Using Path Transforms to Guide the Search for Findpath in 2D

Alexander Zelinsky
Electrotechnical Laboratory
Tsukuba Ibaraki 305
JAPAN

ABSTRACT

This paper will present a new approach to the 2D *findpath* problem for robots which have 3 degrees of freedom of movement (3DOF). This approach is based upon a new extension to distance transform planning, called the *path transform*. Path transforms have the desirable potential fields property of taking into account clearance from obstacles without suffering the problem of local minima. Since path transforms guide the search for a 3DOF path where solution paths are likely to lie, the computational burden of the new approach is low compared with previously reported results. The low computational burden is also due to efficient collision detection. A new heirarchical collision checking based on *distance space bubbles* is presented. The new approach to findpath is guaranteed to find a solution path if it exists. Computer simulation results of solving the findpath problem in non-trivial environments are presented.

1. INTRODUCTION

To solve the findpath problem in 2D for robots can require rotation as well as translation (ie 3 degrees of freedom). Solutions to this problem have been put forward by (Brooks and Lozano-Perez 1985), (Donald 1987), (Ilari and Torras 1990), (Jarvis 1983), (Lozano-Perez 1983), (Noborio *et. al.* 1989), (Schwartz and Sharir 1983) and (Zhu and Latombe 1991). These approaches can be broadly classified into three groupings; global, local and hybrid methods.

The global approach (Brooks and Lozano-Perez 1985), (Jarvis 1983), (Lozano-Perez 1983), (Schwartz and Sharir 1983) and (Zhu and Latombe 1991) construct a "connectivity graph" representing all the collision free path configurations of the mobile robot, and then search this graph for a solution path. In the case of a robot with 3 DOF the graph is 3 dimensional. These approaches have used various methods to construct the connectivity graphs, such as using exact cell decomposition (Schwartz and Sharir 1983), approximate cell decomposition (Brooks and Lozano-Perez 1985, Zhu and Latombe 1991), and visibility graphs (Jarvis 1983 and Lozano-Perez 1983). The drawback of these methods is that it is expensive computationally to construct the "connectivity graph" before the findpath search commences. However the solution put forward by (Zhu and Latombe 1991) addresses the issue of expensive construction of the search graph by using a hierarchical approximate cell decomposition. This technique decomposes the configuration space into rectangloid cells, which are labelled as "empty", "full" or "mixed". The planner tries to find a sequence of cells or "channels" that join the start configuration with the goal configuration. If a channel passes through a region of "mixed" cells, these cells are recursively decomposed into smaller "empty", "full" or "mixed" cells. The search graph is augmented progressively as "mixed" cells are decomposed rather than completely reconstructed. This makes this findpath planner much faster than other global methods.

The local approach (Donald 1987) does not build a graph of all the collision free configurations. Instead a grid is placed over the configuration space. The grid is searched for a solution path using heuristics. The search heuristics are generated from the information about the geometry of the local configuration space. Since heuristic guided searches requires no preprocessing they run faster than global methods. However, heuristic methods can guide the search into unfavourable situations e.g. dead-ends, which can only be escaped with a computational effort that is higher than global methods. One solution is to use potential fields (Khatib 1986 and Krogh 1984) as the guiding search heuristic.

The problems of using potential fields is well documented (Koren and Borenstein 1991). The worst problem affecting potential fields is the local minima problem. One solution to this problem (Arkin 1989) is to execute a random robot motion. Another solution (Adams *et. al.* 1990) is to temporarily relocate the goal. Both these approaches are *ad. hoc.* and neither can guarantee that the robot will not remain caught in a local minima. Khosla *et.al.* (1988) constructed a potential field based on superquadrics to counter the problem of local minima. Using this approach the potential field is less susceptible to local minima, however the problem is not completely eliminated. Constructing potential field without local minima was first reported by (Rimon *et.al.* 1988). The computation effort required to construct the potential field for the (Khosla *et.al.* 1988) and (Rimon *et.al.* 1988) approaches can be high. Neither of these approaches can easily deal with the obstacles or the goal position being moved, since this requires the recalculation of the potential function. Another drawback is that the potential functions can only be calculated for obstacles with simple geometric shapes.

The hybrid approach is a compromise between global and local approaches (Ilari and Torras 1990 and Noborio *et. al.* 1989). Both these approaches compute a coarse path for the robot which is likely to yield a 3 DOF path. This coarse path is then searched for a fine path using a set of heuristics. The (Ilari and Torras 1990) approach assumes that the collision free paths for a mobile robot with 3 DOF are likely to lie in the middle of free space between obstacles. Initially the environment model is preprocessed to find all the global paths. To this network of global paths the start and goal locations are added. Since the (Ilari and Torras 1990 and Noborio *et. al.* 1989) findpath approaches are based on heuristic search they can fail, although the probability of this occurring is much lower compared with purely heuristic methods.

It is obvious from the review of past research of the findpath problem that the bulk of the work (with the exception of potential fields) has concentrated on minimising the distance between the start and goal locations. The shortest path between the start and goal locations may reduce a robot's travelling time or energy requirements. However, the safety of the robot should not be ignored (Thorpe 1984). The safety of the robot becomes important particularly when there are uncertainties in the environment information, such as the exact shape and position of obstacles. This problem is compounded by the uncertainty in the dynamic control of a robot i.e. the precise position of the robot is not always known by the robot's control system. Thus both minimum distance to a goal and safety of the robot need to be considered simultaneously during path planning.

The findpath approaches presented by (Brooks and Lozano-Perez 1985, Donald 1987, Jarvis 1983, Lozano-Perez 1983, and Schwartz and Sharir 1983) do not take into account clearance from obstacles information during the findpath operation. They all assume that the shortest path is the "best" path, and in their findpath endeavours clip the corners of obstacles and run down the edges of obstacles. This is called the "too close" problem (Thorpe 1984). This problem can be countered by expanding the obstacles by an extra amount to avoid such collisions. However, the penalty for this strategy is that possible solution paths are blocked. The (Ilari and Torras 1990) approach can select solution paths that maximise the clearance of the path from obstacles while simultaneously minimising the length of the path to the goal. Since this approach is based on selecting paths through the middle of free space, a drawback is that the solution path may deviate significantly from the shortest path. This is called the "too far" problem (Thorpe 1984).

The (Noborio *et. al.* 1989) and the (Zhu and Latombe 1991) path planners do not take clearance from obstacles into account. The (Noborio *et. al.* 1989) planner searches a quadtree's free space cells for a negotiable path for the minimum width of the robot while at the same time minimising the path length to the goal. The effect of this strategy is the planner will avoid the quadtree's smaller free space cells which are usually found near the boundaries of obstacles. Therefore, the likelihood of suffering the "too close" problem is reduced, but not guarantied. The (Zhu and Latombe 1991) planner can be influenced to avoid "mixed" cells. This has the effect of keeping the robot away from obstacles. However this planner can suffer the "too close" and "too far" problems. The way the environment is decomposed into the approximating cells determines whether these problems will occur or not.

Potential field methods can generate "safe" paths. However, since no global search is undertaken the solution path may neither be the shortest nor the safest. It is simply a negotiable path from the start configuration to the goal configuration. As yet, no robust mechanisms have been developed for handling local minima. Thus, using potential field methods to elegantly solve the findpath problem is difficult.

A novel approach to the problem of findpath using distance transforms (Refer Figure 1).was presented by (Jarvis and Byrne 1986). This approach considers path planning to be finding all paths from the goal location back to the start location. This planning approach propagates a distance wave front through all free space from the goal. From any starting point in the environment the shortest path to the goal is traced by following path of steepest descent. If no downhill path exists then we can conclude the goal is unreachable. Despite the disadvantage of computing all paths from the goal location the distance transforms have a number of advantageous properties. These include; supporting multiple robots and multiple goals. In addition to supporting different types of findpath. For example paths which favour or avoid certain areas in the environment (Jarvis and Byrne 1986) and paths of complete coverage of the environment (Jarvis *et. al.* 1988). Payton (1990) reported work that is similar to distance transforms. Payton uses a grid representation where each grid cell stores the lowest cost to the goal. From the grid a circulation map is derived which is represented with gradient fields to show all paths to the goal. Both the (Jarvis and Byrne 1986) and (Payton 1990) approaches suffer the "too close" problem since they only consider the shortest path to the goal.

