A Parallel Path Planning Algorithm for Mobile Robots

Chang Shu and Hilary Buxton

Department of Computer Science

Queen Mary and Westfield College

University of London, E1 4NS

This paper presents a path planning algorithm for mobile robots. We introduce a parallel search approach which is based on a regular grid representation of the map. The search is formulated as a cellular automaton by which local inter-cell communication rules are defined. The algorithm is made adaptive by utilising a multiresolution representation of the map. It is implemented on AMT DAP 510, which is a SIMD machine.

I. Introduction

A primary task in robotics is to plan collision-free path in cluttered environments. The problem can be stated as: Given the initial and desired final configurations of an object in two- or three-dimensional space, and given the description of the obstacles in the space, determine whether there is a continuous motion of the object from one configuration to the other, and find such a motion if one exists.

Previous methods for path planning can be divided into two categories by the way they model the environment. Methods in the first category are based on geometric reasoning in which a precise geometric model of the environment and of the moving object is required. These kind of models can be obtained from CAD systems. In this paradigm, path planning is an once-only off-line process. A great deal of research has been devoted to this situation, e.g. [1] - [5], [8], [9], [11]. While these methods are basically designed for manipulator path planning which requires precise and repeatable paths, they are not suitable for mobile robots where the requirements are different. A mobile robot is often equipped with some sensors such as a TV camera associated with a computer vision system. Its knowledge of the environment is generally incomplete and imprecise. It has to plan paths for the area it has seen. Therefore, path planning, in this case, is an on-line real-time process.

Methods in the second category are designed for mobile robot path planning. Thorpe[10], Kambhampati and Davis' methods[6] belong to this group. In [10] Thorpe uses a regular grid search method and in [6] Kambhampati and Davis proposed a quadtree representation of the environment. They all use a computer vision system or image data as sources of the world model.

Note that all of the above mentioned methods ultimately reduce the path finding problem to a graph search. For example, in the configuration space method[9], the search space is the visibility graph. Usually, the A^* or Dijkstra's algorithm is used for searching the graph. The main drawback of these methods is they have a big overhead for the construction of the vertex graph. The best algorithm so far constructs the graph in time $O(n^2)$ where n is the number of vertices of the obstacles[15]. Another disadvantage is that when working in a complex environment the search space becomes so huge that it is not applicable to a real-time mobile robot.

The potential field method[7], which also belongs to the second category, uses artificial potential field applied to the obstacles and goal positions and uses the resulting field to influence the path of the robot which is subject to this potential.

Only very few papers in the literature discuss parallel algorithms for the problem. Witkowski[17] suggested a method which is based on a cellular representation of the environment and uses a parallel searching algorithm. More recently, Steels[12] suggested a similar approach which states the problem in a more general way by viewing path planning as a dynamical process[13].

The work described here is an extension to that of Steels. We propose an adaptive algorithm which plans paths on different levels of resolution and implement this algorithm in parallel on a SIMD machine, the AMT DAP510, to give fast execution time. Section II introduces a cellular representation of the environment. Section III describes a parallel algorithm for path planning. In section IV, we give an adaptive path planning method which uses multiresolution representation of the map. Finally in section V, we discuss the implementation of the algorithm and give some experimental results.

In this paper, we focus on two-dimensional path planning and assume a robot with a symmetric cross-section. Work is going on to generalising the approach to the asymmetric case.

II. Space Representation

The algorithm presented here tries to formulate the path planning system as a cellular automaton. Given the binary image representing the map of the environment as well as the robot itself, we divide the map into regular square grids. Each grid represents either a free space area or an obstacle area. Therefore, the map is transformed into a binary array. Instead of converting the array into a graph, as in [10], we define strengths on each grid and then spread the strengths in the whole area according to some predefined rules. In the next two subsections, we shall first give a definition of cellular automata and then define strengths on the grids.

A. Cellular Automata

Cellular Automata are mathematical models of physical systems in which space and time are discrete. They are generally used as a tool for investigation of self-organisation and nonlinear dynamical systems[14][16]. In its simplest form a cellular automaton consists of a line of sites, with each site initially having a value. It evolves in discrete time steps. At each time step, the value of each site is updated according to a definite rule. The new value on each site is specified in terms of the values of its neighbours. One simple example of a cellular automaton rule is

$$a_i^{(t+1)} = a_{i-1}^{(t)} + a_{i+1}^{(t)} \bmod 2$$

where $a_i^{(t)}$ is the value of site i at time step t.

