Path planning in adversarial dynamic environments

Krunal Chande Alex Chang Ajit Krisshna Tim Vergenz

***Abstract*— We demonstrate an algorithm which incorporates uncertainty of the environment's description in planning a path to the goal. We apply this to a game called Humans Vs Zombies, simulated over the Georgia Tech campus map, in which zombies fullfill the role of dynamically uncertain adversaries. Utilizing historical data gathered about the environment, our algorithm allows consideration to be spared for the trade-off between risk avoidance and optimal task completion. We demonstrate it's effectiveness over simpler shortest path planners exercised in the same environment.**

I. Introduction

The motion planning problem for mobile robots is typically formulated as follows. Given a robot and a description of an environment, plan a path of the robot between two specified locations, which is collision-free and satisfies certain optimization criteria. Traditionally there are two approaches to the problem: off-line planning, which assumes a perfectly known and stable environment, and on-line planning, which focuses on dealing with uncertainties as the robot traverses the environment. On-line planning is also referred to by many researchers as the navigation problem. A great deal of research has been done in motion planning and navigation [1], [2].

Path planning in dynamic and uncertain environments is one of the most complicated problems in robot path planning, and is also the most common situation that mobile robots will confront. The additional condition that obstacles are adversarial further restricts the set of acceptable path plans within the environment. Due to the complex, unknown environment, the robot cannot adopt a one-time global motion plan. A static, temporally optimal solution is difficult, if not impossible, to obtain. Additionally, re-planning time for the robot, upon encountering new obstacle information, needs to be short in order to allow sufficient time for locomotion adjustments. In the next section, we discuss prior related research in this area.

II. Related Work

Unlike the situation for path planning in a static environment, limited reports have been found in the open literature discussing optimal path planning in dynamic environments. Complexity and uncertainty increase with the number of the dynamic obstacles. Therefore, traditional path planning algorithms, such as the visibility graphs, the Voronoi diagrams and the grids method, do not perform well in dynamic environments.

Ref. [3] uses a modified A* algorithm to calculate paths for a humanoid robot. The path planning method is applied to real robots rather than simulated on software. A grid of cells is employed to represent the environment. Color cells represent the obstacles and they create a bitmap representing the free spaces and obstacles in the map. The algorithm plans a sequence of foot configurations to navigate toward a goal location based on known static and moving obstacles with predictable trajectories. Ref. [4] discusses a hybrid intelligent approach to path planning for high mobility robots operating in rough environments. It involves the characterization of the environment using a fuzzy logic framework, and a two-stage Genetic Algorithm planner. A global planner determines the path that optimizes a combination of terrain roughness and path curvature. A local planner uses sensory information, and upon encounters with unknown and unaccounted for obstacles, performs on-line re-planning to circumvent them.

Ref. [5] uses a potential field method and applies it to both path and speed planning for a mobile robot in a dynamic environment where the target and obstacles are moving. The robots velocity and trajectory are determined by the relative speed and moving directions of the obstacles and target. The simulation results verify that the method can efficiently track the moving target while avoiding the obstacles along its path.

Many planning approaches utilize two separate algorithms, one for global path planning and another local obstacle avoidance. Two global path planning algorithms which have been successfully applied for the DARPA Urban Challenge are the Hybrid A* algorithm [6] and planning in state lattices [7]. Both algorithms search for an optimal path by successively concatenating short motion primitives until reaching the goal. The Hybrid A* algorithm operates simultaneously in a continuous and discrete state space, which results in the reachable set forming a tree in the state space. We use another method called D* (Dynamic A*) [8]. It is a typical method used in path planning in dynamic environments. It plans optimal paths in real-time by incrementally repairing paths to the robots state as new environment information is known, which makes it possible to greatly reduce the computational cost. When the robot gathers new information about the environment, it will re-plan new paths based on the new information and produce a path for the robot. D* does not require re-planning of the entire path on each occasion new information becomes available. It updates the path arc cost locally when environment changes.