In (Zelinsky 1989 and 1991) an extension to the distance transform called the "path transform" was presented. Using the path transform approach to findpath, instead of propagating a distance wave front from the goal, another wave front is propagated which is a combination of the distance from the goal together with a measure of the discomfort of moving near obstacles. This has the effect of producing a distance transform which has the properties of potential fields *without* local minima.

The distance transform is extended to include safety from obstacles information in the following way. Firstly, the distance transform is inverted into an "obstacle transform" where the obstacle cells become the goals. The resulting transformation yields the minimum distance for each free cell from the boundary of the closest obstacle cell (Refer Figure 2). Lastly, a second distance transform is generated through free space from the goal location using a new cost function. This cost function is refered to as the "path transform" (*PT*). The path transform for a cell c is defined as:

$$PT(c) = \min_{p \in P} \left(length(p) + \sum_{c_i \in p} \alpha obstacle(c_i) \right)$$

where P is the set of all possible paths from the cell c to the goal, and $p \in P$ i.e. a single path to the goal. The function *length(p)* is the length of path p to the goal. The function *obstacle(c)* is a cost function generated using the values of the obstacle transform. It represents the degree of discomfort the nearest obstacle exerts on a cell c. The weight α is a constant ≥ 0 which determines by how strongly the path transform will avoid obstacles.

The value stored in the path transform of each cell represents the minimum propogated cost to the goal. The propogated cost is a weighted sum of the distance to the goal and the cost of approaching obstacles. Since the path transform determines the costs of all paths to the goal from each cell it therefore does not contain local minima. The path to the goal is found by tracing the path of steepest descent. Examples of the path transform with different values of α are shown in Figure 3. Figure 3 (A) shows the path transform with $\alpha = 0.0$, resulting in a distance transform where the shortest path to the goal is best. Figure 3 (B) shows the path transform with $\alpha = 0.5$, resulting in a solution path that takes a safer path to the goal. Figure 3 (C) shows the path transform with $\alpha = 1.0$, resulting in a solution path which takes the safest path to the goal.

The cost function *obstacle(c)* that was found to be most effective was a cubic function that ranged from zero at a fixed distance, set by the user, to a maximum cost at zero distance. Such a cost function has the advantages of: creating a saddle between the repulsion peaks of neighbouring obstacles, ease of computation, and has its effects bounded in a local area.

The (Ilari and Torras 1990) approach to finding safe paths is to construct a Voronoi diagram of the free space and then removing branches in the diagram that are not relevant to planning. Figure 4 shows a complete Voronoi diagram and the planning version after removal of the irrelevant path branches. This figure also shows how sensitivity Voronoi diagrams are to noise. The small triangular obtrusion on the boundary causes an unnecessary deviation in the global path. There is no mechanism to limit the effect of irrelevant obstacles. Ilari and Torras smooth the final path to remedy the deficiencies of the planning process. Also, Ilari and Torras join the start and goal locations to the Voronoi network of paths by finding the shortest straight line to the network which does not intersect with any obstacle. This practice has the side affect of creating non-optimum paths between the start and goal locations.

There has been other work which is similar to the path transform (Barraquand and Latombe 1991). Barraquand and Latombe construct a numeric potential field using a grid representation. This is done in three (3) steps. Firstly, they compute an "obstacle transform" of the free space cells, from which they extract a "distance skeleton" which represents a digitised Voronoi diagram of the free space in the environment. Joining the highest values in the obstacle transform with line segments will yield a distance skeleton. They then connect the goal cell to the distance skeleton. Secondly, they compute a distance transform from the goal cell to all cells that are members of the distance skeleton. Thirdly, they compute another distance transform from the member cells of the distance skeleton to all the remaining free space cells in the environment. The end result is a "numeric" Voronoi diagram of the environment. This method is more complicated to compute than the path transform. Also, since it maximises clearance from obstacles it suffers from the "too far" problem. Another drawback is because Barraquand and Latombe do not specifically consider clearance information to generate their numeric potential, their method can guide the robot through narrow free space channels that are close to the goal there by endangering the robot. They counter this problem by removing channels or branches of the Voronoi diagram that are too narrow. In effect they grow selected obstacles, and thereby lose the completeness of their solution. The path transform does not suffer this drawback.

The path transform can achieve the same objectives sought by (Ilari and Torras 1990) and (Barraquand and Latombe 1991). The obstacle transform represents the distance from each free space cell to the nearest obstacle. Instead of generating a Voronoi diagram a cost function is applied to the obstacle transform which varies inversely with the distance to the nearest obstacle. The cost function exerts its influence over a limited distance. The solution path minimises the distance to the goal while keeping a safe distance from obstacles. If no safer solution path exists only then will the path transform venture into areas that have less clearance from obstacles.

The second stage of the Ilari and Torras planner refines the global path to produce a 3 DOF path by using heuristics. The authors note the drawback of using a heuristic which only minimises the distance to the goal. They propose two types of heuristics to refine the global path. The first heuristic tries to select points in free space that keep the robot close to the goal while not straying from the global path. This heuristic is computed along the whole length of the global path prior to the search for a 3 DOF path. The path transform provides the same facility as this heuristic without the need for preprocessing. The path transform can be regarded as a heuristic which pushes a robot away from obstacles in the direction of the goal. In contrast the Ilari and Torras planner pushes the robot toward the goal by keeping to the global path, rather than trying to stay away from obstacles.

The purpose of Ilari's and Torras's second heuristic is to keep the robot's body closely aligned to the global path. This heuristic requires that every point in free space must be assigned a most suitable orientation value prior to planning. A suitable orientation is an orientation which has the most chance of overcoming, with the fewest reorientations, future bottle-necks along the proposed path. The path transform does not provide an equivalent to this class of heuristic. However in the Section 2 a new mechanism is presented which allows path transforms to perform in an equivalent manner to the Ilari and Torras orientation heuristic. The extended path transform can then be used to solve the findpath problem. Simulation results of the new algorithm are presented in Section 3. Finally in Section 4 conclusions and insights gained are presented.

2. THE APPROACH

The objective in the findpath problem is to move the robot from a start configuration to a goal configuration. The path transform can only guide a point robot to a point goal configuration. The single point path transform specifies a robot with two degrees of freedom (translation only along the x and y axis). Selecting one control point on the rectangular robot does not provide the path transform with enough information to determine whether or not the robot has reached the goal configuration. If we consider the robot to have two control points, one at each end, we have enough information to determine whether the goal configuration has been reached. This is the approach used by (Barraquand and Latombe 1991).

A path transform is computed for each control point on the robot. The path transform will describe a solution path from the start configuration to the goal configuration for each control point. Figure 5 shows a robot with two control points and the path transforms which correspond with each control point. Figure 5 (A) shows a rectangular robot, start and goal configurations (the heavy dot discriminates between the two ends of the robot). The two control points which have been selected on the robot correspond to the mid points of the two ends of the robot. Figures 5 (B) and (C) show the path transforms for each control point. The movement of the individual control points is not independent. Only movements of the control points that do not violate the geometry of the robot are allowed. Consequently situations will arise where all the path transforms drive the robot into "conflict" situations, where all the control points are working against each other. A conflict resolution function is needed to handle this problem. This function selects a single control point, and the path transform values of this control point are used to guide the robot to the next grid cell. The conflict resolution function selects the control point with the largest path transform value, because this is the control point which is furthest from its goal configuration. The chosen control point is then moved closer to the goal. This is done by moving down the steepest descent gradient of the path transform (i.e. the neighbouring cell with the smallest path transform value). This algorithm uses an 8 connected grid. The remaining control point is moved to a grid cell which does not violate the geometry of the robot, and does not cause a collision with obstacles in the environment. In Figure 5 the conflict resolution function selects the control point in Figure 5 (A). The robot moves towards the goal as shown in Figure 5 (D). Once a move has been successfully completed, the conflict resolution function selects the next control point to guide the next movement of the robot. This procedure is repeated until all the control points on the robot reach their respective goal configurations.