In this paper, we use a two-dimensional cellular automaton which involves rules based on values of the four neighbourhoods of each site. An example of a two-dimensional cellular automaton rule is

$$a_{ij}^{(t+1)} = (a_{i-1j}^{(t)} + a_{ij-1}^{(t)} + a_{ij+1}^{(t)} + a_{i+1j}^{(t)}) \bmod 2.$$

Since the rule is applied to each site simultaneously at each time step, it makes a good model for parallel computers.

B. Strength on the Grids

Returning to the environment map, we create a strength space upon the map. Each grid in the map has four strengths in the strength space. They are the strengths to move to the south, north, west, and east(Figure 1).

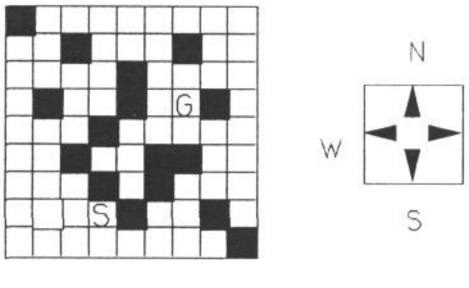

Figure 1. The binary map and strengths on a grid

We can use five bits for each grid to denote its state. The first four bits represent the strengths in the four directions and the fifth indicates whether the grid is in a free area or in a blocked area. For example, if a_{ij} = 10101, it means that the grid a_{ij} has strength 1 in south and west , and strength 0 in north and east, and it is a free point. We shall use a_{ij} (N), a_{ij} (S), a_{ij} (E), and a_{ij} (W) to denote the first four bits of a_{ij}, and use a_{ij} (Free) to denote the fifth bit.

That a grid has a strength in the north means if the robot were at the location of the grid, it may move one step to the north, and similarly for the strengths in the south, west and east. We define the fifth strength as invariant, i.e.

$$a_{ij}^{(t+1)}(\text{Free}) = \quad a_{ij}^{(t)} (\text{Free}), \text{ for all t.}$$

III. Spreading the Strengths

In this section, we describe how to search in parallel in the strength space to find a path. Suppose we have a robot which has a symmetric cross-section with a radius r in terms of the number of grids. In order to make the search easy, we first shrink the robot to a point by growing the obstacles by r grids. The general strategy is that we first assign strengths to the immediate four neighbourhood grids of the goal location, and then diffuse these strengths in parallel in the four directions according to some rules. Initially, the four neighbourhood grids are each assigned a strength which points to the goal grid, as shown in Figure 2(a). The diffusion rules are specified in the following formula:

$$a_{ij}^{(t+1)}(N) = (a_{ij}^{(t)}(N) \vee a_{ij-1}^{(t)}(N) \vee a_{ij-1}^{(t)}(E) \vee a_{ij-1}^{(t)}(W)) \wedge a_{ij}^{(t)}(\text{Free}) \qquad (1)$$

$$a_{ij}^{(t+1)}(S) = (a_{ij}^{(t)}(S) \vee a_{ij+1}^{(t)}(S) \vee a_{ij+1}^{(t)}(E) \vee a_{ij+1}^{(t)}(W)) \wedge a_{ij}^{(t)}(\text{Free}) \qquad (2)$$

$$a_{ij}^{(t+1)}(E) = (a_{ij}^{(t)}(E) \vee a_{i-1j}^{(t)}(E) \vee a_{i-1j}^{(t)}(S) \vee a_{i-1j}^{(t)}(N)) \wedge a_{ij}^{(t)}(\text{Free}) \qquad (3)$$

$$a_{ij}^{(t+1)}(W) = (a_{ij}^{(t)}(W) \vee a_{i+1j}^{(t)}(W) \vee a_{i+1j}^{(t)}(S) \vee a_{i+1j}^{(t)}(N)) \wedge a_{ij}^{(t)}(\text{Free}) \qquad (4)$$

Rule (1) - (4) specify how to compute the strengths on an arbitrary grid in one step of the iteration. Rule(1) indicates that the north strength value on the next step depends on its present value as well as the east and west strength value of its north neighbourhood('$\vee$' and '$\wedge$' represent logical 'or' and 'and' respectively). It is necessary to consider the east and west strength when the north strength is diffused because otherwise the flow of strengths would not be able to avoid the obstacles. Note that grids in a blocked area cannot acquire any strength in any direction. Thus rule(1) implies that for any free grid a_{ij} at a certain step of iteration, it acquires a north strength if (a) its present north strength value is 1, or (b) its north neighbourhood grid has a north strength, or (c) its north neighbourhood grid has an east or a west strength. The implications of rule (2) through (4) should be apparent from the above explanation.

