DYNAMIC WAYPOINT NAVIGATION USING VORONOI CLASSIFIER METHODS

J. Overholt*, G. Hudas, G. Fiorani, M. Skalny, A. Tucker
U.S. Army RDECOM-TARDEC
Robotics Mobility Laboratory
Warren, MI 48397-5000

ABSTRACT

This paper details the development of a dynamic waypoint navigation method which introduces and utilizes *Voronoi classifiers* as the control mechanism for an autonomous mobile robot. A *Voronoi* diagram may be generated by any finite set of points in a plane. For mobile robot control each point in the plane represents a *Voronoi classifier*. The classifiers are used to generate *Voronoi* regions. As a robot comes into a *Voronoi* region the classifier will act as a control input; providing a new waypoint for the vehicle to follow. The robot moves towards the new waypoint unless interrupted by an obstacle or wall. The robot will get a new waypoint from the classifier in the robot's current *Voronoi* region. This process continues until the robot has terminated at a desired position (goal) or runs out of power.

1. INTRODUCTION

Of the many areas of research in the field of autonomous mobile robotics, waypoint navigation has actually enjoyed some measure of success. The U.S. Army has demonstrated the capabilities of waypoint navigation (sometimes known as laying out 'electronic breadcrumbs') in several well publicized experiments (such as Demo III leader-follower activities). Waypoint algorithms can exhibit problems when unknown obstacles or dynamically changing terrain alters the path between consecutive points (**Figure 1**).

Ideally the algorithm should adapt to the new information. Stenz's D* algorithm (Stenz, 1994) computes an initial optimal path to the goal. As the robot moves to the goal a new optimal path is computed every time it finds new information about its environment. This provides optimality in the local sense (only optimal in the global sense if all elements of the environment are known initially and are used in the computation of the initial path). The drawback to this method is that the robot could be spending a significant amount of time computing new optimal paths instead of moving to the goal. In obstacle rich environments a non-optimal robust path planning approach may be more desirable.

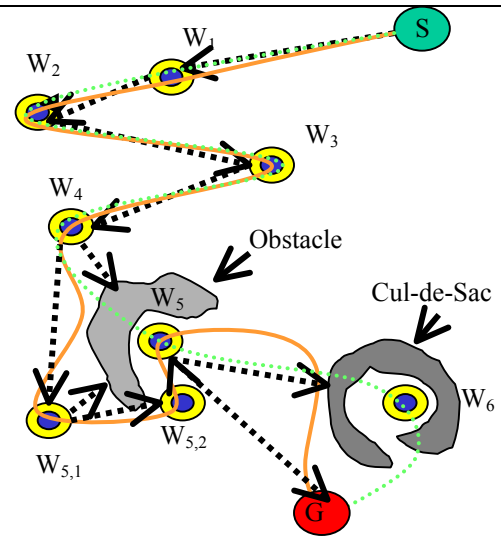

Figure 1: Waypoints used to get to a goal (W_1-W_6). Adaptive algorithm creates new waypoints ($W_{5,1}$ & $W_{5,2}$) and eliminates unobtainable ones (W_6). Final paths to the goal are shown in green (ideal) and orange (actual).

We have been investigating the D* algorithm and have hypothesized an alternative path planning approach based on classification and *Voronoi* diagrams (Fortune, 1995). This research deals with the development of a dynamic waypoint navigation method which utilizes *Voronoi* classifiers as the control mechanism for an autonomous mobile robot.

2. VORONOI CLASSIFIERS

A *Voronoi* diagram may be generated by any finite set of points in a plane. The partitioning of a plane with *n* points into *n* convex polygons such that each polygon contains exactly one point and every point in a given polygon is closer to its central point than to any other.

