2nd International Conference on Autonomous Robots and Agents
December 13-15, 2004 Palmerston North, New Zealand

Efficient Robotic Pursuit of a Moving Target in a Known Environment Using a Novel Convex Region Segmentation

Mohamed Marzouqi and Ray A. Jarvis
Intelligent Robotics Research Centre
Monash University, Melbourne, Australia
{mohamed.marzouqi, ray.jarvis}@eng.monash.edu.au

Abstract

A new method is introduced here that allows real time path planning in pursuit of a moving target in a cluttered known environment. The environment map is represented here by occupancy grids. Two algorithms co-operate to achieve this mission. The first is the well known shortest collision-free path planning, Distance Transform algorithm. The second one is our new map understanding algorithm that is introduced as the Map Segmentation algorithm. The Map Segmentation algorithm divides the free spaces into logically divided, convex regions. In each region, there are number of pre-specified border cells (gates). Each gate at region R is assigned to different region which represents the nearest cell in R to that region. Assuming the region the target stands in at any moment is known, a robot can navigate efficiently from region to another through the gates that are assigned to the current target's region. Test cases are presented.

Keywords: path planning, map segmentation, robotics pursuit.

1 Introduction

The problem of path planning to approach a specified destination in a known environment with obstacles has been satisfied with many algorithms that can give reasonable solutions. Some examples are: Distance Transform algorithm [7], Rapidly-exploring Random Trees [9], Potential Field Based Approach [8], and Visibility Graph [10]. However, most of the above mentioned algorithms do not consider the situation of a moving target, where the path needs to re-planned every time the target moves to a new location (which is time consuming for online missions). The Potential Field approach can consider moving targets but suffers from the problem of local minima entrapment.

For a successful pursuit of a moving target, an efficient robot that can act in real time is needed to cope with target movements and to achieve the assigned mission. This consideration is absent in most methods that solve such problems that need continuous path re-planning. Given a robot with a maximum speed, v, re-planning a path in the case of a moving target will reduce its speed by the time needed to recalculate a new path for every change in the target's position. The speed can be reduced significantly in high resolution environments. This can be critical to achieve a successful mission if the target has the same or faster speed than the robot's speed.

In this paper, a new approach is introduced that allows real time path planning by transforming the time consumed during navigation (online) into a memory table form before the pursuit mission starts (offline). This will allow the robot to operate at its maximum speed.

Two co-operative algorithms are involved. The first is the Distance Transform algorithm [8] which is a well known shortest collision-free path planning tool. The second is our new map understanding algorithm that is introduced here as the *Map Segmentation algorithm.* The environment map is a known two dimensional array represented by exact cell decomposition (non-polygonal).

The Map Segmentation algorithm divides free spaces in the map into natural (i.e. realistic or logically divided), convex regions. A convex region of a set of points has been given a lot of attention in computational geometry, the reason being that it can give a rough idea of the shape or extent of a data set; most geometric problem solutions are simpler and faster on convex objects. A convex region can be defined as a set of points such that each can see any other point in that region. There are many convex region algorithms [2,4] that are used in many applications. However, our algorithm differs from the others in its ability to deal with environment maps with obstacles, where each created convex region is a set of free space points (cells).

To achieve efficient robotics pursuit, the idea in our approach is to pre-specify border cells (gates) in each created region. Each gate at region R is assigned to different region which represents the nearest cell in R (shortcut) to that region. Assuming that the region where the target is at any moment is known (or can be predicted), the robot can navigate from one region to another through the gates that are assigned to the current target's region. Once the robot is at the same region where the target is, it can then navigate directly to the target as all segmented regions are convex and thus without internal obstacles.

In the next section, the two co-operative algorithms, the Distance Transform algorithm and the Map Segmentation algorithm will be described separately. In section 3 the combination of the two algorithms to solve the introduced problem will be explained in detail. An environment simulator was built to test the new method and a number of test cases are presented in section 4, followed by a discussion in section 5.