The diffusion process terminates when the starting grid acquires a strength. Then the robot makes an one-step move

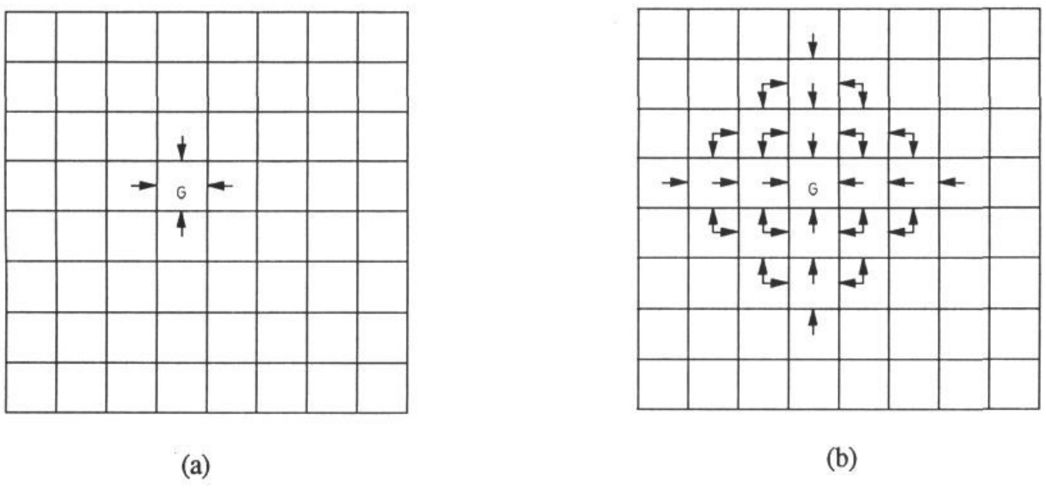

Figure 2. (a) The initial strength space (b) The configuration of the strength space after two steps of diffusion

according to the strength value on its location grid. If the grid has strengths in more than one direction, a random choice is made. Figure2(b) shows the configuration of the strength space after two steps of diffusion. To make the robot move further steps towards the goal, the strength space is cleared and repeats the same process as described above. Figure 3 shows a path found by the algorithm.

IV. Adaptive Path Planning

When discretising the space into regular cells, we always face the problem of resolution, i.e. how big a cell should be. The increase of resolution can be very costly in terms of computation. However, using a coarse resolution may result in the report of a longer path, or even worse, may result in failure report a path when one actually exists. Figure 4 illustrates this situation.

Our adaptive algorithm makes use of a multiresolution representation of the map. First, we apply the diffusion algorithm to a relatively coarse resolution map. If a path is found, the

