An Exploration Algorithm for Autonomous Mobile Robots

Gavin Baker & Andrew Howard
Computer Vision and Machine Intelligence Lab
Department of Computer Science and Software Engineering
University of Melbourne
{gavinb|andrbh}@cs.mu.oz.au

Abstract

In this paper we present an exploration algorithm for mobile robots in unknown environments. During the task of exploration, a robot must have a way of navigating around the environment while making sensible decisions on where to explore next. Our approach uses an algorithm which will navigate the robot toward the largest unexplored region in the environment in order to maximise efficiency. We have tested this approach on a simulator, with good results.

1 Introduction

Traditional navigation and path planning techniques focus on how to arrive at a specified goal, according to some given criteria such as optimal time. But what if the goal is not to *reach* a particular point, but the *task* of exploring the entire environment? This paper describes an algorithm which enables autonomous mobile robots to efficiently explore their environment without an *a priori* goal or map. The technique presented assumes reasonable localisation is provided. Otherwise, it makes no assumptions about the environment (such as rectilinearity) and imposes no constraints other than that the region should be bounded.

The problem we are trying to solve is simply this: given an unknown environment, how do we explore as much of it as possible in an efficient manner? But in an unknown environment there is no single point which is a goal; *the unexplored space is the goal.* So how do we choose where to navigate in unknown space in order to explore? The algorithm presented here is designed to address this very problem.

The solution we present here is to generate a *terrain* which represents the environment, where the explored space is flat, and the unexplored space becomes mountainous. The height of the unexplored space is a function of its size. By continuously heading for the largest unexplored region, the environment is efficiently explored.

2 Related Work

There have been a number of approaches suggested for autonomous exploration [Yamamuchi *et al.*, 1998; Zelinsky, 1992; Kuipers and Byun, 1991; Elfes, 1989; Mataric, 1992], as well as a number of approaches for navigating in unstructured environments. The occupancy grids representation [Elfes, 1989] is commonly used (as in [Thrun *et al.*, 1998]) to map out an environment during exploration.

The distance transform [Byrne, 1986; Zelinsky, 1992] is a widely employed in robot navigation and path planning. The distance transform is represented by a uniform 8-connected grid[1] which covers the environment. The contents of each cell represents the distance from that cell to the goal. Starting from the goal cell, which has a distance of zero, the distance is propagated through free space (and not obstacles) to cover the entire grid. Thus by starting at any point on the map, the goal can be reached by moving to the next cell of a lower distance. This gives the shortest obstacle-free path to the goal, since it is following the path of steepest descent. The algorithm presented here is essentially a variation on the distance transform, but in reverse.

In [Zelinsky, 1992], Zelinsky describes an approach to exploration using the distance transform. By setting all the unexplored cells to goals, the robot will navigate to the nearest goal, and thus eventually explore the entire environment.

On the exploration problem, Yamamuchi asks in [Yamamuchi *et al.*, 1998] : "Given what you know about the world, where should you move to gain as much new information as possible?" His response to this rhetorical question is: "...move to the boundary between open space and uncharted territory." He then describes a technique called *frontier-based exploration*, an exploration algorithm which navigates to *frontiers*, the boundaries between explored and unexplored space.

The net effect of these two approaches is essentially the same; the robot will navigate to the nearest boundary of

[1] Although it can equally be implemented as a quad-tree.

unexplored space. In contrast, our approach looks at the environment as a whole and the unexplored regions in order to choose where to navigate.

3 Algorithm

The solution presented here to the exploration problem is a response to the question posed by Yamamuchi above. The solution is based on the premise that in order to explore the environment, we should *keep exploring the largest region of unexplored space.* Intuitively this behaviour mimics that of a human; when we walk into an unknown house, we do not enter one room and exhaustively examine every nook and cranny before moving to the next room. Instead, we look briefly in each room, one after another, and then come back later to fill in the detail.