2 Algorithms Involved

2.1 Distance Transform Algorithm

The Distance Transform (DT) based path planning algorithm [7] finds the *optimal* collision-free path. Given a known stationary environment map and a destination point as inputs, the output is a two-dimensional array of exact cell decomposition. Each cell has a value that represents a collision-free least distance cost to reach the goal. The main idea of DT has been represented in [12] as numerical potential field technique which is also known as Wavefront algorithm.

Distance Transforms were first popularized as a binary image processing tool [11], especially for shape analysis purposes. The DT algorithm was used to propagate distance inwards from the borders of shapes, the process was constructed as a two pass traversal of a 2D binary image array, the first pass in forward raster (left to right and top to bottom) order, the second is reverse raster (right to left and bottom to top) order.

In [7], the algorithm was extended to develop DTs through the free space of a robot's environment from a goal cell (or goal cells). Distance propagation flows around the obstacles. Initially, the cell that represents the goal location is given a distance cost equal to '0'. All other free cells are given very high value (e.g. a million). The distance cost for each cell is derived based on propagating the distance costs of the surrounding cells. In the forward raster pass, each free space cell is assigned the value of one greater than the least value of the four neighbors previously visited on that pass, however, this assignment occur only if the new value is lower than the previous value at that cell. In the reverse raster pass the same process is done. Both passes are repeated until no further changes to cell assignment occur. Obstacle cells are skipped during these passes.

Each cell in the resulting array contains a number indicating the least number of steps to the goal's cell. Using the 2-D output array, the optimal path can be found for a given starting point by looking at its 8-neighbor cells and choosing a cell that has the lowest distance cost. This process is repeated until there is no neighbor cell that is lower than the current cell, i.e. the goal's cell is reached.

2.2 Map Segmentation Algorithm

The core idea of Map Segmentation algorithm is to divide a given known stationary environment map into different convex regions. Our method to find the convex regions depends on the visibility value of each unoccupied cell in the environment map (as it will be described later in this section). This leads to a 'natural' partitioning of the free spaces in the environment.

The input to the Map Segmentation algorithm is an environment map with cells which are free or forbidden (obstacles). The output of this function is called the Segmented Map, which is a two dimensional array that has the same size as the input environment map. Each free cell in the segmented map has a number (label) that indicates what region it belongs to. This number ranges from one to the total number of regions that are found.

Three examples are shown in the following figures (Fig. 1 to 6). For each pair of the six figures, one represents the environment map, while the other represents the segmented version of that map. In each segmented map, different convex regions are represented in different color. In all maps, obstacles are represented as black, since an obstacle can not be visually penetrated.

Looking at the first environment (Fig. 1), it is obvious how the environment should be divided into 16 simple regions. The segmented map of this environment in Fig. 2 shows how these rooms were recognized perfectly by our algorithm and each one of them is identified with different color. Moreover, the doorways connecting the rooms were treated as separated regions.

The structure of each of the other two environments is not as uniform as the first one. Therefore, using a pen to manually divide each of these environments maps into convex regions can give many different solutions as there is no clear way to divide them. The presented output segmented maps of each of these two environments can be described as one reasonable and accepted solution (at least for the purpose of this paper).

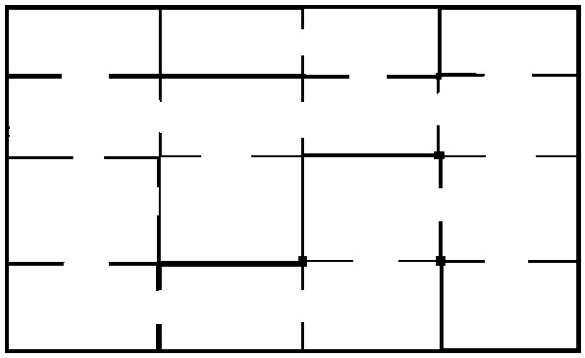

Figure 1: An example of an environment map.