Imagine the plane is a map of a city, and the points are polling locations. Election law requires that each citizen vote at the polling place which is closest to their home. **Figure 2** shows that the resulting election districts form a *Voronoi* diagram for the polling locations.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
00 DEC 2004	**N/A**	**-**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Dynamic Waypoint Navigation Using Voronoi Classifier Methods	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **U.S. Army RDECOM-TARDEC Robotics Mobility Laboratory Warren, MI 48397-5000**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
See also ADM001736, Proceedings for the Army Science Conference (24th) Held on 29 November - 2 December 2005 in Orlando, Florida. , The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**UU**	**6**	

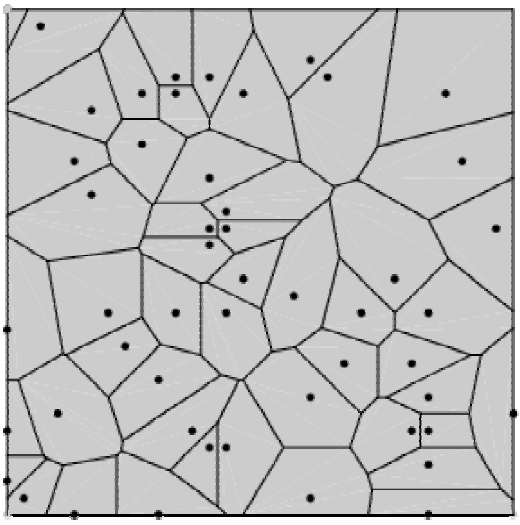

Figure 2: Voronoi regions

We are extending this concept of *Voronoi* diagrams to classification and control for mobile robots operating in a plane (*although the method can be logically extended to 3-dimensional surface representations as well*). For mobile robot navigation we are interested in placing a set of navigation 'beacons' in a plane that provides waypoint commands for a mobile robot. The beacon that is the closest to the robot at any given time will provide the current control input. Each beacon (point) in the plane represents what we call a *Voronoi classifier* (**Figure 3**).

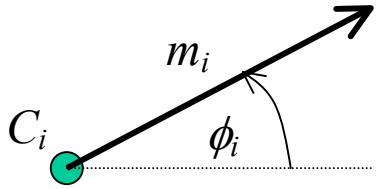

Figure 3: Voronoi classifier C_i; with magnitude m_i and direction (or orientation) ϕ_i.

Each *Voronoi classifier* will define a *Voronoi* region (as described in the preceding paragraphs). The magnitude and orientation of each classifier will be used to generate the next location for the mobile robot to move. The next waypoint, *W(k+1)* can be generated by the simple set of geometric equations (1a) and (1b);

$$W_x(k+1) = W_x(k) + m_i \cdot \cos(\phi_i) \qquad (1a)$$

$$W_y(k+1) = W_y(k) + m_i \cdot \sin(\phi_i) \qquad (1b)$$

The subscript 'i' refers to the robot being in the i^{th} *Voronoi* region. The robot will continue to generate new waypoints unless interrupted by an obstacle or wall. In this case the robot will generate a new waypoint from the classifier in the robot's current *Voronoi* region. This process continues until the robot has terminated at a desired position (goal) or runs out of power.

Figure 4 shows a collection of *Voronoi classifiers* (C_1-C_5). Two initial starting points are also shown (b_0 and y_0).

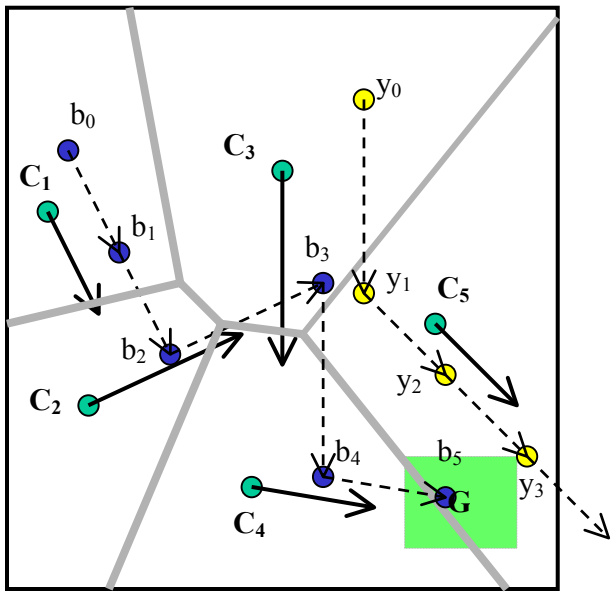

Figure 4: Trajectories on a playing field generated by Voronoi classifiers. The blue trajectory terminates at the goal. The yellow trajectory leaves the playing field.

Starting with the initial positions b_0 and y_0, each trajectory is generated using the following 3 steps;

1. If you are at the goal, end the run (*successful*).
2. If you are off the playing field, end the run (*unsuccessful*).
3. If you are not at the goal and on the playing field move in the direction and magnitude of the classifier contained in your current *Voronoi* region.