This approach uses a grid-based representation, and uses three grids in our initial implementation. The first is the occupancy grid, which is generated from sensor data using Bayesian probabilistic techniques [Elfes, 1990]. From this data, the *terrain* is generated, using the algorithm described below. A third grid is generated as a distance transform to plot a course to the goal. (Once the goal is chosen however, other path planning algorithms could equally be employed).

The largest blob of unexplored space is chosen by generating a terrain (see Figure 1b), and then setting the goal as the highest peak in the terrain. The terrain consists of *water* and *land*; water represents the explored space, while land represents unexplored space. As the unexplored space (land) gets further away from explored space (water) it grows in height, like a mountainous island rising out of the sea[2]. As a result of the way the terrain is generated, the height of the mountain is effectively a function of the size of the area it represents. Thus heading for the highest peak will result in exploring the largest region of unexplored space.

Once the goal is chosen, the robot can use an existing path-planning technique such as the distance transform [Jarvis, 1985] in order to navigate toward the goal. We make the optimistic assumption (as stated in [Zelinsky, 1992]) that the unexplored areas are free, in order to optimise the path. However as the robot further explores the environment, obstacles will be revealed and the path will need to be regenerated.

The robot will keep trying to reach the goal until either (a) it reaches the goal, or (b) it proves it cannot reach the goal. At this point, it will choose another goal, which is simply the next highest peak in the terrain. In order to prove it cannot reach a goal, it must explore enough of the environment such that the goal is surrounded by obstacles. When the terrain is generated, it will not propagate past obstacles, and thus any closed region (convex *or* concave) will result in a *void* (an unreachable area bordered by obstacle cells). A goal is therefore deemed unreachable if it lies within a void.

The net result of this technique is an environment which is completely explored, and the terrain generated results in a map containing the obstacles.

3.1 Terrain Generation Algorithm

The algorithm is similar in concept to the distance transform [Jarvis, 1985], but in reverse. Here we begin at the robot, and propagate the numbers outward. The difference is that the distance (or height if you think of it in 3D as a terrain) is held at zero (or sea level) through all the explored regions. At the boundary of explored and unexplored space, the terrain begins to rise. The larger the unexplored region is, the higher it will grow. As the terrain is being generated, the highest point, called the *peak*, is tracked. This peak must represent the largest unexplored region in the environment, and in terms of the high-level behaviour, represents the *next* goal for exploration.

A queue-based algorithm is presented below:

```
// Initialise the terrain
for y := 0 to height
    for x := 0 to width
        terrain[x][y] := unknown;

// Start with the robot's position
queue.push( Point( rx, ry, 0 ) );

// Generate the terrain
while not queue.empty() do
begin
  p := queue.pop();

  h := p.height + 1;

  for each direction do

      if terrain[x][y] == unknown or
         terrain[x][y] > h then

          switch status[x][y] do

              case occupied:
              case avoid:
                terrain[x][y] := void;

              case free:
                terrain[x][y] := sea;
                queue.push( Point( x, y, 0 ) );

              case unknown:
                terrain[x][y] := h;
                queue.push( Point( x, y, h ) );

                if d > peak.height then
```

[2]This of course, begs the question: does the robot need to be an amphibious vehicle? Fortunately, the robot does not have to get wet for correct operation of the algorithm.