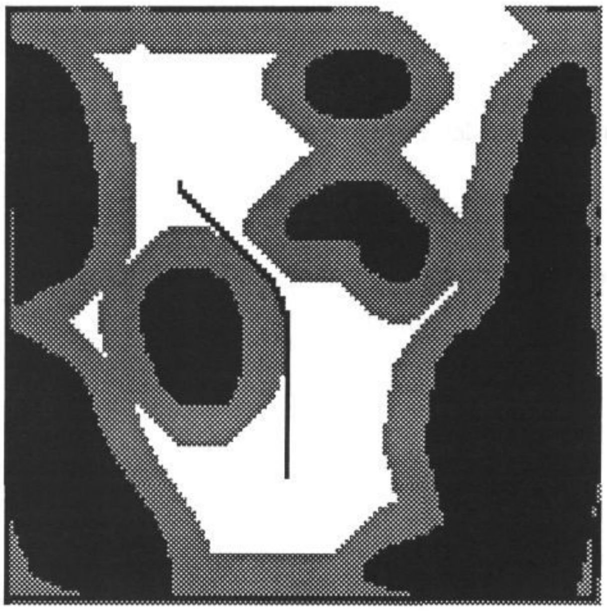

Figure 3. A path found by the algorithm

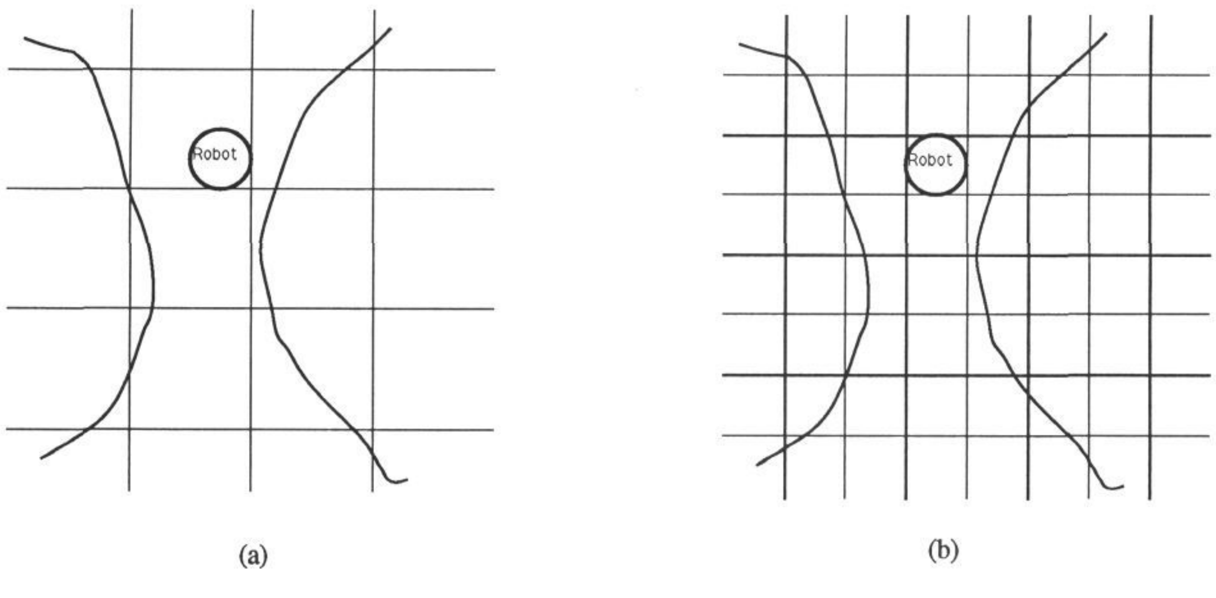

Figure 4. (a) A blocked path in a coarse resolution (b) An open path in a finer resolution

algorithm terminates and reports the path. If not, there are two possibilities; either there is actually no path, or there is a path in a finer resolution. Instead of going on to work in a finer resolution immediately, we can decide whether it is possible to find a path in a finer resolution by working on the present resolution. This is achieved by growing the obstacles by r-1(r is the radius of the robot cross-section measuring by the number of grids), and then apply the algorithm again. If a path is found, that means it is possible to find a path in a finer resolution and so we move on to work in a finer resolution. On the other hand, if no path is reported, we come to the conclusion that there is really no path. To see this clearly, we need only to note that by growing the obstacles by r-1, we get rid of those cells which are partially occupied by the obstacles but are represented as blocked cells due to the relatively lower resolution level. The same process is repeated at each resolution level. Note that it is worth checking if a path is possible at the next resolution by repeating the algorithm as above since the time to do the coarser resolution is only a small friction of the time to do the finer one.

The above exposition is outlined in Figure 5 where n is the resolution level.

```
Procedure   Adaptive_Path_Planning(Map,n,r)
    while n > 0 do begin
        Map ← Grow(Map, r);
        Path ← Diffusion(Map);
        if Path ≠ Nil then
            Output(Path); Return;
        else begin
            Map ← Grow(Map, r-1);
            Path ← Diffusion(Map);
            if Path ≠ Nil then
                n ← n-1
            else
                report no path
            end
    end
```

Figure 5. Adaptive Path Planning Algorithm

V. Implementation and Experimental Results

The algorithm has been implemented on AMT DAP 510 which is a SIMD machine and which is an ideal structure for implementing cellular automata. The AMT DAP 510 has a 32×32 array of processors each connected to its four nearest neighbours(north, south, east, and west) with each processor having its own memory(Figure 6)

The program is written in DAP Fortran which exploits the bit serial nature of the individual processors to give fast execution time for the binary representations. Many computations are conveniently expressed with the built-in function 'shift', which allows all processors simultaneously get access to their four neighbours. For example, growing the obstacles by r simply means shifting r times in four directions.