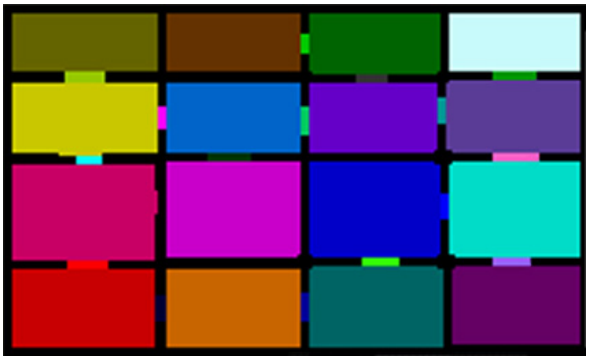

Figure 2: The segmented map of the environment in Fig 1. It is divided perfectly to regions including even those that represent door ways.

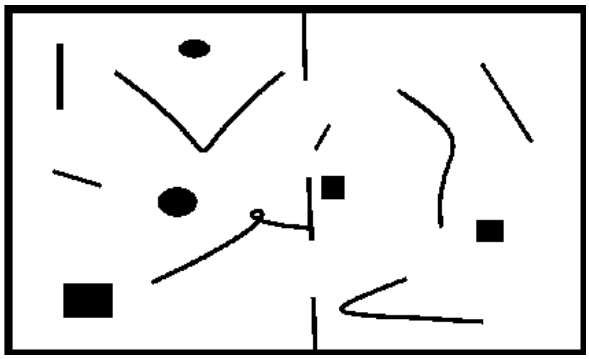

Figure 3: An example of unstructured environment.

Figure 4: The Segmented Map of the environment in Fig 3. The regions are presented with different colors for clarity.

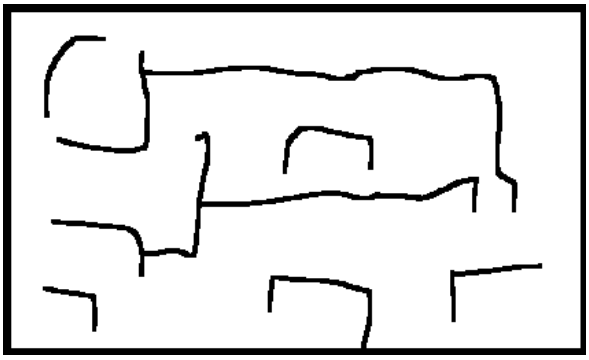

Figure 5: Another example of an environment map.

Figure 6: The Segmented Map of the map in Fig 5.

Starting with the results before giving the details of Map Segmentation should make the following description much easier to be understood.

Given a known stationary environment, the Map Segmentation approach goes into three stages to create a segmented map of that environment. In the *first stage* the Visibility Map Algorithm is applied to the environment map [1]. The Visibility Map Algorithm measures the visibility at each possible non-obstacle location and assigns the measured value to that location. The visibility value of a free cell is the number of the other free cells in the environment that can observe this cell. The visibility values are stored in the Visibility Map which is a two dimensional array of the same size as the environment map.

In the *second stage*, a segmented map array is created where all cells are initiated with a value less than '1' (e.g. '0'). To form the first region, the visibility map is used to find a cell with the lowest visibility value. This cell is given a value '1' in the segmented map and it is treated as a seed cell for region number one. The next step is to add the *neighbor* cells (that are not been added yet to any region) to the current region if and only if the neighbor cell can observe all cells that has already joined the region. This process can be accelerated by keeping a list of the growing region's border cells. In order to allow a neighbor cell to join the region then only its visibility to the border cells will need to be tested. If a cell can see all the border cells then it can see all the cells in the region (due to its convexity). Moreover, the testing process of a cell is terminated once it is shown that it does not observe at least one of the border cells. The joined cells are marked with the same value as the seed cell (region number).

To continue the process to form the second region, the visibility map is used again to find the next seed cell, given that a seed cell should not belong to any of the created regions. The selected seed cell is given value '2', and the same process is repeated to find all cells belonging to that region. The whole process is repeated until no more seed cells are found. At this stage, each free cell in the segmented map has a value between '1' and the number of found regions. It is

clear that each region is a convex as all cells of each region are visible to each other.

Choosing the seed cells with the lowest visibility values is our main idea that leads to create a set of natural regions (i.e. each region represent a separated physical space in the environment such as a room or a hallway). The reason is that a cell with the lowest visibility value will only observe cells that mostly belong to one separated physical space, this will lead to form regions with a high chance that each represents a different physical space. This becomes clear if a seed cell with maximum visibility is chosen, this will lead to form a region with cells which would most likely belong to more than one room or separated space in the environment.

