# Navigation Master: Design and Implementation of Path Planning Algorithm for a known robot in a dynamic environment

Lingamgunta Nikhilesh, Ashish Kulkarni
School of Computer Science and Engineering (SCOPE), VIT CHENNAI

**Abstract**— Robotics is rapidly gaining traction in our daily lives, as well as in modern industry automation and cyber-physical applications. This necessitates the incorporation of intelligence into these robots in order to ensure (near)-optimal task execution solutions. As a result, a slew of robotics-related research difficulties have arisen, including path, motion, and mission planning, work distribution issues, navigation, and tracking. We concentrated on the path planning research challenge in this project.

**Index Terms**— Path Planning, Mapping environment, A* algorithm research, genetic algorithm research, obstacles path, coordinates, nodes.

## 1 Introduction

Navigation is one of the fundamental abilities that mobile robots must have in order to perform high-level tasks autonomously in a specific environment. The following actions can be taken to effectively address this issue. First, the robot must perceive the environment in which it must move and extract relevant information from it. Second, the robot must be able to solve the problem of localization within this environment. With this information, a trajectory to the target points must be planned, and the vehicle must be guided along this trajectory in a reactive manner, taking into account potential changes/interactions with the environment/with the user. Some onboard sensors, such as laser rangefinders and visual systems, can be used to perceive the environment. This perception task can be performed either before or after the navigation task begins, while the robot moves through the environment, and the result is a model or map of the environment. The localization task must be designed with several factors in mind, including the available sensors, the structure of the map, and the movement constraints that the robot imposes. Furthermore, integrated exploration systems take all of these issues into account, developing trajectory planning and control as well as estimating the robot's position and orientation within it.

## 2 Problem Statement

A robot travels from one position to another position with obstacles having in the middle of its way to reach the destination. We need to find the path for the mobile robot in such a way that is shortest path to robot with respective to the coordinators of the given space. Also, it should be the shortest path with least cost within every coordinates.

## 3 Proposed Model

### 3.1 Introduction to Proposed System

For humans, getting from one place to another is a simple undertaking. In a split second, one selects how to move. Such a simple and basic task is a huge difficulty for a robot. Path planning is an important subject in autonomous robotics. The common task is to safely navigate a robot from a starting point to a goal position, whether it's a vacuum cleaning robot, a robotic arm, or a magically flying object. The task at hand is to find a path from a starting point to a destination point. Depending on the environment model and the type of robots, this topic has been addressed in a variety of methods in the literature, nature of applications etc. Because the quality of the generated path has such a large impact on the robotic application, a safe and effective mobile robot navigation requires an efficient path planning algorithm. The minimizing of the travelled distance is typically the primary goal of the navigation process, as it has an impact on other metrics such as processing time and energy usage. This project gives a complete overview of global path planning for mobile robots as well as the relevant background information. It gives a taxonomy of global path planning problems and describes the various global route planning categories.

### 3.2 Path Planning Approaches

Several approaches to path planning for a mobile robot have been proposed. The approaches differ depending on the environment, the type of sensor, and the capabilities of the robot, and they progress toward better performance in terms of time, distance, cost, and complexity. Path planning, for example, was classified as an optimization problem by Al-Taharwa [7] based on the definition that, in a given mobile robot and a description of an environment, a plan is required between start and end points to create a path that is free of collisions and meets certain optimization criteria such as shortest path.This

definition is correct if the goal of path planning is to find the shortest path because most new approaches are introduced to find the shortest path. Looking for the shortest path does not guarantee that the time taken will be shorter because the shorter path may require a complex algorithm, causing the calculation to produce a longer output.

### 3.3 Properties of Path Planning

According to the type of environment, algorithm, and completeness, mobile robot path planning has a few main properties. The properties are whether it is static or dynamic, local or global, complete or heuristic, and complete or heuristic. Static path planning refers to an environment with no moving objects or obstacles other than a navigating robot, whereas dynamic path planning refers to an environment with dynamic moving and changing objects, such as moving obstacles. Meanwhile, local and global path planning are dependent on an algorithm that determines whether or not information about the environment is a priori to the algorithm. Assuming that the path planning is global, the robot has access to information about the environment via a map, cells, grid, etc. If the path planning is local, the robot does not have access to information about the environment and must sense the environment before deciding to move in order to avoid obstacles and generate trajectory planning toward the target.