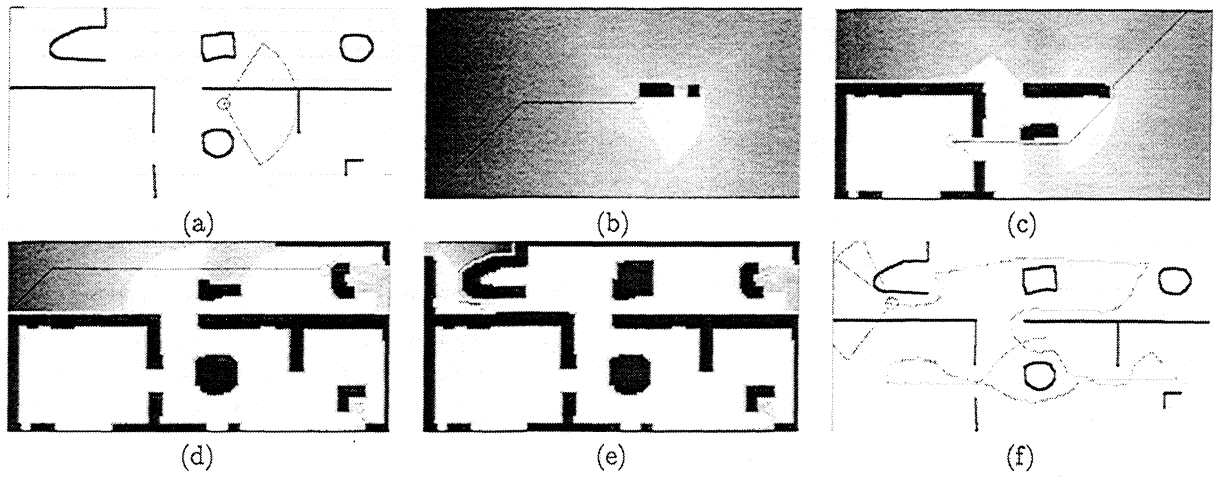


Figure 1: The simulator in action. Image (a) is the starting state. Images (b) – (e) show the terrain as the environment is progressively mapped. Image (f) shows the path taken by the robot during exploration.

```
        begin
            peak.x      := x;
            peak.y      := y;
            peak.height := h;
        end
end
```

3.2 High Level Algorithm

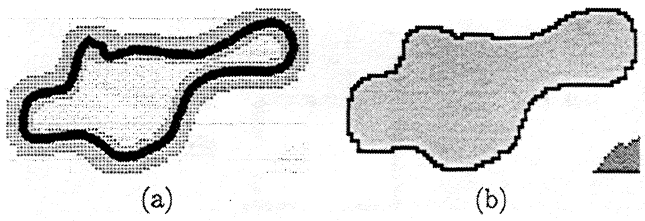
The algorithm described above describes how the terrain is generated. This section deals with the higher-level aspect of the exploration behaviour. The high-level algorithm is a loop, as follows:

1. **Update position.** Maintain an accurate position of the robot's location within the environment.
2. **Update occupancy grid.** Using all available sensor data, update the map with obstacles and free space.
3. **Generate the terrain.** Using the above algorithm, generate the terrain.
4. **Evaluate the goal.** If the goal ends up in a void (an unreachable part of the map) choose a new goal. If the occupancy status of the goal cell becomes free then the goal has been reached, so choose a new goal. The new goal is obtained by choosing the highest peak which was generated in the terrain.
5. **Generate exploration path.** Plot a path from the robot to the goal. This can easily be achieved using standard techniques such as the distance transform.
6. **Control the motors.** Based on the chosen path, send the appropriate commands to the servo motors to guide the robot along.

3.3 Properties

The approach presented here has a number of desirable properties. For example:

- **Inexpensive.** The processing overhead is low, and the memory requirements are low, making it eminently suitable for embedded and autonomous robots.
- **Efficient.** The terrain can be regenerated on an as-required basis (such as finding a new goal); it does not necessarily require constant or frequent updating like an occupancy grid does. It is efficient in that a robot can quickly explore a complex environment, since it is continuously exploring the largest unexplored region.
- **Robust.** The algorithm has been simulated in a wide variety of environments, where it performs equally well. It makes no assumptions about the environment itself (such as rectilinearity), and requires no simplifications of the environment or the system to operate effectively.
- **Integrated.** The algorithm integrates extremely well with existing established techniques, such as occupancy grids and the distance transform, and can be used to supplement other behaviours.
- **Global.** Unlike most other approaches which operate on a local level, this approach works at the global level. When choosing where to navigate, the entire environment is considered, instead of the nearest border between explored and unexplored space.
- **Flexible.** The goal selection and evaluation algorithm can be varied separately from the terrain generation, and thus can potentially be optimised



(a) (b)

Figure 2: Standard mapping results in obstacles which still contain unexplored cells, as in (a). Terrain generation fills this in as a void (b).

for different scenarios, and even for multi-agent systems. This aspect is currently being investigated.

