strategy is shown in Alg. B.3. Clearing paths are generated in order to clear sets of levels. The clearing paths that are generated are stored in

Algorithm B.3 clearingStrategy

```
Input: env, environment
 1: levels = env →getCurrentLevelsSearching()
 2: CPC = ClearingPathCollection()
 3: CPC.init()
 4: outgoing = getOutgoingNodes( levels )
 5: while notEmpty( outgoing ) do
 6:     gL = node in outgoing
 7:     newCP = new Clearing Path( starting from gL )
 8:     newCP→setGuardPath()
 9:     mark nodes cleared
10:     CPC.add( newCP)
11: end while
12: while levelsNotCleared(level) do
13:     cpMax = generateClearingPathOfMaxClearing( CPC )
14:     CPC.add( cpMax )
15:     mark nodes cleared
16: end while
```

a structure we call a ClearingPathCollection. Given a set of levels to be searched, guard locations are set where agents will guard nodes that lead to outgoing nodes (i.e., nodes that are not in the set of levels to be searched). Nodes are then marked as clear given the set of guard locations generated. The next step is to generate clearing paths of maximum clearing until all the levels are cleared. Clearing paths of maximum clearing are started from the nodes on the frontier of the current clearing path collection. This is shown in Alg. B.5.

In Alg. B.4, a clearing path has a set of nodes that are initially covered and a set of nodes being covered which are being built up. A move can be taken (from an outgoing edge), if it is valid. Once a valid move is taken, a new clearing path is created with the new move added as part of a clearing path. The build clearing path algorithm is then recursively called until no more valid moves exist.

We consider a valid move to be one that prevents covered nodes from becoming contaminated. This can be by either previous clearing paths or if the edges outgoing with nodes that are covered remain covered. In this way, nodes that have edges out going will remain covered if a move is taken. Examples of valid and invalid moves within our framework is shown in Figure 9.

3. Pathway Assignments

The process of pathway assignment is critical when attempting to guarantee that contamination of a covered region does not occur. As a simple example, a guard agent at a guard location on level i whose group of agents is progressing to level $i+1$ might naturally be the guard of the exit associated with the stairwell connecting levels i and $i+1$. This may not happen if an agent is present whose pathway to the guard location at level $i+1$ is shorter. In the assignment process, shown in Algorithm B.6, all agents have pathways generated from their current location to one

Algorithm B.4 buildingClearingPath

Input: **nCP**, starting clearing path; **bestCP**, best clearing path

```
 1: nCP has start pt, node id and initial path
 2: nodesCov: nCP→getNodesCov()
 3: nodesOut: nCP→getNodesOut()
 4: potentialMoves: nCP→getOutgoingEdges()
 5: for pm ∈ potentialMoves do
 6:     if moveIsValid( nCP, pm ) then
 7:         new_nCP = addMove(nCP,pm)
 8:         buildOnClearingPath( new_nCP, bestCP )
 9:         if new_nCP→Better(bestCP) then
10:             bestCP = new_nCP
11:         end if
12:     else
13:         if nCP→Better(bestCP) then
14:             bestCP = nCP
15:         end if
16:     end if
17: end for
```

Algorithm B.5 generateClearingPathOfMaxClearing

```
Input: CPC, clearing path collection
1: FN = generate frontier nodes from CPC
2: bestCP = NULL
3: for fn ∈ FN do
4:     newCP = new Clearing Path( starting from fn )
5:     buildOnClearingPath( newCP, bestCP )
6:     if newCP→Better(bestCP) then
7:         bestCP = new_nCP
8:     else
9:         delete nCP
10:    end if
11: end for
```

of the goal locations generated for the next level. The goal locations are then assigned by the leader agent based on the path distance. This is an iterative process where the shortest path is selected from all the pathways. Once an agent and goal location pair has been selected, its associated pathways are removed from the full pathway list as are the pathways that are associated with the goal location assigned. In this way, we guarantee that an agent guarding a stairwell at level i leading to level $i+1$ is selected to guard the exit at level $i+1$.

Algorithm B.6 Assign Pathways

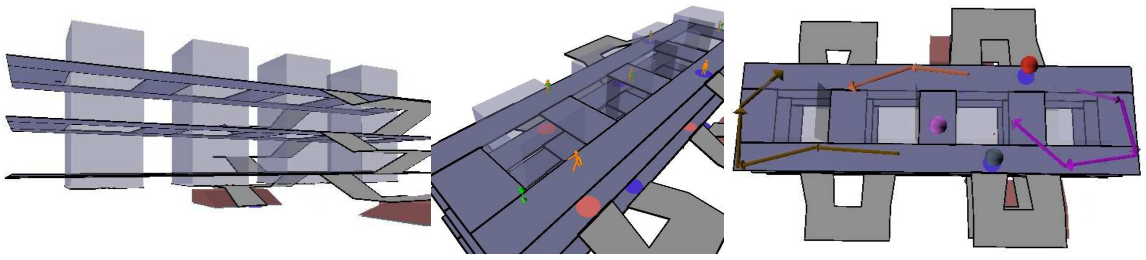
```
Input: environment $env$, roadmap $rdmp$, current level searching $i$, $group$ of agents
    searching
1: $pathWays \leftarrow \emptyset$ # to store all possible paths
2: $goalLocations \leftarrow env$.getGoalLocationsFromStrategy($\cdots$)
3: for all a $\in group$ do
4:     #generate all pathways
5:     for all gL $\in goalLocations$ do
6:         $path_i$ = findPath($a$.getPos(), gL)
7:         $pathWays += path_i$
8:     end for
9: end for
10: Assign goal locations to agents based on path distance
```

4. Dependence on Mapping

The main aspects of this approach, generating coverage locations, generating clearing paths, and pathway assignments, have an inherent dependence on the roadmap. When attempting to achieve full coverage, we assume that the roadmap gives a good coverage of the environment. One way we attempt to ensure this is by preventing samples or configurations from being generated too close to one another. In this way, the nodes representing the free space in the environment are spread out. The assignment of pathways also assumes that valid pathways exist between guard locations, through stairwells, and between levels. We ensure that the roadmap is generated in such a way that these pathways do exist although it is a known limitation to our approach. Due to the inherent dependence on the roadmap, our searching agents can be made more or less intelligent based on the quality of the mapping. This will be shown in the results. Agents that have a better mapping will have a better chance of fully clearing an environment whereas agents with a simpler mapping may miss portions of the environment. This may in fact be useful in games when trying to provide varying capabilities to adversarial agents.

5. Level Search Examples

We present results in simulated building-like environments, shown in Figures 10(a) and 12. Each of these environments consists of multiple levels with stairwells connecting adjacent levels. In these problems, stairwells do not exist between non-adjacent levels. We present results showing the average number of agents that would be needed to cover each level of the environment given the visibility constraints, which are varied by changing the view radius (VR) distances. The number of agents needed to clear the environment would be the maximum needed over each of the individual



(a) Building Example (b) Coverage Locations (c) Clearing Paths

Figure 10: (a) A building example with four levels and multiple passages between each level. (b) Agents clearing a level by using coverage locations. (c) Clearing paths generated for top-level given agent visibility restrictions.

floors (which is dependent on the roadmaps generated). Results are generated using roadmaps that adequately cover the environment, which includes valid pathways between open areas, hallways and exits. The effect of agent view radius restrictions is not taken into account during the roadmap construction phase. Results presented are averaged over ten trial runs, unless specified otherwise.

a. 4 Floor, 5 Hallways

This environment, shown in Figure 10(a), consists of four levels. The first level consists of an open area with stairwells leading to the second level. The second level and each subsequent level is divided by four obstacles, creating five hallways off the main two hallways. The upper levels are connected by two stairways between the levels. The environment dimensions are 230 units by 140 units. We tested our search algorithm with a number of agent view radius capabilities (10, 30, 60, 120 and 240 units); results are shown in Figure 11. Results are omitted for view radius capabilities of 240 units since they match those for agents with a view radius of 120 units. The

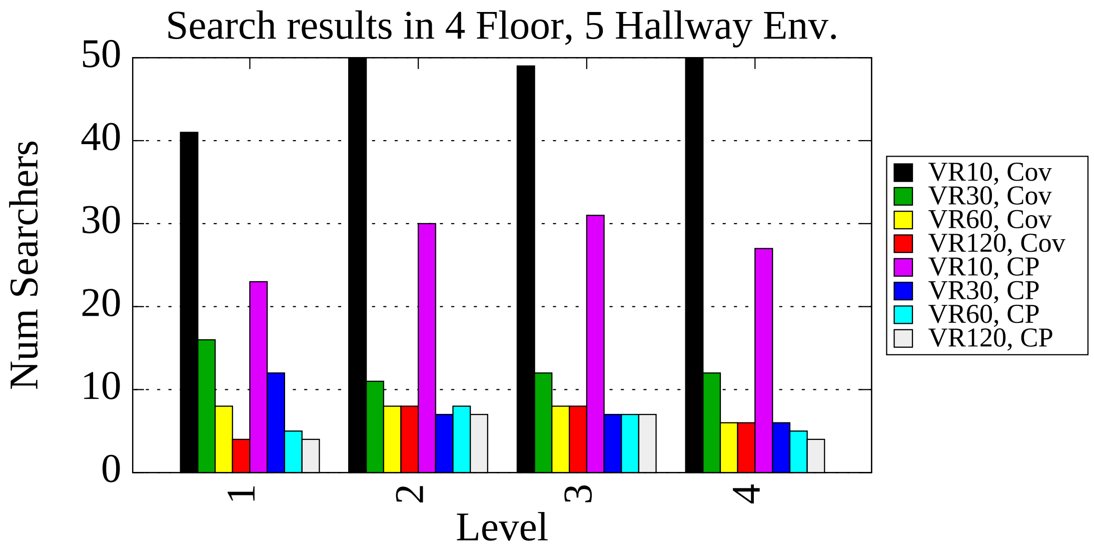


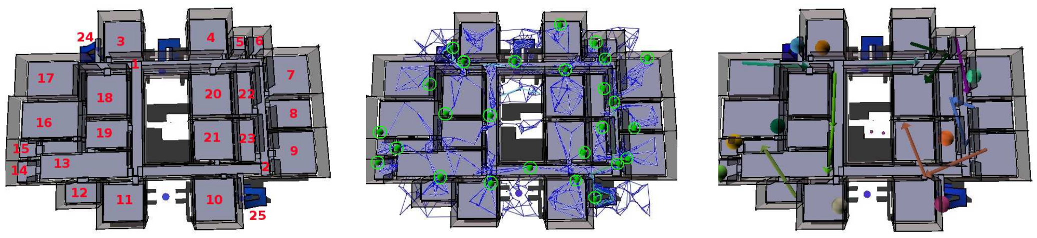
Figure 11: Search results showing number of agents needed in 4 floor, 5 hallway environment.

maximum number of searchers needed to clear each level is shown.

As the view radius increases, the number of agents needed to cover each level decreases. In fact, once the view radius gets large enough, the average number of agents necessary to cover an environment levels off. In this environment, this happened when the view radius was 120 or larger, i.e., it required the same number of agents at each level for view radii of 120 and 240. For a view radius of 60 units and higher, the upper levels all require the same number of searchers. This corresponds to guard agents blocking off the exits and hallways and agents required for guarding the hallways.

The coverage of the environment can be analyzed as coverage locations are added. The coverage gradually increases until complete coverage is achieved. Given N agents available to cover and clear an environment, the first N coverage locations would be used as goals to maximize coverage. If coverage were only required to a certain level, our approach could give insight into the number of agents necessary to achieve this coverage. For example, in the case of agents with a view radius 60, as coverage locations are added to cover the second level the percent of the roadmap covered progresses like: [22.3, 44.0, 56.0, 79.5, 88.6, 95.2, 98.8 and 100]. If only 98 percent coverage were required then only 6 agents would be needed as goal locations 7 and 8 would be unnecessary for the first three levels.

The number of agents needed when generating clearing paths in this environment is much fewer than are needed when trying to cover each level with agents. This is because the agents can build off the areas that are previously covered to expand the area that they clear while preventing contamination of that region. In fact, when agents have smaller view radius values using the clearing paths strategy requires far fewer agents to clear the levels.



(a) Coverage Locs. (By Simple Inspection) (b) Roadmap-Based Coverage Locations (VR 60) (c) Clearing Paths (VR 60)

Figure 12: Office example.

b. 3 Floor Office Building

The second environment we performed experiments in is shown in Figure 12. This environment is much more complex than the previous one. It consists of three levels, with many rooms on each level with walls obstructing the view between rooms and hallways. There are three stairwells between the first and second floor and two between the second and third floor. This environment has dimensions of 355 units by 275 units. We tested our search algorithms with a number of agent view radius capabilities (30, 60, and 300 units); results are shown in Figure 13. The first floor has 900 samples to cover the inside and outer portions of the building while floors 2 and 3 have 250 samples. This minimal sampling is good enough to have expected pathways through the environment. As before, for each view radius tested, the average number of searchers needed per level is shown.

When agents are equipped with a view radius of 300 units they have the ability to see across almost the entire environment. This results in much fewer searchers needed to guarantee a clearing of the roadmap. The first level generally requires more searchers because the open space around the building is also considered an area

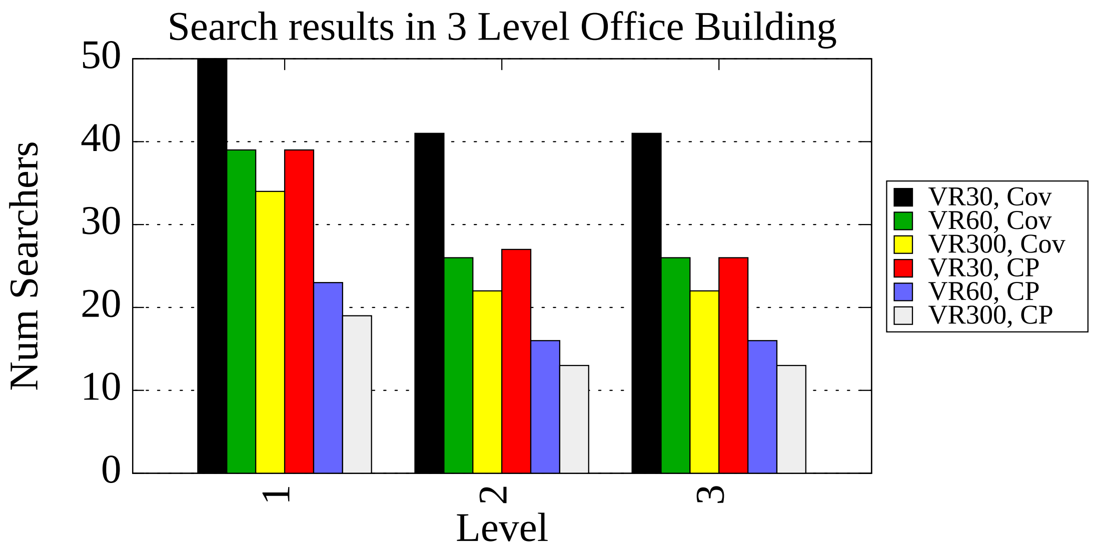


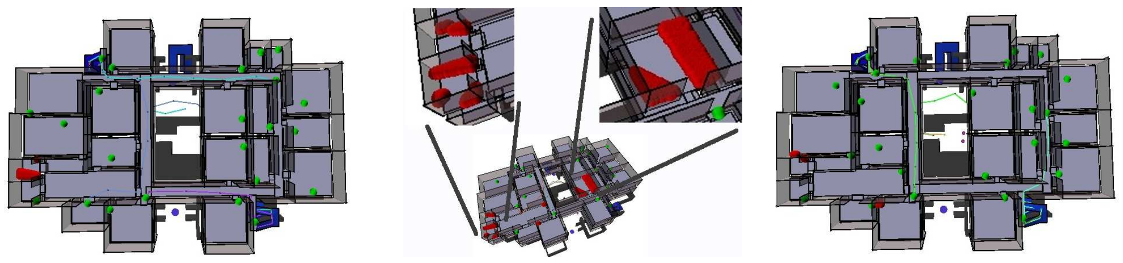
Figure 13: Search results in 3 level office building

to cover in the problem. When agents have a view radius of only 30 units, the number of searchers needed to cover the roadmap is nearly twice the number needed when the view radius is 300 units.

The number of agents needed to cover the roadmap with a view radius of 60 units is similar to when they have a view radius of 300. Although we do not have the ability to compute the optimal geometric coverage locations, we look at the number of locations needed to guarantee coverage on the third floor of this environment. By simple inspection, the number of coverage locations needed is shown in Figure 12(a) when an infinite view radius is used. These locations correspond to two searchers for the main four hallways (Locations 1 and 2), one for each main room (Locations 3–23) and two to guard the exits (Locations 24 and 25). These locations would guarantee a geometric coverage of the third level (with guarding of the exits). With a view radius of 60 units, an example of locations that are generated is shown in Figure 12(b). For the most part, the locations shown correspond to those in Figure 12(a) although one room (corresponding to Location 19) is not guaranteed to be covered. However, the underlying roadmap was covered.

Our approach results in fewer agents needed when the view radius is set to 300 units. This is because while the roadmap may be covered, the geometric representation of the environment is not guaranteed to be covered. An example of both good and missed coverage locations is shown in Figure 14. In this example, we generated 100,000 random samples on the third level of the environment. Samples that were not covered by the coverage locations are drawn in red. These locations are generally in the corners of rooms, however cases can occur where a room seems to be covered based on the roadmap nodes covering that room which may leave a large portion of the room uncovered (Figure 14(b)). If more complete coverage is desired, a more densely sampled map can be generated as in Figure 14(c).

We are able to use clearing paths in this complex environment to clear each level. It can be seen in Figure 13 that fewer agents are needed when using the clearing paths strategy. This is true at each level of the environment but especially when the agents have the more limited view radius (VR 30) in which nearly 30 fewer agents are needed to clear the first level. An example of the clearing paths generated can be seen in Figure 12(c).