The problem of mobile robot navigation can be broken down into three subtasks: mapping and modelling the environment, path planning, and path traversal with collision avoidance [8]. Because the navigation problem varies depending on the approach used to solve the problem, mobile robot navigation problems cannot be decomposed into fixed subtasks. For example, the bug algorithm solves the navigation problem without the need to map and model the environment and instead responds only to the output of the contact sensor.

### 3.4 Local Path Planning

Local path planning is path planning that requires a robot to move in an unknown or dynamic environment where the path planning algorithm responds to obstacles and changes in the environment. Local path planning can also be defined as real-time obstacle avoidance using sensory-based information about contingency measures that affect the robot's safe navigation. To plan a robot's local path, the robot is typically steered along a single straight line from its origin to its destination. The robot follows the line until it encounters an impediment. It then avoids obstacles by diverging from its path, while updating crucial information such as new distance from present position to target point and obstacle departure point, among others. If the robot is going to reach its goal properly, it must always know where it is in relation to the target point.

One well-known local path planning technique is the potential field method. The robot is viewed as a particle moving under the influence of an artificial potential generated by the goal configuration and the obstacles in this path planning method. A potential function's value can be thought of as energy, and its gradient as force. The goal configuration has an appealing potential, while the obstacles all have a repulsive potential.

A robot's journey in a closed area can be planned in a variety of ways. There are a variety of ways to choose a path, including landmark navigation, reactive planning, and other path planning algorithms. One way to do this would be to use an algorithm to retrieve a set of coordinates that the robot can follow in an environment that has been predetermined. It is common for algorithms to be tested by creating a grid of a predetermined size indicating which parts of the map can be traversed. We may safely presume that the robot can reach all grid edges during testing. Within the scope of this project, we will also assume that a solution is always possible from a given starting point.

## 4 Overview of Robot Path Planning Problem

We're on the cusp of a robotics revolution, and it's about to happen. Automated systems have been developed for a variety of tasks in areas such as smart homes and manufacturing laboratories as well as airports and shopping malls. If you want a robot that can perform (near) optimally and efficiently, you'll need to give it artificial intelligence (AI). Intelligence in mobile robots, on the other hand, requires a number of research difficulties to be resolved. For a robot to navigate correctly, it must know where it is in relation to its goal. Aside from that, he has to evaluate the threats of the surrounding environment and alter his activities in order to maximize the chances of reaching the destination on time.

Whatever the difficulty of the path planning problem, three major issues must be considered: efficiency, accuracy, and safety. Any robot should be able to find its way around in a short amount of time while using as little energy as possible. A robot should also avoid any obstacles in its path in a safe manner. It must also precisely follow its ideal, obstacle-free path.

Because of the complexity and time-consuming nature of planning a path in large-scale environments, robotic applications that require a real-time component are hindered. In this project, we're trying to figure out the best way to solve the path planning problem so that we can find the shortest route in the shortest amount of time possible. On the other hand, we had to take into account the fact that the robot is operating in a vast, complex environment with many obstacles of varying sizes, shapes, and locations.

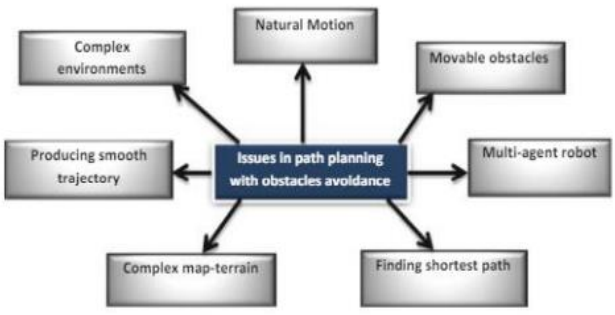

Fig. 1 Different issues of Path Planning

## 5 Proposed System

It is based on A* and genetic algorithm research for global path planning known as dynamic environment and communicate. The A* algorithm starts from a specific node in a graph and tries to find the shortest path to the given goal node. With higher speed, A* determines the heuristic function's value just prior to colliding instead of first. This results in significant processing time savings.

### 5.1 Evolution of Environment Nature

The path planning problem can be solved in both static and dynamic environments: In a static environment, the source and destination positions are set, and impediments do not move over time. However, in a dynamic environment, the position of obstacles and the goal may alter during the search. Path planning in dynamic environments is usually more difficult than in static ones due to the variety of the environment. As a result, the algorithms must adapt to any unanticipated changes, such as the introduction of new moving obstacles in the preplanned path or when the target is constantly shifting.

Since the path planning problem must react to both goal and obstacle motions in real time, the path planning problem becomes significantly more difficult when both obstacles and targets are moving. Algorithms for path planning in static environments do not work in dynamic environments.