The *third stage* is the refinement step where the region borders are smoothed. A number of border cells in some regions might be mostly surrounded with a neighbor region's cells rather than its own region's cells. To smooth the borders, such cells are identified and added to one of the neighbor regions, where it fits best, if and only if it can observe all the cells in that region.

An early stage that is not mentioned yet in the Map Segmentation process is to skip all free cells that are adjacent to obstacles (i.e. at the end of the segmentation process those cells will not belong to any region). The reason of doing this step is the nature of the occupancy grids representations where the obstacles might be in any shape and size (in comparison to polygonal representations), this usually leads to create a huge number of convex regions, even small in size as one cell (specially near the walls). Therefore, allowing this step will smooth the obstacles' walls and minimize the number of created regions significantly. In the figures shown above the free cells adjacent to the walls were shown in black as obstacles, while in Fig. 7 (another version of Fig. 6), they are represented in white for clarity.

The following is the pseudo code of the Map Segmentation algorithm:

- *Find the visibility value for each free cell*
- *do*
  - *Find free cell with minimum visibility (seed)*
  - *Give the found seed next label value i*
  - *do*
    - *for all neighbors of region i*
      - *if the neighbor cell can see all region i cells then give it value i*
  - *until no neighbor was added to the region*
- *until no new seed found*
- *Smoothing regions' borders*

3 Real Time Path Planning Method

In our approach both DT and Map Segmentation algorithms are combined to achieve real time path planning to reach a moving target. The process is divided into two phases: offline phase and online phase.

In the *offline phase* a lookup table is created for each region and stored to be used later in the online phase. To create the lookup table for region R_i, a 2-D environment map is created where all free cells belonging to region R_i are given a distance value of zero while keeping the other cells equal to a high value. The DT algorithm is applied to this map so that all cells belonging to other regions will get a value representing its transformed distance (shortest collision-free distance) to the border of R_i. The lookup table of R_i will be stored as a set of pairs in the form: [region ID, nearest cell to R_i (gate)]. The gate represents a cell position in the assigned region. This cell is the nearest cell to R_i in that region, i.e. it has the lowest transform distance value among other cells within the same region. If more than one cell were found in the same region with the same lowest distance value, the center location cell is chosen as a gate. To find the lookup tables for other regions the process is repeated in the same manner for each region.

Naturally, the gates are always located at the region's borders, therefore we can imagine each region as a castle with gates each labeled by other castles IDs as a shortcut junctions to them. Fig.8 shows the gates of each region as white dots. It is noticeable that some regions have less number of gates comparing to the total number of regions, the reason is that one gate maybe allocated for two or more different regions.

In the online phase, the path can be planned in real time using the lookup tables. To allow a robot to follow or capture a target that may be moving in the same or different speed, the robot can use the lookup table of the region where the target is standing in (assuming the target's current region is known all the time). Using that table the robot can identify and navigate to the cell gate in its region that is the nearest to the target region.

When the robot reaches the gate cell, it moves to the neighbor region cell and use the target's region lookup table again to find the gate in its new region that leads to the target's region. While the robot is navigating within a region to the appropriate gate, if the target moves to another region, the robot will stop and use the lookup table of the target's current region to find the gate that relates to it.

In the time both the robot and the target are in the same region, due to the convexity of all regions, the robot can navigate in straight line to the target, unless the target moves to different region. (However, if the robot is equipped with a vision system that can allocate the visible target position then it can still navigate in straight line to the target which maybe in a different region, while only using our approach when the target disappears behind an obstacle where a path planning strategy is needed.)

Figure 7: adjacent free cells to walls are in white.

Figure 8: The gates of each region are the white dots.

Using our approach, it is clear how a planner can change its path in a real time manner during the online phase by reading stored information. Comparing that to the original DT algorithm, the process of path planning should be repeated for every movement of the target which consumes a lot of time that increases in proportional to the environment map resolution. The next section will compare both our method and the original DT algorithm with respect to the distance optimality.