Like A*, D* operates on a cost graph. The main idea of the method is illustrated as follows: From the initial state, the method repeatedly selects the neighbour with the minimum cost until it propagates to the goal. Each small cell in the map is called a state. Each state X has the arc cost from the state X to the goal given by the path cost function $h(X, G)$. From the start point (start state), all neighbour states of the current state are listed on the open list. From the open list, the method calculates the arc cost of the states by $h(X, G)$. Then, it selects the state with the minimum $h(X, G)$, goes to this state, and adds new neighbours to the open list.

III. Methods

Humans vs. Zombies (HvZ) is a massive, live-action game that is played at many college campuses across the nation, with players numbering in the hundreds. All players begin as humans with a single random player selected as the first zombie. A game of tag, over the course of weeks, ensues in which humans tagged by zombies become zombies themselves. At each site, the indoors of specific buildings are designated as 'safezones' where zombie attacks are invalid. The game completes when all humans at a site have become zombies or all zombies have perished after having not tagged a human for 48 hours.

The HvZ environment provides both a convenient and motivating context in which to apply our pathfinding algorithms. The zombie-infested environment is both changing and uncertain; zombies are, by definition, dynamically uncertain and adversarial obstacles. Additionally, statistical information comprising frequency and locations of zombie attacks is derivable from prior HvZ event records.

The HvZ pathfinding algorithm determines a probabilistically viable path through an environment populated with adversarial and dynamic flesh-eating zombies. We illustrate it in two parts, an offline component and an online component. Off-line components of the algorithm consist of data processing and environment establishment. The product of this ultimately serves as an input, or a scenario, in which the online components of the algorithm are executed.

A. *Offline Algorithm Components*

Off-line components of the algorithm are further subsubdivided into tasks related to zombie density map generation, zombie instantiation as well as parameter configuration. The latter is a user-oriented task, allowing configuration of various inputs to the algorithm that may change the efficiency and optimality of the path plans generated.

Zombie Density Map: Data from previous seasons of Humans Vs. Zombies at Georgia Tech. is available as a list of zombie attacks and their locations. By discretizing the Georgia Tech. campus map into a 401 x 294 square grid, we are able to compile this data into a histogram of zombie attacks that have occurred at each discretized position on the Georgia Tech. campus, over the past 1.5 years. We designate this data representation as the 'zombie frequency map'. In order to facilitate the exploitation of this data by a path finding algorithm, we take inspiration from the concept of potential fields. A 2-dimensional gaussian distribution is generated with standard deviation, σ, and a skirt size, N$_{gauss}$ (both in units of discretized position). Both of these gaussian distribution parameters also serve as user-adjustable inputs to the overall algorithm. The gaussian distribution is then convolved across the zombie frequency map, producing a topology similar to that illustrated in Fig. 1, where gaussian filter parameters have been set to, σ = 6 and N$_{gauss}$ = 24. In spirit of the reality that no position within the map is completely 'safe' (aside from obstacles and safe-zones), and hence should not hold a value of '0', a small bias is added to all positions in the 'zombie density map'. The bias is currently arbitrarily defined to be 1% of the spatial average of 'zombie density map' values and is another tunable parameter of the overall algorithm.

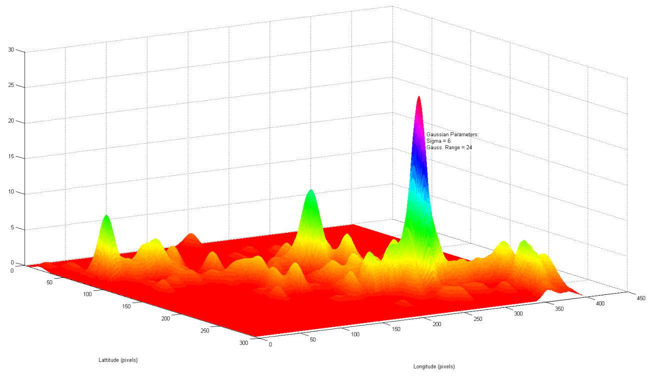


Fig. 1: Zombie density map: smoothed topology representing HvZ zombie attack frequency on GT campus (σ = *6*, N_{guass} = *24*)

We designate this data representation as the 'zombie density map' across the Georgia Tech. campus. Fig. 2, provides a color intensity plot of the 'zombie density map', overlayed on top of the Georgia Tech. campus map. Immobile obstacles (eg. building walls) are displayed in black, HvZ safe zones (eg. indoors of buildings) in gray and zombie density values, at each position, as an intensity from red to yellow, corresponding to low to high density values, respectively. An obstacle that is a single discretized grid cell (pixel) thick, serves to bound the entire map in order simplify path planning.