Checking if a proposed move produces an illegal geometric configuration of the robot can be done in a straightforward manner. The control points must always be separated from each other by the same fixed distance. After each proposed move the distances between all the control points are calculated, and compared to the distances between the control points of the initial robot configuration.

A collision check must be made every time the robot is moved, because the obstacles in the environment were not expanded by the radius of the cylinder approximating the robot. This is not a trivial task and may seem to be an expensive computation. However a hierarchical collision testing procedure is used to implement collision testing based on "distance space bubbles". Using distance space bubbles the robot is placed inside a bubble which is of sufficient radius to encapsulate the entire robot. The distance to the closest obstacle is measured from the centre of the bubble. If this distance is greater than the radius of the bubble then no collision has occurred. If on the other hand this distance is less than the radius of the bubble, then a collision is possible and further testing is necessary. The bubble encapsulating the robot is burst and the robot is then enclosed with two smaller bubbles, and the test for collision is repeated once more. This process continues until it can be confirmed that no collision has occurred, or the bubbles reach a resolution, beyond which no further testing is done. In this case the conclusion is made that a collision has occurred. Refer to Figure 6 for an example of collision detection using the bubbles hierarchy. This figure shows from left to right the series of tests for collision between a rectangular robot and an obstacle. The first figure shows that a collision has occurred with the enclosing bubble. The robot is split along its longest side, giving two smaller rectangles. These rectangles are enclosed by bubbles, and these bubbles tested for collision.

This hierarchical collision checking procedure suits this path planning method for the following reasons. During the computation process of the path transform, the obstacle transform is calculated. The obstacle transform gives for each free space cell the distance to the closest obstacle. Thus, the information necessary for checking the safety of a space bubble is already available. Secondly, this planner deliberately steers the robot away from obstacles, approaching obstacles only when it is necessary. Thus, the bubble hierarchy is very suitable since checking for collisions is done primarily at the highest levels. This gives a cheap and efficient collision detection mechanism. The distance space bubbles are computed prior to path planning and is stored in a binary tree. The creation of the distance space bubbles can generate superfluous bubbles i.e. bubbles which only cover interior parts of the robot. It is only necessary to store the distance space bubbles which enclose the outside surfaces of the robot. The distance spaces bubbles concept can be applied to robots of various shape and dimensions. Barraquand and Latombe (1991) use a divide and conquer technique in which is based on checking on the two ends of a line segment representing the robot. Conceptually, their method can be equated to our method except that at each end of robot Barraquand and Latombe place a distance space bubble whose radius is length of the line segment. The disadvantages of their technique is that it performs twice as many tests and uses an unnecessarily large distance space bubble. Barraquand's and Latombe's collision detection procedure is more expensive than our's particularly as the shape of robots become more complicated.

Following the steepest descent path for control points with the largest path transform values will not necessarily guarantee that the robot will reach its desired goal configuration. The robot can get caught in local minima in configuration space situations. This produces an undesirable oscillating behaviour. Figure 7 shows an example of this behaviour. Figure 7 (A) shows a rectangular robot parked in a garage. The task is to repark the robot to face in the opposite direction. Two control points are selected on the robot; these correspond to the mid points of the two ends of the robot. Figures 7 (A) and (B) show the path transforms for each control point. The algorithm selects to move the control point in Figure 7 (A). The resulting motion is shown in Figure 7 (C). The algorithm then selects to move the control point in Figure 7 (B). This results in the motion shown in Figure 7 (D).

The solution to this problem is use the A* (Hart *et. al.* 1968) algorithm as do (Barraquand and Latombe 1991). The A* algorithm keeps a record of all the moves made by the robot. The A* algorithm uses the straight line distance to the goal, as the heuristic to guide the robot to the goal. Our algorithm uses the path transform values as the heuristic to guide the robot to the goal. The path transform values are a much more powerful heuristic than the straight line distance to the goal, and therefore will guide the search through configuration space much more efficiently. The path transform produces a potential field valley between obstacles, through which the robot moves. By choosing suitable control points on the robot, the robot will naturally orientate itself with the potential contours of the path transform. Thus using the path transform with two control points creates an orientation heuristic similar to the (Ilari and Torras 1990) orientation heuristic but without the extra work. At the end of path planning the A* "open" list shows all the intermediate moves of the control points between the start and goal configurations.

Since the path transform acts as a heuristic to guide the robot to the goal, the robot can be guided into situations where no solution path exists. For example consider the case of moving a long rectangular robot down a narrow corridor, and around a sharp bend. If the robot is too long, there is no solution path, even though the path transform indicates there is a path to the goal for a point robot. The robot is trapped. The only option is to backtrack along the open list. This is same way the A* search escapes dead end situations. Figure 8 shows an example of the A* algorithm backtracking. Figures 8 (A) - (D) show the A* search algorithm backtracking out of a dead end.

If a solution path for the mobile robot to move between the start and goal configurations does not exist, for example if the robot is surrounded by impassable narrow corridors, it should be understood that this algorithm will search the entire solution space. Although this is a drawback, a path planner is usually invoked only when it is known that a solution path between the start and goal configurations exists. Due to the exhaustive search properties of this approach it should be clear that the algorithm will find the solution to a findpath problem if it exists.

The situation can arise that a solution path is unnecessarily long. The open list is checked for any configuration of the robot earlier on in the solution path which could directly reach the current robot configuration. If the solution path can be shortened, the robot configurations which have been by-passed are deleted from the open list and are added to the closed list. Figure 9 shows an example of a solution path being shortened. Figure 9 (A) shows the planner rotating the robot about one control point. After the robot rotated to the location shown with the broken line in Figure 9 (A), the planner deduced that this location could also be reached with less effort by rotating in the opposite direction. This is shown in Figure 9 (B).

To Grow or Not to Grow?

The strength of this method is that the configuration space is never explicitly computed. Therefore collision detection must be performed every time we perform a move. This can lead the robot into situations of searching for solutions which do not exist. For example the goal is close and it can be seen through a small gap between two obstacles. The path transform can leak through the gap but the robot can not physically pass (Refer to Figure 10). The A* algorithm will waste time filling in this "local minima" of this impossible path. (Barraquand and Latombe 1991) deal with the problem by removing narrow paths from their distance skeleton. Even paths that are negotiable by the robot are removed. To maintain completeness of the solution we should only grow the obstacles in the environment to remove the impossible paths. This is done by not growing by the mimimim circle that encloses the robot. Rather, we grow the obstacles by the maximum circle that can be enclosed inside the robot. The maximum enclosed circle can be computed using the distance transform. The sides of the robot become goals. A distance transform is then computed inside the robot. Then we search the cells inside robot for the largest value. This value represents the radius of the maximum enclosed circle. Refer to Figure 11 for an example of maximum and minimum circles.

3. THE RESULTS

The new findpath approach was implemented in the C programming language on a Macintosh II Microcomputer. To demonstrate the capabilities of the new findpath approach a variety experiments were performed. Figure 12 shows a series of experiments performed in an environment which progessively grows in complexity. Figure 12 (A) shows a simple environment containing one polygonal obstacle. Figure 12 (B) shows the same environment used in Figure 12 (A), but with the edition of a thin wall along the top edge of the environment. Since the findpath algorithm is influenced by clearance from obstacles, the solution path shown in Figure 12 (B) differs from the path shown in Figure 12 (A). Figures 12 (C) - (E) show the environment growing in complexity and resulting findpath solutions. Figure 12 (D) shows the new approach to findpath selects a longer path once the shorter path becomes cluttered with obstacles. However, Figure 12 (E) shows the planner reverting to a shorter path once the longer path becomes cluttered. The planning statistics for the five findpath experiments are shown in Figure 12 (F). The statistics show the planner spends more computation time in the more complex environments. In Figure 12 (F) planning time signifies the effort needed to compute the path transform for the environment. The search time shows the work that is required to findpath using the path transform as the guiding heuristic.