- **Dynamic & Adaptive.** As the robot explores the environment, it uncovers obstacles which it then routes around, adapting its optimal path to the environment as it goes.
- **Oscillation-free.** Unlike many other robotic behaviours, the algorithm does not suffer from oscillations. Once it chooses a goal, it persists in reaching it until it either succeeds, or it proves that it cannot reach it.
- **Map building.** Once the entire environment has been explored, the result is a map which completely describes the area. Moreover, the map resulting from the terrain will represent enclosed obstacles as voids, rather than simply a border of obstacles containing unexplored cells (see Figure 2). This can be used for subsequent navigation tasks.

4 Experiments

The technique described above has been implemented in a robot simulator called ArSim [Howard, 2000]. The robot in the simulator has a laser range finder to detect obstacles, and uses odometry and landmarks feeding into a Kalman filter for localisation. ArSim introduces noise and random errors into the simulation in order to be as realistic as possible. It was tested using a variety of environments; cluttered office-like environments, mazes, corridors, open spaces and irregular shaped obstacles (see Figure 1 for an example). The technique performed well in all cases.

An image of a terrain when the robot first starts exploring is shown in Figure 1a. The light area surrounding the robot is the water, or explored space. A line from the robot leads to the goal in the top right corner, representing the currently planned path. The dark blobs are obstacles, while the gradient shows the mountains of unexplored space rising away from the water.

As the robot further explores the environment (Figure 1 b–c) the terrain turns into a map of the environment and the obstacles within. The path taken by the robot during a sample run is shown in Figure 1d.

4.1 Aim

The aims of the experiments carried out were to:

1. Measure the efficiency of the exploration algorithm
2. Compare the performance to other established techniques
3. Capture the terrain for visualisation

We intend to show that our exploration algorithm will explore a greater proportion of the environment over the same distance, as compared to the other techniques.

4.2 Method

The measurements were all performed on the same simulator, using the same environment, initial conditions and sensor properties. The algorithm was the only aspect varied. We tested our algorithm and compared it with the distance transform.

The measurements taken were:

- Percentage of environment explored versus distance traveled.
- Snapshots of the generated terrain, taken at regular intervals over time.

4.3 Results

The performance results from the experiments are shown in Figure 3. The graph shows three distinct regions, marked by the vertical lines. In the first stage, both algorithms perform fairly closely. In the second stage, our approach makes a significant gain. Finally in the third stage the distance transform finishes slightly ahead while the terrain approach catches up filling in the smaller areas it missed along the way. Overall, our algorithm explored 75% of the environment after 44 metres compared to 50 metres for the distance transform, translating to a 12% improvement. However, the distance transform finished first.

Some time-lapse snapshots of the terrain data in 3D are shown in Figure 4. This clearly shows the unexplored mountains being flattened as the robot explores the environment. Screen shots of the simulator in action are shown in Figure 1.

4.4 Discussion

Somewhat surprisingly, irrespective of the algorithm employed, when a robot starts out in an unknown environment, almost anywhere it moves will uncover new areas. Thus almost any non-trivial motion around an unexplored area will result in roughly linear performance, when examining percentage explored versus distance traveled. This explains the very close results in the first stage of the graph.