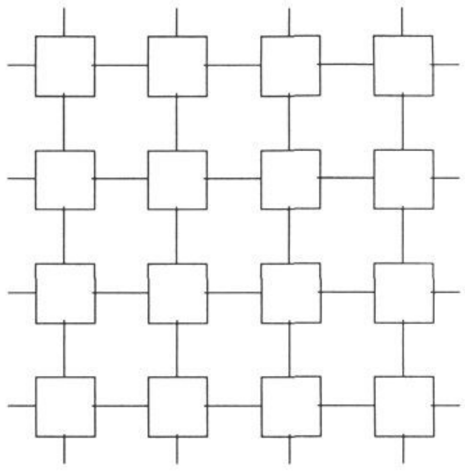

Figure 6. The structure of the DAP

When working on a map with a resolution of 32×32, each grid of the map is directly mapped onto the corresponding processor. When working on resolutions which are greater than 32×32, the map is represented by a few 32×32 matrices. Parallel instructions are applied on single matrix, while between matrices, instructions have to be executed repeatedly.

Figure 7 shows a 'worst case' example of path found only at the finest of the three resolutions. Table 1 presents the timing results for the running of the algorithm in three resolutions for the problem shown in Figure 3.

Table 1. Timing summary of the experiment(second)

Resolution	Data conversion	Diffusion	Total
32×32	0.0004	0.0110	0.0116
64×64	0.0031	3.6276	3.6321
128×128	0.0179	42.5864	42.6093

VI. Conclusions

The purpose of this paper has been to demonstrate the utility of the analogical representation of the environment for an instance of the path planning problem. The representation scheme is simple because it is directly transformed from the input image data and makes no use of explicit geometrical features of the environment. This leads to a parallel algorithm based on a SIMD structure. The algorithm is modelled by a cellular automaton which clearly specifies the inter-processor communications. The algorithm is adaptive so timings vary from problem to problem but typical times are between 0.01 - 46 seconds in a 3 resolution level version. The algorithm has applications not only to mobile robot navigation but also to surveillance problems involving motion analysis where some element of fast prediction of paths is required.

