Directed Exploration using a Modified Distance Transform

Trevor Taylor, Shlomo Geva, Wageeh W. Boles
Queensland University of Technology
T.Taylor@qut.edu.au

Abstract

Mobile robots operating in unknown environments need to build maps. To do so they must have an exploration algorithm to plan a path. This algorithm should guarantee that the whole of the environment, or at least some designated area, will be mapped. The path should also be optimal in some sense and not simply a "random walk" which is clearly inefficient. When multiple robots are involved, the algorithm also needs to take advantage of the fact that the robots can share the task. In this paper we discuss a modification to the well-known distance transform that satisfies these requirements.

1. Introduction

Exploration of unknown environments is a common task for robots. Apart from the issues of mapping and localization, the process also requires an exploration or path planning algorithm to ensure that the entire surrounding area is explored. As far as possible, the algorithm should produce an efficient path.

Several methods have been used for path planning in the past, but many of these assume that a map is already available. We use a Distance Transform [10] to repeatedly calculate paths towards unknown space.

The motivation for developing a modified distance transform is to set the preferred initial directions of exploration for multiple robots. This should allow multiple robots to explore with minimal overlap, thereby speeding up the exploration process.

1.1. Experimental Setup

In our experimental environment, the robot is a Yujin soccer robot, which is a cube 7cm on each side. It is remotely controlled by a PC via a wireless modem. A wireless colour camera on top of the robot is used to locate obstacles in the surrounding environment [13]. By performing a pirouette (spinning around on the spot) the robot builds a 360° view of its surroundings.

The resulting Radial Obstacle Profile (ROP) is similar to a sonar sweep, and it is suitable for producing a map based on an occupancy grid. Moravec and Elfes [6] first suggested this approach and it is widely used.

Several occupancy grid maps are shown in this paper. By convention, unknown space, i.e. areas not yet explored, is shown as grey in an occupancy grid, which corresponds to a probability of being unoccupied of 50%. Obstacles are shown as black (0% unoccupied means the cell is definitely occupied) and free space is white (100% unoccupied).

Due to limitations on the size of our physical environment, the examples in this paper are based on a simulation in order to provide a larger working area. The code uses OpenGL to create simulated camera images from the perspective of the robot and the same vision processing is applied as for the real robot. The principles are therefore still applicable to the real robot. See [14] for an example map from the real robot.

1.1. Related Work

Jarvis [4] first recognized that the Distance Transform (DT) could be used as an exploration algorithm. For this purpose, all of the unknown space cells in the map are marked as goals for the transform.

Distance Transforms have been used widely for path planning in exploration by single robots, e.g. [7, 15]. However, we wish to use multiple robots.

In most of the literature a square grid is used, although the tessellation does not need to be square. For example, a triangular grid is used in [8]. Regardless of the grid shape, it is still necessary to define the "distance" from one cell to another.

In some applications, the exact Euclidean distance is required. For instance, [11] uses a Weighted Distance Transform as one way to calculate Euclidean distances.

An exact Euclidean DT is not necessary for our purposes. The only requirement is that the DT must

Proceedings of the Digital Imaging Computing: Techniques and Applications (DICTA 2005)
0-7695-2467-2/05 $20.00 © 2005 IEEE

produce a surface that contains peaks where the obstacles are, but elsewhere it is monotonically decreasing towards unknown space. A simple downhill search can then be used to find a path from the robot to the nearest unexplored area.

Chong and Kleeman [2] developed an explore-local-first behaviour using a DT to force the robot to explore nearby areas before moving further away. They also created a local path validator to prevent collisions within a cell when a large cell size is used.

Their approach results in a pattern of exploration that is roughly a spiral. This is not desirable for multiple robots starting from the same location because they would be constantly running into each other or following the same paths.

Frontier-based exploration [16] is conceptually similar to the distance transform approach in that it locates the advancing frontier of unknown space, i.e. "the boundary between open space and uncharted territory". However, Yamauchi uses a depth-first search of the grid to find the shortest obstacle-free path to a frontier. In contrast, we use a modified DT to create a path to the nearest unexplored space.