(a) Good coverage with Basic Map

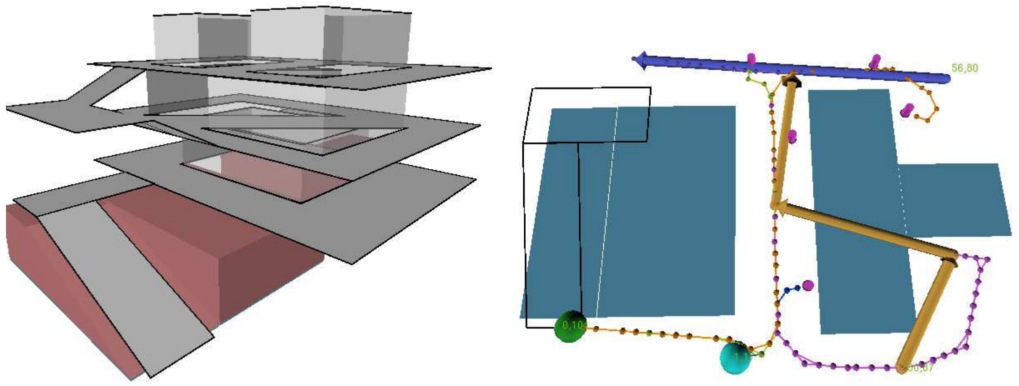
(b) Missed Coverage

(c) Good Coverage Oversampled Map

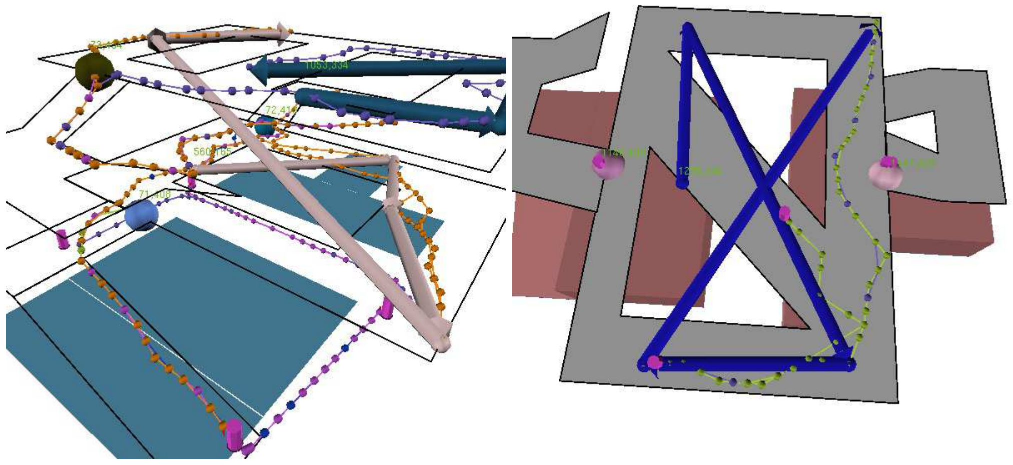
Figure 14: Coverage examples of 3D office example. Goal locations shown in green. Missed samples (out of 100,000 samples) shown in red for third floor.

c. Extended Examples

The level search strategies can be executed without requiring an underlying serial environmental graph which restricts clearing from one level to the next in either a bottom-up or top-down manner. This extension is done by starting from a set of initial levels being cleared and once cleared, traversing the environmental graph in a



(a) Example General Env. (b) Base Level and Strategy



(c) Level 1 and 3 (d) Level 2

Figure 15: Basic example showing the general level search. A connection is allowed in the environmental graph between the base level and the top (third) floor.

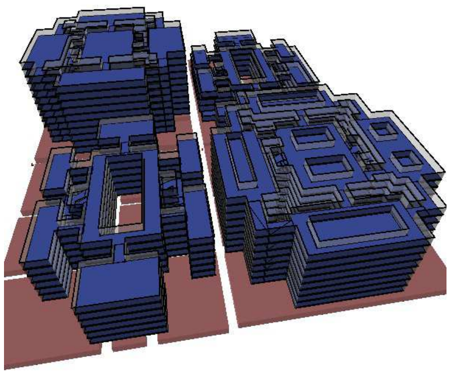
breadth first way to get the next set of levels to clear. In this way the environment is traversed until the strategies have been created at each set of levels which would guarantee a clearing based on the underlying roadmap.

In Figure 15(a), an example environment is shown which we then run the clearing strategy on to obtain a clearing. In this example, a connection is allowed between the base level and the third level with agents starting at the base level. In this way, after the base level is cleared (with the overall strategy for that level shown in Figure 15(b)) the next set of levels searched are levels 1 and 3. The strategy for these levels is shown in Figure 15(c). An interesting aspect to note is that one agent would be responsible for clearing a portion of level 1 and then continuing on to level 3. The final level searched would then be level 2, shown in Figure 15(d).

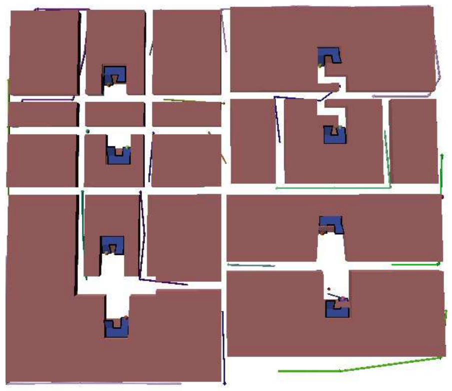
A much more complex example is shown in Figure 16(a). This environment is composed of four buildings. Connections are made in the underlying environmental graph between the base level and the second level of each building. The clearing strategies generated are shown in Figure 16(b)-(h). The strategies are built up until all the buildings are cleared. In Figure 16(h), the building with the fewest levels no longer has any more levels to search and the remaining buildings are left to be cleared. This is a much more complex and computationally intensive environment in which to generate these clearing paths, although our approach is still able to generate clearing strategies.

C. Agent Tracking Probability Model

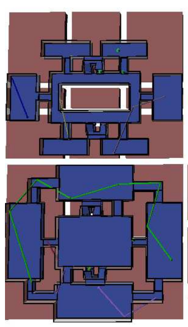
This section describes a probability model that can be used in order to track an agent of interest. The model exploits the roadmap representation of the environment to model the potential movements of the agent of interest.



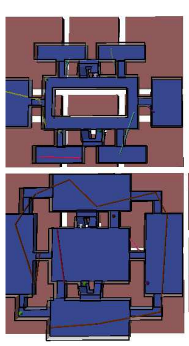
(a) 4 Building Env.



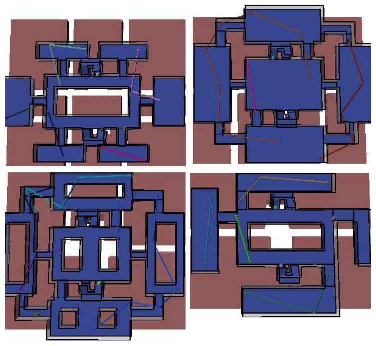
(b) Base Level and Strategy



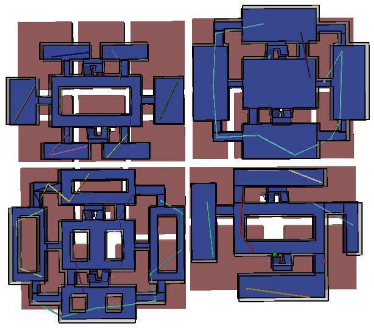
(c) Level 1



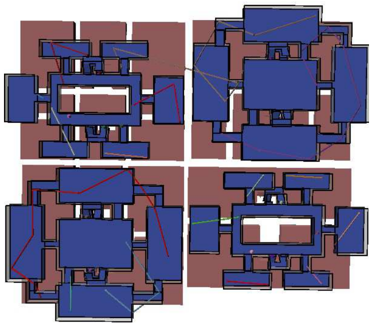
(d) Level 2



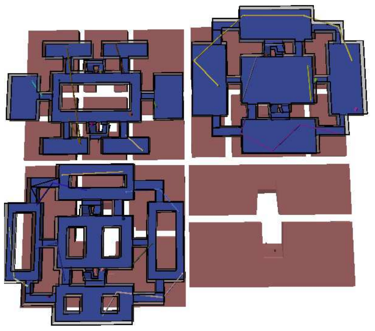
(e) Level 3



(f) Level 4



(g) Level 5



(h) Level 6

Figure 16: Example with four buildings. The underlying environmental graph as connections allowed between the base level and the first level of all the buildings.

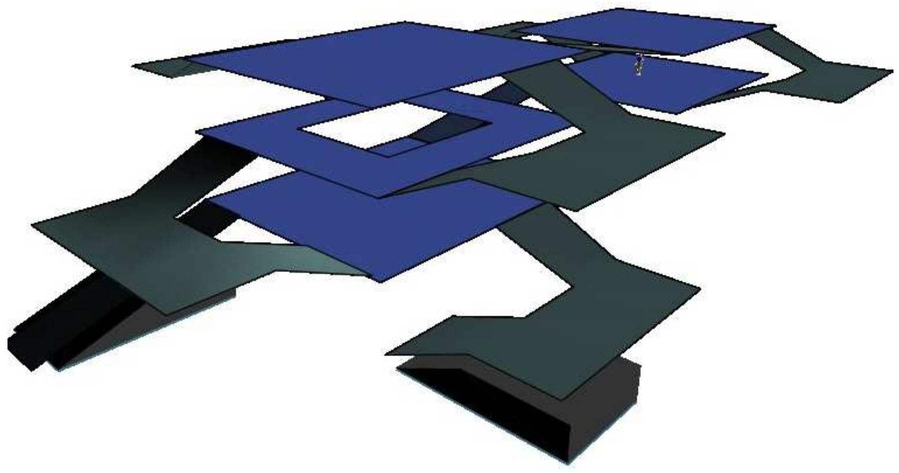


Figure 17: A multi-level building example with two columns of levels with passageways connecting the columns of levels.

This differs from prior approaches that use a probability model. First, we use an arbitrary roadmap that is not dependent on a regular discretization of the environment. Instead, since nodes are randomly sampled, the discretization is dictated by the roadmap. Secondly, the roadmap approach allows more realistic assumptions about agents' capabilities and limitations, considering where the agent believes the agent being tracked may move.

A Voronoi region is a partitioning of a space in which each region consists of points that are closer to the object (often a point) representing that partition than to any of the other objects [114]. We can consider each node of our roadmap to be the object representing a Voronoi region. Also, each node has a probability associated with it representing the probability that the agent being tracked is residing in that region.

We can consider each node of the roadmap to represent a disjoint subset of points: every point in the free space of the environment maps to the node closest to it. This mapping partitions the environment into regions represented by the roadmap nodes. The edges of the roadmap represent the available transitions an agent can make between regions. There is also an implicit self-edge, meaning an agent can remain in the region during a transition. Due to agent capabilities, agents in a common region represented by a node need not be visible to one another and agents visible to one another are often not in the same cell.

An agent in the environment exists in one Voronoi region at any time. Since the agent has bounded speed, it moves a finite distance in a single time step. After its movement, it is either in the same region it was previously in, or it has moved to an adjacent region. Each region can be thought of as a state. After every time step, an agent continues to be in its previous state or has transitioned to a neighboring state. Since each region is represented by a roadmap node, we can consider an edge between any pair of nodes to represent a transition between two states.

We can therefore model the movements of an unseen agent by assigning transition probabilities to each node n and its neighbors (called successors here) S. Probability is distributed in a simple manner. Some amount of the probability of a node, $P_{n,before}$ remains on a node, $P_{n,new} = P'_{n,new} + 1/(|S|+1) * P_{n,before}$. The rest will be distributed equally among its successors, $P_s = P'_s + 1/(|S|+1) * P_{n,before}$, where P_s is the new probability associated with the node for the updated model. This probability model is updated at each time step in our simulation. We define the temperature of a node n to be $T(n) = P_n + \sum_{s \in succ(n)} P_s$, where P is the probability at a node. This node is called a hotspot if $T(n) > \epsilon$, ϵ being a threshold value.

1. Tracking Examples

In this section, three basic comparative studies are shown: a) a comparison of search strategies, b) a basic comparison when using probability model, and c) a more complex scenario in a two level office building. All scenarios were allowed to run a finite number of time steps in our framework. There was no other stopping criteria for the simulation.

a. Scenario: Immobile Targets

This scenario takes place in a two column building, shown in Figure 17 with results shown in Table I. This scenario involves five pursuers searching for 100 immobile targets. The immobile targets are randomly scattered throughout the environment. The pursuers' roadmap is sufficiently sampled and connected to represent the free space of the environment. The pursuers use one of the searching behaviors as described in Section A. For this scenario, since the evaders are immobile we recorded the number of captures, and the percent of time these evaders stay hidden (averaged over all evaders).

Table I: We show how each searching behavior performed by comparing the number of captures and the percent of time the evaders stay hidden in a multi-level building example with two columns of levels. Results are run in the environment shown in Figure 17.

Search Behavior	Num. Captures	Per. Time Hidden
Basic	93.0	0.25
Frontier	99.6	0.21
Level	94.6	0.26
Probabilistic	83.0	0.31

In this scenario, we see how the various behaviors perform. The frontier behavior performed the best for this scenario. It captured on average 99.6 out of 100 targets. The frontier behavior performed better than the other various behaviors because it uses implicit communication to better explore the environment. Both the level and basic search strategies perform about the same. These do not use any communication, so some of the agents end up re-exploring areas already explored by their fellow pursuers. This explains the increase in percent of time hidden as compared to the frontier behavior, 0.26 and 0.25 as compared to 0.21 for frontier. The probabilistic search performs poorly in this scenario because it is modeling moving agents, and has no communication to share models between agents. Since the behavior models moving agents, it re-explores areas which it has visited more often than the covering or the level search might do.

b. Scenario: Tracking

In this scenario, we compare a basic search with and without the probability model to show the base effectiveness on one pursuer. One pursuer tries to capture 10 evaders. The pursuer is given a slight advantage of speed, while the evaders have an improved view radius. This experiment was done in a simple three level building environment, shown in Figure 18 with results in Table II. Here we compare not only the number of captures but also the total time spent chasing agents.

Table II: A comparison of search with and without the probability model in a three-level building example which is shown in Figure 18.

Search Strategy	Prob. Model	Num Captures	Total Chase
Basic	N	6.8	2264
Basic	Y	8.6	3630
Probabilistic	Y	5.9	3011

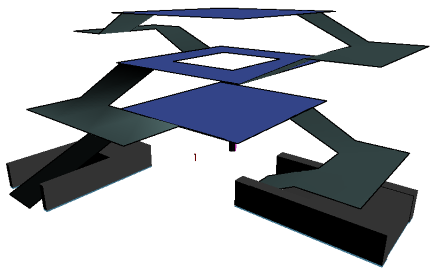


Figure 18: A three-level building example with multiple stairwells connecting successive levels.

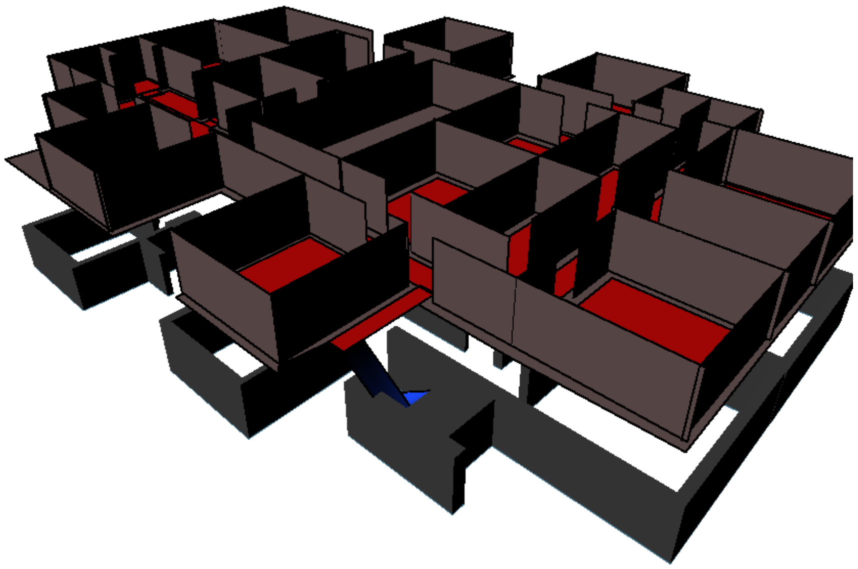
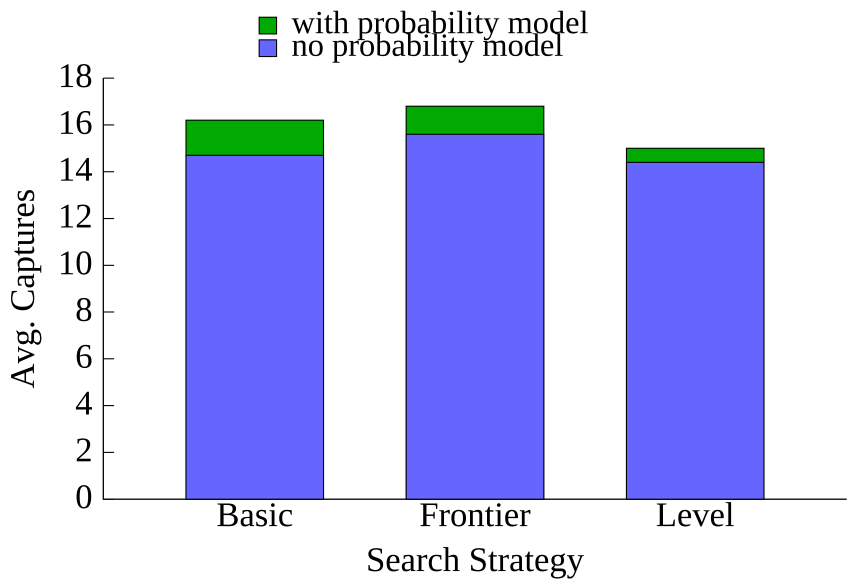


Figure 19: The complex two level office building example used to in testing the agent tracking probability model.

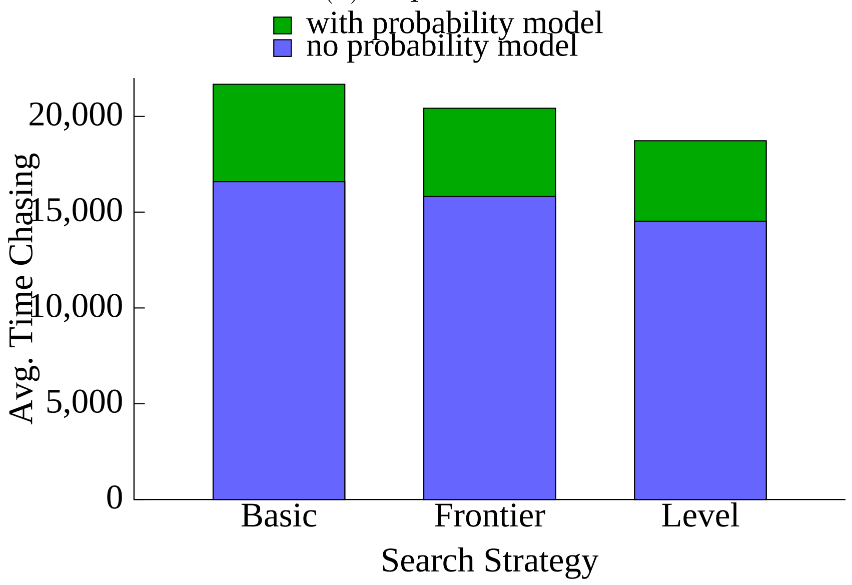
From this experiment we can see the base effectiveness of the probability model for tracking. The advantage of the evaders in view radius allows them to move more effectively to get out of view of the pursuer, allowing the model to show its effectiveness. We see an average improvement in the number of captures of 1.8. Also seen is an improvement in the total chase time. Agents with the probability model can chase agents better, and we see that Basic and Probabilistic searches using the model have better chases than without the model, 3630, and 3011 as compared to 2264.

c. Scenario: Office Building

In the office building scenario, five pursuers attempt to capture twenty evaders. Evaders have a speed advantage, but the view distances of the agents are equal.



(a) Captures



(b) Chasing

Figure 20: Comparison for searching behaviors with and without the agent tracking probability model in a complex office building, shown in Figure 19. In (a), the average number of captures is shown where using the probability model results in more captures. In (b), the time the pursuers spend chasing is shown with the probability model increasing the time agents end up chasing an evader.

Evaders can effectively move around corners and out of view readily, making captures highly difficult. This scenario takes place in a two level office building with many rooms shown in Figure 19 and results in Figure 20. We compare the number of captures and the total chasing time of the three search strategies with and without the probability model.

The effectiveness of the tracking probability model can be seen in Figure 20. In all searching behaviors, using probabilistic tracking improves the number of captures and the chase time. The level search did not see as much improvement in the number of captures as the others did, but it still increased the total chase time. This increase in chase time shows the pursuers are better able to predict where the evaders may be so they can resume chasing.

From each of the experiments shown, we see that while using the probability model alone may not result in a good searching strategy, when combined with traditional searching strategies, it becomes an effective method of tracking and capturing, even in highly complex multi-level environments.