The blue trajectory terminates at the goal in 5 steps (*successful*). The yellow trajectory leaves the playing field in 4 steps (*unsuccessful*). This simple example shows that the current set of classifiers will not generate trajectories that will all terminate at the goal. Underlying the classifier set is another performance representation which will show areas of the playing field that will terminate at the goal (*stable*) or leave the surface (*unstable*) called the *color map*.

3. Voronoi Classifier Color Maps

The *Voronoi classifier* set from **Figure 4** is re-examined in **Figure 5**. The classifiers produce some *unstable* regions (denoted by the red areas on the map).

There are also *stable* regions on the map as well (denoted by the green areas). Some of the *unstable* regions were found through simple geometric means. The goal region was propagated backwards through the playing field to find some of the *stable* regions (a much more difficult process than stated). The remaining regions are unknown and can be found by brute force or approximate measures (*i.e. initialize a trajectory at an unknown location and see where it terminates*).

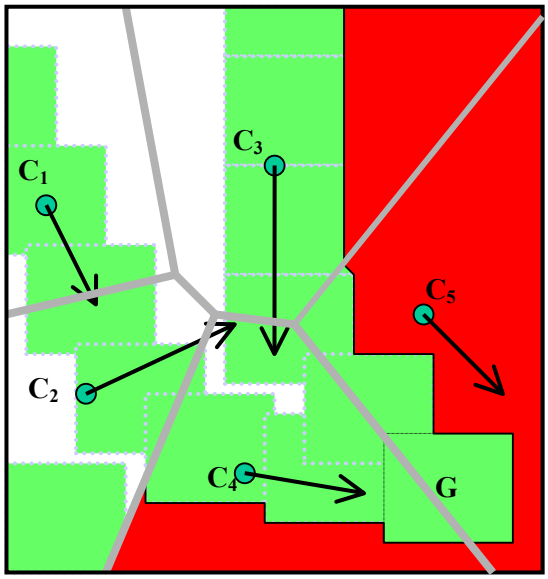

Figure 5: Partial color map of a Voronoi classifier set. Green (stable) regions found by propagating the goal region through the playing field. Red (unstable) regions found through geometric and superposition means.

The *Voronoi classifier color map* is a visual, multi-colored representation of the cost associated with every point on the playing field. Cost can be assigned using any number of cost functions, as long as the actual cost values can be grouped or form a gradient. A simple cost function that we use is based on the number of classifier queries made (or steps taken), plus large penalties for hitting an obstacle and larger penalties for leaving the playing field. For the color map, colors are assigned by group for the following 3 conditions;

Trajectories reach goal → green
Trajectories leave playing field → red
Trajectories stay on the playing field after a fixed number of steps or collide with an obstacle → yellow

A critical issue that arises when calculating the color map is exactly how all the costs are calculated. The naïve way is to start from every point on the playing field, and calculate the number of queries that are made before the goal is reached, an obstacle is hit, or the playing field bounds are exceeded. This approach is infeasible, because there are infinite points on the playing field, so an estimate must be used instead. Our current approach is to break the playing field into a grid, and use one or more sample points within each grid square as a representative for that grid square. The drawback of this approach is that a limited number of points are being used to compute the entire color map, but it does correlate well to the pixilated nature of computer screens.

As is seen in **Figure 6**, the color map provides an easy way to visualize the cost associated with every point on the playing field. The color map is also easy to represent for use by a computer algorithm. This versatility and ease of use makes the color map the key component for editing and tuning the classifier set. For a computer algorithm, the color map represents a set of cost values that can be used to qualify and quantify the effectiveness of a set of classifiers. With a human in the loop, adjusting the classifiers to get a better solution is as simple as making sure any classifier changes lead to more green areas, or conversely, less red and yellow areas. If the underlying mathematical relationships could be found and the right software tool (*PTMS-discussed in the next section*) were developed, a user could place, alter, add or remove (PAAR) classifiers from a playing field and get immediate feedback on its overall effect on the stability of the changes.