Pei and Horng [9] developed a modified DT algorithm to take account of the shape and directional constraints involved in driving a car through an obstacle field. This is not necessary in our case because our robot is pseudo-holonomic, i.e. it can turn on the spot and thereby move in any direction (although not instantaneously nor whilst in motion). We limit our robot to motions made up solely of turns and straight forward moves, which we refer to as piece-wise linear.

Paths of complete coverage, where the robot visits every cell in the map, have been developed using distance transforms [17]. Applications such as vacuum cleaning [8] or lawn mowing require the robots to traverse every accessible cell in the map to complete their task. In contrast, our objective is to build a map. By using vision, the robot does not need to enter a cell to see into it. Furthermore, it can see into cells that are not accessible to the robot because it is too large.

Exploration using robotic swarms has been investigated by Tang and Jarvis [12]. Their approach requires knowledge of the locations of all the other robots at any given instant in time, although it does minimize the overlap. Our concept will allow the robots to be "let loose" and operate autonomously with minimal interaction. Of course, a global map must be produced eventually, but in the initial stages the robots will tend to spread out independently of each other regardless of whether or not they can communicate reliably at all times.

2. Description of the Algorithm

The basic process can be described as follows:

1. Perform a pirouette and update the Occupancy Grid map using the resulting Radial Obstacle Profile.
2. Calculate a modified Distance Transform to find the shortest path to the nearest unknown space. If no path can be found, then stop.
3. Travel along this new path, up to a specified maximum number of moves, updating the Occupancy Grid along the way.
4. Go to step 1.

The procedure for obtaining an ROP and creating a map has been described in our previous work [13]. Therefore step 1 above is not covered here.

Exploration continues while there is unknown space that is reachable. Our ultimate objective is to use multiple robots to speed up the exploration process, and this is why we have developed a modification to the Distance Transform which is used in step 2.

As the robots explore, they will send new map information to a central server that will combine it into a global map. The modified Distance Transform ensures that they start out in different directions. Subsequently, they tend to move outwards, unless their path is blocked, e.g. a walled in area.

Notice that in step 3 the number of moves along the path is limited. Most other exploration algorithms in the literature allow the robot to move all the way to the end of the newly calculated path. We have found this to be undesirable, for reasons explained below.

2.1. Distance Transform Path Planning

The basic concept of a Distance Transform is simple. Consider the map as a pool of water with the obstacles sticking out. Throw a pebble into the pool and take snapshots as the ripples spread out. The advancing wave front will wrap around obstacles. (Ignore any "reflections" that occur.) As time progresses the wave front gets further and further away from its origin, i.e. the distance increases. All points on the wave front at any given instant are equidistant from the origin.

For a square grid, there are both 4-connected and 8-connected versions of the DT. We have chosen the 8-connected version which requires two types of moves – move forward the distance of one grid cell for north-south and east-west moves, and the square root of two times this distance for diagonal moves.

Therefore, the use of the DT only requires the robot to rotate on the spot and move forward by two different

amounts. There is no requirement to move in real time or make steering decisions on the fly. This is a significant advantage of the approach because it is not dependent on the computational resources available. (Clearly, the fastest possible processor is desirable, but it is not essential to the task.)

To perform the DT, obstacles in the map are set to "infinity", and hence ignored during the calculations. Free space is filled in with a "big" value, which is necessary so that the transform will correctly update the cells. Of course, it is assumed that the total cost in a cell can never reach "big", but this is a fair assumption for practical purposes even when using a 32-bit integer as the distance. Unknown space, which is the goal, is filled in with zeros.

Our modification to the DT is to superimpose an influence map, or potential field, over the transform map during the calculation of the DT. The values in the influence map are simply added to those in the transform map when the cost in each cell is calculated.