CHAPTER V

APPLICATION FOCUS: PURSUIT-EVASION*

A. Motivation, Goals and Objectives

The chase between a pursuer and an evader is an exciting display of dynamic interaction and competition between two groups, each trying to outmaneuver the other in a time-sensitive manner. When this is further extended to include teams of pursuers, this problem combines cooperative problem solving with fast-paced competition in scenarios that are often seen in the real world. Both of these aspects have a wide range of applications, and are one reason for the prevalence of the pursuit/evasion problem in a wide range of disciplines.

While a great deal of work has been done in the area, each category of the pursuit/evasion problem has a set of restrictions and assumptions about the environment or the agents involved to make it more manageable. For example, in the case of graph-based approaches, the agents are limited to moving on the nodes and the edges [93]. In geometry based approaches, the environment is often limited to being polygonal [18]. In fact, the types of visibility restrictions that agents can handle is often one of the main limiting factors [104, 105]. While providing a means for developing a more complete solution, these assumptions limit the scope of the problem and consequently the applicability of the techniques.

*Part of the data reported in this chapter is reprinted with the kind permission of Springer Science+Business Media from "Toward Simulating Realistic Pursuit-Evasion Using a Roadmap-Based Approach" in *Lecture Notes in Computer Science*, Vol. 6459, by S. Rodriguez, J. Denny, T. Zourntos, and N. M. Amato, 2010, Springer, Berlin. Copyright 2010 by Springer. [22] Part of the data reported in this chapter is reprinted with the kind permission of IEEE from "Toward Realistic Pursuit-Evasion Using a Roadmap-Based Approach" in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, by S. Rodriguez, J. Denny, J. Burgos, A. Mahadevan, L. Murray, A. Kodochygov, T. Zourntos, and N. M. Amato, 2011. Copyright©2011, IEEE. [23]

A contribution of this work is our versatile approach for agent-based pursuit-evasion games that extend the range of applications that we are able to handle. We are able to extend the scenarios to 3D-scenarios that require more complex visibility checks that consider obstacles, surfaces, and agents in the environment. This 3D extension is critical for handling the more realistic situations we are interested in studying. The only work of which we are aware that uses 3D visibility during pursuit evasion is a recent approach using height maps on terrains [108].

In our framework, agents are equipped with a roadmap (used for navigation), a set of capabilities, and behaviors which define the actions they will take given their knowledge of the environment. The agents and behaviors are versatile and tunable so that the kinds of agents and the behaviors being applied can be adjusted with a set of parameters that define them. This general representation of the problem allows us to use the same approach for pursuit-evasion in very simple environments as well as a range of more complex scenarios that have received much less attention. We demonstrate our approach on mobile robots and in simulated scenarios involving pursuit-evasion and tag on terrains, in multi-story buildings and in crowds.

B. Problem Definition

The traditional pursuit/evasion problem that we consider consists of one set of agents, the pursuers H, attempting to capture another set of agents, the evaders E. An evader $e \subset E$ is considered captured if a non zero set of pursuers $h \subset H$ is within some predefined distance $d \geq 0$ of e. The pursuers and evaders may have different levels of environmental knowledge, sensing abilities and capabilities, which affect the overall time to capture. The pursuit we study is a complete chase which consists of searching for a target agent and maintaining visibility until a capture can occur. The evading agents take adversarial actions which involve fleeing from the pursuing agents that have been detected and improving on hiding locations when undetected.

1. Our Approach

In an attempt to explore a range of the entire pursuit/evasion spectrum we have developed an approach that allows for quick and easy development of strategies for different versions of the problem. Our approach gives the user full control over a number of parameters including agent and behavioral properties. This ability allows us to explore a wide spectrum of problems and provides the basis for a framework that provides flexibility and control to the user over a range of interesting pursuit/evasion games and simulations.

C. Behavior: General Pursuit

Our general pursuit strategy, shown in Algorithm C.1, has four stages: location of the target, creation of a pursuit plan, acting upon the plan, and then evaluating pursuit status. A wide range of pursuit strategies can be employed by enabling certain customizations of the general pursuing strategy. As an example, the most basic pursuit strategy will chase a target, once located, in a direct path until it either captures the target or the target is no longer visible.

A target may be available to an agent through either direct observation or through communication with nearby agents (communication being a capability that may be enabled or not). Determining which potential target to pursue involves selecting the target agent from all potential targets which is most likely to result in a successful capture. While we typically use distance as the criteria for selecting a target to pursue, other factors could be used such as the size, health, or stamina of

Algorithm C.1 General Pursuing Algorithm

```
Input: Agent s and perceived data (neighboring agents)
1: if a target exists then
2:     Determine which potential target to pursue
3:     Create a pursuit plan
4:     Selection individual pursuit roles
5:     Execute plan and pursue the target
6:     Determine status — terminate on success or failure
7: else
8:     search for target
9: end if
```

the agent. In creating a pursuit plan, agents may take into account behavioral factors, described further in the enhancements section. The plan created should take the agent from its current location to a location that will improve its chances of capturing the evading agent. The plan is executed until it is no longer valid (e.g., the target agent is no longer within range of the final goal of the plan or the target is no longer available).

1. Enhancements

The basic pursuit strategy consists of an agent chasing a target toward the target's current location. Many improvements to this basic pursuit strategy can be enabled which potentially improve the success rate of a group of pursuing agents. Pursuing agents can head off an evading agent by planning their path to intercept it; we refer to this as an intercepting behavior. Pursuing agents can also avoid over re-planning by only creating a new pursuit plan when the evading agent has moved far enough away from the previously planned pursuit route. This can prevent erratic motion of the pursuing agent.

Another enhancement that cooperating pursuing agents can be given is the ability to flank an evading agent. Pursuing agents attempt to flank by spreading out behind the evading agent while chasing. This restricts the evaders motions when an evasive action is needed (for example near an obstacle or boundary). Our flanking enhancement allows agents in a pursuit group to plan their path to flank positions around a target; this corresponds to selecting individual roles from Algorithm C.1.

Enabling communication between pursuing agents can improve a pursuing agent's effectiveness. For example, communication can alert a pursuing agent to the location of an evading agent. Communication between agents can also allow for one agent, the leader, to request other agents to follow that agent when searching the environment. This type of communication potentially allows for more coordination between pursuing agents which is necessary in some pursuit scenarios.

D. Behavior: General Evasion

Our general evasion strategy, shown in Algorithm D.1, attempts to reduce the likelihood of being or staying visible to opposing agents. In this behavior, an agent generates hiding locations, usually within a certain range, in the environment. These positions are then evaluated, or scored. Different scoring functions are used depending on whether or not pursuers are present. A new hiding location can then be selected if one is found such that the score of the new location achieves some predefined percentage increase over the score of the current hiding location being used. The new goal and path are updated if a location is found that sufficiently improves the score.

Algorithm D.1 General Evasion Algorithm

```
Input: Agent s and perceived data (neighboring agents)
1: if Evasive plan is needed then
2:     Determine strategy for evasion
3:     Begin execution of the strategy
4:     Adapt strategy to current situation
5: else
6:     attempt to improve hiding location
7: end if
```

1. Scoring Hiding Locations

In the case of an evading agent being undetected by pursuing agents, the goal of the evading agent is to decrease its likelihood of being detected. While the framework is flexible and can accommodate many options, in the current implementation, the score for each hiding location evaluated is determined by the visibility to all other potential hiding locations. A hiding location with low visibility is preferred over one with high visibility, i.e., a location that is blocked by objects in the environment is preferred.

When an evading agent detects one or more pursuing agents, five criteria are used for scoring hiding locations:

- **Distance to Pursuers (α):** This value will determine how much an agent prefers locations that are further away from the visible agents. The distance a hiding location is from all visible pursuing agents can be computed and then scaled by the evading agent's view radius.

- **Direction to Pursuers (β):** The direction to a hiding location is computed and scored so that directions away from pursuing agents are preferred.

- **Distance To Boundary (γ):** The distance to the boundary of a hiding location can be factored in to the final score by weighting the factor when the potential hiding location is within some predefined distance to the boundary. This factor will allow agents to prefer hiding locations away from the boundary and reduce the likelihood of being trapped by the boundary.

- **Visibility Restricted Surfaces (δ):** The weight of this factor can be determined by how much a hiding location is obscured by surfaces in the environment.

- **Visibility Restricted by Other Agents (ϵ):** Other agents can be used when scoring hiding locations. A hiding location can be weighted such that an agent prefers a location that has other agents in between pursuing agent locations and the hiding location.

An evasion strategy can be defined to be a weighted set of these factors, $< \alpha, \beta, \gamma, \delta, \epsilon >$ so that evading agents can be tuned to prefer certain kinds of evasion strategies. These tunable factors allow for our adversarial evaders to exhibit potentially very different behaviors and are different from many approaches that assume the evading agents have a set evasion strategy. As an example, for two evasion strategies, S1 and S2, with all factors being equal except $\gamma_1 \ll \gamma_2$ would result in S2 avoiding the boundary more than S1.

E. Experimental Results

We present results for pursuit/evasion scenarios that have not received as much attention by other researchers. These involve more complex environments where the visibility of the agents is restricted by the environment. We also show examples where some interesting interaction takes place between the groups of agents being simulated. In the following, we show results in terrains, crowds, and a multi-level environment. We also show an example of interacting behaviors where agents play a game of tag with varying pursuit and evasion behaviors. Results show the average proportion/number of captures, the average time to capture, minimum and maximum time spent chasing among the pursuing agents, minimum and maximum time hidden for the evading agents and the proportion of time hidden. These values give insight into how involved each agent was in the pursuit process both in the chasing aspect and in the searching of the environment. The maximum number of time steps used throughout any simulation is 5000.

1. Terrain

The terrain environment, shown in Figure 21, consists of multiple hills and valleys with dimensions of 195 by 150 units and a maximum height 18.5 units. This provides numerous hiding locations for the evading agents. In the first scenario, 5 pursuing and 5 evading agents react to one another in the pursuit evasion game until the pursuing agents capture the evading agents. The pursuit behaviors tested in this environment are: basic pursuit, intercepting behavior, a flanking behavior and with sharing targets between pursing agents which are also intercepting. Pursuing agents have a maximum velocity of 4, a view radius of 40 and heights of 6 units. Evading agents have a maximum velocity of 6, a view radius of 50 and heights of 3. In this

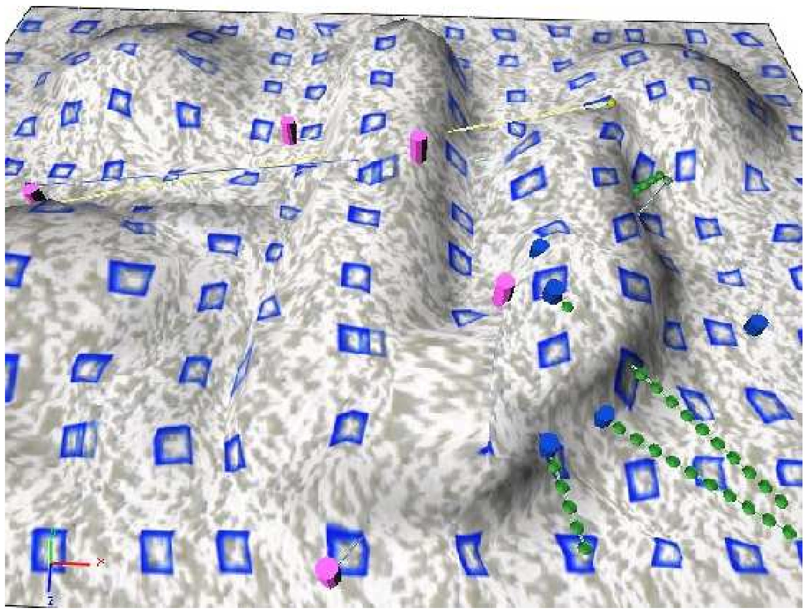
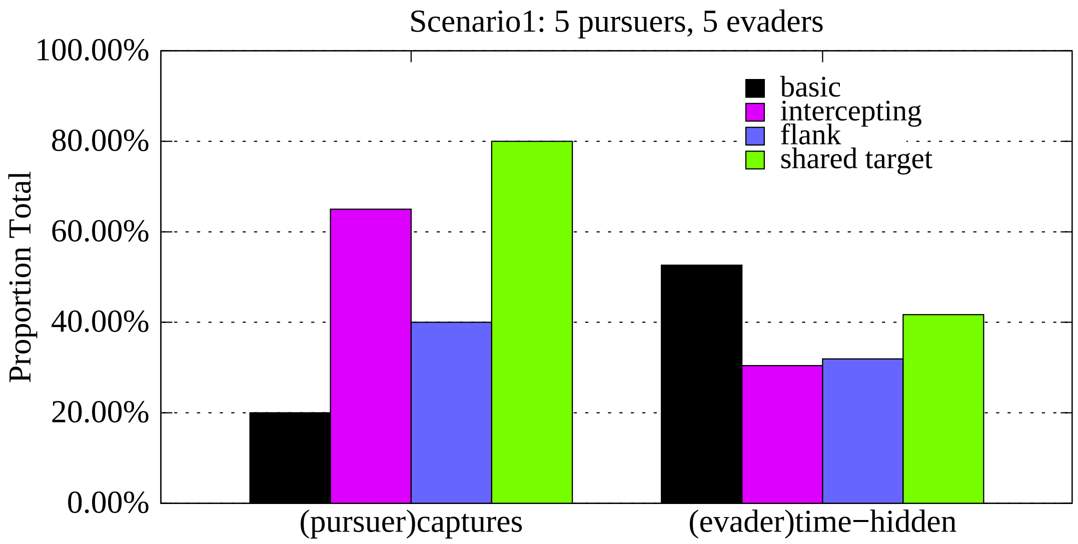
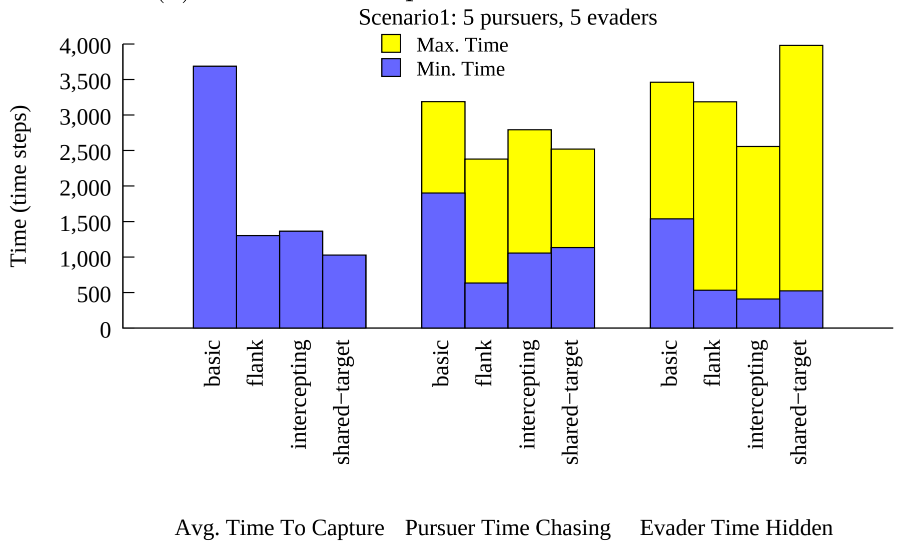


Figure 21: Terrain environment used to compare behavior types, with hills and valleys which can obscure visibility in the environment.

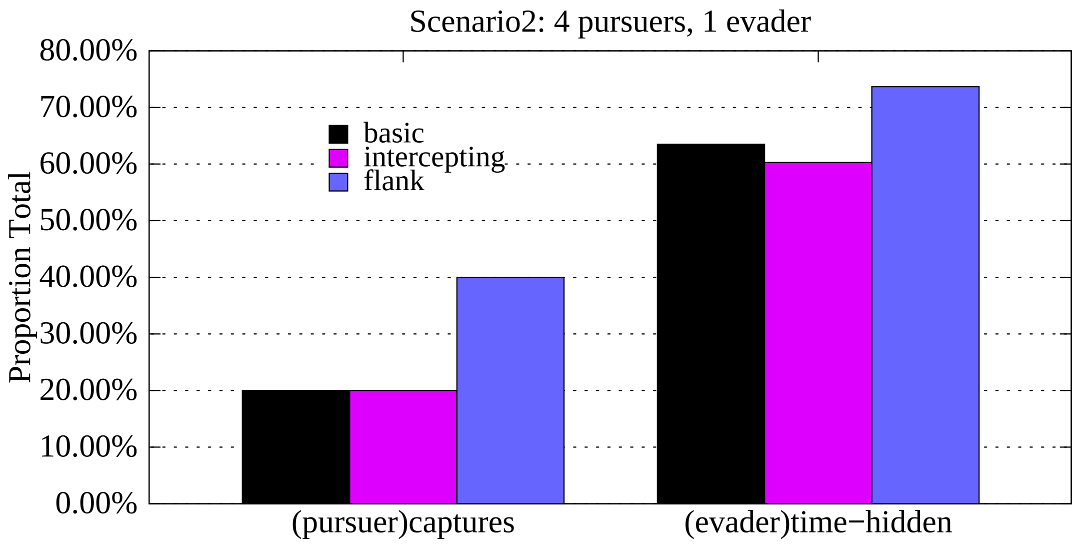


(a) Scenario 1: Captures and Time Hidden

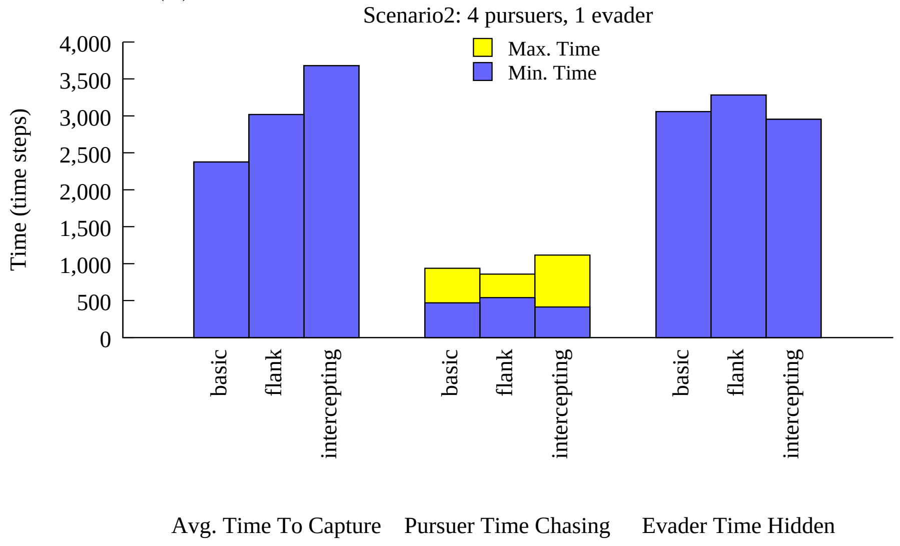


(b) Scenario 1: Times

Figure 22: Pursuit/Evasion on a terrain with hills and valleys obscuring visibility in scenario 1 with 5 pursuers and 5 evaders. Environment shown in Figure 21. (a) Illustrates proportion of agents captured and time hidden . (b) The average time to capture and minimum/maximum times for chasing and time hidden.