A high-level walkthrough of construction of the zombie density map is illustrated below:

1) Import past HvZ zombie attack location data on the GT campus.

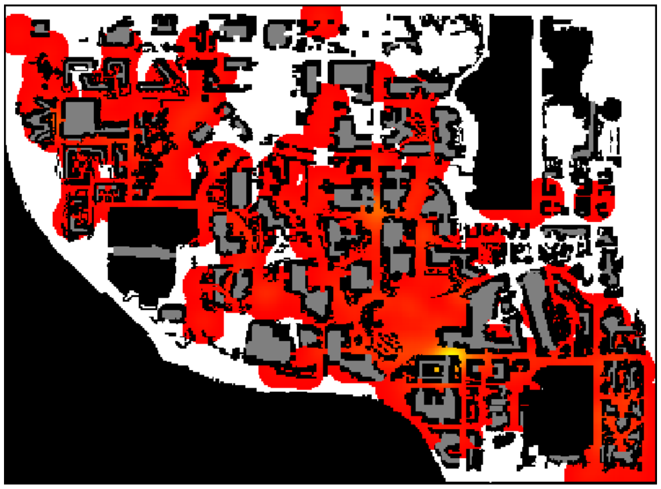


Fig. 2: Overlay of zombie density map over Georgia Tech campus (401 x 294 pixels)

2) Compile data into a position-referenceable tally of zombie attacks to produce ’zombie frequency map’.
3) Generate gaussian distribution using user-defined σ and N$_{gauss}$.
4) Convolve gaussian distribution against the ’zombie frequency map’ from Step 2, to construct the ’zombie density map’.
5) Export the ’zombie density map’ as CSV file.

Zombie Entity Instantiations: Upon constructing the zombie density map, dynamic zombie entities are instantiated to operate in the simulated, discretized GT campus environment. Any given position in the discretized map will have a probability of a zombie instantiation based on its value in the zombie density map. Positions with higher values (i.e. locations where a greater number of zombie attacks have historically been reported) will yield higher chances of a zombie instantiation. The number of zombies to instantiate for a scenario is an adjustable input parameter to the overall algorithm.

B. Online Algorithm Components

Using the scenario established by the offline components of the algorithm, online processing can now be executed to move the agent from its start configuration to its goal configuration. The configuration of our agent consists solely of its position in the discretized Georgia Tech. campus map (401 x 294 square grid). Both the agent and all adversarial zombies are restricted to Manhattan mobility in this discretized environment.

Fig. 3, illustrates an annotated snapshot of the HvZ simulator GUI. Zombies are represented as single pixel yellow dots. The current agent and goal configurations are represented as blue and green dots, respectively, with a transparent blue circle representing the agent’s visibility range skirt. A red heat map is overlayed on top of the campus map to illustrate current ’zombie density map’; greater red opacity is used to represent higher zombie density map values at a particular location.

Goal-Oriented Locomotion: Locomotion of our agent from its starting configuration to its goal configuration is accomplished through D* Lite path planning, using the ’zombie density map’ to determine grid cell costs across the Georgia Tech. campus map. The decision to utilize the D* Lite path planning algorithm was due to its efficiency in path re-planning after deviations or unexpected obstacles are encountered. Although, our algorithm doesn’t currently allow for re-active re-planning upon encounters with zombies, this is a highly anticipated enhancement for the future. The D* Lite implementation used is part of an open source library of utilities, ’robotutils’, made available by Pras Velagapudi of the Robotics Institute of Carnegie Mellon University [9]. A high-level walk-through of the locomotion planning algorithm follows:

1) Establish agent start and goal configurations
2) Generate map defining the cost for each discretized position, weighted based on ’zombie density map’ values at those positions.
3) Calculate D* path from the start to goal configuration
4) Agent traverses the D* path
5) If agents current position is the goal, exit the algorithm with success.
6) If a zombie is present within the agent range skirt:
    - If zombie’s current position is the agent’s current position, exit the algorithm with failure.
    - Attempt to stun zombie
    - Go to beginning of Step 6
7) Go to Step 4

Zombie Locomotion: Zombie movement toggles between two distinct behaviors. The default zombie behavior is to execute a random walk. The intent is to allow zombies uncertain mobility but in such a way that they will maintain the integrity of the ’zombie density map’. After zombies are instantiated (based on the ’zombie density map’), a random walk will have the tendency to keep the zombies within the same area in which they were instantiated. Additionally, each zombie is assigned a movement speed determined by a uniform distribution between 0.5 and 1.5 discretized grid cells per turn. In implementation this movement speed manifests as 0, 1 or 2 steps per turn, but in such a way that the intended uniform distribution is maintained.

Zombie Stunning: In accordance with the rules of actual HvZ events, a human can stun a zombie by hitting it with a sock. We have modeled this behavior by allowing a player to attack zombies within its visibility range skirt. The success of this attack is modeled by a probability that is greatest at a position adjacent to the agent and linearly decreases until it reaches its least value four grid cells away from the agent. There is no limit on the number of times an agent can attempt to stun a zombie, however, a penalty is implicitly incurred because the agent must remain immobile for the time step during which it attacks a zombie. Once a zombie is stunned, it is removed from the map. The probability values for successfully stunning a zombie are given by the formula

$$P(\text{hit}) = 0.6 - 0.1 * (\text{Manhattan distance})$$

These values have been determined through empirical tuning.

C. Parameter Configuration

This algorithm entails several user-adjustable parameters which impact the performance of the path plan that is generated. We experiment with some of these parameters and present our analysis in a subsequent section. Several of the parameters are listed below for convenience:

1) Agent start/goal configurations
2) Number of zombies (*zombie entity instantiations*)
3) Standard deviation of the gaussian filter, σ (*zombie density map*)
4) Discrete radius of gaussian filter, N$_{gauss}$ (*zombie density map*)
5) Range skirt size (i.e., agent’s sensory range)

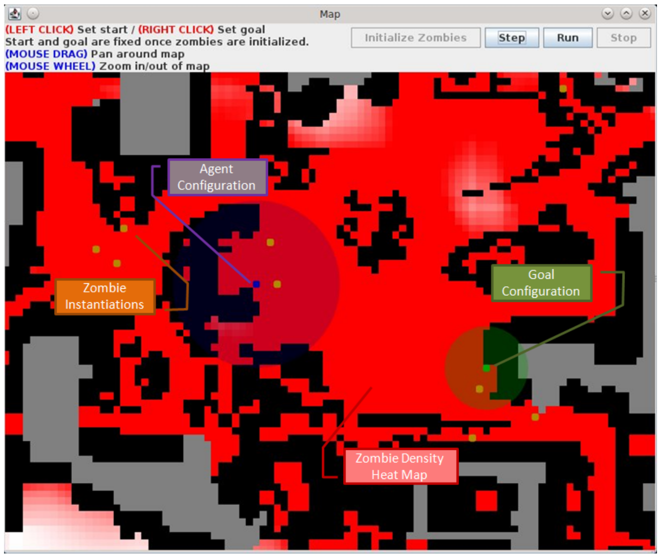

(LEFT CLICK) Set start / (RIGHT CLICK) Set goal
Start and goal are fixed once zombies are initialized.
(MOUSE DRAG) Pan around map
(MOUSE WHEEL) Zoom in/out of map

Fig. 3: HzV simulator GUI (agent = blue; zombies = yellow; goal configuration = green; heat map = red)

IV. Experiments

The environment in which the algorithm executes is inherently uncertain. Locations of zombie instantiations are determined probabilistically and zombie motion is a random walk. As a result, experimentation requires execution of several hundred or more trials, for a given set of test parameters, and metrics are measured as averages over all trials. In our experiments, we measure algorithm performance through the metric of percentage goal attainment over a given number of trials. This measures how often the agent, in simulation, is able to reach its goal configuration using the planned path, generated with specified test parameters. The results of our algorithm are compared against a control planner being tested in identical experiments.