The DT "wave" emanates from the zero cells (unknown space) and travels "uphill" until it encounters an obstacle and has nowhere else to go.

The net effect is that following a steepest descent path through the DT from the robot's current location (or in fact starting from any non-obstacle cell) will lead to the nearest goal, i.e. unknown space.

Fig. 1 shows an example with a path from the robot (the round object near the middle of the diagram) to unknown space in the top left.

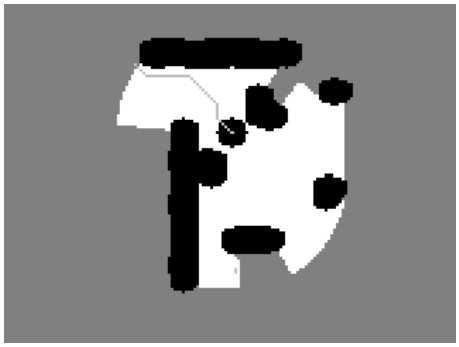


Figure 1. Path from a Distance Transform

While calculating the path it sometimes happens that there are two adjacent cells with the same distance value. In order to minimize the number of turns, the direction that is chosen is the one that the robot is currently facing if this is possible.

When creating the path, if a cell is reached which has no surrounding cells with a lower value, and this cell is not an unknown space (zero), then there are no more paths and exploration is complete. This is a very useful feature of the DT.

2.2. Collision Avoidance

In some of the Distance Transform literature, the robot is assumed to occupy a single cell. In practice, this is often not the case. If the robot is larger than one cell, then it can potentially collide with obstacles as it tries to drive past them. In fact, Zelinsky [17] has pointed out that the DT generates "too close paths" which tightly hug the walls.

The solution to this problem is very simple, and uses an old concept called Configuration Space, or C-Space for short. In C-Space all of the obstacle cells are enlarged by the size of the robot, thereby preventing it from coming too close to an obstacle. Basically this is a morphological dilation of the obstacles in the map by a structuring element that is the size of the robot.

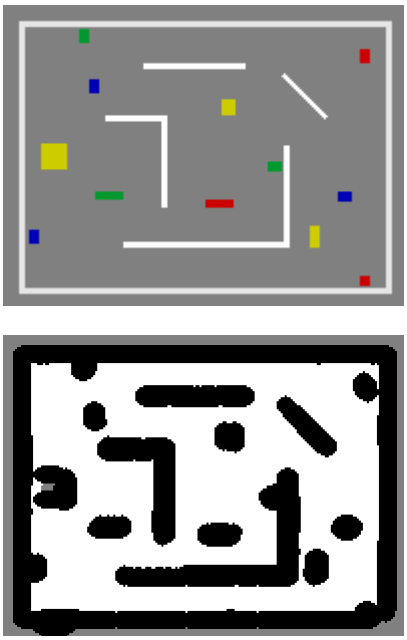


Figure 2. (a) World Model and (b) Configuration Space

An example is given in Fig. 2 of a world model (or ground truth) and the C-Space corresponding to a completed map. Notice that the obstacles in C-Space have been expanded.

Our robot is cubic in shape, but we use a round structuring element that has a radius half the diagonal width of the robot. (Using a square structuring element would not be correct because the robot's orientation can change. For instance, it could rotate to "squeeze" through small gaps, but this is not taken into account.) The result is a slightly conservative C-Space map.

Several methods exist for calculating a DT. We use one developed by Borgefors [1] which is commonly referred to as being similar to a convolution consisting

of a forward pass and a backward pass. For many environments this is all that is required, but it has been noted [5] that under certain circumstances multiple passes might be necessary. It depends on how many small "channels" there are in the image and can occur in mazes with a lot of twists and turns.

Our code is based on the Intel Open Computer Vision (OpenCV) library [3]. However, it should be noted that OpenCV (in version Beta 4) only performs one forward-backward pass and therefore the resulting DT might not be correct. In contrast, we iterate until there are no changes, which always results in one pass too many, but this is unavoidable. In addition, the modifications to the DT that we incorporate often require another pass or two to reach the final result.