In Figure 13, three (3) experimental results are presented which show how effectively the path transform reduces the search space during the course of path planning. Figure 13 (A) shows path planning in which the path transform does not use clearance from obstacles information. In this example the path transform is simply a distance transform i.e. the guiding heuristic is the distance from the goal. The resulting solution path is one that clips the corners of the obstacles, causing the planner to spend considerable time doing collision checking and searching for alternate safe moves. In this figure the timing results show (in seconds) the time needed to compute the Path Transform - (Planning Time), the time needed to find the final solution path - (Search Time), and the total computation time (Total Time). This figure also shows the number of collisions that occurred during the search for a path, the number of moves examined during the search, and the number of moves contained in the final solution path. Figure 13 (B) shows experimental results for the same findpath problem posed in Figure 13 (A) but this time using a mild Path Transform function which influences the solution path up to three (3) cells from an obstacle. The solution path is more graceful and computed much quicker than the solution path shown in Figure 13 (A). The speedup is due to searching a reduced space in which collisions with obstacles are less likely. Figure 13 (C) presents a using a stronger Path Transform function which influences the solution path up to five (5) cells from an obstacle. In this experiment the solution path is computed the quickest. The search for a solution path occurs in the area where collision are least likely. The statistics show that no collisions were encountered and that search space matched the final solution path.

The capabilities of the new findpath approach operating in cluttered environments is presented in the experimental results shown in Figure 14. Figure 14 shows a series of experiments performed in two cluttered environments. Figure 14 (A) shows a highly cluttered environment containing bubble shaped obstacles. Figure 14 (B) shows the same environment shown in Figure 14 (A), but with the edition of a set of thin walls along the edges of the environment. Since the findpath algorithm is influenced by clearance from obstacles, the solution path shown in Figure 14 (B) differs from the path shown in Figure 14 (A). Figures 14 (C) show the findpath solution to the same problem posed in Figure 14

Error!

In several of these experiments we can see different solution paths to the same findpath problem. This is due to the ability of our planner to varying the strength of the path transform. The (Barraquand and Latombe 1991). planner does not have same property since the digitised Voronoi diagram is static once it is computed. Also our planner behaves elegantly in uncluttered environments i.e. does not suffer the "too far" problem by following the spine of the Voronoi diagram.

4. THE CONCLUSIONS

This paper presented a new approach to the findpath problem for robots with 3 degrees of freedom. This approach is based upon selecting two control points on the robot and then constructing a path transform of the robot work space for each control point. The path transform functions as a heuristic to guide the search through the work place for a solution path. The path transform is an effective guide since it steers the search into areas of the work space where solutions are more likely. The path transform has the additional benefit of functioning as an orientation heuristic. The control points are naturally driven into the valleys of the path transform, thus minimising potential collisions. Collision detection is implemented using a hierarchy of distance space bubbles. Space bubbles allow for the quick testing of collisions between robot and obstacles in the environment. Experimentation showed that the new algorithm yields solutions to non-trivial problems that can be computed quickly.

It was shown by (Ilari and Torras 1990) that their path planner is in the order of 100 to 200 times faster in search than path planners which only use distance to the goal as the search heuristic. Since the algorithm reported in this research uses similar heuristics to those used by Ilari and Torras, the new approach should perform with a comparable efficency. However the new approach has a number of advantages over the former. The new approach does not require the two stage search used by Ilari and Torras to find a solution path. By using a single stage search the new algorithm is able to elegantly backtrack if the planner chooses to explore a path to the goal which has no solution. Backtracking with the Ilari and Torras algorithm is awkward since this planner keeps the robot on the global path, and the planner cannot explore alternatives. The Ilari and Torras approach to backtracking requires checking whether the robot is trapped. If the robot is trapped then the current global path is removed from the path network and a search for a new global path is performed. However the most important advantage of the new algorithm has over the Ilari and Torras algorithm is the quality of the solution path. The new algorithm does not suffer from the "too far" problem and the final solution path does not require a path smoothing process.

It was shown by (Barraquand and Latombe 1991) that their path planner was the extremely fast. Since our algorithm uses a similar technique of two control points to guide the search for findpath, we obtain similar performance. However, our approach is more powerful and useful for the following reasons. Firstly, the path transform is more easily computed than the (Barraquand and Latombe 1991) numeric potential function. Secondly, the path transform is a superior search heuristic since our solution paths do not suffer from the "too far" problem. Thirdly, our approach generates a complete solution by removing only the impossible paths from the search space. Finally, our hierarchical collision checking procedure using distance space bubbles is more efficient than (Barraquand and Latombe 1991) divide and conquer method.

ACKNOWLEDGMENTS

The author extends his thanks to the anonymous reviewers for their constructive criticism that greatly improved the presentation of this paper. The author also extends special thanks to Dr. Yun Hui Liu for his helpful suggestions during the preparation of this paper and to Dr. Philippe Pignon for suggesting the idea of minimal growing.

THE BIBLIOGRAPHY

(Adams *et. al.* 1990) Adams, M.D., Hu, H. and Probert, P.J. 1990. Towards a Real-Time Architecture for Obstacle Avoidance and Path Planning in Mobile Robots. *Proc. IEEE International Conference on Robotics and Automation*, Cincinatti. pp584-589.

(Arkin 1989) Arkin R.C., 1989. Motor Schema - Based Mobile Robot Navigation, *Int J. Robotics Res.* 8(4), pp92-112.

(Barraquand and Latombe 1991)
Barraquand, J. and Latombe, J.-C. 1991. Robot Motion Planning: A Distributed Representation Approach. *Int. J.. Robotics Res.* 10(6), pp628-649.

(Brooks and Lozano-Perez 1985)
Brooks, R.A. and Lozano-Perez T. 1985. A Subdivision Algorithm in Configuration Space for Findpath with Rotation, *IEEE Trans. on Systems, Man and Cybernetics*, 15(2), pp224-233.

(Donald 1987) Donald, B.R. 1987. A Search Algorithm for Motion Planning with Six Degrees of Freedom". *Artifical Intelligence* 31(3), pp295-353.

(Hart *et. al.* 1968) P.E. Hart , N.J. Nilsson and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths, *IEEE Trans. of Systems Science and Cybernetics*, 4 (2), pp 100-107.

(Ilari and Torras 1990)
Ilari, J. and Torras, C. 1990. 2D Path Planning: A Configuration Space Heuristic Approach. *Int. J. Robotics Res.* 9(1) pp75-91.

(Jarvis 1983) Jarvis, R.A. 1983. Growing Polyhedral Obstacles for Collision Free Paths. *Australian Computer J.* 15(3), pp103-111.

(Jarvis and Byrne 1986)
Jarvis, R.A. and Byrne, J.C. 1986. Robot Navigation: Touching, Seeing and Knowing. *Proc. Australian Conf. on Artificial Intelligence*. Melbourne, Australia.

(Jarvis *et. al.* 1988) Jarvis, R.A., Byrne, J.C., and Ajay, K. 1988. An Intelligent Autonomous Guided Vehicle: Localisation, Environment Modelling and Collision-Free Path Finding. *Proc. 19th ISIR The Int. Symp. and Exposition on Robots*. pp767-792.

(Khatib 1986) Khatib, O. 1986. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots", *Int. J. Robotics Res.* 5(1). pp90-98.

(Khosla *et. al.* 1988) Khosla, P. and Volpe, R. 1988. Superquadric Artificial Potential for Obstacle Avoidance and Approach. *Proc. IEEE Int. Conf. on Robotics and Automation*. Philadelphia. pp1778-1784.

(Koren *et. al.* 1991) Koren, Y. and Borenstein, J. 1991. Potential Field Methods and their Inherent Limitations for Mobile Robot Navigation, *Proc. IEEE Int. Conf. on Robotics and Automation*. Sacramento. pp1398-1404.

(Krogh 1984) Krogh, B.H. 1984. A Generalised Potential Field Approach to Obstacle Avoidance Control. *First World Conf. on Robotics Research..* SME.

(Lozano-Perez 1983) Lozano-Perez, T. 1983. Spatial Planning: A Configuration Space Approach. *IEEE Trans. on Computers*. 32(2). pp108-120.