Although the algorithm accommodates several user-adjustable parameters, we choose a subset of these with which to experiment, and document their impact on the algorithm's performance. We begin by testing the algorithm's performance using pre-determined agent start and goal configurations. In the first experiment we execute 1000 trials of the algorithm, producing path plans from Woodruff Dormitory to Clough Undergraduate Commons. A subsequent experiment is run using the same setup, but with the start and goal configurations set to Clark Howell Hall and Clough Undergraduate Commons, respectively; a shorter euclidean distance than the former experiment. We also run 1000 trials using randomized start and goal configurations. In all cases the numbers of zombies instantiated on the map is held at a constant value of 50 zombies. Finally, another four sets of 1000 trials is run for the case of randomized start and goal configurations, where the population of zombies is set to 40, 50, 60 and 70 zombies.

A. Agent Start/Goal Configurations

In summary, three different start and goal agent configurations are tested and the algorithm's percentage success is captured:

1) Start Configuration: Woodruff Undergraduate Dormitory, Goal Configuration: Clough Undergraduate Commons
2) Start Configuration: Clark Howell Hall, Goal Configuration: Clough Undergraduate Commons
3) Start Configuration: Random, Goal Configuration: Random

B. Zombie Population

For each of the start/goal configurations above, we maintain the size of the zombie population instantiated at the start of the simulation to be constant (50 zombies). Additionally, for the scenario of randomized start and goal configurations, we run 1000 trials with the zombie population set to each of 40, 50, 60 and 70 zombies.

C. Control Group

As a control, we compare the performance of our algorithm with a non-risk based generic D* algorithm, which finds the minimum distance from start to goal. This shortest path planner ignores the information available from the 'zombie density map' and treats all grid cells to be of uniform cost. It is run against a set of experiments identical to those used to test our HvZ path planning algorithm above.

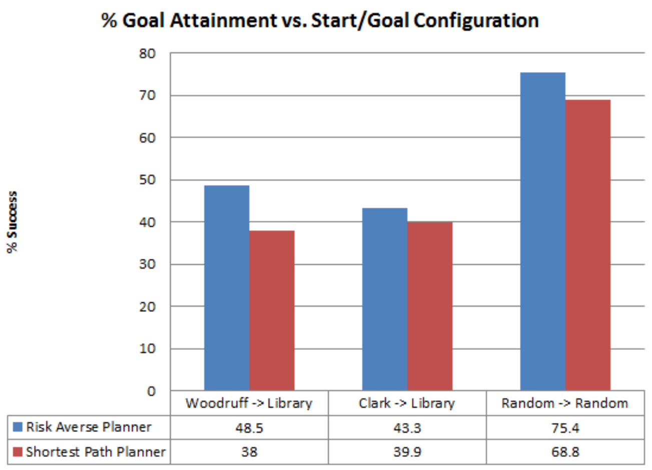
V. Analysis

A. Start and Goal Configuration Variation

Our HvZ algorithm was found to yield a higher success rate in each of the start and goal configurations tested. The Woodruff Dormitory to Library (Clough Undergraduate Commons) scenario was meant to serve as a higher risk endeavor where the agent would need to cross nearly the entire map as well as regions comprising high 'zombie density map' values (higher probabilities of zombie presence). The Clark Howell Hall to Library (Clough Undergraduate Commons) scenario was intended to represent a lower risk scenario where the euclidean distance between start and destination were relatively much smaller. Finally, the randomized start and goal configuration scenario represents a more comprehensive test potentially covering various types of paths over the course of 1000 trials. Fig. 4 demonstrates the percentage of trials in which the agent successfully reached its goal using both the our HvZ planner (labeled as the 'Risk Averse Planner') and the 'Shortest Path Planner' described previously, for each start/goal test configuration. Table I illustrates the data collected from this testing.