(a) Scenario 2: Captures and Time Hidden



(b) Scenario 2: Times

Figure 23: Pursuit/Evasion on a terrain with hills and valleys obscuring visibility in scenario 2 with 4 pursuers and 1 evader. Environment shown in Figure 21. (a) Illustrates proportion of agents captured and time hidden . (b) The average time to capture and minimum/maximum times for chasing and time hidden.

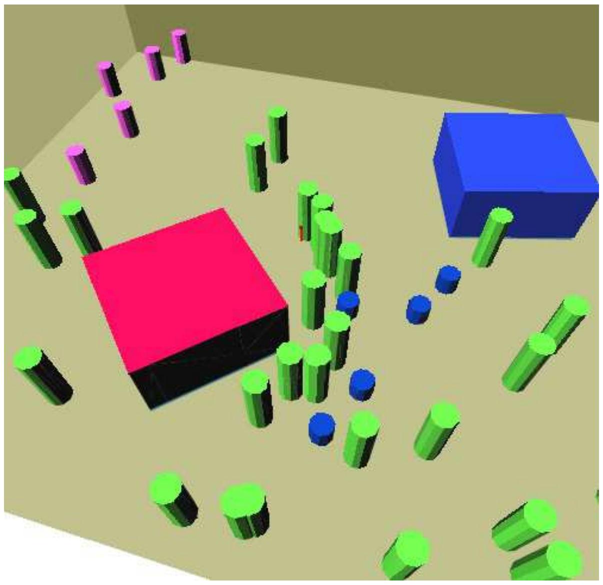
scenario, a slight advantage is given to the evading agents where they are faster and can sense more of the environment than the pursuing agents.

In Figure 22, for Scenario 1, it can be seen that using only basic pursuit results in fairly poor performance since the evading agents can easily outmaneuver the pursuers. In this table, the proportion of captures is an average of the proportion of captures per run. When pursuing agents attempt to intercept or head off the evading agents, the performance improves overall resulting in more captures and lower time to capture. In this case, adding the flank behavior does not improve results over using an intercepting behavior. Also, sharing a target among the pursuing agents greatly improves the pursuing agents' effectiveness.

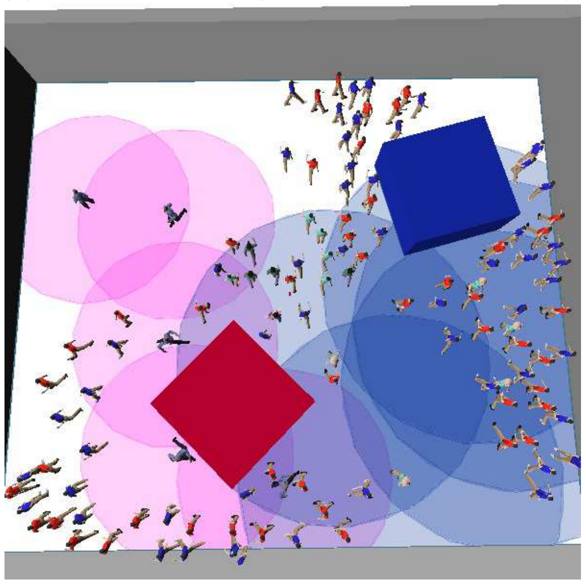
In Figure 23, results are shown for the second scenario where 4 pursuers attempt to capture a single evading agent. A much greater advantage is given to the evading agent by giving it a faster maximum velocity. Pursuing agents have a maximum velocity of 2, view radius of 80 and height of 6 units. Evading agents have a maximum velocity of 9, view radius of 80 and height of 3. In the table, the proportion of captures refers to an average of successful captures within the time limit over all runs. In this scenario, both sets of agents have similar sensing abilities, being able to detect a large portion of the environment that is not blocked by obstacles/surfaces. Both basic and intercepting behaviors have fairly low captures with the flank behavior outperforming both. This is an example where coordination and communication between agents is needed given the evading agent's superior maneuverability. In each scenario, utilizing some effective enhancements allows agents to restrict the number of evasive actions.

2. Crowd

In an environment with a crowd, the evading agents can hide among crowd members. While the test environment is very basic, including the crowd in the problem adds



(a) Simple Crowd (Computational Model)



(b) Complex Crowd (Animated Model)

Figure 24: (a) Pursuit/Evasion in a crowd of other agents (5 pursuing, 5 evading, 30 in crowd). Pursuing agents shown in Top Left corner of image, evading agents shorter, blue cylinders at center and crowd are tall, green cylinders. (b) A more complex model shown with a crowd of 100 members. Pursuers are shown with smaller view radius (pink) and evaders have a larger view radius (blue).

Table III: Pursuit/Evasion in a crowd of other agents (5 pursuing, 5 evading, 30 in crowd). Pursuing agents shown in Top Left corner of image, evading agents shorter, blue cylinders at center and crowd are tall, green cylinders. Environment shown in Figure 24.

Scenario	Num Capt.	Time To Capture	Min/Max Time Chasing	Min/Max Time Hidden	Prop. Time Hidden
basic	3	1013.2	19/376	1152/4727	0.5889
flank	4.5	770.5	144/458	911/3959	0.4418
intercepting, share target	5	448.4	126/543	418/1744	0.4104

another level of complexity. The environment dimensions are 110 by 100 units with 30 members of the crowd tall enough to obscure visibility, 5 pursuing and 5 evading agents, shown in Figure 24(a). The pursuit behaviors tested are: a) basic pursuit, b) a flank behavior and c) using an intercepting with shared target behavior. Results are shown in Table III for the various behaviors tested. The basic pursuit again performs the worst with agents that perform a flank behavior being even more effective. As in the previous example, using an intercepting behavior along with a shared target performs the best. Both enhanced behaviors result in the much better performance than basic pursuit with the highest number of captures and lower average time to capture. These behaviors among pursuing agents also alert other agents to a target that may end up hiding in the crowd, but the subsequent searching in the crowd often resulted in a capture.

A more complex version of the problem can be seen in Figure 24(b). In this example, the crowd is composed of 100 members, again with 5 pursuing and 5 evading agents. The larger crowd acts as dynamic obstacles for the evading agents to hide behind. An animated screen shot is shown in Figure 24(b). The pursuers are shown with a smaller, pink view radius and evaders with a larger, blue view radius. This shows how much more complex the scenario can become and the flexibility of our system.

In Table IV, results are shown for a scenario similar to the one described in Figure 24(b) except with pursuing and evading agents having the same view radius. This allows both sets of agents to react when the other becomes visible. We compare the evading agents with and without using the crowd in scoring their hiding locations. It can be seen that with a crowd of this size, filling up a majority of the environment, using the crowd to hide can be useful. This can be seen in the overall average time to capture being higher when using the crowd in scoring hiding locations. The other interesting aspect is that evaders using crowd hiding spend a smaller proportion of the time being hidden than evaders that do not use crowd hiding. However, overall, the agents using crowd hiding take longer to be captured as they can get lost in the crowd making finding them more difficult for the evading agents.

3. Multi-Level Environment

The multi-level environment shown in Figure 25 is a very complex environment consisting of two main floors connected to each other in two ways. One connection is a long ramp, in the background in the environment shown in Figure 25. The other connection consists of multiple surfaces ramping up to the second floor. Two basic

Table IV: Pursuit/Evasion in a crowd of other agents (5 pursuing, 5 evading, 100 in crowd). Results are shown with and without evading agents using the crowd to hide in.

Scenario	Time to Capture	Time Chasing	Prop. Time Hidden
with crowd hiding	2187	2365	0.227
without crowd hiding	1656	2446	0.284

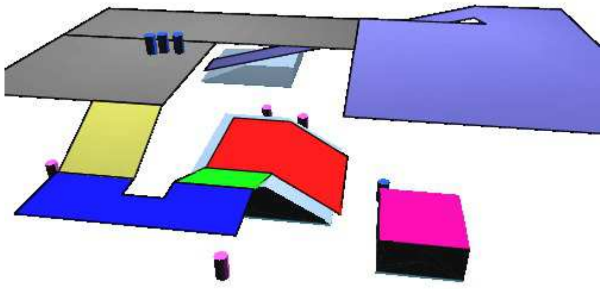


Figure 25: Multi-level environment used to compare behavior types

Table V: Pursuit/Evasion in a multi-level building example shown in Figure 25

Scenario	Num Capt.	Time To Capture	Min/Max Time Chasing	Min/Max Time Hidden	Prop. Time Hidden
basic	5	784.8	1103/2453	728/1349	0.2428
shared target	3	1236	284/1267	1457/3921	0.5082

pursuit behaviors tested are shown in Table V, a basic pursuit and pursuit with a shared target. It is interesting to note that in this example, sharing a target among pursuing agents does not result in better performance. In this environment, agents independently searching the environment increase the chance that the pursuing agents will trap agents on the upper floor. This reduces the potential for the evading agents to find valid evasive actions. Agents that are independently searching also reduce the chance of an evading agent staying in a hiding location for very long.

4. Tag Game

The game of tag played in this scenario is a very simple extension of the standard pursuit/evasion game. In this scenario, two sets of pursuing agents with the same capabilities start out with predefined pursuit behaviors. The 25 evading agents in the environment, shown in Figure 26, have the goal of avoiding both sets of pursuing agents. Once an evading agent is captured, it becomes part of the capturing agent's team and begins executing the same pursuit behavior. In Figure 27, pursuit behavior comparisons are shown for basic vs basic pursuit, intercepting vs intercepting pursuit, intercepting vs basic pursuit, and flank vs basic pursuit behaviors in the environment. We initialize each set of pursuing agents with two pursuers. The numbers shown in Figure 27 reflect (a) the final number of agents performing the behaviors, and (b) the average time to complete the game.

When each type of pursuit behavior is tested against the other, on average the same number of agents ends up in each pursuit group. When comparing intercepting against basic pursuit behaviors, it can be seen that many more agents end up in the intercepting behavior group which shows that using a intercepting behavior is much more effective in this scenario. Another interesting aspect is that when both pursuit behaviors use an intercepting behavior, the average time until the game is complete

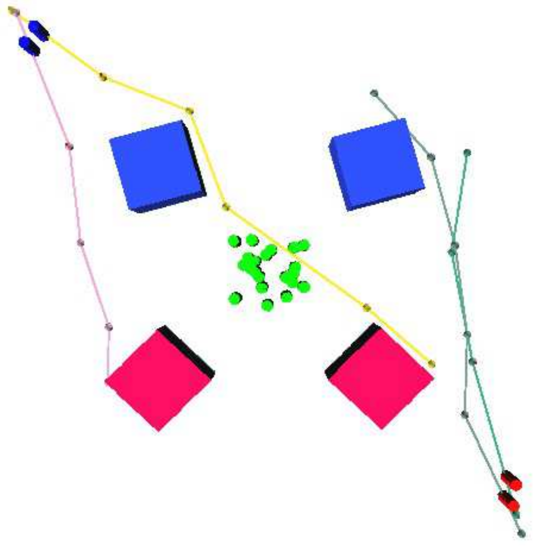
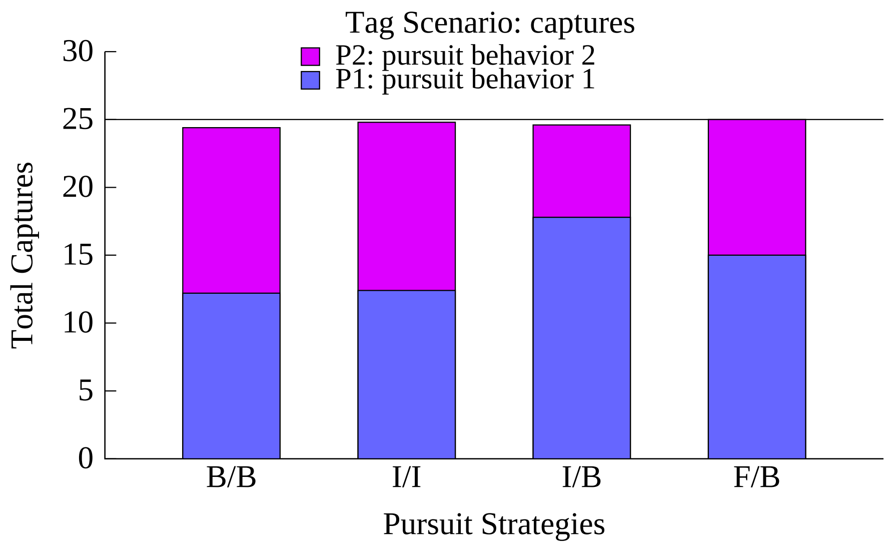
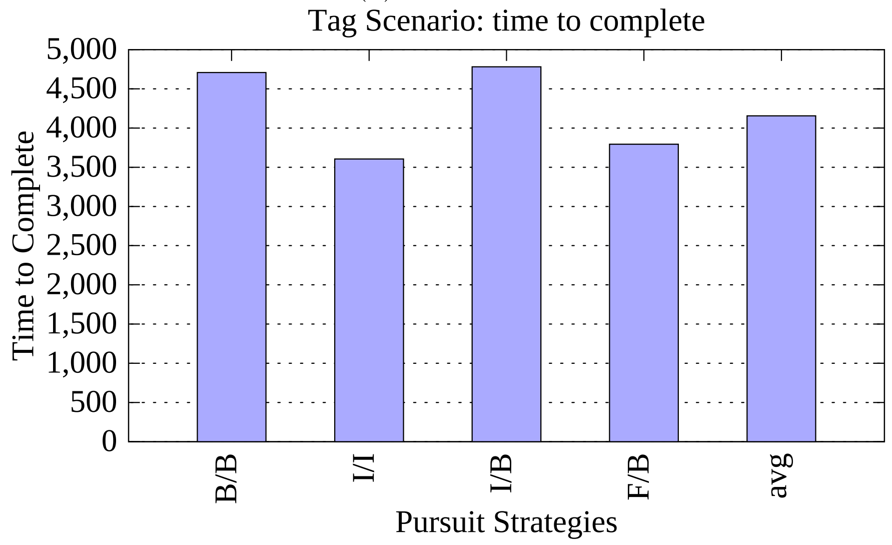


Figure 26: Pursuit/Evasion applied in a game of tag. Two pursuing agents are initialized at the corners of the environment and start in each type of behavior. They then attempt to capture an evading agent which when captured becomes part of the pursuing agent's team, executes the same behavior and can be used to capture.



(a) Captures



(b) Time to complete

Figure 27: Pursuit/Evasion applied in a game of tag. Two pursuing agents start in each type of behavior and attempt to capture an evading agent (initially 25 agents are evading). An evading agent that is captured becomes part of the pursuing agent's team and becomes part of that group and executes the same behavior. The pursuit behaviors shown are: (B) basic, (I) intercepting, and (F) flank. (a) The final number of agents in each pursuing agent behavior. (b) The time at which behaviors finish capturing agents before the time limit is reached. Environment shown in Figure 26.

(all agents are caught) is much lower, also showing the effectiveness of this pursuit behavior.

While using an intercepting off behavior results in the most captures, there are cases where agents are left evading. When agents are equipped with a flank behavior more agents end up performing a flank behavior than basic pursuit. These examples show the flank behavior results in better performance overall. This can be seen in leaving no agents evading and a low time for completion of the game. This is in part due to agents performing a basic pursuit being assisted by the agents performing the flank behavior. While some agents may be flanking an evading agent, basic pursuit agents are able to pursue and take advantage of the flanking agents restricting the evading agents potential evasive actions.

CHAPTER VI

APPLICATION FOCUS: EVACUATION PLANNING*

A. Motivation, Goals and Objectives

Behavioral based evacuation simulations allow for someone to study agents performing pre-defined evacuation behaviors without having to see these behavior in practice. The simulated behaviors should, however, be based on actual behaviors that have been observed. In the case of evacuation planning, the evacuation of an environment can be studied with different behavioral, environmental and interactive conditions. Our work in evacuation planning has focused on incorporating interaction with control or directing agents which may influence how the evacuating agents perform the evacuation. An important focus of this work is to develop these control behaviors, where one group of agents actively tries to control or direct the movement of another group of agents, which allows for more complex evacuation scenarios to be studied. The usage of our general framework also enables us to study more complex environments than most systems can handle.

The overall goal is to develop an interactive planning and training tool for crowd-based behaviors of large numbers of interacting agents that supports a variety of evacuation situations. Realistic simulations for evacuation planning must consider the behaviors of the (groups of) agents and how these entities interact with each other. The behavior of agents and groupings of agents can also add a level of accuracy and detail, which is necessary when studying real world situations. Our focus on the

*Part of the data reported in this chapter is reprinted with the kind permission of IEEE from "Behavior-Based Evacuation Planning" in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, by S. Rodriguez and N. M. Amato, 2010. Copyright©2010, IEEE. [26]

control behaviors allows us to consider some aspects of evacuation scenarios that have so far not been extensively studied.

While a great deal of work has been done on large-scale multi-agent behavior and evacuation planning, we present some novel techniques for this problem. The main contributions of this work include:

- A motion planning inspired formulation of an Evacuation Planning Problem;
- An integrated behavioral agent-based and roadmap-based motion planning system that supports interaction with control and directing behaviors;
- Support for customization based on grouping and environmental factors;
- Evacuation in complex 3D environments using a roadmap-based approach;
- Heterogeneous agent populations;
- Incorporating different forms of communication to improve environmental usage.

Other approaches to the evacuation planning problem are often restricted to certain aspects of the problem. We define the problem in a general way to include evacuation behaviors that can be used, area types that should be considered, grouping restrictions that may exist and interaction that may occur. One set of behaviors that we focus on, controlling behaviors, requires cooperation between the directing agents and the agents that are cooperating. We allow for many kinds of interaction including intra- and inter- group interactions. This interaction between agents is needed for many behavioral types and allowing a variable level of cooperation between groups and coordination within groups can vastly affect the outcome of a scenario being studied. Our behavior framework is dynamic and includes having behaviors general enough to be applicable to a single agent or a group of agents. This will enable the group of agents performing the behavior to work cooperatively and share goals from the behavior. Our group hierarchy, which organizes the agents and subgroups, has been extended such that behaviors can be applied at any point in time and to groups at any level of the hierarchy. This can help in achieving cooperation and in reducing computation, where groups at a lower level of the hierarchy can benefit from computation done at the higher levels and behaviors can be applied periodically.

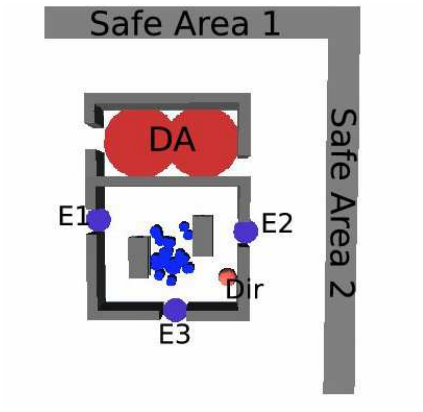
The system we describe is tunable so different environmental and behavioral conditions can be tested as well as varying the population types in complex environments. We expect that by looking at this problem from a behavioral, evacuation planning aspect, it will allow us to further explore cooperative and complex behaviors. We are able to simulate these evacuation scenarios in complex 3D environments by utilizing a general roadmap-based approach.

B. Overview of System

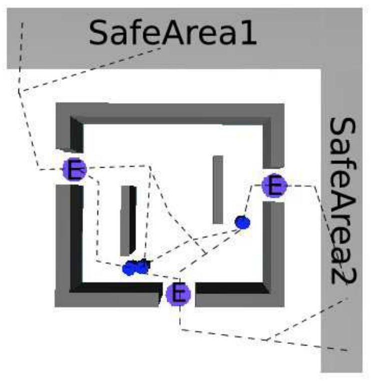
1. Problem Definition

We propose a generalized approach for studying the evacuation planning problem. We first describe the basic problem and then some extensions.