### 5.2 Map Knowledge

Using an existing map as a reference, mobile robot path planning determines the start and destination locations, as well as the relationship between them. When it comes to path planning, the amount of information in the map has a big impact on the algorithm design. It's possible to divide path planning into two categories based on the robot's understanding of the environment. To begin with, there is a map that shows what the robot already knows about its surroundings. As a result, it is referred to as a global or deliberative path planning.

The robot in the second sort of path planning has no prior knowledge of its surroundings (i.e., uncertain environment). As a result, in order to avoid barriers and discover a viable path to the goal state, it must detect the positions of barriers and create an estimated map of the environment in real time during the search process. Obstacles of various forms and sizes are put in an inclined orientation in real-world robot environments, which are often unstructured. In this circumstance, the path planning problem is quite difficult. The algorithm must choose a feasible and optimal path. For unstructured road navigation, we proposed real-time autonomous vehicle path planning.

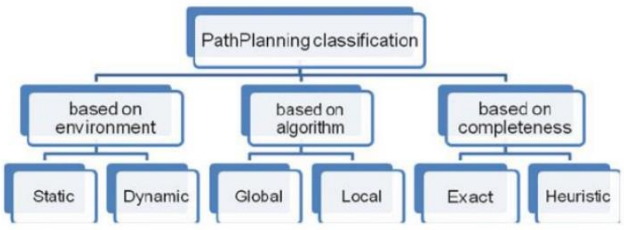


Fig. 2 Path Planning Categories

### 5.3 Softwares and Components Used

- A* Algorithm
- Genetic Algorithm Research
- A Dynamic environment
- Pygame package
- Python3
- C++

## 6 Algorithms

### 6.1 A* Algorithm Research

Using point S as the source and the destination as the aim, the A* method is employed to discover the shortest path. The algorithm now uses the established A* technique for path planning, which is why the phase is called changeover. The straight line path from source to point S and the A* algorithm resultant path from point S to target make up the whole path. When specific circumstances are met, the above-mentioned approach yields the best results. The work environment is deemed to be static. The robot's geometry and position are known. Prior to using the suggested approach, the source, goal, and robot's coordinates or relative positions are known.

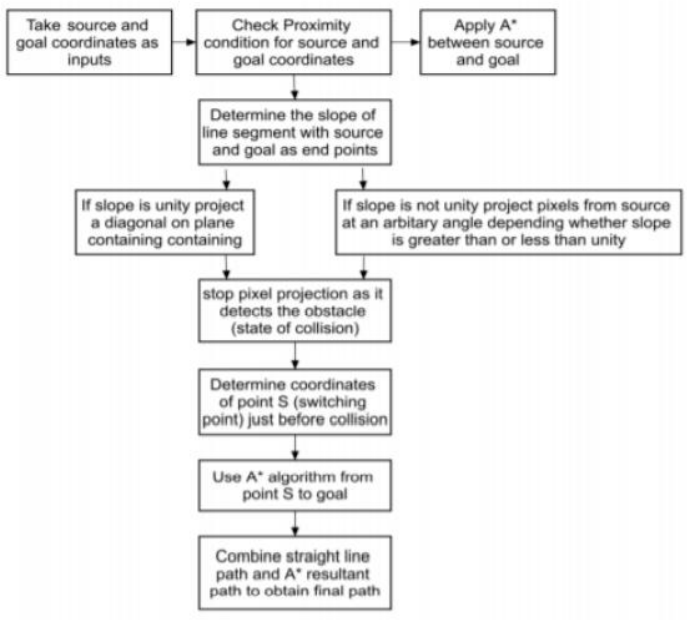


Fig. 3 Flow Chart of proposed time-efficient A* algorithm

### 6.2 Genetic Algorithm Research

This is a popular search-based optimization tool that follows Bremermann's 1958 discovery of the genetics and natural selection principles. Holland was the first to present it in the discipline of computer science in 1975. It now has a wide range of applications in science and technology, including robot navigation. It is concerned with the optimization of complex situations in which the objective function value must be maximised or minimised under given restrictions. The

population (various individuals identified by genes) must be apportioned for the given problem in this strategy, and each member of the population is awarded a fitness value based on the problem. Since the path planning problem must react to both goal and obstacle motions in real time, the path planning problem becomes significantly more difficult when both obstacles and targets are moving. Algorithms for path planning in static environments do not work in dynamic environments.