(Noborio *et. al.* 1989)

Noborio, H., Naniwa, T. and Arimoto S. 1989. A Feasible Motion-Planning Algorithm for a Mobile robot on a Quadtree Representation. *Proc. IEEE Int. Conf. on Robotics and Automation*. Scottsdale. pp327-332.

(Payton 1990) Payton, D.W. 1990. Internalised Plans, *Robotics and Autonomous Systems*. 6(1-2) pp 89-103.

(Rimon *et. al.* 1988) Rimon, E. and Koditschek, D.E. 1988. "Exact Robot Navigation using Cost Functions", *Proc. IEEE Int. Conf. on Robotics and Automation.* Sacramento. pp1791-1796.

(Schwartz *et. al.* 1983)
Schwartz, J.T. and Sharir, M. 1983. On the Piano Movers Problem, the Case of a Two Dimensional Rigid Polygonal Body Moving Amidst Polygonal Barriers. *Comm. on Pure and Applied Mathematics*. 36. pp345-398.

(Thorpe 1984) C.E. Thorpe 1984. FIDO: Vision and Navigation for a Robot Rover, Phd dissertation, Carnegie Mellon University, Department of Computer Science.

(Zelinsky 1989) Zelinsky, A. 1989. Navigation with Learning. *Proc. IEEE/RSJ Conference on Intelligent Robot Systems*. Tsukuba, Japan. pp331-338.

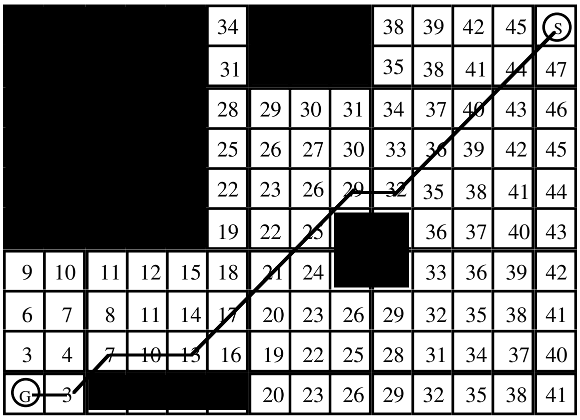
(Zelinsky 1991) Zelinsky, A. 1991. Environment Exploration and Path Planning Algorithms for a Mobile Robot using Sonar. Phd dissertation, Wollongong University, Australia. Department of Computer Science.

(Zhu and Latombe 1991)
Zhu, D. and Latombe, J.-C. 1991. New Heuristic for Efficient Hierarchical Path Planning for Mobile Robot, *IEEE Trans. Robotics and Automation*. 7(1). pp9-20.

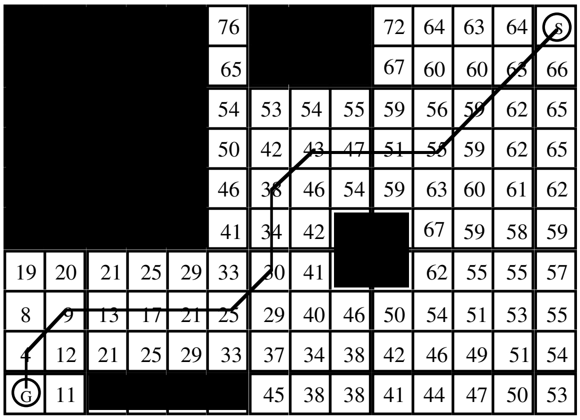
Figure Captions

			4	3	2	2	2	2	2
			4	3	2	1	1	1	2
7	6	5	4	3		1	Goal	1	2
7	6	5	4	4		1	1	1	2
7			5	5				2	2
8			6	6	5	4	3	3	3

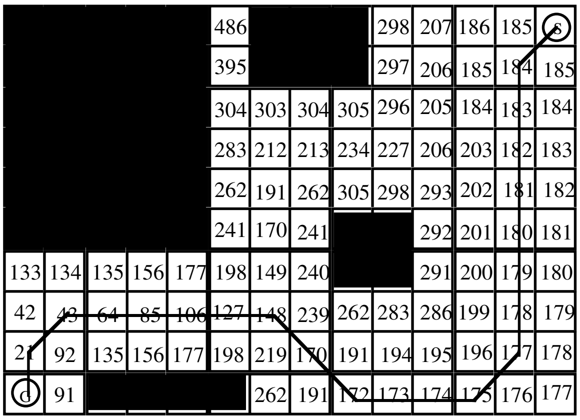
3	2	2	2	2	2	2	2
3	2	1	1	1	1	1	2
3	2	1		1		1	2
3	2	1		1		1	2
3	2	1		1		1	2
3	2	1	1	1	1	1	2