The basic problem can be stated as follows: Given an environment composed of polygonal obstacles, surfaces and N agents, $A = \{a_1, a_2, ..., a_N\}$, in an enclosed area ($EA$), find a valid evacuation plan for each $a_i \in A$ satisfying given constraints. These constraints (some shown in Figure 28) deal with areas that should be avoided and areas that evacuation routes can and should pass through. The physical representation of our environment consists of obstacles in the environment, surfaces that the agent can move on, valid transitions between these surfaces and a bounding box restricting



(a) Labeled Example



(b) Example Routes

Figure 28: In (a) an example environment is shown where an evacuation planning problem can be studied. Evacuation agents, shown in blue, are clustered in the lower room. Exits are labeled E# along walls of the rooms. The safe areas are labeled along the boundaries. A dangerous area is labeled DA in the second room. A directing agent is placed in the bottom, right of the lower room. In (b) potential evacuation routes to safe areas are shown.

the available area. The simulation is initialized with different sets of agents, which are deployed at predefined locations and equipped with different behaviors (i.e., evacuation or direction). In our simulations, we can define agents with varying capabilities and initial conditions.

This most basic form of the problem involves each agent finding a path through the environment from their starting location through an available exit and finally, to a safe area. This is similar to the motion planning problem where the objective is to find a path from a start to goal configuration that avoids obstacles. The evacuation route that an agent selects when only considering potential exits may be based solely on distance. A safe area is a region, outside of EA, that is used when generating a final goal location.

Dangerous areas may exist in the environment that the agents should avoid if possible. When evaluating paths, while considering these areas, a potential route that passes through a known dangerous area should be considered less desirable than a path that is clear of the area. In the case that dangerous areas are unavoidable, then routes that minimize the intersection with these areas should be most desirable.

In this form of the problem, we assume the agents have shared knowledge. This includes their knowledge of the environment, areas present and a shared roadmap, which the agents may use when generating evacuation routes.

Dynamic areas. Dynamic areas can appear at any time and their shape can alter as the scenario progresses. In actual evacuation scenarios these areas exist and should be considered in a general framework. One example dynamic area is a congestion that may occur as too many agents are present. Another example is a toxic spill, an area which may expand. While we do not consider these areas in this paper, it is something we are interested in studying.

Grouping. During an evacuation agents may be grouped with other agents.

This is a key constraint often overlooked in many approaches. This constraint requires that agents that are grouped to stay within some predefined range of one another while moving through the environment. This can represent a familial tie or assistance provided between agents.

Direction. Another key aspect that is often not supported in many evacuation planning approaches is the ability for one group of agents to direct the evacuating agents. The directing agents can represent a number of direction types including barriers or agents that can provide local or global information. Local information can consist of a nearby exit or safe area to avoid while global information can be many areas to avoid.

Heterogeneous Agent Knowledge. The agent's knowledge of each of these areas and their communication with other agents will effect the evacuation routes that are generated. An agent can compute an evacuation route with it's current knowledge but this route should be updated when the agent's knowledge changes. The knowledge an agent has about the environment can be either predefined, observed or communicated. This can also include having different knowledge about how to navigate through an environment, which can be represented by varying roadmap quality.

These different aspects are key to being able to create a general evacuation planning framework. In the following we will describe in more detail the approaches we have attempted and think should be supported.

C. Behavior: Evacuation

Agents planning a path to a safe location have to take into account different environmental aspects. This includes considering available safe exits, potential dangerous regions, locations that are considered safe and any other subgoals that must be reached along an evacuation route. The roadmap is well suited for finding these kinds of paths. Using path evaluation and roadmap re-weighting, paths can be found that satisfy constraints that are known to an agent.

1. Route Selection

A roadmap can be used to extract a safe path through the environment. A path can be extracted from the roadmap with the lowest weight (where weights can represent anything from distance to hazard levels).

Selecting a safe route through the environment, can be done with consideration to exits and safe areas in the roadmap. An overview of this process is shown in Algorithm C.1. An illustration of this process in a simple environment is shown in Figure 28(b).

An agent performing an evacuation behavior uses the safe planning and route selection techniques. Part of the evacuation behavior involves updating an agents' information about known dangerous areas. An agent updates it's own information when discovering new areas, either by observation or from communication with other agents. The lowest weight paths are selected. In this way, areas can be avoided by assigning higher edge weights to areas that have been marked as areas to avoid. This will prevent routes that pass through these areas from being selected. Groups of agents can also perform the same evacuation behavior. The agents within this group then only need to follow along the group evacuation route.

Algorithm C.1 Route Selection

```
Input: Agent $s_i$, known exits $E$, known safe areas $SA$
1:  for all $e \in E$ do
2:      if $s$.hasMarkedExitAsAvoid($e$) then
3:          continue
4:      end if
5:      $P1$ = findPath( $s_i$.getPos(), $e$.posInArea() )
6:      $nSA$ = nearestAvailableSafeArea( $e$, $s_i$ )
7:      $P2$ = findPath( $e$.posInArea(), $nSA$.posInArea() )
8:      $P3 = P1 + P2$
9:      $score$ = evaluateExitRoute( $P3$ )
10:     if $score$ of $P3$ is best then
11:         Save $P3, score$
12:     end if
13: end for
```

D. Behavior: Direction

The goal of our direction behaviors is to simulate a range of forms of direction that may exist in a real evacuation. These forms of direction can provide evacuating agents either local or global information, depending on the capability of the director and the kind of direction being modeled.

There are a number of ways that direction can be given to agents. Specifically, as agents are evacuating they may be interested in areas that may be considered dangerous, exits that should be avoided, and exits that should be preferred. In real evacuation situations this can be seen in the form of exits routes posted in buildings or lights representing the direction to evacuate. These could also be physical barriers preventing passage such as a moveable barrier or police tape. Another example of this could be cones or flares set up to direct or alert the agents. These forms of information are easy for humans to process but more difficult to simulate.

Here we describe two of the main mechanisms necessary to direct or steer the agents:

- **Local:** Barrier or Agent blocking an exit
- **Global:** Relaying global or other non-local information

Local direction can be either a barrier or other locally perceived information provided by an agent. There are two ways to achieve local direction. The simplest form of local direction is an obstruction in the environment which can be modeled with an obstacle or obstruction present (i.e., physically preventing passage). In the second form of local direction, a directing agent can represent a barrier to an exit by being placed nearby. It may also represent a sign to indicate an unsafe exit or area nearby. The evacuating agent can then no longer use this exit. An agent is only aware of a barrier when within range of the barrier.

Global direction can be information provided such as a goal location or route guidance beyond the local sensory range of the agent. We model global direction by allowing the directing agents to provide the evacuating agents with a subset of full information about specified exits, safe areas or known dangerous areas. The directing agents are equipped with information about the areas from which they are directing away. The directors should also be placed to be able to effectively disperse this direction information. This could be a human or robot directing the evacuating agents.

By being able to give the evacuating agents that encounter directors more global and complete information, better evacuation routes can be selected. We model giving complete information by having the directors make the evacuating agents aware of all the different areas in the environment of which the encountered director is aware. A director may be giving full information to the evacuating agents, however the director may not be aware of all information.

1. Cooperation Between Directors

We attempt to model a realistic form of cooperation by allowing a certain probability for direction to be accepted. This enables us to simulate agents that may not trust or accept certain forms of direction. This can include ignoring a barrier or sign that exists in the environment or even ignoring a police agent. To handle this situation, a directing agent may have potentially useful information for an evacuating agent, g_i, but a probability, P_d, is assigned which describes the chance that g_i will accept this information. This value, P_d is determined from the effectiveness of the directing agent and the cooperation value from g_i. Depending on the type of interaction that has taken place (accepting the information or not), g_i remembers the encountered director for a predefined number of time steps. The interaction is either restricted by the evacuating agent's sensory range or by the directional agent's alert distance.

We also model two forms of director communication. An *alert* involves providing information to the evacuating agent and allowing the agent to use the information as needed (update plan if a better one exists). *Assignment* allows for a director to guide the evacuating agent as desired by the director. This is a more active form of direction and allows for the directing agents to cooperate and coordinate in order to improve evacuation and better utilize environmental resources (i.e., exits). The interaction occurring between a director and an evacuating agent along with the cooperation probability is shown in Figure 29.

An example of cooperative weighted assignment of exits is shown in Algorithm D.1. In this algorithm, a director attempts to select a good exit for an evacuating agent. The director considers usage of each exit and computes paths to the exits that the director is biasing evacuation toward. A weight is then computed for each exit which considers both the usage of the exits and distance it would take to reach the exit (with the director's knowledge of the environment). The director then probabilistically selects the assigned exit based on the weights generated. Coordination between directors comes in the form of communicating exit information about the assigned exit usage. The coordination can also be predefined by the user of the evacuation planning system by specifying which set of exits each director is biasing evacuation toward.

Algorithm D.1 Cooperative Weighted Exit Assignment

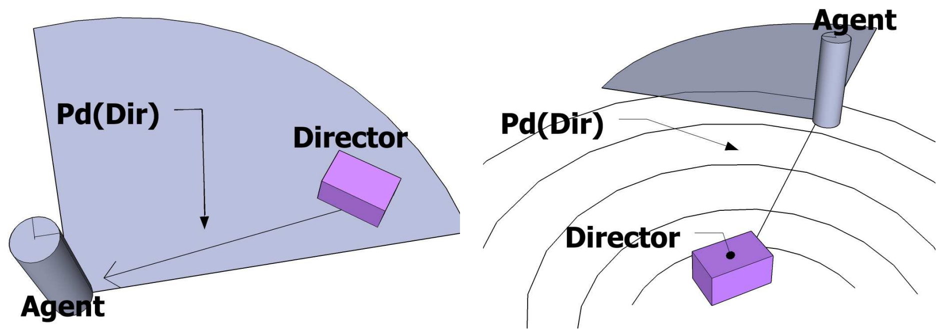
```
Input: $Director_j$, Agent $g_j$ to direct
 1: $U_{total} = \sum_{i \in Ex} usage(e_i)$
 2: for  $e_i \in Director_j \rightarrow getExits()$  do
 3:     $ue_i = \text{Usage}_{assigned}(e_i, U_{total})$
 4:     $Dist_i[e_i] = \text{restrictedPathToExit}(e_i,\ g_j)$
 5:     $Dist_{total} += Dist_i[e_i]$
 6: end for
 7: compute weights, $W_i$ for $i \in EX$, where
 8: $W_i = ue_i + \left(\frac{Dist_{total} - Dist_i[e_i]}{Dist_{total}}\right)$
 9:  select exit probabilistically based on weight
10: re-weight exit usage based on exit selection
```

E. Evacuation Behavior Extensions

1. Heterogeneous Populations

We attempt to model a range of agent characteristics, shown in Table VI, each affecting some aspect of the evacuation behaviors. While this is not a complete set of agent characteristics, it is a substantial enough set to give the resulting simulation very dynamic characteristics. Agent capabilities such as speed and sensory characteristics affect how an agent moves and what is sensed. The environmental knowledge of each agent will affect the evacuation routes selected. The areas known to an agent potentially alter the routes as will the roadmap used by the agent.

There are many behavioral variables that affect an agent. Some of these include



(a) Assign+Probabilistic (b) Alert+Probabilistic

Figure 29: Forms of direction modeled. (a) An assignment is made by the director. (b) The director alerts the agent to environmental information.

the amount of cooperation allowed and weighting of dangerous areas by agents.

A number of navigation abilities are given to the agents. The agents use the roadmap as a guide through the environment, however they do not need to follow the paths exactly. Reactive avoidance is employed as a low level force rule to avoid the nearest agent. This is similar to the basic flocking rule [29], considering only separation to the nearest agent. The level of path optimization done by each agent, reducing the potential length of the route, is a parameter we allow to be variable between agents. This also prevents agents that are following a similar path from appearing to follow the *same* path.

2. Roadmap Re-weighting

We utilize roadmap re-weighting in a number of ways. This can be used to bias agents away from certain areas or prevent passage through other areas. A general roadmap re-weighting algorithm can be seen in Algorithm E.1. In this example algorithm, the nodes in the roadmap within range of any of the areas in the area list will have the

Table VI: Heterogeneous agent properties considered in this work.

Agent capabilities:
- speed capabilities: velocity, acceleration - sensory capabilities: view radius, angle
Behavioral variables:
- cooperation – between/with directors - roadmap sharing - dangerous areas weighting (bravery parameter)
Environmental Knowledge:
- Known areas (exits, safe areas, dangerous areas) - Roadmap information: mapping, quality, etc
Navigation abilities:
- agent reactive avoidance - near agent avoid - path optimization – urgency

edge weight to all neighboring nodes modified by W. The modification to the roadmap edges can be by either a predefined value or some scaling factor of the edges previous weight. The agents use this re-weighting scheme to avoid nodes in the roadmap near dangerous areas or exits marked as to avoid that are known to the agent.

Algorithm E.1 Roadmap Re-weight

```
Input: Roadmap $Rdmp$, area list $A_L$, weight $W$
1: for all $r \in A_L$ do
2:     for all $node \in Rdmp \rightarrow getNodes()$ do
3:         if $r \rightarrow inRange(node)$ then
4:             $Rdmp \rightarrow modifyWeight(node, W)$
5:         end if
6:     end for
7: end for
```

3. Weighted Path Parameterization

Another re-weighting parameter we look at is the cost of passage. Whereas the normal weighting of dangerous areas is to make the weights in the roadmap so high that passage is infeasible, there are situations where passage through a dangerous area is needed. In this situation, portions of the roadmap that pass through the dangerous areas are scaled by a factor (unique to each agent). A low scaling factor will make passage through these areas slightly less desirable where as a high scaling factor makes passage unlikely.

4. Path Sharing

A shared roadmap between evacuating agents is beneficial to better utilize the environment. This can be used to help the agents make intelligent decisions that human agents may make with little effort. These can be about areas of the environment that are overused or potential exits that may lead to quick evacuation. In using roadmap re-weighting to select better paths, an agent that has selected an evacuation route will then scale that route in the roadmap by some pre-defined value. This will make the edges along that route seem somewhat more or less desirable (depending on the scaling factor) to the next agent selecting a route.

5. Path Optimization

After an agent extracts an evacuation route, the path can be smoothed to make it more natural and reduce the distance the agent will have to travel through the environment. We define the amount that the agents attempt to smooth the paths through agent parameters. In this way, agents can be given a high sense of urgency by heavily smoothing paths whereas agents with less urgency may do less path smoothing.

F. Experimental Results

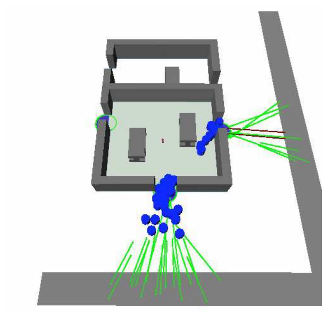
We have selected a number of examples to show the versatility of our evacuation planning techniques. The examples range from simple examples used to illustrate our planning potential to more complex and intricate planning scenarios.

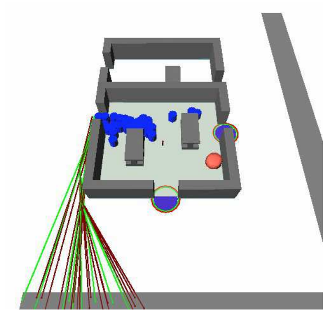
1. 2D Examples

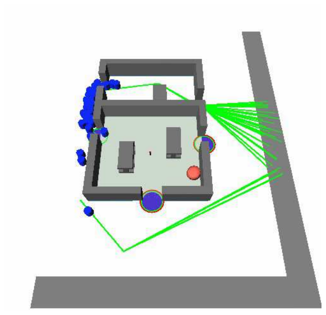
The time reported is the number of time steps required to have all agents either evacuate the area or reach a predefined safe location averaged over ten trial runs.

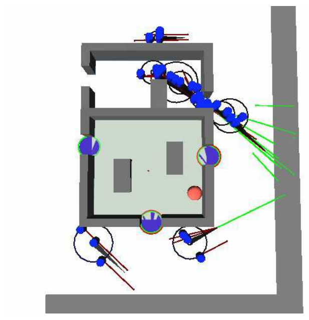
a. Rooms Environment

The examples, shown in Figure 30 with results in Table VII, illustrate many of the capabilities of our evacuation planning system, some unique to our approach. This environment consists of two rooms with a number of area types available, depending


(a) Full Evacuation


(b) With direction away from exit


(c) With direction away from exit and safe area


(d) Same as (c), with grouping

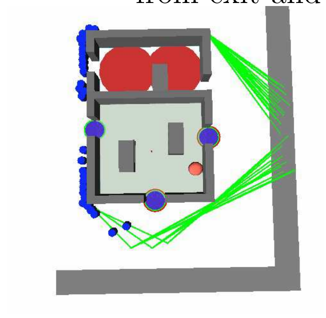

(e) With direction away from dangerous area

Figure 30: Different forms of evacuation problem

Table VII: Different forms of evacuation problem

Scenario	Evac. Time	Reach Time	$\frac{M}{S}$EX	$\frac{M}{S}$SA	$\frac{D}{S}$SA
(a)	112	164	1.0933	1.1457	1.0010
(b)	176	326	1.1165	1.1454	2.0849
(c)	176	497	1.1138	1.1588	3.1219
(d)	168	688	1.0652	1.1404	3.1955
(e)	176	510	1.0897	1.1499	3.3723

on the scenario. The area types include three exits in the main room, two safe areas (at the bottom and right of the environment), and a potential dangerous area in the second room. Thirty evacuating agents begin the simulation clustered at the center of the lower room. A full evacuation is shown in scenario (a) and in (b)-(e) a director is present to guide evacuation.

In scenario (b) the director prevents passage from two exits (on the lower and right walls of the room) resulting in the agents selecting the lower safe area. The director in (c) prevents passage from the same two exits but also relays global information of the lower safe area no longer being available. Scenario (d) is the same as (c) but with grouping of agents. The director in (e) is the same as in (c) and (d), but also relays global information about the dangerous area. While the number of time steps required to evacuate the initial room does not vary much, the amount of time needed to reach the safe area does change depending on the environmental parameters.

The fourth and fifth columns in Table VII compare the ratio of minimum distances to the nearest exit or safe area when finding a path using the agent roadmap

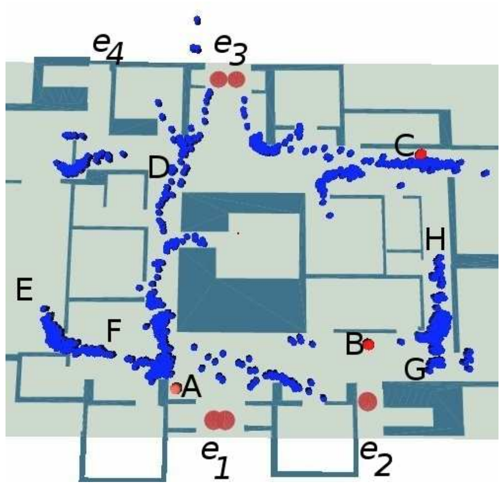


Figure 31: The first floor of a building to test variants of evacuation.

versus an approximation of the shortest distance. This approximation is computed from a roadmap whose nodes are densely sampled on a grid. The last column is a ratio of the actual distance traveled by the agents compared to the approximate shortest distance to any safe area. This shows that the agents plan in order to satisfy increasing constraints. These ratios also show that the roadmap is adequate in approximating potential paths through the environment.

b. Evacuation Variants

This experiment shows how evacuation can be effected by varying the parameters of the planning problem. We show evacuation results for agents evacuating a building under different conditions. The test environment is the first floor of a building at Texas A&M University, Figure 31. Evacuating agents are randomly placed in rooms throughout the building. The agents try to evacuate the building, utilizing the roadmap to find paths to safe areas. We are able to look at many different scenar-

Table VIII: Evacuation scenarios with varying forms of direction.