They maintain population variety and prevent premature convergence because of their mutation. Finaly, if the population has converged, the algorithm will be shut down. Although the GA is somewhat random in nature, it performs better than a random local search because it may take advantage of previous knowledge as well. When a polygonal obstruction is present, simulation results are used for the analysis. Shing et al. developed a real-time path planner because traditional approaches to searching and optimization are very slow in real-time. To search efficiently in an unknown environment, GA is a robust search method that requires very little information about the environment itself. They've shown that the outcomes can be achieved in the presence of both static and dynamic impediments. Many studies have found that the GA has some drawbacks, such as a sluggish convergence rate, no guarantee of getting the best solution, a time-consuming method for determining the parameters for mutation rate and population size, and so on.

The robot can readily monitor the moving impediment and moving goal in this manner, and arrives at the destination in a short amount of time. For defence equipment, the GA method is a widely employed intelligent technique. Creaser et al. provided a missile control demonstration based on a mix of the GA method and fuzzy logic. The GA is crucial in the development of the missile's guiding law. Iyer et al. present a novel GA-based methodology for military and ocean monitoring applications. They employed GA to safeguard a high-value military asset and to determine the best positioning plan for underwater sensor network deployment.

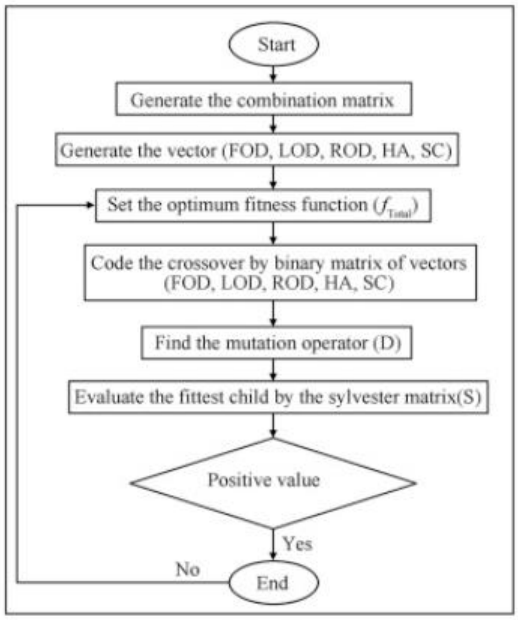


Fig. 4 Flowchart

### 6.3 PointBug Algorithm

PointBug, a newly designed robot navigation system, navigates a robot point in a planar unknown environment populated with stationary objects of any type. It calculates where the next point to go from a starting position to reach the target. The output of the range sensor, which detects a sudden change in distance from the sensor to the nearest obstacle, determines the next point. Inconstant distance readings are defined as rapid changes in range sensor output, whether they are growing or decreasing. It can be from infinity to a specific value, or from a specific value to infinity, or from a specific value to a specific value, as long as the difference value, d, is defined. A sudden point is defined as a reading from a range sensor that detects a difference in range of 1cm or more.

The algorithm can operate in dynamic environment as well because information about the environment can be obtained immediately from the range sensor during the movement of the robot. The performance of the algorithm depends on total sudden points detected. The less number of sudden points detected is better. Thus, whether it outperforms other bug algorithms depends on obstacles in the environment as if the obstacle is a circle, it will produce less sudden point since a circle has no vertex. The total vertex in obstacle affects the total sudden points

To scan the environment, the robot uses a range sensor that rotates at constant speed between zero and three thousand revolutions per minute (rpm). The robot's initial posture is to face directly toward the goal point, and then it spins left or right in search of a new location in the target area. In order to determine the robot's rotation direction after finding the initial sudden point, it is necessary to determine the position of a straight line between the current sudden point and goal point or the dmin line. The robot's rotation is always in the direction of the dmin line's position. This distance is measured every time the robot hits a new abrupt point and is recorded as dmin. At 1800 degrees of rotation, the sensor reading is always ignored so that it doesn't cause the robot to return to its prior abrupt point. As soon as 3600 rotations have been completed, the target is deemed unavailable and the robot stops instantly.