4 Experiment And Results

A simulator has been built to test the previously described method. The simulator inputs are an environment map and both the pursuit robot and the target locations. The environment is assumed to be known and stationary which means that its map is clearly specifies both obstacle and free spaces which do not change. The robot and the target are both assumed to be the size of one cell. The target's current region is assumed to be known at all times. For each of the figures below (Fig. 9 to 12), the robot's start location is presented by a yellow triangle, the target's start location is presented by red 'X' sign, and the obstacles are in black.

The Fig. 9 and 10 represent a comparison between two paths. The red path (dark gray in B&W printouts) is generated by our method, while the green path (light gray) is generated by the original DT algorithm. The target is kept stationary to focus on comparing both paths.

Both paths are nearly identical; this shows how our method is inheriting the nature and the distance optimality from the DT algorithm that it uses. However, this is not always the case where the arrow in Fig. 10 shows how our (red) path is getting away for a while from the optimal (green) path. The reason is that both the robot and the target exact locations are not taken into account when they are in different regions, only what matter is in which region they are standing. Thus, the chosen sequence of gates which generate the path may not always be along the actual optimal path. On the other hand, the method performs well in general, and this drawback is a payback for transforming the time consumed during online path planning into stored information.

In Fig. 11 and 12, the robot pursues a randomly moving target whose path is shown in green. In the first case, Fig. 11, the robot's speed is faster than the target and it has successfully captured it at the position indicated by the arrow. In Fig. 12 the robot is slower than the target, and it is trying to cope with the target's movements in order to reach it.

Using a laptop computer with Pentium 4 processor, the time required for the offline process (creating the convex regions and the lookup tables) ranges from 8 to 17 seconds, depending on the obstacle arrangement in the environment (the environment maps is 200x120 cells). The time consuming burden can be accepted, since all the calculations are done only once for a stationary environment before navigating. In addition, the memory consumed is acceptable as the storage complexity is $O(n^2)$, n: the number of map regions.

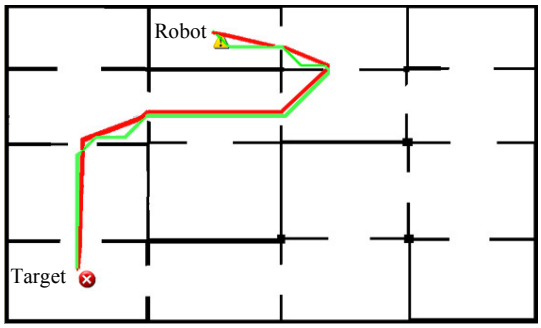

Figure 9: Our (red) path and the DT algorithm path are nearly identical.

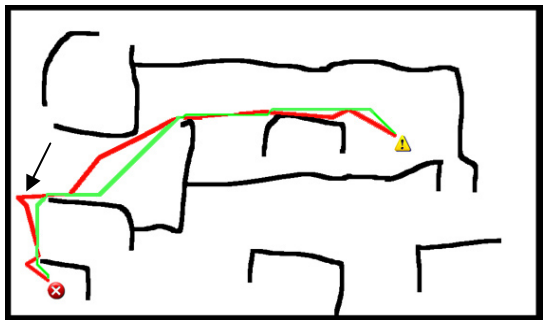

Figure 10: The sequence of gates are not always along the actual (green) optimal path, this leads to generate a path with less distance optimality.

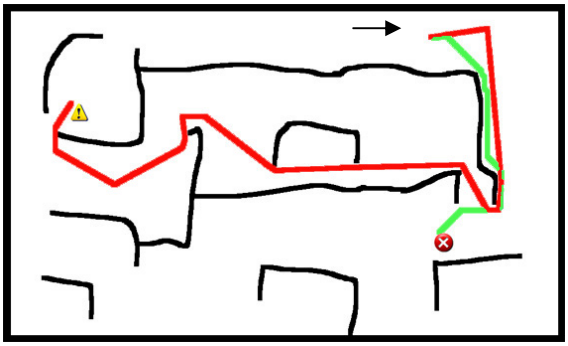

Figure 11: The robot (red path) pursuits a slow moving target (green path). The robot has successfully captured the target at the arrow position.