The DT grid size need not be the same as the map grid size, but an integer multiple is best. Our map grid is 1cm square and the DT grid is 5cm square, which is about half the size of the robot.

A large DT grid size results in smoother paths and minimizes the number of turns involved. Because turns are a large source of localization errors, it is highly desirable to keep them to a minimum.

When adjusting the grid size, it will occasionally happen that the robot's cell will be marked as occupied by an obstacle. This is due to quantization effects and clearly is not correct. Therefore the robot's cell is always set to free space before the DT is run.

2.3. Initial Direction Control

To control the initial direction that the robot heads off in, an Influence Map (or potential field) is created and overlaid on the distance transform. By controlling the initial direction of exploration, we can force multiple robots to spread out.

Several different influence maps have been tested. For example, Fig. 3 shows a cardioid shape and a pie slice (a.k.a. pacman) that have a preference for a north-easterly direction. In this figure, darker areas correspond to higher values. Notice that the pie only has two values, whereas the cardioid has a range.

All the different influence maps that we have tried had the desired effect, but the pie slice is the simplest to calculate. The radius of the pie can be adjusted to suit the environment, and the centre of the slice can be pointed in the desired direction.

Figure 3. Examples of Influence Maps

As noted above, the values in the influence map are simply added to the "distance" in each cell during the calculation of the DT.

3. Discussion

One of the primary advantages of using the Distance Transform is that it is very easy to determine when the exploration is complete – there will be no reachable goals, i.e. unknown space. Furthermore, the DT cannot become trapped like some other algorithms.

However, a DT-based exploration algorithm can exhibit oscillations, especially when a directional field is applied. These oscillations result if the maximum allowed path length is less than the distance necessary to reach the nearest free space in two different directions. In this case, the DT alternates between two paths.

Fig. 4 shows a partial map where oscillations have started in the top right. The influence map was centered on the robot, rather than fixed at the centre of the map. Using a robot-centric influence map accentuates this problem, and so we have discarded this approach.

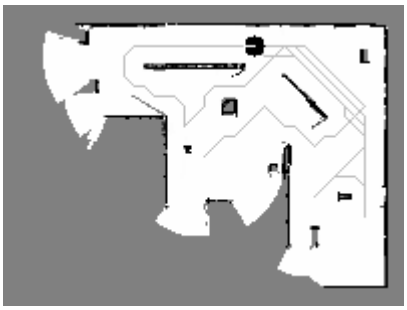


Figure 4. Oscillations in the DT Path

It is easy to detect oscillations and all that is required to break the cycle is, for instance, to halve the distance moved along one of the paths. On the next iteration the robot generally reaches unknown territory.

Another issue with the use of influence maps is that they tend to force the robot to take an indirect route to the nearest free space once it falls on the opposite side of the map. The robot then follows a path that moves alternately from one side of the map to the other as it winds its way around the central potential "hill".

To reduce this effect, the influence map can be made to decay over time. Its purpose is only to direct the robot in a particular direction initially, so it is not required in the longer term.

Proof that the directional control does work is apparent in Figs. 5 and 6 which show the final maps where the specified initial direction was south and

north-east respectively. The robot began in the centre of the map in both cases and it is obvious that it started out in the correct directions.

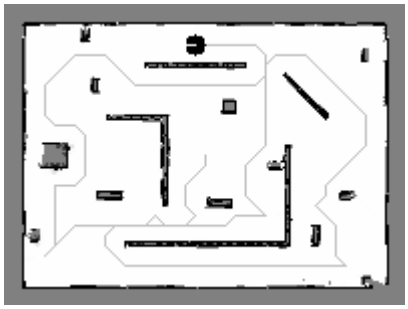
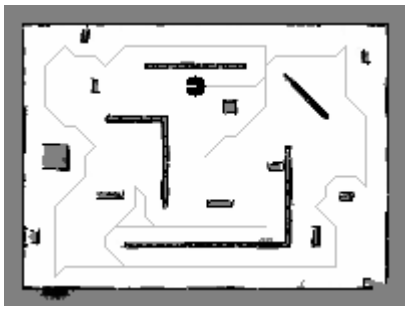