Scenario	GE	LE	Evac. Time
E4, D0			1568
E3, D0			1585
E3, D1, LNE1		A	2051
E3, D2, LNE1		A,B	4009
E3, D2, GNE2	A,B		3574
E3, D4, GNE2	A,B,C,D		2215
E3, D5, GNE2	A,B,C,D,E		2182
E3, D3, 1-GNE2, 2-LNE1	F	A,B	3074
E3, D5, 1-GNE2, 4-LNE1	F	A,B,C,H	2188

ios. We show experimental results with 500 agents evacuating the first floor, varying the number of available exits, the amount and type of direction given during the evacuation and which exits, if any, are restricted.

In Table VIII, evacuation under different conditions are shown. E# represents the number of exits available for the evacuating agents to select from. D# is the number of direction points present in the environment directing the agents away from exits (essentially making some exits unavailable). NE# is the number of exits that the direction points are steering evacuating agents away from with L/G used to denote whether the directing agents give local or global information. The GE and LE columns indicate the locations of the global and local directing agents, respectively.

Exit locations are shown, labeled e_{1-4}. All the scenarios use e_{1-3} and the first scenario also uses e_4.

Here we try to highlight some of the main results. In the second scenario (noted with LNE1), local direction is provided to evacuating agents in the form of barriers. The information is to avoid the nearest exit. In the case of one directing agent (D1), the barrier blocks one of the main exits and with two directing agents (D2), both lower exits are blocked which results in evacuation through the last available exit. By only giving local information, evacuating agents may end up selecting two bad exits before learning that the last exit is the only exit available. The effect on evacuation can be seen as the evacuation time increases greatly.

We also tested the effects of having directing agents provide global information to the evacuating agents (noted with GNE2). These directing agents inform the evacuating agents of the two exits to avoid. In this way, they are able to act as intelligent directing agents and direct the evacuation to the correct exit in the environment. This creates a better flow during evacuation. The benefits of increasing the number of directing agents can be seen as the evacuation time decreases.

We also tested the evacuation with a mixture of directing agents that provide global and local information. It is interesting to note that by placing local direction at certain locations in the environment and placing the global directing agent in a high traffic area we are able to come close to an evacuation time where five global directors are present.

2. Regrouping to Safe Areas

In this scenario, agents are dispersed around an environment (shown in Figure 32) and regroup to safe locations defined in a building. Evacuation results are shown in Table IX for four different scenarios. In the first scenario, the agents regroup to the

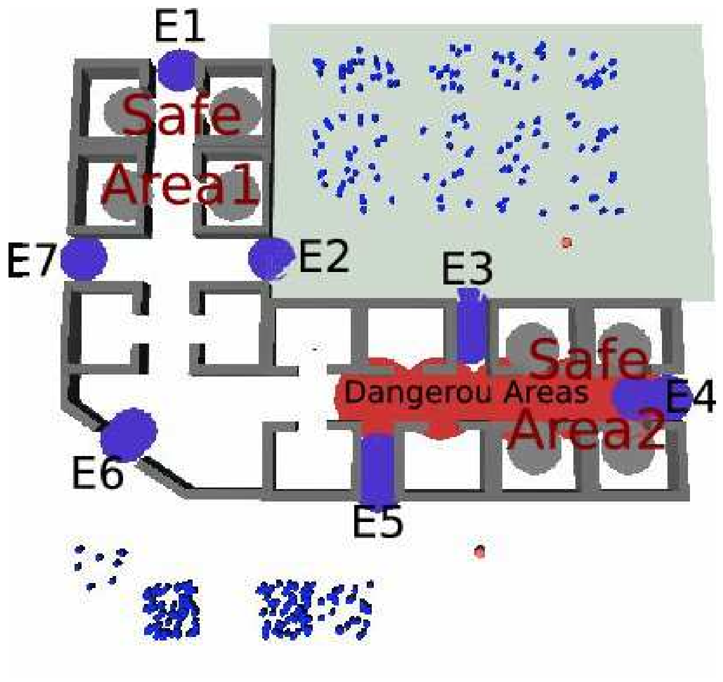


Figure 32: Test environments used for regrouping example.

Table IX: Regrouping to predefined locations

Scenario	Evac. Time	Reach Time
Basic Regroup	845	1475
Partial blockage of Lower Corridor	1425	2306
Full Block of Lower Corridor	1383	2490
Full Block of Lower Corridor (Grouping)	1448	2747

nearest safe area. In the second example, directing agents are placed at exits $E3$ and $E5$ to simulate a partial blockage of the corridor. The result is that the agents still use both safe areas while not passing through the blocked corridor. In the third and fourth scenario, two advanced directing agents are placed at the locations shown in Figure 32 near $E3$ and $E5$. These agents, with a larger alert radius inform evacuating agents of the dangerous areas present in the corridor. The result is that the agents regroup at the safe areas near the upper corridor in the environment.

3. Complex 3D Examples

We present results in two example environments under a number of simulation conditions. The first environment is a symmetrical three level building and the second environment also consists of three levels but is much more complex. All examples show 500 agents evacuating and considering environmental and interaction conditions dictated by the scenario. Evacuation times reported are in number of simulated time steps averaged over ten trial runs. Agents equipped with heterogeneous parameters have values centered around specifications of the homogeneous agent parameters.

4. Basic 3 Level Building

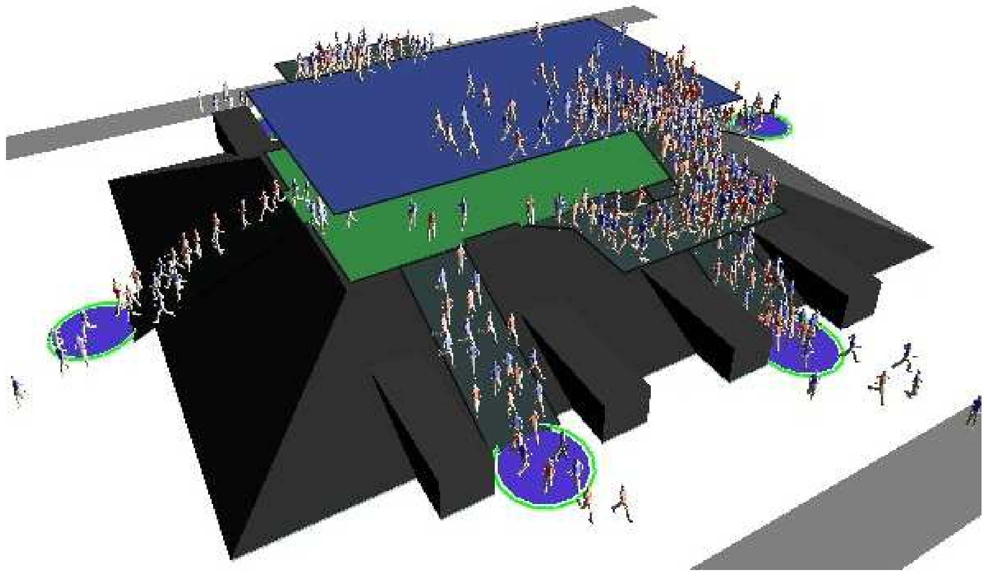


Figure 33: A basic 3-level building example.

The first environment tested is shown in Figure 33. The third level of this environment has two stairways to the second level which has six available exits and three safe areas. Due to the nature of the environment, agents are biased toward exits 1–4 because safe areas are nearby and the stair wells lead to areas near the exit. A safe area is also located near exit 5. Exit 6 is somewhat out of the way with no safe area located near by.

The different aspects considered in this example are:

(A) homogeneous agents,

(B) heterogeneous agents,

(C) shared roadmap between evacuating agents,

(D) heterogeneous agents with differing knowledge,

(E) 2 blocking agents,

(F) 2 cooperating directors, and

(G) 2 cooperating directors with cooperation parameter of 0.95.

In all the scenarios considered, each agent is equipped with full environmental knowledge except in scenario (D). The scenarios with directors are blocking exits 4 and 5. Resulting evacuation time and exit usage for each scenario is shown in Table X.

It is interesting to note that in scenario (A), with homogeneous agents selecting the shortest possible route, the evacuation time is minimized. This could be considered ideal conditions, however the agents almost completely ignore an available exit. Just equipping the agents with heterogeneous parameters, (B), causes the evacuation time to increase greatly. While agents sharing a roadmap, (C), causes the evacuation time to increase, it also results in a more even usage of the environment, shown by the exit usage. Similar usage occurs when the agents have differing knowledge, representing conditions where agents in an environment may not always know a building's floor plan but rather have knowledge of areas they passed through while in the building. In (E), agents that encounter the blockade then proceed to their nearest available exit. Another interesting aspect is that when cooperating directing agents are included, a more even usage of the environment is seen. This can be seen in the exit usage results shown in Table X and Figure 34 .

Table X: Results showing evacuation under different scenarios in basic 3-level building shown in Figure 33.

Scenario	Evac. Time	Exit Usage					
		1	2	3	4	5	6
(A)	2040	115	70	52	152	112	0.1
(B)	2814	107	71	54	148	112	0.1
(C)	3605	97	96	76	122	72	37
(D)	3260	91	84	72	113	102	39
(E)	3382	163	77	235	0	0	26
(F)	3961	175	143	118	0	0	64
(G)	3943	171	140	114	8	6	61

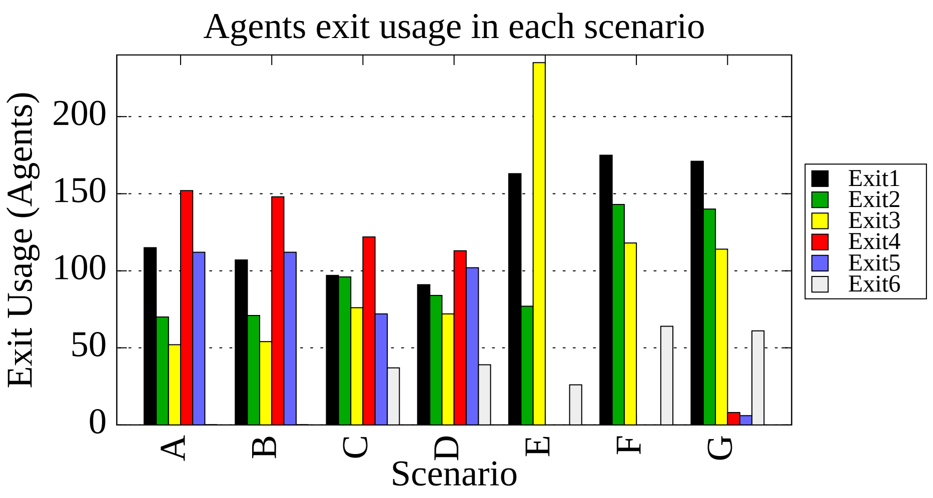


Figure 34: Exit usage in basic 3-level building example.

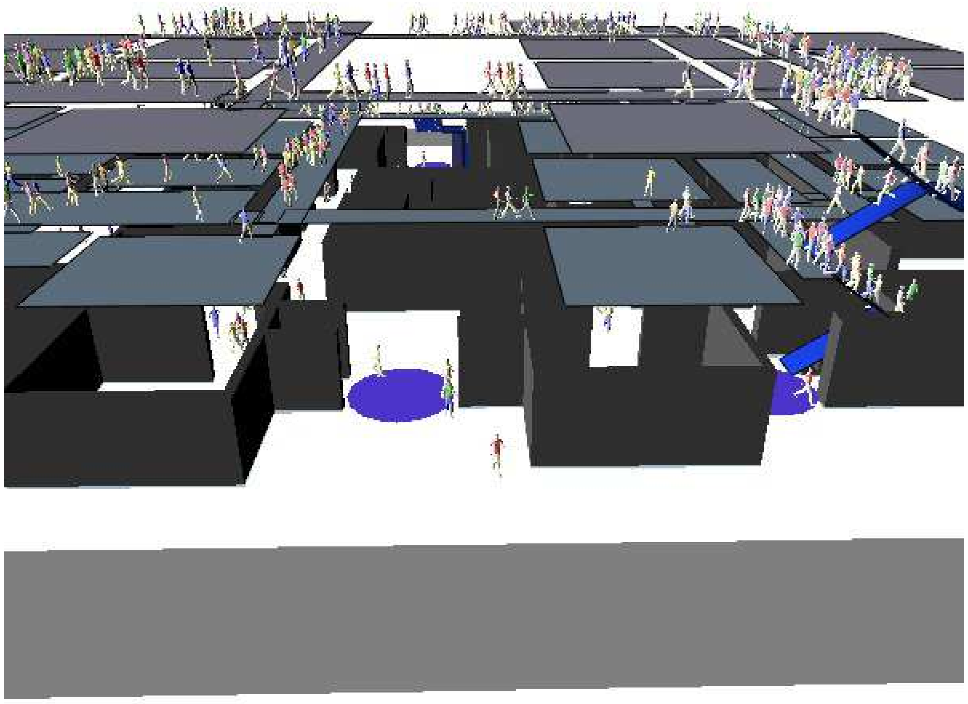


Figure 35: A complex 3-level building example.

5. 3 Level Building

This environment, shown in Figure 35 without walls, consists of many rooms, passages and stairwells between levels. There are also five exits to two available safe areas. Exit 2 and 4 are the most heavily used as these are the main passages to an exit for agents in the upper levels in the environment. Resulting evacuation time and exit usage is shown in Table XI and Figure 36. There is a stairwell near Exit 1 between the first and second floor.

The scenarios considered here are:

(A) homogeneous agents,

(B) heterogeneous agents,

(C) shared roadmap, heterogeneous agents, and

(D) 3 cooperating directing agents blocking off one exit, heterogeneous agents.

All agents are equipped with full environmental knowledge. In (D), the directing agents (one per floor) are all guiding agents away from an exit near a corner stair well. Again, we can show that a heterogeneous set of agents evacuates slower than a group of agents with the same capabilities. This represents scenarios where some agents are in a hurry and some unaffected by the situation. Agents sharing information (C) are able to make better usage of the environment, also a common situation seen in the real world where some people are willing to take a longer route in order to avoid congestion as shown in Figure 36. In fact, many agents select routes that avoid the more heavily used exits (2 and 4). Including directors (D), allows the agents to avoid the exit in question (Exit 4) however some agents do get near enough that it is considered reached in our usage metric.

Table XI: Results showing evacuation under different scenarios in complex 3-level building example shown in Figure 35.

Scenario	Evac. Time	Exit Usage				
		1	2	3	4	5
(A)	3577	25	176	92	205	0
(B)	4375	22	209	77	185	6
(C)	6653	138	158	99	68	35
(D)	9850	107	297	86	1.4	6

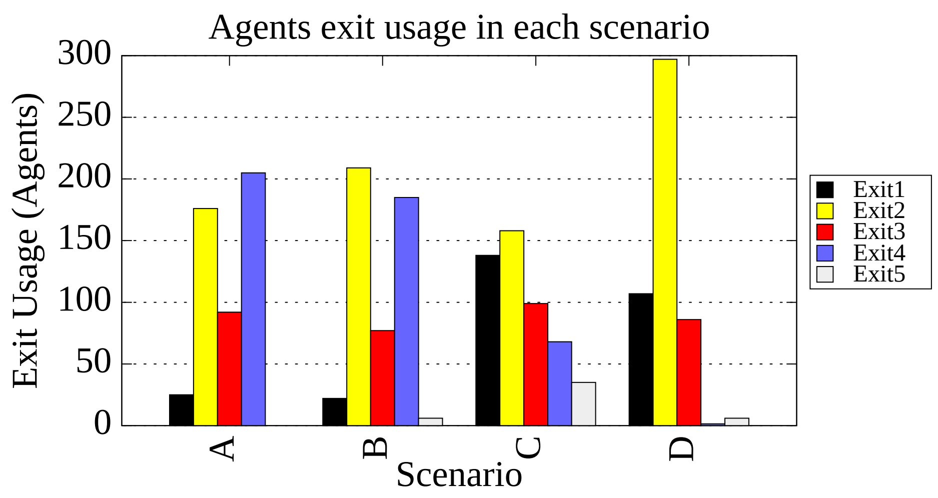


Figure 36: Exit usage in complex 3-level building example.

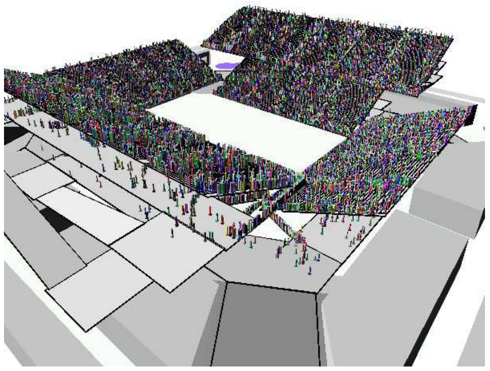
6. Stadium Example

To show how our system can extend to much more complex scenarios, an example evacuation of a stadium is shown in Figure 37. This scenario consists of 20,000 agents evacuating from a stadium using our roadmap-based approach. The agents initially start on the bleachers and plan paths to areas outside of the stadium, on the boundary of the environment. Screen shots as the evacuation progresses is shown in Figure 37(a)-(d). This example is interesting because it is of a much larger scale than many of our other examples and shows an evacuation of a very complex structure. In fact, some approaches focus solely on evacuation of a stadium but simulate this on a very simplified network graph structure with just over 200 links representing the entire environment [20]. An important aspect is that our shared path re-weighting scheme extends to crowds of this size. Our roadmap-based approach is also flexible enough to adequately map an environment of this complexity.

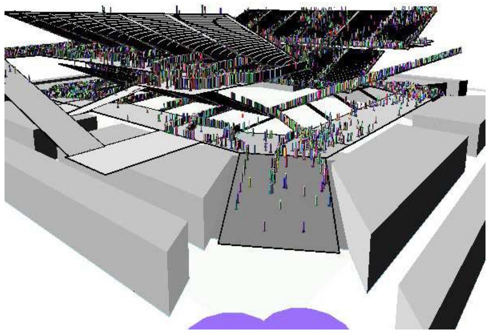
7. Examples of Impact of Environment on Pedestrian Flow

In this section we show simulation results when varying portions of the environment. Our system can be used to evaluate the result of environmental changes on the evacuation process. This capability would allow for our system to be used as a validation tool in the design process of buildings and other structures. It could also be extended to show how an environment is used when agents are performing normal everyday behaviors and give insight as to how the environment can be improved for better utilization. This is an important application which further expands the usefulness of our system as a design tool. We present results for two simulated environments, a stadium and office building.