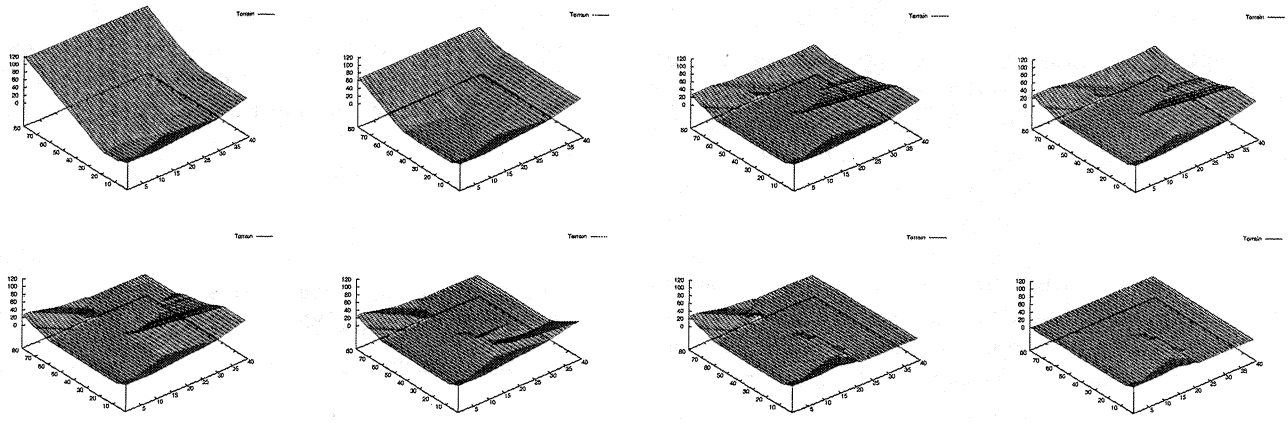


Figure 4: Plots of the terrain generated over time.

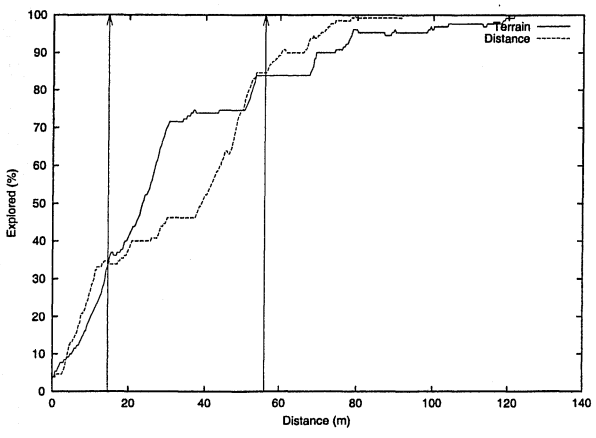


Figure 3: Exploration performance – the distance traveled versus the percentage of the environment explored.

The nature of the distance transform is to perform a detailed exploration of a region, since even a single unexplored cell nearby will cause the robot to navigate to that point before continuing on. Our approach however, effectively works with large regions rather than cells so in the big picture it does not try to fill in all the detail first. This explains why there was a large performance gain in the second stage, as our terrain algorithm was forging ahead exploring new regions while the distance transform was busy filling in the detail.

The final stage shows the distance transform finishing slightly ahead of our algorithm. This is because using our algorithm, the robot had to go back and explore the smaller regions it missed the first time through. Even still, it managed to explore three-quarters of the environment 12% faster than the distance transform.

Since this algorithm tends to explore larger regions first, it may skip smaller regions nearby in favour of larger ones further away, thus missing some of the "detail" in the early stages. This is of course entirely dependent on the environment, and is usually (but not always) desirable. It will eventually get back to exploring these smaller regions, but by adapting the goal selection algorithm the technique could potentially be adapted to cover this situation as well.

The simulation failed on several occasions, in that the robot got stuck too close to an obstacle and could not plot a collision-free path out of harm's way. However this failure was a result of two things combined: first, the rather simplistic path planner in the test implementation had a tendency to plot a route close to obstacles, and second the motor control had a habit of cutting corners. Since the path-planner would never plot a path through an obstacle, once the robot within the configuration space of the obstacle it would get stuck. This failure is not related to the algorithm presented in this paper; it is merely an implementation issue which will be addressed by adding more sophisticated control routines in the next version. No failures were encountered with respect to the exploration algorithm itself.

The design of the algorithm effectively requires that it operate in a bounded environment. However this is not really an issue, since standard windowing techniques can be used to employ this technique in unbounded situations. The current implementation assumes the environment is static, although it could be extended to cope with moving obstacles, such as other robots or humans.

This approach makes no attempt to address the localisation issue; it instead relies on existing techniques (such as odometry and landmarks) to provide it with accurate positional information. This reliance is a potential weakness, however localisation was not the problem we set out to solve.