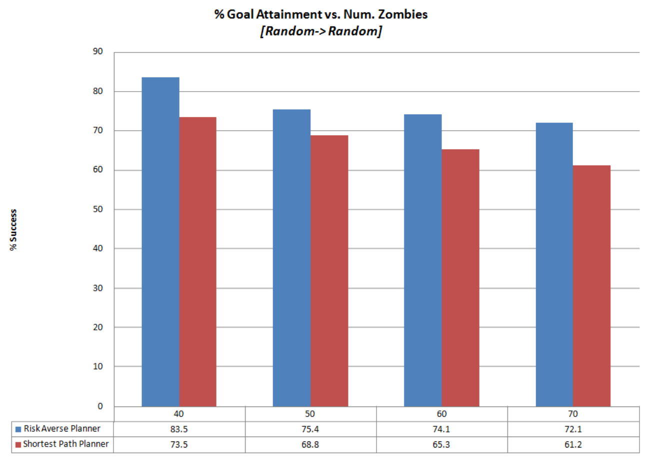
TABLE I: Comparison of success rates in Risk Averse and Shortest Path Planners for a zombie population of 50

	Risk Averse Planner	Shortest Path Planner
Start Config. $->$ Goal Config.	Success Rate (%)	Success Rate (%)
Woodruff$-$ >Library	48.5	38
Clark$-$ >Library	43.3	39.9
Random$-$ >Random	75.4	68.8



	Woodruff -> Library	Clark -> Library	Random -> Random
🟦 Risk Averse Planner	48.5	43.3	75.4
🟥 Shortest Path Planner	38	39.9	68.8

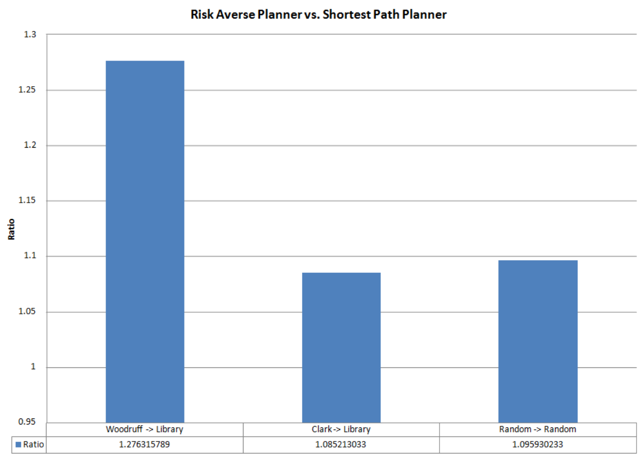
Fig. 4: Performance of Risk Averse Planner vs. Shortest Path Planner over all start/goal test configurations (zombie population = 50)



	40	50	60	70
🟦 RiskAverse Planner	83.5	75.4	74.1	72.1
🟥 Shortest Path Planner	73.5	68.8	65.3	61.2

Fig. 6: Performance of Risk Averse Planner vs. Shortest Path Planner using randomized start/stop goals and varying zombie populations

Although it may be premature and require further future investigation, we notice what may be a trend in which larger euclidean distances between the start and goal configurations yield a larger ratio between our 'Risk Averse Planner' success rate and the 'Shortest Path Planner' success rate. The scenario of randomized start and goal configurations seemed to yield a ratio between that of the other two scenarios, that was 'middle of the way'. Fig. 5 provides a chart illustrating this trend in the ratio of success rates between the two planners.



	Woodruff -> Library	Clark -> Library	Random -> Random
🟦 Ratio	1.276315789	1.085213033	1.095930233

Fig. 5: Ratio of Risk Averse Planner to Shortest Path Planner success rates over all start/goal test configurations

B. Zombie Population Variation

In our final experiment, we exercised the algorithm using randomized agent start and goal configurations, assigning the zombie population with each of values of 40, 50, 60 and 70. 1000 trials were run using each of the zombie population values. Fig. 6 illustrates the results; the 'Risk Averse Planner' exhibits greater success rates than the 'Shortest Path Planner' for all zombie planner values.