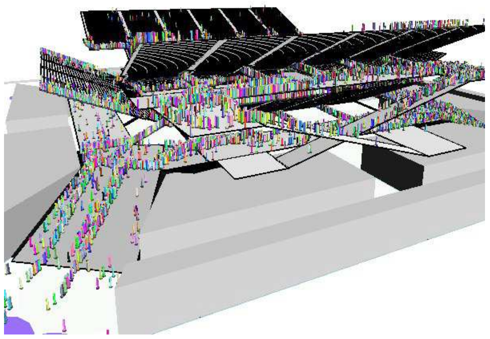
In the first enviornment, the stadium example, we tested varying the environment



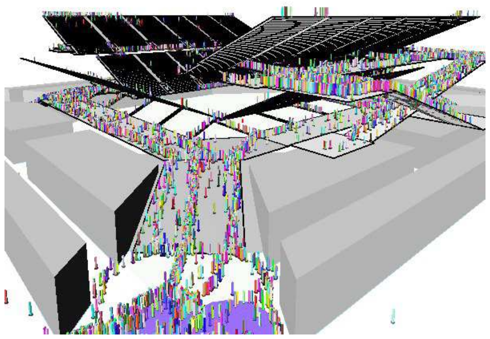
(a) Initial Setup



(b) time step: 432

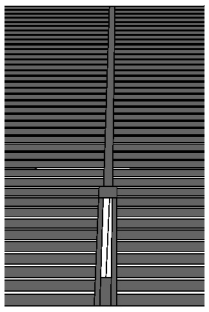


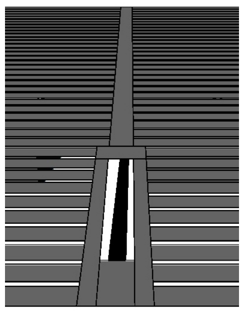
(c) time step: 854

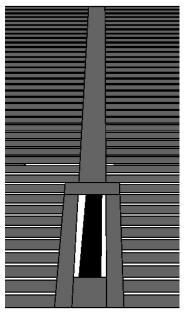


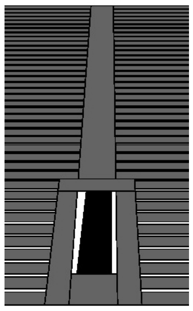
(d) time step: 1506

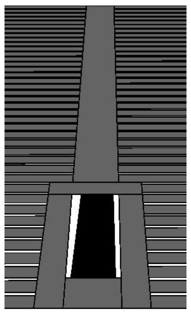
Figure 37: Stadium evacuation of 20,000 agents with heterogeneous parameters.

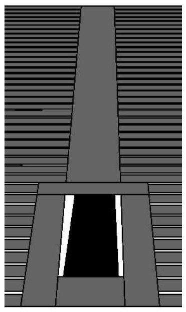

(a) width 1


(b) width 2


(c) width 3


(d) width 4


(e) width 5


(f) width 6

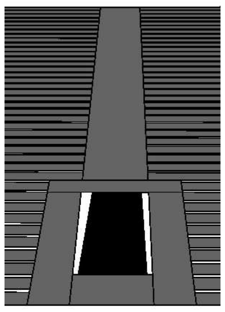

(g) width 7

Figure 38: Different versions of the stadium with varying access widths (between 1 and 7 units). This is an example where the design of the environment impacts the speed at which the agents can travel.

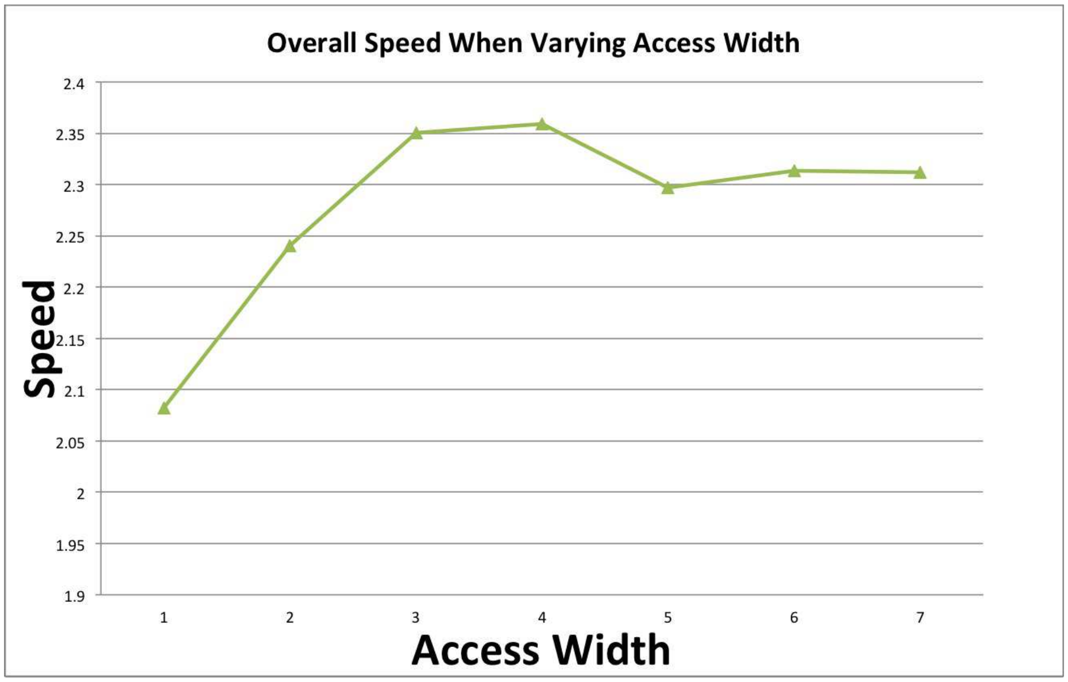
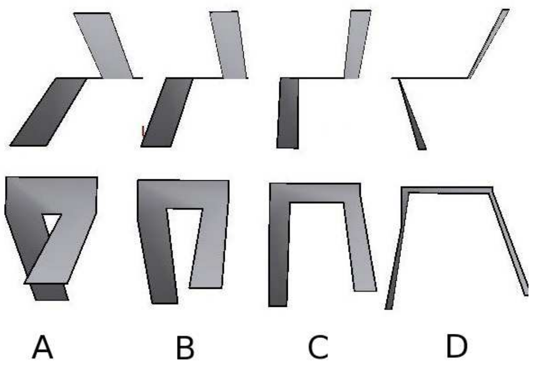


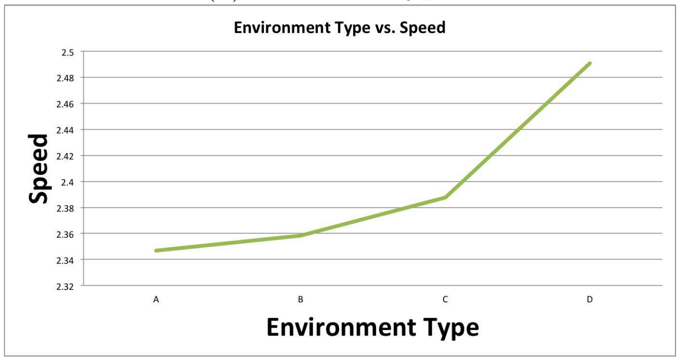
Figure 39: Stadium evacuation of one side of the stadium varying the access width from 1–7 units.

where 4000 agents would attempt to evacuate one side of a stadium under different conditions. The agents have maximum velocities between 2 and 4 units per time step. In Figure 39, the speed of agents evacuating one side of a stadium (for a single access) shown under varying environmental conditions. An example of the different access sizes is shown in Figure 38(a)–(g) with the smallest access size having a width of 1 unit and the largest access size having a width of 7 units. In Figure 39, it can be seen that as the size of the access width increases, the speed that agents can navigate through the access increases until a width of 4 units, at which point the average speed decreases and then levels off. In this environment, this shows that at a certain point, increasing navigable space does not necessarily increase the speed at which agents can move over the surface.

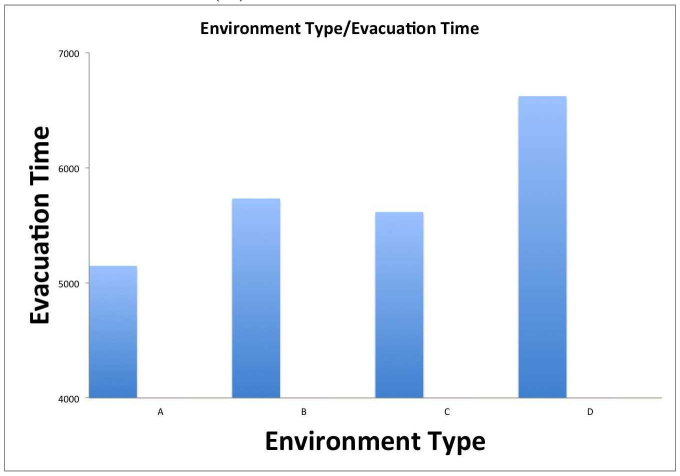
Varying the environment was also tested in an office building example where a single stairwell is used for evacuation and the size of the stairwell is progressively decreased. In this example 200 agents were placed on the second level of the office building and would evacuate to a safe area on the boundary of the environment, requiring them to travel through the only available stairwell. The different stairwell sizes are shown in Figure 40(a). The speed at which the agents can navigate over the stairwell surfaces is shown in Figure 40(b) with Stairwell A being the widest and Stairwell D being the most narrow. It is interesting to see that the speed that agents navigate over the surface actually increases as the width decreases. This is due to the fact that the agents must organize themselves before they use the stairwell as shown in Figure 41. When the stairwell is wide enough, as in Stairwell A, more collision is occurring between agents trying to move at faster speeds causing the overall speed of agents to be reduced over the surface being measured. This is also supported in Figure 40(c) in that the overall evacuation time shows an increasing trend when the simulation is run varying between Stairwell A to D.



(a) Stairwell Types



(b) Type vs Speed



(c) Type vs Evacuation Time

Figure 40: Office building example with varying types of stairwells (a) (varying the width of a single stairwell) to see effect on (b) overall average speed of the agents and (c) evacuation time.

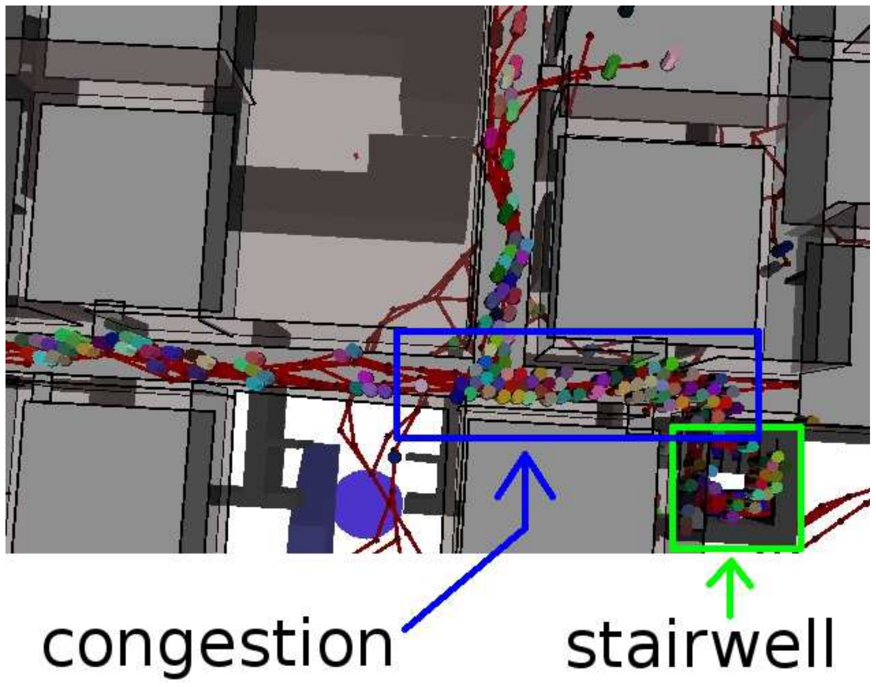


Figure 41: Congestion occurring before agents enter stairwell. This congestion is in part due to agents having to organize themselves before traveling through the available stairwell.

CHAPTER VII

CONCLUSION AND FUTURE WORK

In this work, a roadmap-based approach to simulating multi-agent group behavior is presented. The roadmap is a flexible, abstract representation and allows for easily handling complex environments. Our system includes aspects of multi-agent behavior required for the applications we focus on. This includes dynamic behaviors, visibility restricted by the environment, heterogeneous parameters for agents (for varying agent populations) and utilization of the roadmap beyond simple navigation. Our application focus is on pursuit-evasion and evacuation planning. These scenarios allow us to investigate different aspects of multi-agent group behavior which require cooperation and coordination between agents.

In Chapter III, we reviewed aspects of our multi-agent system. This included parameters that define the system including agents, behaviors, forces on agents, the entire simulation cycle and our visibility model. Each of these components affects the system and is highly tunable to allow for each aspect to be studied and varied as needed. The roadmap in our system allows us to handle complex environments by encoding the problem in a general way so that the agents and behaviors operate in the same way regardless of the complexity of the environment. The way paths are extracted and used affects the movement of the agents but can be optimized as needed. Metrics were also described for validating our agent motion model.

Search, clearing and tracking strategies are described in Chapter IV. The search and clearing strategies allow us to test how well an environment can be searched by a group of agents and the coordination required to do this. The level search strategies are dependent on the quality of the roadmap which will dictate where and how well each level is covered and the paths that can be taken through the roadmap. Our tracking model is also dependent on the mapping of the environment but allows the agents to make intelligent decisions about where to search for agents that have left their field-of-view.

The pursuit-evasion application, covered in Chapter V, includes search, pursuit and evasion behaviors. Our pursuit and evasion behaviors can be used in complex 3D environments as well as simple 2D environments. The extended scenarios we have been able to study include buildings, terrains, and crowds. We include an evasion behavior description which is important because many approaches assume only a clearing problem where the evader can move arbitrarily fast and are only interested in generating pursuit strategies which guarantee to capture the evader if one exists. In our framework, we can vary the capabilities of both the pursuing and evading agents to compare types of agents and add another level or realism. The level of capabilities can also be varied by the roadmap, for example in the level search strategies to clear buildings.

Our evacuation framework was described in Chapter VI. This included evacuation and controlling agent behaviors. The way we model cooperation in our framework is an important aspect. Between evacuating agents, the roadmap can be used to cooperate by sharing and re-weighting paths that an agent is planning to use. The directing agents are able to cooperate both within the directing group and with the evacuating agents. The directors attempt to guide agents to use the environment evenly. A cooperation parameter allows agents to decide whether or not they will accept direction information. Overall, we have been able to study complex evacuation scenarios in large scale environments where many approaches are restricted to 2D scenarios.

In the future, our system would ideally be used as a training system to develop strategies (search, pursuit, and evacuation strategies) and for building validation to ensure proper design (evacuation planning). Studying the pursuit-evasion application would allow search strategies to be developed for search and rescue where the environment needs to be cleared or an unknown number of targets need to be found. It can also be used to develop strategies for guarding an area, for example a building or museum. Pursuit-evasion has natural extensions to games, graphics, virtual environments and studying natural hunting scenarios. The evacuation planning scenarios can be used for a number of strategies as well. It can be used in evacuation or emergency training in order to train for the best plan of action given the personnel available. It can also be used in planning or design of buildings to improve and validate design. Our system is flexible enough to handle these scenarios. Given our pluggable behavioral framework, our system can be used anytime a simulation is needed that requires agents in a realistic environment, cooperating and performing complex tasks. In the future, we would like to extend our system to many other virtual settings.

REFERENCES

[1] V. Isler, D. Sun, and S. Sastry, “Roadmap based pursuit-evasion and collision avoidance,” in *Proc. Robotics: Sci. Sys. (RSS)*, 2005.

[2] K. E. Bekris, K. Tsianos, and L. E. Kavraki, “A decentralized planner that guarantees the safety of communicating vehicles with complex dynamics that replan online,” in *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, 2007.

[3] L. Heigeas, A. Luciani, J. Thollot, and N. Castagné, “A physically-based particle model of emergent crowd behaviors,” *The Computing Research Repository (CoRR)*, vol. abs/1005.4405, 2010.

[4] A. Treuille, S. Cooper, and Z. Popovi, “Continuum crowds,” *ACM Trans. Graph.*, vol. 25, no. 3, pp. 1160–1168, 2006.

[5] J. van den Berg, S. Patil, J. Sewall, D. Manocha, and M. Lin, “Interactive navigation of multiple agents in crowded environments,” in *Proc. Symposium on Interactive 3D Graphics and Games - I3D’08*, 2008.

[6] H. Kitano, S. Tadokoro, I. Noda, H. Matsubara, T. Takahashi, A. Shinjou, and S. Shimada, “Robocup rescue: Search and rescue in large-scale disasters as a domain for autonomous agents research,” in *Proc. IEEE Trans. Sys., Man, Cybern.*, 1999.

[7] Q. Yu and D. Terzopoulos, “A decision network framework for the behavioral animation of virtual humans,” in *SCA ’07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, New York, NY, USA, 2007, pp. 119–128, ACM Press.

[8] R. Loftin, M. Scerbo, F. McKenzie, and J. Catanzaro, "Training in peacekeeping operations using virtual environments," *Projects in VR*, pp. 2–5, 2004.

[9] M. Brenner, N. Wijermans, T. Nussle, and B. de Boer, "Simulating and controlling civilian crowds in robocup rescue," in *Proceedings of RoboCup 2005: Robot Soccer World Cup IX*, 2005.

[10] N. Pelechano and N. Badler, "Modeling crowd and trained leader behavior during building evacuation," *IEEE Computer Graphics and Applications*, vol. 26, no. 6, pp. 80–86, November 2006.

[11] N. Pelechano, J. Allbeck, and N. Badler, "Controlling individual agents in high-density crowd simulation," in *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2007.

[12] G. Lammel, M. Rieser, and K. Nagel, "Bottlenecks and congestion in evacuation scenarios: A microscopic evacuation simulation for large-scale disasters," in *5th Workshop on Agents in Traffic and Transportation*, 2008.

[13] N. P. Waterson and E. Pellissier, "The steps pedestrian microsimulation tool a technical summary," Steps software - technical summary, Mott MacDonald Limited, Croydon, UK, July 2010.

[14] C. Neumann and R. Neunteufel, "Evacuation simulation at Linz Central Station usefulness during design, approval and start-up," in *Pedestrian and Evacuation Dynamics 2005*, N. Waldau, P. Gattermann, H. Knoflacher, and M. Schreckenberg, Eds., pp. 333–339. Springer, Berlin, 2007.

[15] D. Helbing, I. Farkas, and T. Vicsek, "Simulating dynamical features of escape panic," *Nature*, vol. 407, pp. 487–290, 2000.

[16] D. Helbing, L. Buzna, A. Johansson, and T. Werner, "Self-organized pedestrian crowd dynamics: Experiments, simulations, and design solutions," *Transportation Science*, vol. 39, pp. 1–24, February 2005.

[17] S. J. Guy, J. Chhugani, S. Curtis, P. Dubey, M. Lin, and D. Manocha, "Pledestrians: A least-effort approach to crowd simulation," in *SCA '10: Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Madrid, Spain, 2010, Eurographics Association.

[18] L. J. Guibas, J. C. Latombe, S. M. Lavalle, D. Lin, and R. Motwani, "Visibility-based pursuit-evasion in a polygonal environment," in *International Journal of Computational Geometry and Applications*. 1997, pp. 17–30, Springer-Verlag.

[19] J. W. Durham, A. Franchi, and F. Bullo, "Distributed pursuit-evasion with limited-visibility sensors via frontier-based exploration," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2010, pp. 3562–3568.

[20] Z. Fang, Q. Li, Q. Li, L. D. Han, and D. Wang, "A proposed pedestrian waiting-time model for improving space-time use efficiency in stadium evacuation scenarios," *Building and Environment*, vol. 46, pp. 1774–1784, September 2011.

[21] A. Varas, M.D. Cornejo, D. Mainemer, B. Toledo, J. Rogan, V. Munoz, and J.A. Valdivia, "Cellular automaton model for evacuation process with obstacles," *Physica A*, vol. 382, pp. 631–642, 2007.