Figure 5. Exploration starting to the South



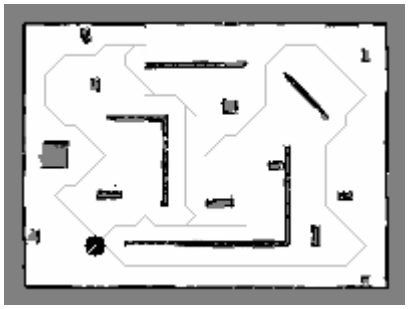


Figure 6. Exploration to the North-East with (a) 50 move maximum (b) 10 move maximum

Note that these diagrams are for a single robot. Due to the effects of the influence maps, the initial paths for multiple robots will not differ much from the paths for single robot because the influence maps have a significant effect on the DT path. Over time, the robots will be far enough away from the centre that their nearest unknown space will tend to be in their direction of exploration. Certainly, it will not be via the centre of the map unless their direction of exploration is completely blocked.

Notice in the two maps of Fig. 6 that the initial path is the same (due to the influence map) but as the influence map decays the robots move off in different directions due to the different maximum number of moves allowed at each stage.

Limiting the number of moves towards the goal is beneficial because the robot performs a pirouette before doing a new DT calculation. This tends to fill in the narrow spaces between obstacles and the walls. Also, in a multi-robot environment, following a very long path could prove to be a waste of time if another robot reaches the unexplored area first, so shorter paths with more frequent re-calculation should be beneficial.

3.1. Efficiency

Efficiency can be measured in a variety of ways, e.g. based on time, energy expended, distance traveled, etc. We have not approached this in a quantitative way, but the diagrams above show minimal overlap. In other words, the DT seems to minimize overall path length.

The distance transform will, in general, provide the shortest path from a specified location to a goal. Our algorithm uses the DT at each stage, and therefore we expect that the overall result should be a reasonably efficient. However, we have found in practice that the amount of duplicated effort is related to the maximum allowed length of the paths. It is also affected by the geometry of the environment and is therefore difficult to predict.

In particular, planning a path in an unknown environment will often lead to paths which will be seen to be sub-optimal in retrospect. An obvious example is a cul-de-sac. A robot entering one of these will have no idea that it is a dead end until it is too late, and it will have to back-track. On a larger scale, it is obvious that some duplication of effort is inevitable.

DTs tend to produce paths that closely follow obstacles and/or zig-zag. This effect can be reduced by using a larger cell size for the DT than the map cell size. However, this has the disadvantage that it might render some locations inaccessible because the robot is not sitting in the centre of a (large) grid square and therefore cannot align itself with a door or corridor.

4. Future Work

The robot currently misses some small areas when obstacles are too close together in C-Space. A DT could be done on the actual map using the map grid size, i.e. not the C-Space map. This would allow the small regions that have been missed to be located.

Because the robot uses vision, it does not have to enter a cell to determine if it is empty or not. An intelligent algorithm could be developed that would allow the robot to “look behind” obstacles using the information generated from the original map. Of course, collision avoidance would still require the use of the C-Space map in producing the paths.

Another area to be addressed is what to do when robots run into each other whilst exploring. Applying influence maps centred on the location of the rendez-vous would force them off in different directions again. However, it might happen that the obstacle geometry does not permit this, e.g. a long corridor.

Testing with multiple robots is still to be done.

5. Conclusion

This paper has outlined a novel modification to the Distance Transform to force robots to explore in a particular direction initially. This will be important to minimize the overlap when multiple robots are used. Our future work on collaborative mapping will use this algorithm to control the robots.

We have also investigated the use of a grid size that is more suited to the size of the robot and therefore differs from the grid size used for the map.