C. Qualitative Observations

Our simulator interface allows monitoring of the agent as it traverses the Georgia Tech. campus map, from start to goal configuration, as well as its interactions with zombies encountered along the way. This capability has yielded some useful insights into the workings of our risk averse algorithm versus the shortest path planner which is serving as our control. In the paths that the risk averse planner produces, we can observe that paths through safezones are preferred whereas the shortest path planner doesn't apply any weight to one type of area versus another. Additionally, we observe that our risk averse algorithm tends to produce paths resembling straight paths, but distorted in order to curve around regions of intense red, corresponding to high zombie density map values. More intuitively, the paths often resemble straight lines that have been modified to deviate and circumvent areas where the probability of zombie presence is greater, based on historical HvZ data.

VI. Discussion

While devising an algorithm to address the problem of path planning and real-time navigation through an uncertain, dynamic and adversarial environment we came to the conclusion that the scope of the problem attempting to be tackled was too large for the given time frame. Instead, a decision was made to partition the problem and address a smaller scope. As a result, the capability of the agent to stun zombies within its range skirt was reserved for later consideration. This removed an aspect of the problem whose solution which we anticipated to involve integration of a local decision-making algorithm, such as Markov Decision Processes (MDPs), with global path planning.

From an algorithmic perspective, we expect future enhancements can be made to lift the restriction on agents and adversarial zombies to Manhattan mobility, perhaps through the application of a hexagonal grid. The effect of map discretization and granularity on algorithm performance, optimality and other characteristics also presents itself as an interesting area of experimentation. Additionally, to bring the algorithm closer to real-world application, work is available to integrate the possibility of fighting (versus simply fleeing/avoidance) through stun actions, which HvZ event rules allow. We anticipate that this addition to the algorithm will entail policy-making and weighing of the long-term value of actions, a problem potentially well-suited for Markov Decision Processes (MDPs).

From the perspective of practical application, wed like to see the algorithm integrated into a mobile app utilized by future HvZ participants on the Georgia Tech. campus. This endeavor would potentially allow for measurement of the effectiveness of the algorithm, and tuned parameters, in a real-world scenario. Results would be compared against control groups not relying on the mobile app to provide path planning and reactive guidance.

ACKNOWLEDGMENT

We would like to thank Prasanna Velagapudi for providing an easy to use Java library called “’robotutils’. It really helped us with easier UI design and the D* implementation.

REFERENCES

[1] Latombe, J.C., Robot Motion Planning. Kluwer Academic Publishers, 1991.

[2] Yap, C.-K., Algorithmic Motion Planning, In Advances in Robotics, Vol.1: Algorithmic and Geometric Aspects of Robotics, J.T. Schwartz and C.-K. Yap Ed., pp.95143, Lawrence Erlbaum Associates, 1987.

[3] J. Chestnutt and M. Lau, Footstep Planning for the Honda ASIMO Humanoid, IEEE International Conference on Robotics and Automation, pages 629-634, Apr 2005

[4] M. Tarokh, Hybrid Intelligent Path Planning for Articulated Rovers in Rough Terrain, Fuzzy Sets and Systems, vol 159, pages 2927 2937, Nov 2008

[5] L. Huang, Velocity Planning for a Mobile Robot to Track a Moving Target A Potential Field Approach, Robotics and Autonomous Systems, vol 57, pages 55 63, Jan 2009

[6] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, Path planning for autonomous vehicles in unknown semi-structured environments, The International Journal of Robotics Research, vol. 29, no. 5, pp. 485501, 2010.

[7] M. Pivtoraiko, R. A. Knepper, and A. Kelly, Differentially contrained mobile robot motion planning in state lattices, Journal of Field Robotics, vol. 26, no. 3, pp. 308333, 2009.

[8] A. Stentz, Optimal and Efficient Path Planning for Partially-Known Environments, Proceedings of the IEEE International Conference on Robotics and Automation, pages 3310 3317, May 1994.

[9] “robotutils” - Java library for simple robotics applications http://www.snowbotic.com/about