[22] S. Rodriguez, J. Denny, T. Zourntos, and N. M. Amato, “Toward simulating realistic pursuit-evasion using a roadmap-based approach,” in *Proc. of the 3rd Intern. Conf. on Motion in Games, 2010, Lecture Notes in Computer Science (LNCS)*, November 2010, pp. 82–93.

[23] S. Rodriguez, J. Denny, J. Burgos, A. Mahadevan, K. Manavi, L. Murray, A. Kodochygov, T. Zourntos, and N. M. Amato, “Toward realistic pursuit-evasion using a roadmap-based approach,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2011, pp. 1738–1745.

[24] S. Rodriguez, J. Denny, A. Mahadevan, J. Vu, J. Burgos, T. Zourntos, and N. M. Amato, “Roadmap-based pursuit-evasion in 3d structures,” *Transactions on Edutainment (to appear)*, 2011.

[25] S. Rodriguez and N. M. Amato, “Roadmap-based level clearing of buildings,” in *Proc. of the 4th Intern. Conf. on Motion in Games, 2011, Lecture Notes in Computer Science (LNCS)*, November 2011, pp. 340–352.

[26] S. Rodriguez and N. M. Amato, “Behavior-based evacuation planning,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2010, pp. 350–355.

[27] S. Rodriguez and N. M. Amato, “Utilizing roadmaps in evacuation planning,” in *Proc. of the 24th Intern. Conf. on Computer Animation and Social Agents (CASA), 2011, in Intern. J. of Virtual Reality (IJVR)*, 2011, pp. 67–73.

[28] C. W. Reynolds, “Steering behaviors for autonomous characters,” in *Proceedings of Game Developers Conference*, 1999, pp. 763–782.

[29] C. W. Reynolds, “Flocks, herds and schools: A distributed behavioral model,” in *Proceedings of the 14th Annual Conference on Computer Graphics and*

Interactive Techniques, New York, NY, USA, 1987, SIGGRAPH '87, pp. 25–34, ACM.

[30] D.C. Brogan and J.K. Hodgins, "Group behaviors for systems with significant dynamics," *Intelligent Robots and Systems, IEEE/RSJ International Conference on*, vol. 3, pp. 3528, 1995.

[31] T. Y. Li, Y. J. Jeng, and S. I. Chang, "Simulating virtual human crowds with a leader-follower model," in *Proceedings of 2001 Computer Animation Conference*, 2001, pp. 93–102.

[32] R. A. Metoyer and J. K. Hodgins, "Reactive pedestrian path following from examples," in *CASA '03: Proceedings of the 16th International Conference on Computer Animation and Social Agents (CASA 2003)*, 2003, pp. 149–156.

[33] J. Funge, X. Tu, and D. Terzopoulos, "Cognitive modeling: Knowledge, reasoning and planning for interlligent characters," in *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, 1999, SIGGRAPH '99, pp. 29–38.

[34] W. Shao and D. Terzopoulos, "Autonomous pedestrians," in *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, New York, NY, USA, 2005, pp. 19–28, ACM Press.

[35] M. Petty, F. D. McKenzie, R. C. Gaskins, and E. W. Weisel, "Developing a crowd federate for military simulation," in *Proceedings of the Spring 2004 Simulation Interoperability Workshop*, 2004, pp. 483–493.

[36] F. D. McKenzie, M. D. Petty, P. A. Kruszewski, R. C. Gaskins, Q. H. Nguyen, J. Seevinck, and E. W. Weise, "Integrating crowd-behavior modeling into

military simulation using game technology," *Simulation and Gaming*, vol. 39, pp. 10–37, 2008.

[37] F. McKenzie, M. Petty, and J. Catanzaro, "An experimental application of a trait-based personality model to the simulation of military decision-making," *Information and Security: An International Journal: Special Issue in Modeling and Simulation*, vol. 12, pp. 75–92, 2003.

[38] L. J. Moya, F. D. McKenzie, and Q. H. Nguyen, "Visualization and rule validation in human-behavior representation," *Simulation and Gaming*, vol. 39, pp. 101–117, 2008.

[39] P. Torrens, "Cellular automata and multi-agent systems as planning support tools," in *Planning Support Systems in Practice*, pp. 205–222. Springer-Verlag, London, UK, 2002.

[40] W.A. Griffin, S. Schmidt, A. Nara, P. Torrens, J. Fewell, and C. Sechler, "Integrating ABM and GIS to model typologies of playgroup dynamics in preschool children," in *Proceedings of the Agent 2007: Complex Interaction and Social Emergence*, 2007, pp. 17–24.

[41] R. Narain, A. Golas, S. Curtis, and M. Lin, "Aggregate dynamics for dense crowd simulation," in *Proceedings of SIGGRAPH Asia, in ACM Transactions on Graphics*, 2009, pp. 1–8.

[42] S. Curtis, S. J. Guy, B. Zafar, and D. Manocha, "Virtual tawaf: A case study in simulating the behavior of dense, heterogeneous crowds," in *Proc. of 1st IEEE Workshop on Modeling, Simulation and Visual Analysis of Large Crowds*, 2011.

[43] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Proceedings of Robotics Research: The 14th International Symposium, ISRR*, Madrid, Spain, 2011, pp. 1–16.

[44] S. J. Guy, J. Chhugani, C. Kim, N. Satish, M. Lin, D. Manocha, and P. Dubey, "Clearpath: Highly parallel collision avoidance for multi-agent simulation," in *SCA '09: Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 2009, pp. 177–187, Eurographics Association.

[45] S. J. Guy, M. Lin, and D. Manocha, "Modeling collision avoidance behavior for virtual humans," in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 2*, Richland, SC, 2010, AAMAS '10, pp. 575–582.

[46] M. Sung, M. Gleicher, and S. Chenney, "Scalable behaviors for crowd simulation,," *Computer Graphics Forum*, vol. 23, no. 3, 2004.

[47] J. M. Lien, O. B. Bayazit, R. T. Sowell, S. Rodriguez, and N. M. Amato, "Shepherding behaviors," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, April 2004, pp. 4159–4164.

[48] J. M. Lien, S. Rodriguez, J. P. Malric, and N. M. Amato, "Shepherding behaviors with multiple shepherds," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, April 2005, pp. 3413–3418.

[49] C. Vo, J. F. Harrison, and J. M. Lien, "Behavior-Based Motion Planning for Group Control," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2009)*, 2009, pp. 3768–3773.

[50] J. F. Harrison, C. Vo, and J. M. Lien, "Scalable and Robust Shepherding via Deformable Shapes," in *Proceedings of the Third International Conference on Motion in Games*, R. Boulic and Y. Chrysanthou and T. Komura, Ed. 2010, pp. 218–229, Springer.

[51] O. B. Bayazit, J. M. Lien, and N. M. Amato, "Roadmap-based flocking for complex environments," in *Proc. Pacific Graphics*, Oct 2002, pp. 104–113.

[52] O. B. Bayazit, J. M. Lien, and N. M. Amato, "Better group behaviors in complex environments using global roadmaps," in *Artif. Life*, Dec 2002, pp. 362–370.

[53] O. B. Bayazit, J. M. Lien, and N. M. Amato, "Better flocking behaviors using rule-based roadmaps," in *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, Dec 2002, pp. 95–111.

[54] L. E. Kavraki, P. Švestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Automat.*, vol. 12, no. 4, pp. 566–580, August 1996.

[55] N. M. Amato, O. B. Bayazit, L. K. Dale, C. V. Jones, and D. Vallejo, "OBPRM: An obstacle-based PRM for 3D workspaces," in *Robotics: The Algorithmic Perspective*, Natick, MA, 1998, pp. 155–168, A.K. Peters, Proc. Third Workshop on Algorithmic Foundations of Robotics (WAFR), Houston, TX, 1998.

[56] S. A. Wilmarth, N. M. Amato, and P. F. Stiller, "MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 1999, vol. 2, pp. 1024–1031.

[57] R. Geraerts and E. Schager, "Stealth-based path planning using corridor maps," in *Proc. of Computer Animation and Social Agents (CASA'10)*, 2010.

[58] R. Geraerts, "Planning short paths with clearance using explicit corridors," in *Proc. of IEEE International Conference on Robotics and Automation (ICRA'10)*, 2010, pp. 1997–2004.

[59] J. J. Kuffner and S. M. LaValle, "RRT-Connect: An Efficient Approach to Single-Query Path Planning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2000, pp. 995–1001.

[60] S. M. LaValle and J. J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," in *New Directions in Algorithmic and Computational Robotics*, pp. 293–308. A. K. Peters, 2001, book contains the proceedings of the International Workshop on the Algorithmic Foundations of Robotics (WAFR), Hanover, NH, 2000.

[61] D. Hsu, J-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 1997, pp. 2719–2726.

[62] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 1999, pp. 473–479.

[63] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, May 2001.

[64] J. Bruce and M. Veloso, "Real-time randomized path planning for robot navigation," in *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, Switzerland, 2002, pp. 2383–2388.

[65] J.M. Phillips, N. Bedrosian, and L. Kavraki, "Guided expansive spaces trees: A search strategy for motion- and cost-constrained state spaces," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2004, pp. 3968–3973.

[66] R. Kindel, D. Hsu, J. Latombe, and S. Rock, "Kinodynamic motion planning amidst moving obstacles," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2000, pp. 537–543.

[67] L. Jaillet and T. Simeon, "A PRM-based motion planner for dynamically changing environments," in *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, 2004, pp. 1606–1611.

[68] J. van den Berg and M. H. Overmars, "Roadmap-based motion planning in dynamic environments," *IEEE Trans. Robot. Automat.*, vol. 21, pp. 885–897, 2005.

[69] J. van den Berg and M. Overmars, "Planning the shortest safe path amidst unpredictably moving obstacles," in *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, 2006, pp. 885–897.

[70] T. Fraichard and H. Asama, "Inevitable collision states - a step towards safer robots?," in *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, 2004, pp. 288–393.

[71] S. Petti and T. Fraichard, "Partial motion planning framework for reactive planning within dynamic environments," in *Proc. of the IFAC/AAAI Int. Conf. on Informatics in Control, Automation and Robotics*, September 2005.

[72] K. E. Bekris and L. E. Kavraki, "Greedy but safe replanning under kinodynamic constraints," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*,

2007, pp. 704–710.

[73] A. Sud, R. Gayle, S. Guy, E. Andersen, M. C. Lin, and D. Manocha, "Real-time navigation of independent agents using adaptive roadmaps," in *Proc. ACM Symp. on Virtual Reality Software and Technology*, 2007, pp. 99–106.

[74] R. Gayle, A. Sud, M. C. Lin, and D. Manocha, "Reactive deformation roadmaps: Motion planning of multiple robots in dynamic environments," in *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, 2007, pp. 3777–3783.

[75] W. G. van Toll, A. F. Cook IV, and R. Geraerts, "Navigation meshes for realistic multi-layered environments," in *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'11)*, 2011, pp. 3526–3532.

[76] M. van den Akker, R. Geraerts, H. Hoogeveen, and C. Prins, "Path planning for groups using column generation," in *Proc. for Motion in Games (MIG'10), Springer Lecture Notes in Computer Science (LNCS)*, 2010, pp. 94–105.

[77] G. Santos and B. E. Aguirre, "A critical review of emergency evacuation simulation models," in *Workshop on Building Occupant Movement During Fire Emergencies*, 2005, pp. 27–52.

[78] P.A. Thompson and E.W. Marchant, "A computer model for the evacuation of large building populations," *Fire Safety Journal*, vol. 24, pp. 131–148, 1995.

[79] E. Galea, J. Perez-Galparsoro, and J. Pearce, "A brief description of the exodus evacuation model," in *Proceedings of the International Conference on Fire Safety*, 1993, pp. 149–162.

[80] A. Braun, S. R. Musse, L. P. L. De Oliveira, and B. E. J. Bodmann, "Modeling individual behaviors in crowd simulation," in *Proceedings of the 16th International Conference on Computer Animation and Social Agents (CASA-03)*, 2003, pp. 143–148.

[81] A. Braun, B. E. J. Bodmann, and S. R. Musse, "Simulating virtual crowds in emergency situations," in *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, 2005, pp. 244–252.

[82] N. Pelechano, J. Allbeck, and N. Badler, *Virtual Crowds: Methods, Simulation, and Control*, Synthesis Lectures on Computer Graphics and Animation, Morgan & Claypool, 2008.

[83] A. Rahman, A. K. Mahmood, and E. Schneider, "Using agent-based simulation of human behavior to reduce evacuation time," in *Proceedings of the 11th Pacific Rim International Conference on Multi-Agents: Intelligent Agents and Multi-Agent Systems*, Berlin, 2008, PRIMA '08, pp. 357–369, Springer-Verlag.

[84] S. C. Pursals and F. G. Garzon, "Optimal building evacuation time considering evacuation routes," *European Journal of Operational Research*, vol. 192, pp. 692–699, 2009.

[85] P. Kailiponi, "Analyzing evacuation decisions using multi-attribute utility theory (MAUT)," in *Proc. of 1st Conference on Evacuation Modeling and Management*, 2010, pp. 163–174.

[86] K. Christensen and Yuya Sasaki, "Agent-based emergency evacuation simulation with individuals with disabilities in the population," *Journal of Artificial Societies and Social Simulation*, vol. 11, no. 39, 2008.

[87] C. W. Johnson and L. Nilsen-Nygaard, "Extending the use of evacuation simulators to support counter terrorism," in *Proc. of the International Systems Safety Conference*, 2008.

[88] A. Adamatzky, *Dynamics of Crowd-Minds: Patterns of Irrationality in Emotions, Beliefs and Actions*, World Scientific Series on Nonlinear Science, Series A, University of the West of England, Bristol, U.K., 2005.

[89] D. Kong, S. Lu, Q. Xie, S. Lo, and Q. Kang, "Fuzzy risk assessment for life safety under building fires," *Fire Technology*, pp. 1–15.

[90] H. Furuta and M. Yasui, "Evacuation simulation in underground mall by artificial life technology," in *Proceedings of the Fourth International Symposium on Uncertainty Modeling and Analysis*, 2003, pp. 345–350.

[91] N. Pelechano and A. Malkawi, "Evacuation simulation models: Challenges in modeling high rise building evacuation with cellular automata approaches," *Automation in Construction*, vol. 17, pp. 377–385, 2008.

[92] Transportation Research Board, "Highway capacity manual," in *Transportation Research Circular: Special Report 209*, 2000.

[93] T. Parsons, "Pursuit-evasion in a graph," in *Theory and Applications of Graphs*, Yousef Alavi and Don Lick, Eds., vol. 642 of *Lecture Notes in Mathematics*, pp. 426–441. Springer, Berlin, 1978.

[94] S. M. Lavalle, D. Lin, L. J. Guibas, J. C. Latombe, and R. Motwani, "Finding an unpredictable target in a workspace with obstacles," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 1997, pp. 737–742.

[95] B. Yamauchi, "Frontier-based exploration using multiple robots," in *Proc. of International Conference on Autonomous Agents (Agents '98)*, 1998, pp. 47–53.

[96] J. O'Rourke, *Art Gallery Theorems and Algorithms*, Oxford University Press, New York, 1987.

[97] Y. Amit, J. S. B. Mitchell, and E. Packer, "Locating guards for visibility coverage of polygons," in *Proceedings of the Workshop on Algorithm Engineering and Experiments (ALENEX)*, 2007.

[98] A. Kolling and S. Carpin, "Pursuit-evasion on trees by robot teams," *Trans. Rob.*, vol. 26, no. 1, pp. 32–47, 2010.

[99] A. Kolling and S. Carpin, "Multi robot pursuit evasion without maps," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2010, pp. 3045–3051.

[100] A. Kolling and S. Carpin, "Cooperative observation of multiple moving targets: an algorithm and its formalization," *Int. J. Rob. Res.*, vol. 26, pp. 935–953, September 2007.

[101] B. Tovar, R. Murrieta-Cid, and S. M. LaValle, "Distance-optimal navigation in an unknown environment without sensing distances," *IEEE Transactions on Robotics*, vol. 23, no. 3, pp. 506–518, 2007.

[102] J. Yu and S. M. LaValle, "Probabilistic shadow information spaces," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2010, pp. 3543–3549.

[103] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun, "Collaborative multi-robot exploration," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2000, pp. 476–481.

[104] V. Isler, S. Kannan, and S. Khanna, "Randomized pursuit-evasion with limited visibility," in *Proc. ACM-SIAM Symposium on Discrete Algorithms*, 2004, pp. 1060–1069.

[105] B. P. Gerkey, S. Thrun, and G. Gordon, "Visibility-based pursuit-evasion with limited field of view," *Int. J. Robot. Res.*, vol. 25, no. 4, pp. 299–315, 2006.

[106] S. Bopardikar, F. Bullo, and J. Hespanha, "Cooperative pursuit with sensing limitations," in *American Control Conference (ACC)*, July 2007, pp. 935–953.

[107] H. J. Kim, R. Vidal, D. H. Shim, O. Shakernia, and S. Sastry, "A hierarchical approach to probabilistic pursuit-evasion games with unmanned ground and aerial vehicles," in *Proc. IEEE Conf. on Decision and Control*, 2001, pp. 634–639.

[108] A. Kolling, A. Kleiner, M. Lewis, and K. Sycara, "Solving pursuit-evasion problems on height maps," in *IEEE International Conference on Robotics and Automation (ICRA 2010) Workshop: Search and Pursuit/Evasion in the Physical World: Efficiency, Scalability, and Guarantees*, 2010.

[109] A. Sud, E. Andersen, S. Curtis, L. Ming, and D. Manocha, "Real-time path planning for virtual agents in dynamic environments," in *Proc. of IEEE Virtual Reality Conference*, 2007, pp. 91–98.

[110] T. Oskam, R. Sumner, N. Thuerey, and M. Gross, "Visibility transition planning for dynamic camera control," in *Proc. of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 2009, SCA '09, pp. 55–65, ACM.

[111] C. Lino, M. Christie, F. Lamarche, G. Schofield, and P. Olivier, "A real-time

cinematography system for interactive 3d environments," in *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Aire-la-Ville, Switzerland, 2010, SCA '10, pp. 139–148, Eurographics Association.

[112] C. Vo and J. M. Lien, "Following a Large Unpredictable Group of Targets Among Obstacles," in *Proceedings of the Third International Conference on Motion in Games*, R. Boulic and Y. Chrysanthou and T. Komura, Ed. 2010, pp. 134–145, Springer.

[113] B. Pusharev and J. Zupan, *Urban Space for Pedestrians*, MIT Press, New York, 1975.

[114] F. Aurenhammer, "Voronoi diagrams-a survey of a fundamental geometric data structure," *ACM Comput. Surv.*, vol. 23, pp. 345–405, September 1991.

VITA

Samuel Oscar Rodriguez received his B.S. in computer engineering with a minor in mathematics from Texas A&M University in December 2002. He graduated Magna Cum Laude with University and Foundation Honors. He started research with Dr. Nancy Amato as part of the Undergraduate Summer Research Program. During his graduate career he received the National Physical Science Consortium (NPSC) Fellowship (Fall 2005–Spring 2010), LSAMP Bridge to the Doctorate Fellowship (Fall 2003–Spring 2005), and University Graduate Merit Fellowship (Summer 2003–Fall 2003). His work is focused on multi-agent systems and motion planning algorithms. He received his Ph.D. in Computer Science from Texas A&M University in May 2012.

More information about his research and publications may be found at http://parasol.tamu.edu/people/sor8786. He may be reached at: Parasol Lab, 301 Harvey R. Bright Bldg, 3112 TAMU, College Station, TX 77843-3112.