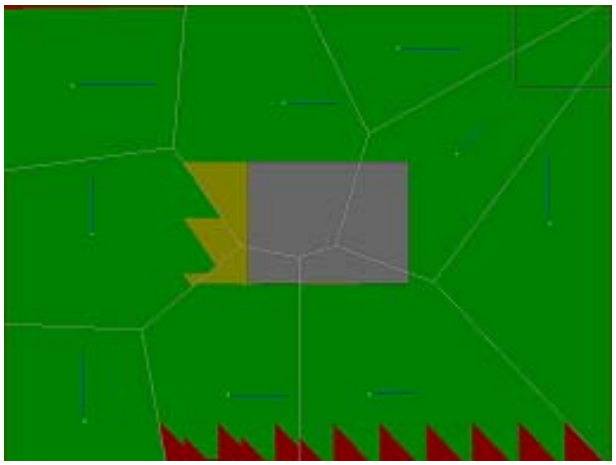

Figure 6: Color map using three different colors to represent cost groups (screen capture from PTMS software). A large rectangular obstacle is in the middle of the playing field. A majority of the trajectories will terminate at the goal (upper right hand corner). Others will stagnate at the obstacle or leave the playing field (red saw-tooth structure at the bottom).

We recognize the utility of the color map and its close coupling to the *Voronoi classifiers* and regions. The user could intercede or an automated procedure could be developed that maximizes the stable regions of the map. **Figure 7** shows a possible scenario on how the *Voronoi* diagrams could be used in conjunction with a

classic path finding scheme, such as A* (Pearl, 1984), for developing stable classifier sets.

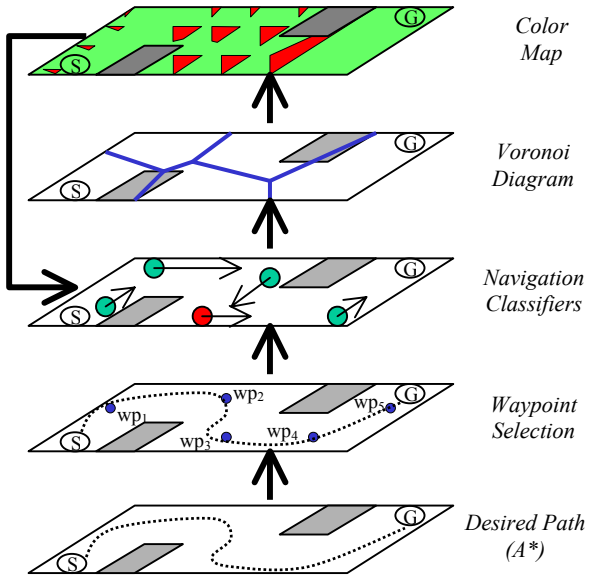

Figure 7: Layered waypoint navigation maps based on A* initialization and Voronoi classifiers.

1. The initial path to the goal is computed using some known path planning technique (e.g. A*).
2. A minimal set of waypoints are set along the path.
3. A minimal set of *Voronoi classifiers* are placed on the playing field that will move the robot successfully to all the desired waypoints.
4. The *Voronoi* diagram is computed.
5. The color map of the classifier set is found.
6. If the color map stability is not at some desired level PAAR changes are needed in the classifier set.

Moving the *red* classifier in *layer 3* slightly to the right (*place*) will result in 100% stability in the color map (i.e. all trajectories, regardless of the initial starting point, will terminate at the goal).

4. POINT-MAGNITUDE SIMULATION (PTMS)

We have developed a *Voronoi classifier* simulation tool to aid us in our research. Originally, the software was meant to be a simple simulation of an agent following the path given by the classifiers and was named the Point-Magnitude Simulation (PTMS), but it has evolved into more of a visualization tool. The software currently runs under Microsoft Windows and uses the OpenGL graphics library.

PTMS has *Microsoft Paint*-style toolbars that allow the user to draw classifiers, obstacles, and goals on a playing field (**Figure 8**). The software computes and displays the *Voronoi* diagram, the Delaunay triangulation, and the color map in near real-time as the user manipulates objects on the field. The user can also place a number of points on the field that will be used as starting points for test paths. When a user places one of these points, PTMS will trace and display the trajectory that an agent would take from the specified starting point, using the instructions of the *Voronoi classifiers*.