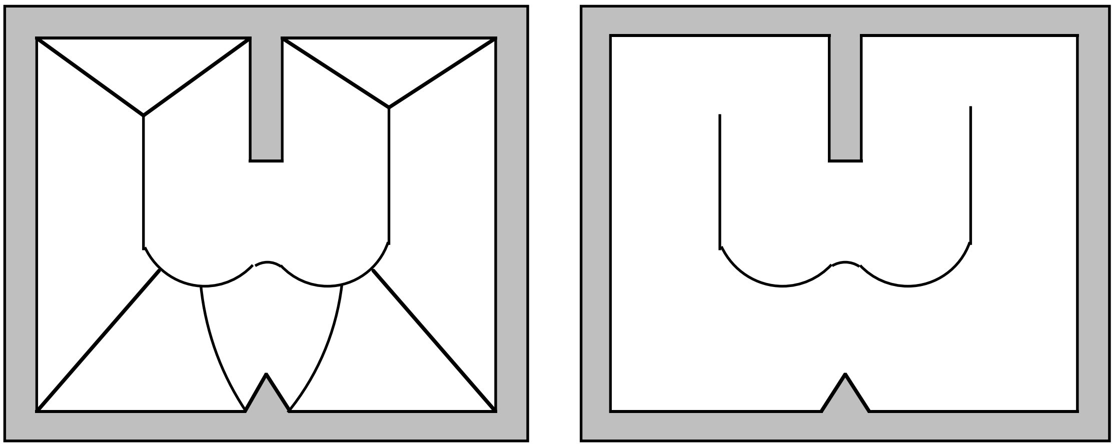
A



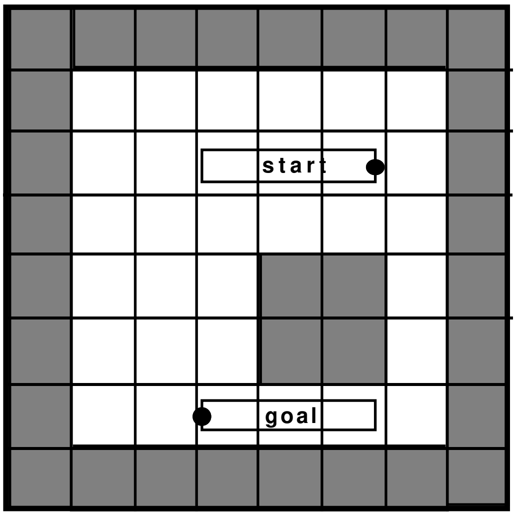
B



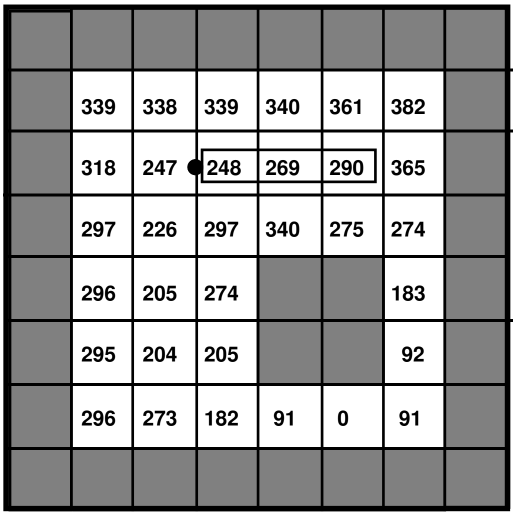
C



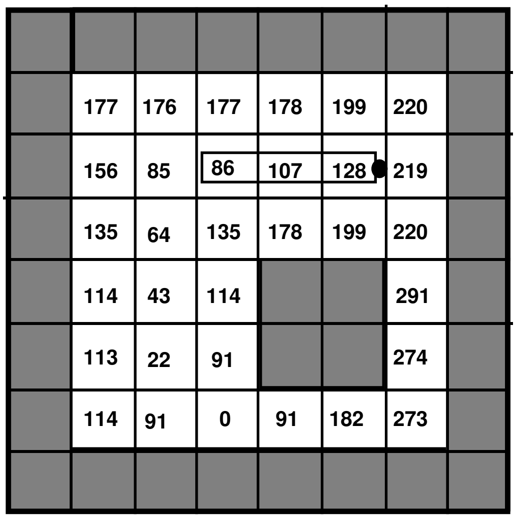
A B


start
goal

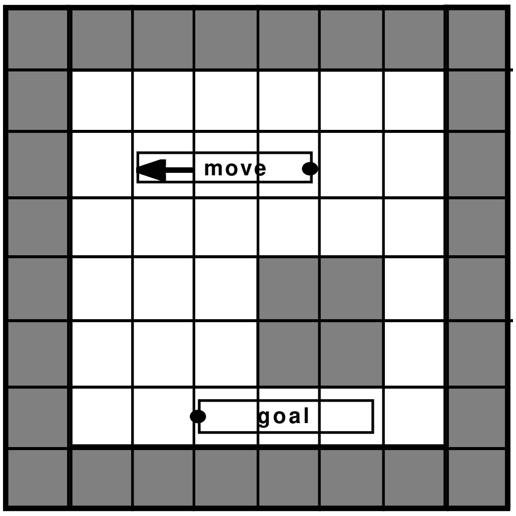
A


339 338 339 340 361 382
318 247 248 269 290 365
297 226 297 340 275 274
296 205 274 183
295 204 205 92
296 273 182 91 0 91

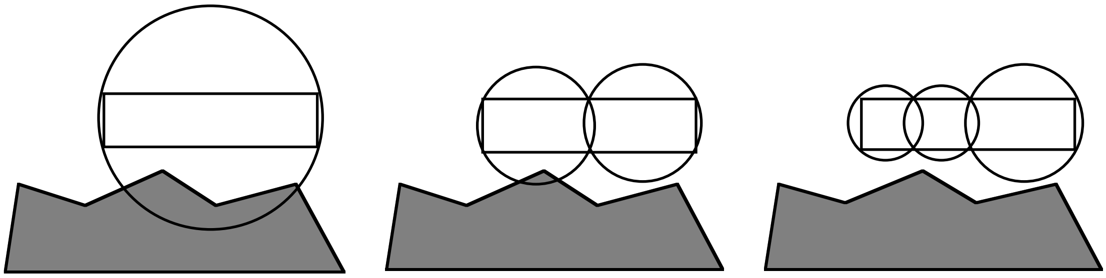
B

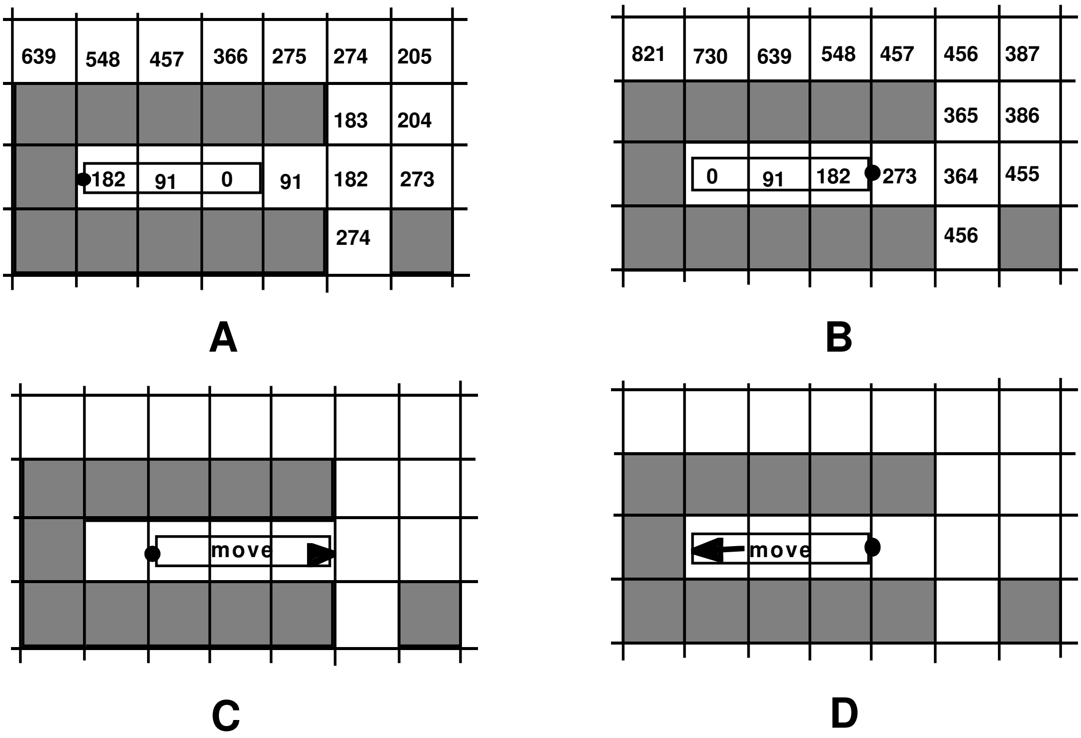

177 176 177 178 199 220
156 85 86 107 128 219
135 64 135 178 199 220
114 43 114 291
113 22 91 274
114 91 0 91 182 273

C

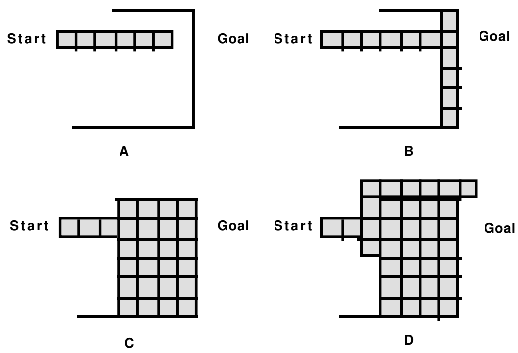

move
goal

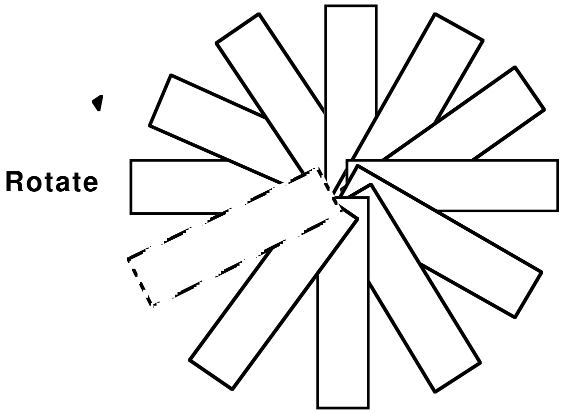
D



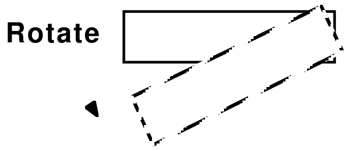

639 548 457 366 275 274 205
183 204
182 91 0 91 182 273
274
821 730 639 548 457 456 387
365 386
0 91 182 273 364 455
456
move
move