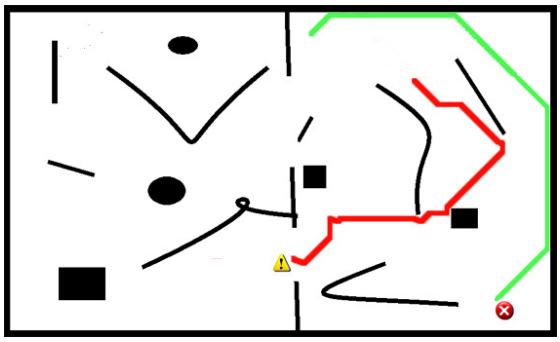

Figure 12: The robot is slower than the target, it is trying to cope with the target's movements to reach it.

5 Discussions and Future work

The results of the new method are promising and more research should be done to achieve efficiency and path optimality at the same time. This can be done by finding a way to take the locations of both the robot and the target into account during the online phase. Moreover, knowing the direction of the target and its destination region can also lead to more clever pursuit.

Knowing the target's exact location all the time is *not* a necessary assumption in this approach as long as the exact (or the predicted) region where the target stands is known. The only time that the exact location is assumed to be known is when both are in the same region, which is a practical assumption as the robot can be equipped with vision system to determine the target's exact location.

To extend this research, in case the target's current region is unknown and can not be predicted, the convexity of the regions can be an advantage to perform efficient environment search. A searching trip can be planned by visiting at least on cell of each region (as each region can be fully observed from any of its cells). This problem may be solved as the Traveling Salesman Problem (TSP) [3], which is a NP-Hard combinatorial optimization problem that aims to find the shortest path between a set of points. A powerful method to solve TSP problems is Genetic algorithms [5, 6], it shows the ability to give a good solution in a small amount of time.

6 Conclusion

We have shown a real time path planning method using convex regions Map Segmentation algorithm with the aid of the Distance Transform algorithm. This method is particularly efficient when using high resolution environment maps to navigate to a moving target; it can reduce significantly the time needed to re-plan a path. The Map Segmentation approach which was introduced here has shown one useful applications in this paper. The advantages of Map Segmentation can be extended beyond this research to be used in many other applications especially where map under-standing is required.

7 References

[1] Mohamed Marzouqi and Ray A. Jarvis, "Covert path planning for autonomous robot moving in known environment," *Proc. Australasian Conference on Robotics and Automation*, Brisbane, (2003).

[2] W. Eddy, "A new convex hull algorithm for planar sets," *ACM Trans. Math. Software* 3(4), pp. 398-403, (1977).

[3] Garey, M. R. and Johnson, D. S., "Computers and interactability: a guide to the theory of NP-completeness," Freeman San Francisco, (1979).

[4] R. Jarvis, "On the identification of the convex hull of a finite set of points in the plane," *Information Processing Letters*, 2:18-21, (1973).

[5] Grefenstette, J., Gopal, R., Rosmaita, R., and Gucht., D., "Genetic algorithms for the traveling salesman problem," In Proceedings of *the First International Conference on Genetic Algorithms*, pp. 160-168, (1985).

[6] Holland, J., "Adaptation in natural and artificial systems," *The University of Michigan Press*, Ann Arbor, (1975).

[7] R. A. Jarvis, "Collision-free path trajectory using distance transforms," Proc. *National Conference and Exhibition on Robotics* – Melbourne, (1984).

[8] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *International Journal of Robotics Research,* 5(1), (1986).

[9] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," TR 98-11, Computer Science Dept., Iowa State University, October (1998).

[10] N.J. Nilsson, "A mobile automation: an application of artificial techniques," Proceedings of *the 1st International Joint Conference on Artificial Intelligence*, pp.509-520, (1969).

[11] A. Rosenfeld and J.L. Pfaltz, "Sequential operations in digital image," Processing, J.A.C.M., Vol.13. No.4 pp.471-494, Oct (1966).

[12] J. Barraquand, B. Langlois, & J.C. Latombe. "Numerical potential field techniques for robot path planning," Report No. STAN-CS-89-1285, Stanford University, (1989).