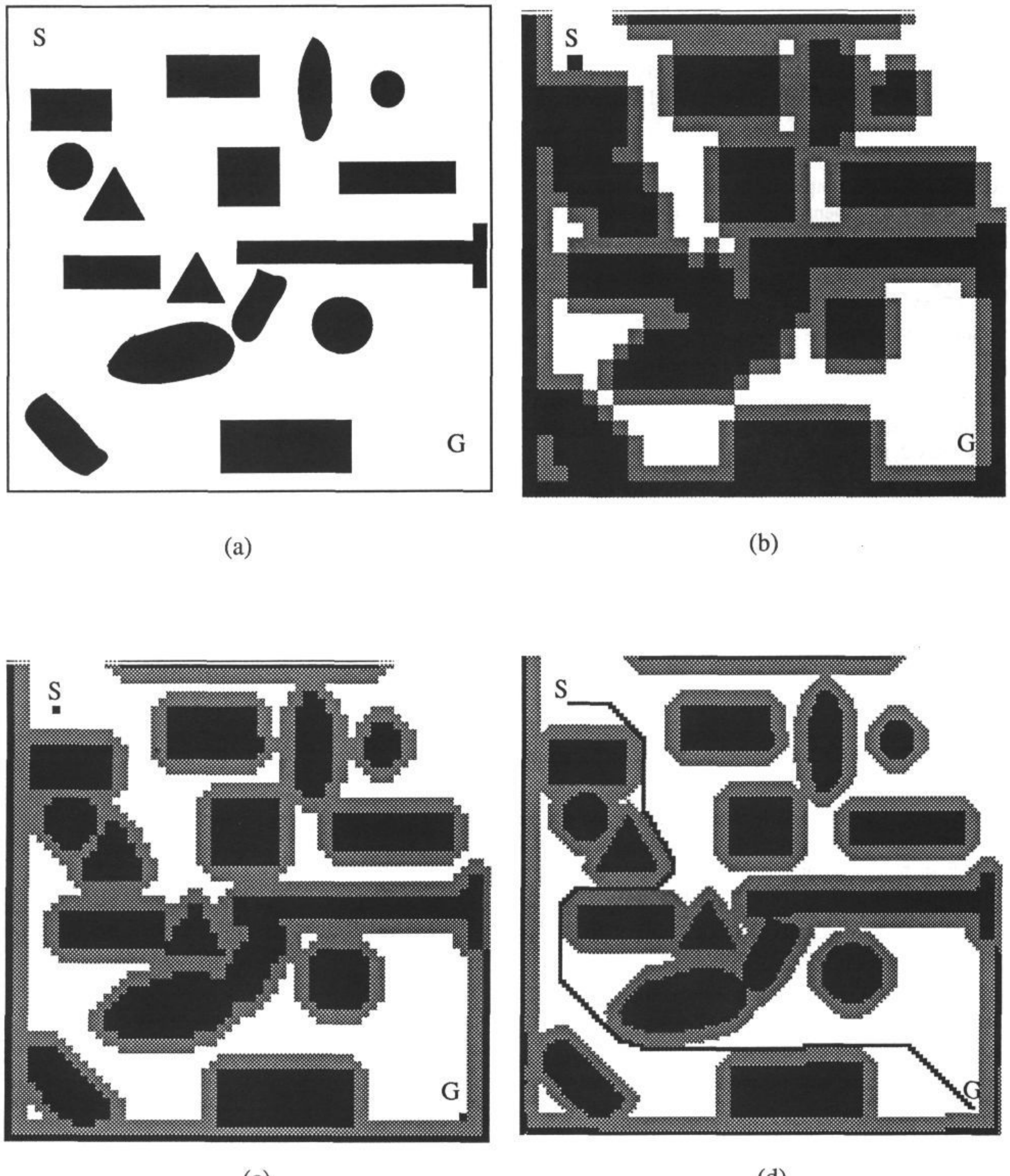

Figure 7. (a) The original map (b) Blocked in resolution 1 (c) Blocked in resolution 2
(d) Found a path in resolution 3

References

[1] Brooks, R.A., Solving the find-path problem by good representation of free space, IEEE Trans. Syst. Man Cybern. 13(2) pp. 190-197 (1983).

[2] Brooks, R.A. and Lozano-Perez,T., A subdivision algorithm in the configuration space for find path with rotation, IEEE Trans. Syst. Man Cybern. 15(2) pp. 224-233 (1985).

[3] Canny, J.F., The complexity of robot motion planning, Ph.D. Thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA (1987).

[4] Canny, J.F., On detecting collisions between polyhedra, Proceedings of ECAI84, pp533-542(1984)

[5] Canny, J.F., Collision detection for moving polyhedra, IEEE Trans. PAMI-8, pp200-209 (1986).

[6] Kambhampati, S. and Davis,L.S., Multiresolution path planning for mobile robots, IEEE J. Robotics and Automation, vol. RA-2, no.3, (1986)

[7] Khatib, O., Real time obstacle avoidance for manipulators and mobile robots, International Journal of Robotics Research, 5(1), pp.90-98,(Spring 1986)

[8] Lozano-Perez, T. and Wesley, M.A., An algorithm for planning collision-free paths among polyhedral obstacles, Comm. ACM 22 (10) pp560-570 (1979).

[9] Lozano-Perez, T., Spatial planning : A configuration space approach, IEEE Trans. Comp. vol. C-32, no.2, pp. 108-120 (1983).

[10] Thorpe, C. E., FIDO: Vision and navigation for a robot rover, CMU-CS-84-168, (1984)

[11] Schwartz, J.T. and Sharir, M., On the piano movers problem II. General techniques for computing topological properties of real algebraic manifolds, Courant Institute of Mathematics, Rept. No. 41, New York(1982).

[12] Steels, L., Steps towards common sense, VUB AI Lab. Memo 88-3, Brussels (1988).

[13] Steels, L., Artificial Intelligence and complex dynamics, VUB AI Lab. Memo. 88-2, Brussels (1988a).

[14] Toffoli, T.,and Margolus,N., Cellular Automata Machines. A new environment for modelling. MIT Press, Cambridge MA (1987)

[15] Welzl, E., Constructing the visibility graph for n line segments in $O(n^2)$time, Inf. Process. Lett. 20 pp.167-171 (1985).

[16] Walfram, S., Statistical mechanics of cellular automata, Review of Modern Physics, vol. 55, no.3 pp. 601-644.(1983)

[17] Witkowski, C.M., A parallel processor algorithm for robot route planning, Proceedings of IJCAI83, pp827-829, (1983)