A B C D

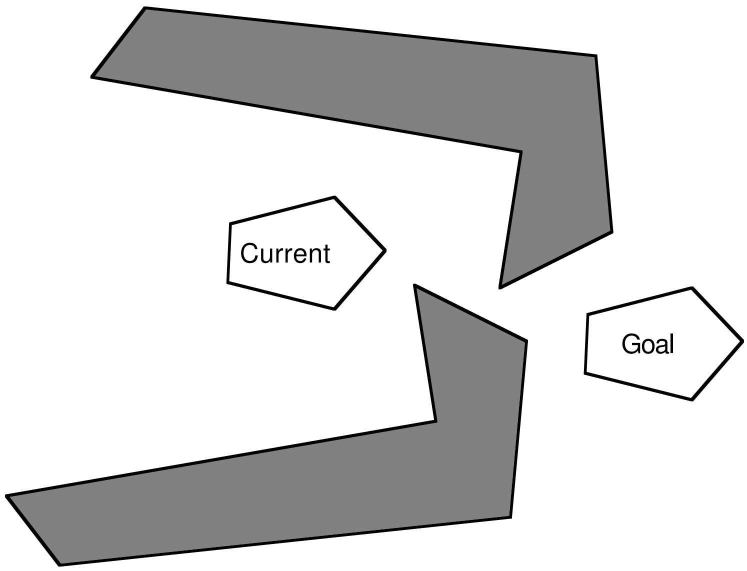

Start
Goal
A
Start
Goal
B
Start
Goal
C
Start
Goal
D

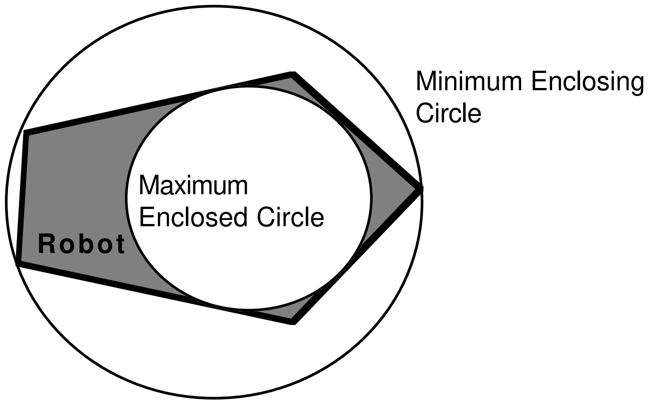


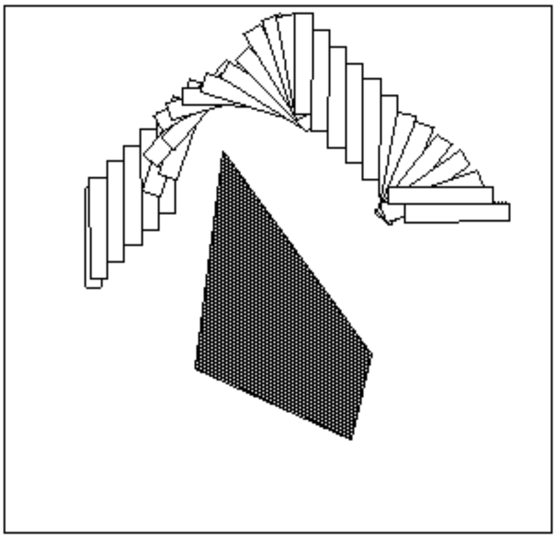
A



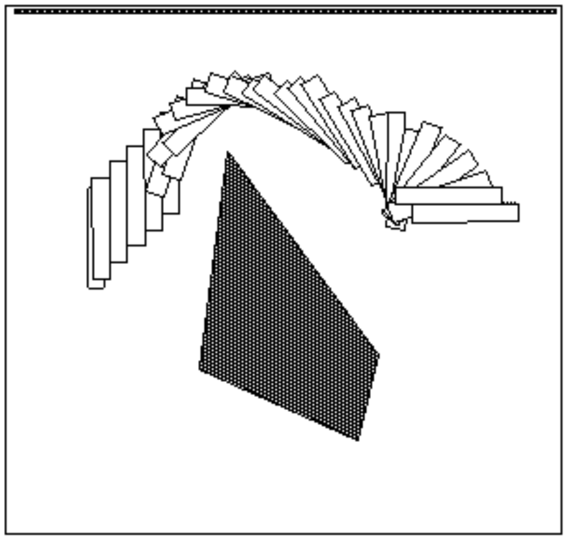
B


Current
Goal

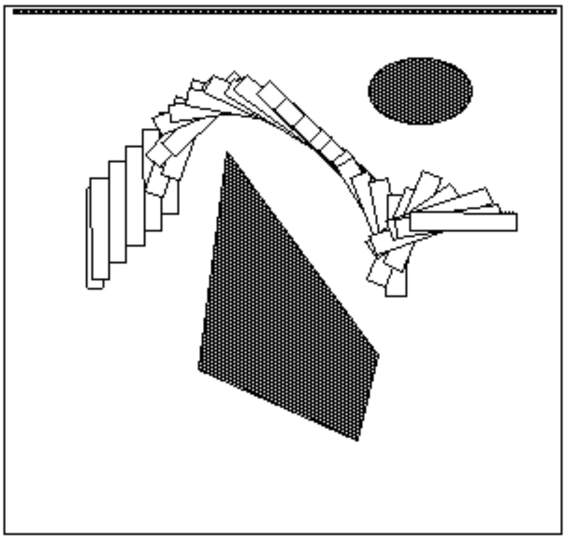

Minimum Enclosing
Circle
Maximum
Enclosed Circle
Robot



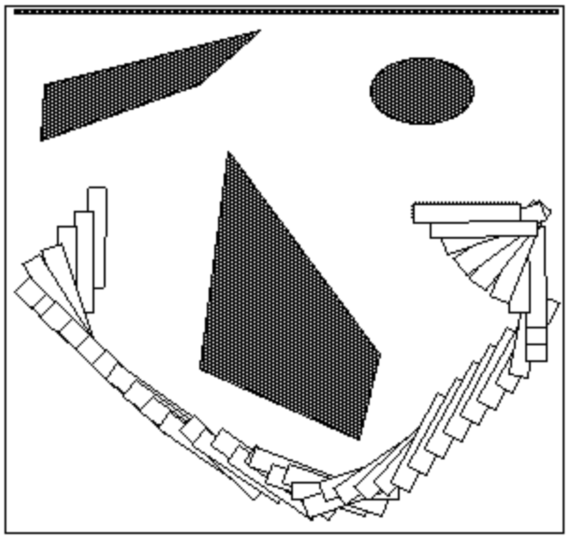
A



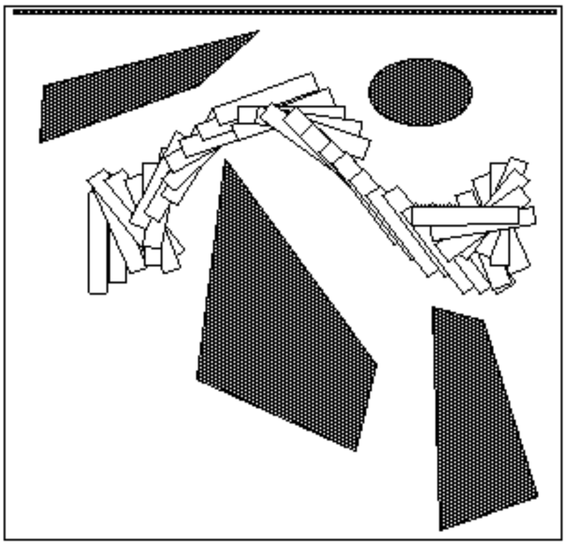
B



C



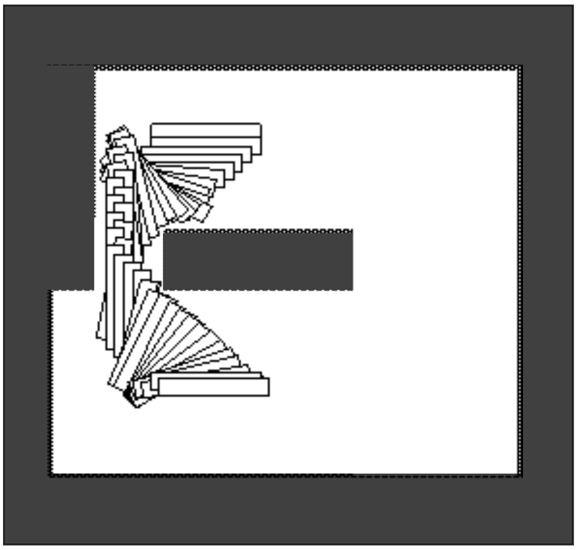
D



E

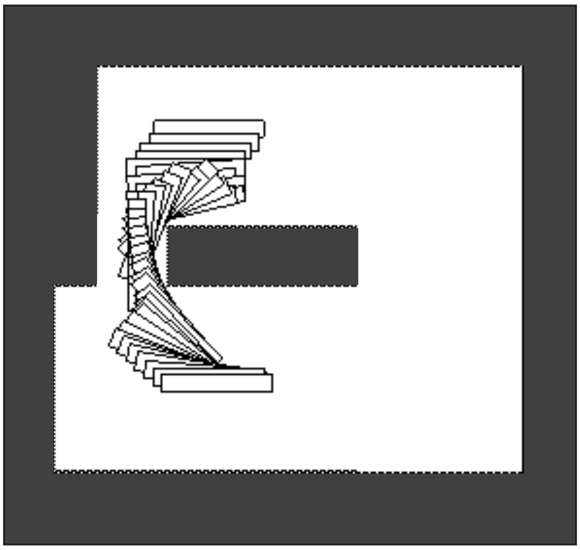
	Planning Time	Search Time	Total Time
A	8	5	13
B	7	6	13
C	7	7	14
D	7	12	19
E	7	26	33

F