Variations in the length of the paths followed at each stage of the exploration have been shown to produce different paths, but essentially the same maps. Limiting the length of the paths is expected to be beneficial in a multi-robot situation because it will allow the robots to reassess their paths more frequently.

6. References

[1] G. Borgefors, "Distance Transformations in Digital Images," *Computer Vision, Graphics and Image Processing*, vol. 34, pp. 344-371, 1986.

[2] K.S. Chong and L. Kleeman, "Indoor Exploration Using a Sonar Sensor Array: A Dual Representation Strategy", *In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2, pp. 676-682, 1997.

[3] Intel Open Computer Vision (OpenCV) Library, Beta 4, available from http://sourceforge.net/projects/opencvlibrary, accessed 25th August, 2004.

[4] R.A. Jarvis, "Collision-free Trajectory Planning Using Distance Transforms", *In Proceedings of the National Conference and Exhibition on Robotics*, Melbourne, Australia, pp. 20-24, 1984.

[5] R.A. Jarvis, "Distance Transform Based Path Planning For Robot Navigation," In: Zheng YF (ed) *Recent Trends in Mobile Robots*, Vol. 11 of *Robotics and Automated Systems*, World Scientific, pp. 3-31, 1993.

[6] H.P. Moravec and A. Elfes, "High Resolution Maps from Wide Angle Sonar," *In Proceedings of the International Conference on Robotics Automation*, St. Louis, MI, USA, pp. 116-121, 1985.

[7] D. Murray and C. Jennings, "Stereo vision based mapping and navigation for mobile robots", In Proceedings of the IEEE International Conference on Robotics and Automation, vol. 2, Albuquerque, New Mexico, USA, pp. 1694-1699, 1997.

[8] J.S. Oh, Y.H. Choi, J.B. Park and Y.F. Zheng, "Complete Coverage Navigation of Cleaning Robots Using Triangular-Cell-Based Map", IEEE Transactions on Industrial Electronics, vol. 51, no. 3, pp. 718-726, 2004.

[9] S-C. Pei and J-H. Horng, "Finding the optimal driving path of a car using the modified constrained distance transformation," *IEEE Transactions on Robotics and Automation*, Vol. 14, pp. 663-670, 1998.

[10] A. Rosenfeld and J.L. Pfaltz, "Sequential operations in digital picture processing," *Journal of the Assoc. Computing Machinery*, vol. 13, no. 4, pp. 471-494, 1966.

[11] I-M. Sintorn and G. Borgefors, "Weighted distance transforms in rectangular grids," *In Proceedings of the 11th International Conference on Image Analysis and Processing*, pp. 322-326, 2001.

[12] K.W. Tang and R. Jarvis, "A Simple and Efficient Algorithm for Robotic Swarm Exploratory Tasks", Technical Report MECSE-11-2003, Department of Electrical and Computer Systems Engineering, Monash University, 2003.

[13] T. Taylor, S. Geva, and W.W. Boles, "Vision-based Pirouettes using the Radial Obstacle Profile," *In Proceedings of the IEEE Conference on Robotics, Automation and Mechatronics*, Singapore, pp. 147-152, 2004

[14] T. Taylor, S. Geva, and W.W. Boles, "Early Results in Vision-based Map Building," *Accepted for the 3rd International Symposium on Autonomous Minirobots for Research and Edutainment (AMiRE)*, Fukui, Japan, 2005.

[15] L.C. Wang, L.S. Yong, and M.H. Ang Jr., "Hybrid of global path planning and local navigation implemented on a mobile robot in indoor environment," *In Proceedings of the IEEE International Symposium on Intelligent Control*, pp. 821-826, 2002.

[16] B. Yamauchi, "A Frontier-Based Approach for Autonomous Exploration", *In Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*, Monterey, CA, USA, 1997.

[17] A. Zelinsky, "A mobile robot exploration algorithm," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 6, pp. 707-717, 1992.