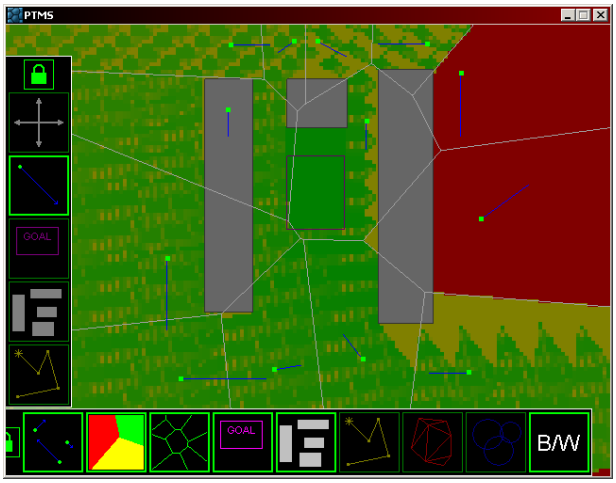

Figure 8: ***PTMS screenshot showing classifiers, obstacles, Voronoi diagram and color map. Note the large red (unstable) region of the playing field on the right side.***

Currently, the software supports only rectangular obstacles and goals and computes the color map by an iterative approximation. We plan to increase functionality by adding support for a number of features, including but not limited to:

- Arbitrary polygonal obstacles/goals
- Multiple obstacle types (e.g. hole vs. tree stump)
- More efficient computation algorithms
- Floor plan importing
- Terrain map importing
- Kinematics and dynamics based simulations

We also plan to make PTMS a feature-rich tool for performing *Voronoi classifier* simulations in realistic virtual environments with accurately simulated robots.

5. USING PTMS FOR VORONOI NAVIGATION

This section looks at two examples of using the PTMS tool to implement and visualize a *Voronoi classifier* set that can be used for navigation. The first example goes step-by-step through the process of placing classifiers in a simple single-obstacle environment. The second shows a more complicated, hallway environment and demonstrates the capabilities of the *Voronoi classifier* method.

5.1 Using Voronoi classifiers in a single-obstacle environment

We start off with an initial environment as shown in **Figure 9**. There is a single obstacle in the middle, with the goal just beyond it. Three initial classifiers have been placed. The color map shows that a majority of the playing field is unstable (red). A trajectory starting on most parts of the playing field will end up off the playing field or running into the obstacle and stalling (yellow).

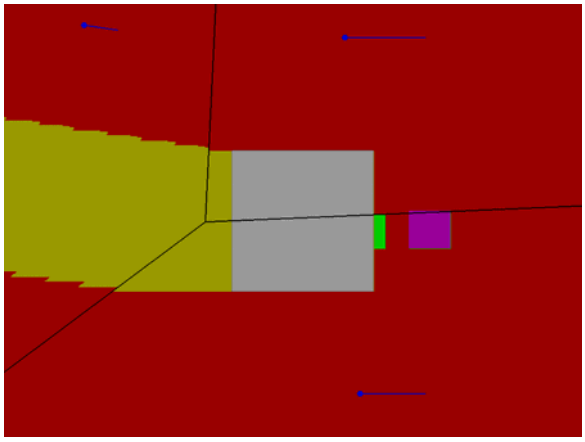

Figure 9: Initial classifier placement with color map.

The next step is to begin adding classifiers that increase the area that leads to the goal – adding more green to the color map. We are still investigating strategies for placement and tuning of classifiers, but a generally good approach is to add classifiers to red areas and make them point towards the goal or towards other green areas. **Figure 10** shows that we have added two classifiers that point towards the goal, making their associated *Voronoi* regions green.

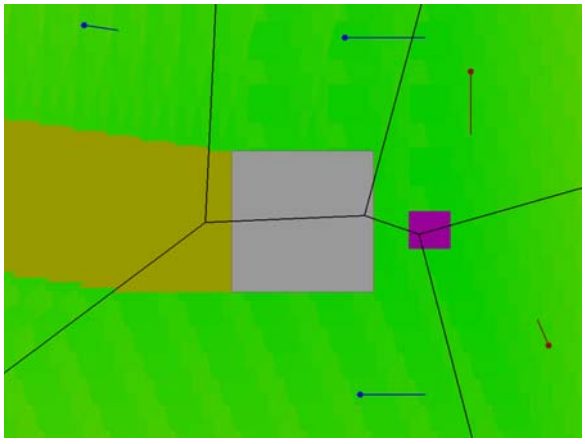

Figure 10: Two new classifiers (shown in red) added on the right side that point to the goal. A majority of the color map changes to green (stable).

Also a consequence of these new classifiers is that the original classifiers' regions also become green, because they lead to the new stable regions.