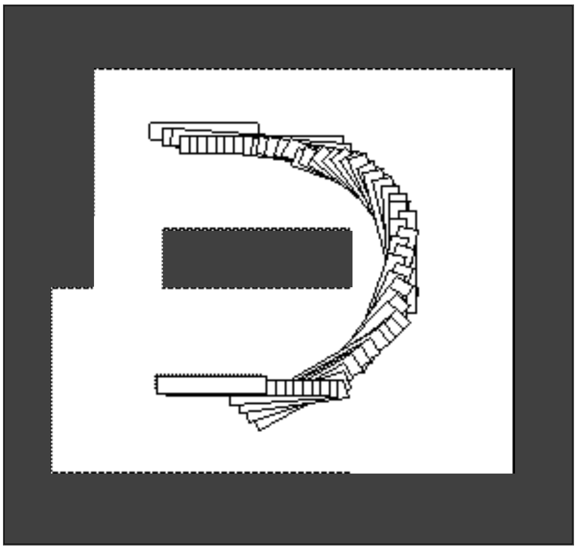
Planning Time	Search Time	Total Time
21	184	205
Collisions	**Search**	**Solution**
107	226	46

A



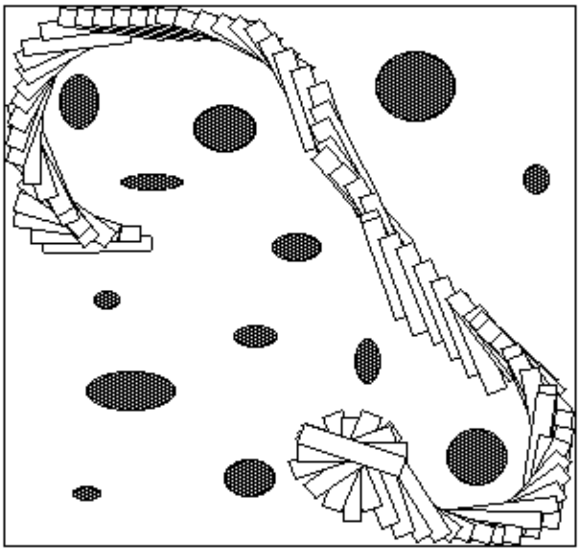
Planning Time	Search Time	Total Time
21	45	66
Collisions	**Search**	**Solution**
24	71	43

B

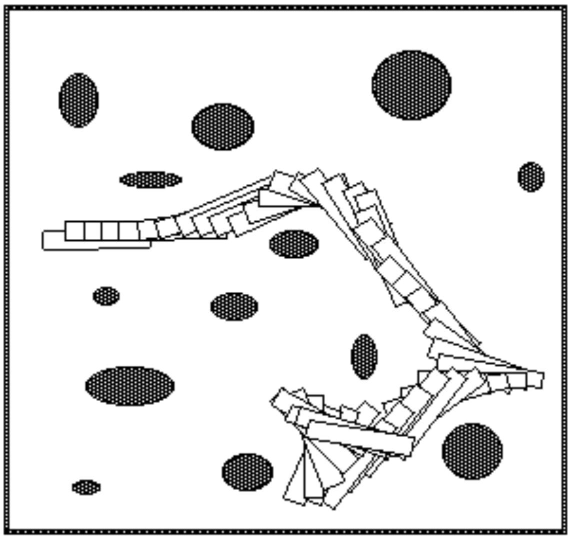


Planning Time	Search Time	Total Time
21	15	36
Collisions	**Search**	**Solution**
0	60	60

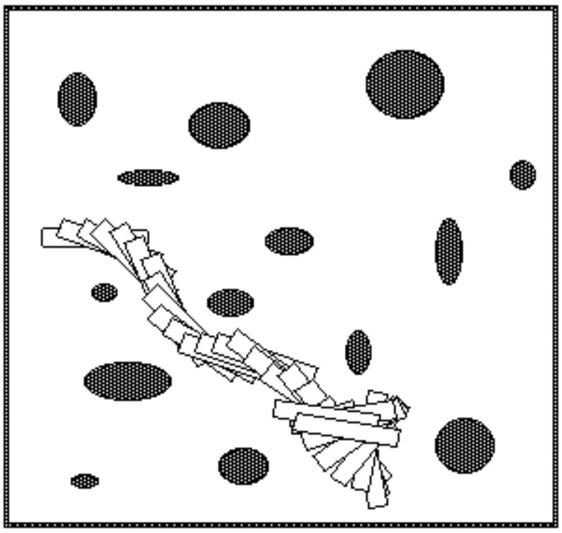
C



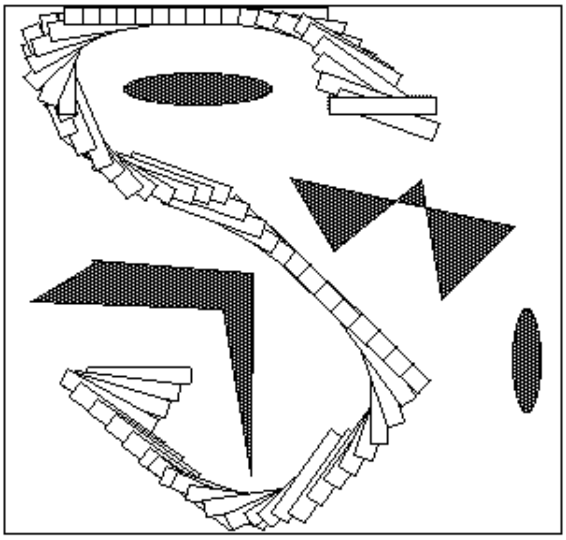
A



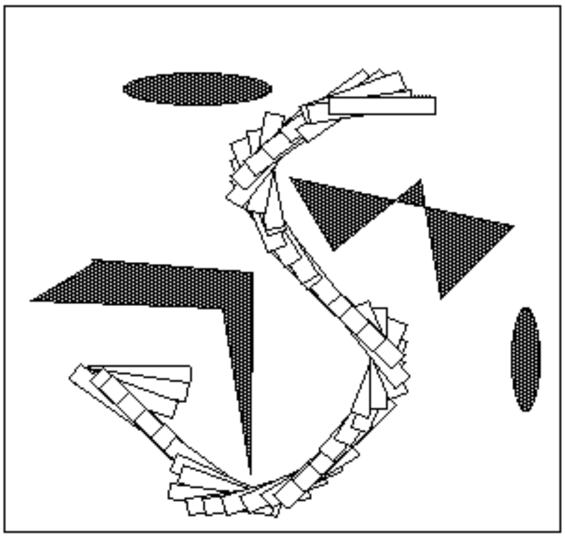
B



C



D



E

	Planning Time	Search Time	Total Time
A	8	48	56
B	8	25	33
C	8	16	24
D	7	19	26
E	7	26	33

F