5 Conclusion & Further Work

This algorithm builds on the strengths of the distance transform, however it introduces a novel approach which is worthy of further study. Its basic performance compares favourably with the distance transform, and has some additional desirable properties. Intuitively, were a human wish to explore an environment, we would surely wander around the environment, seeking new and unseen rooms to explore rather than stay in the one room until we had observed every single detail. This behaviour is represented in our approach, which works as a process of refinement, and we hope to further improve its performance and extend it to multi-agent systems.

The next step is to prove the algorithm is viable on a real system. Since the simulator replicates the inherent noise and inaccuracies of the real world, the results should provide a good indication of how it will perform on a real robot. In addition, the algorithm should be exercised in different settings and in conjunction with other approaches. This algorithm is amenable to different path-planning strategies, environment mappings, representations, and architectures.

The major aspect we intend to explore is the extension of the exploration to co-operative map-building with multi-agent systems. The nature of this algorithm is to build a global picture of the environment and then make an intelligent choice for a goal. Furthermore, the goal selection algorithm is independent of the terrain generation. We believe these two factors combined will allow us to readily extend this approach to support multiple robots by adapting the goal selection algorithm.

References

[Arkin and MacKenzie, 1994] Ronald Arkin and Douglas MacKenzie. Temporal coordination of perceptual algorithms for mobile robot navigation. *IEEE Transactions on Robotics and Automation*, 10(3):276–286, June 1994.

[Arkin and Murphy, 1990] Ronald Arkin and Robin Murphy. Autonomous navigation in a manufacturing environment. *IEEE Transactions on Robotics and Automation*, 1990.

[Borenstein, 1991] Johann Borenstein. The vector field histogram — fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 1991.

[Byrne, 1986] RA Jarvis & JC Byrne. Robot navigation: Touching, seeing and knowing. In *Proceedings 1st Australian Conf Artificial Intell*, Nov 1986.

[Elfes, 1989] Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, June 1989.

[Elfes, 1990] Alberto Elfes. Occupancy grids: A stochastic spatial representation for active robot perception. In *Proceedings of the Sixth Conference on Uncertainty in AI*. Morgan Kauffman Publishers, July 1990.

[Howard, 2000] Andrew Howard. Arsim programmer's guide v1.3. Technical report, University of Melbourne, 2000.

[Jarvis, 1985] R Jarvis. Collision-free trajectory planning using distance transforms. *Mechanical Engineering Transactions*, 1985.

[Kuipers and Byun, 1991] Benjamin Kuipers and Yung-Tai Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Journal of Robotics and Autonomous Systems*, 1991.

[Lozano-Pérez, 1981] Tomás Lozano-Pérez. Automatic planning of manipulator transfer movements. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-11(10), October 1981.

[Lozano-Pérez, 1983] Tomás Lozano-Pérez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, 32(2), February 1983.

[Mataric, 1992] Maja Mataric. Integration of representation into goal-driven behaviour-based robots. *IEEE Transactions on Robotics and Automation*, 8(3):304–312, June 1992.

[Thrun *et al.*, 1998] Sebastian Thrun, Arno Bücken, Wolfram Burgard, Dieter Fox, Thorsten Fröhlinghaus, Daniel Hennig, Thomas Hofmann, Michael Krell, and Timo Schmidt. Map leaning and high-speed navigation in rhino. In David Kortenkamp, R Peter Bonasso, and Robin Murphy, editors, *Artificial Intelligence and Mobile Robots*, pages 21–50. MIT Press, 1998.

[Yamamuchi *et al.*, 1998] Brian Yamamuchi, Alan Schultz, and William Adams. Mobile robot exploration and map-building with continuous localisation. In *Proceedings of the 1998 IEEE Conference on Robotics and Automation*, pages 3715–3720, May 1998.

[Zelinsky, 1992] Alexander Zelinsky. A mobile robot exploration algorithm. *IEEE Transactions on Robotics and Automation*, 8(6):707–717, 1992.