The final step in getting an entirely green color map is to eliminate the yellow region in which the agent will be directed into an obstacle. This is the same process as the previous step – add classifiers in the yellow area that point to green areas. **Figure 11** shows the final classifier set with a totally green color map after the addition of the final classifier. All trajectories, regardless of the initial location will eventually stop at the target.

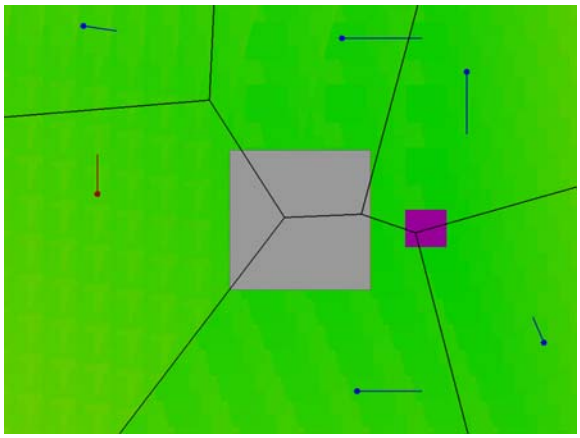

Figure 11: One last classifier (shown in red) added to the left-center of the playing field to make the entire color map green.

5.2 Voronoi navigation in a closed hallway

Figure 12 shows a more complicated example of using *Voronoi classifiers* and the color map to navigate a hallway. This shows that using the same process as in the single obstacle example, more complicated environments can be successfully navigated using a set of *Voronoi classifiers*. The underlying color map is shown in **Figure 13**.

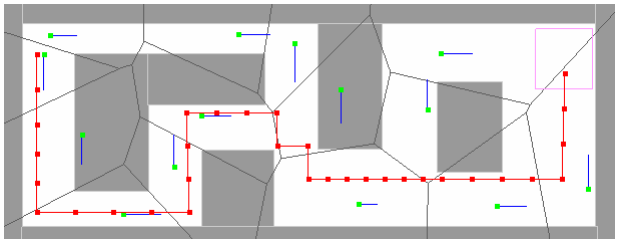

Figure 12: A set of Voronoi classifiers used to negotiate a structured indoor environment to a goal location. The coverage of this set is nearly 100%. Dark gray areas are walls or impassable objects.

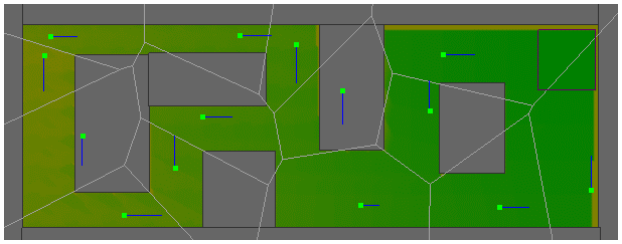

Figure 13: Color map for hallway waypoint navigation using Voronoi classifiers.

CONCLUSIONS

We have developed the theory and the framework for a robust navigation system where waypoints to a goal can be generated and guaranteed to terminate at the desired final location. We have also developed a rudimentary software tool for interactively designing *Voronoi classifier* sets in cluttered, obstacle strewn environments. In addition, an underlying color map can be generated in near real-time as a function of changing parameters of the current classifier set and/or adding and removing *Voronoi classifiers*. The color map can be used be the designer as an intuitive feedback for the effect of changes to the classifier set.

We are currently developing automated methods of finding the optimal classifier set for any given obstacle/goal playing field. This includes a genetic algorithm-based method for automatically finding a feasible set of *Voronoi classifiers* that will stabilize *most* of the playing field. Research continues in the development of heuristics for instituting PAAR changes to the classifier set in the presence of finding previously unknown obstacles in the playing field. We anticipate that these methods will result in more efficient dynamic path planning capabilities then current state-of-the-art methods.

REFERENCES

Fortune, S., 1992: *Voronoi* diagrams and Delaunay triangulations in *Computing in Euclidian Geometry, World Scientific*, 193-233.

Pearl, J., 1985: Heuristics: Intelligent Search Strategies for Computer Problem Solving, Addison-Wesley.

Stenz, A., 1994: Optimal and Efficient Path Planning for Partially-Known Environments in *Proceedings IEEE International Conference on Robotics and Automation*.