The pseudo code of the algorithm as follows:

```
While Not Target
  If robot rotation <= 360
    Robot rotates right of left according to position of dmin
     If sudden point
        If 180 degree rotation
          Ignore reading /* to avoid robot return to previous point */
        Else
          Get distance from current sudden point to next sudden point
          Get angle of robot rotation
          Move to new point according to distance and rotation
```

angle
Record New dmin value
Reset rotation
End if
End if
Else
Robot Stop /* No sudden point and exit loop */
End if
While end
Robot Stop /* Robot successfully reaches target */

## 7 FLOWCHART & ARCHITECTURE MODEL

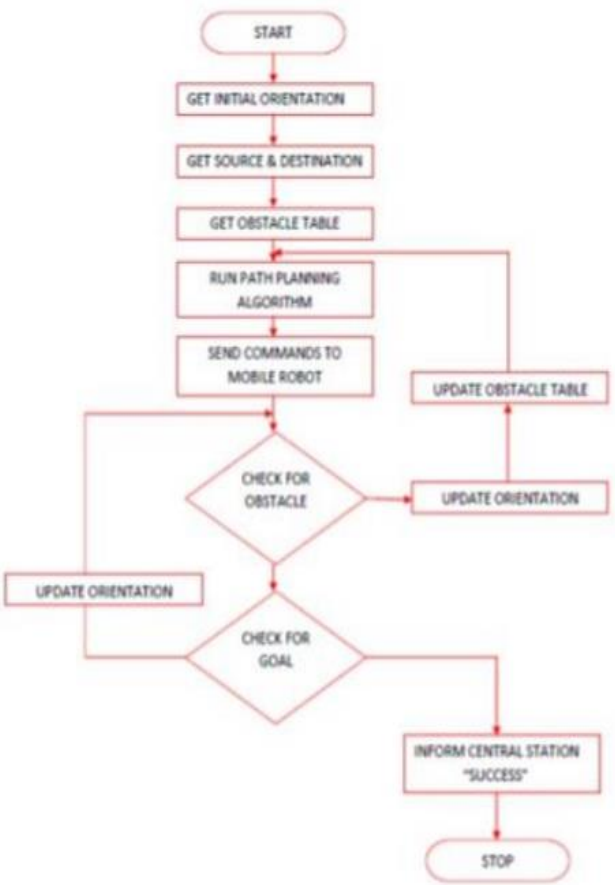


Fig. 5 Flow Chart of Navigation Master

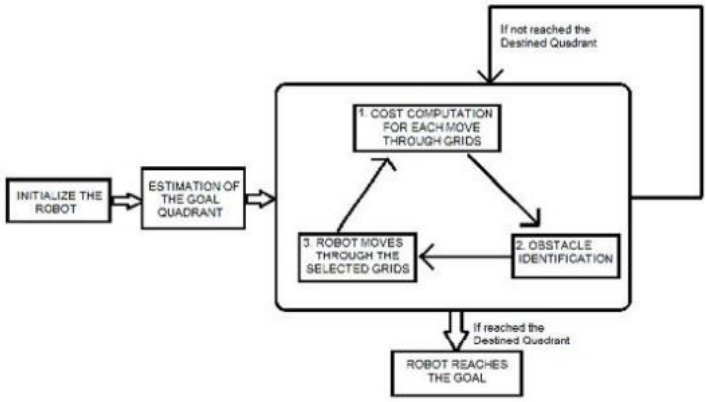


Fig. 6 Architecture Model of Navigation Master

## 8 WORKING MECHANISM

The working of this project mainly includes with finding the shortest path finding for robots and also the time complexity plays an important role in finding the path. The time complexity makes your robot accurate as with the least time and high speed the proposed system can be more efficient. The proposed Time Efficient A* algorithm fetches all nodes but calculates the heuristics function's value only before collision phase. This reduces the processing time so that the robot can perform its work quickly. Also Genetic Algorithm helps in finding the best route updating whenever it got to a new position and recognizing the shortest path to the destination. Firstly, we import out required packages for the project such as "Pygame" which helps in projecting a platform for which we can see the virtual working of the robot. Also we install python3 in prior for better understanding and executing of the scripts.

1. "map.txt" contains the workspace in terms of 1s and 0s where 1 is a free cell and 0 is an occupied cell.
2. Run the below line in terminal
   ***python3 graphics.py***
3. Then we'll be asked to give the dimensions of the map accordingly.
4. Create map with the help of Pygame creates (whites are free spaces, red/selected cells are blocks).
5. Then after creating the map, you are free to set some of the obstacles (blocking some coordinates) in the map. Red/selected blocks will be blocked once you select the obstacle/blocks. After this press Enter for going to next phase.
6. Here, we should give the coordinates of starting point and destination point of robot in the map.
7. Once you click Enter, it automatically takes the shortest path with green line which is the solution path

too.

8. Use pause button to pause in the process of travelling and you can update map using Pygame window and then press enter to continue to travel towards destination. Also while the robot is taking its green colored solution path, we can add some obstacles where the robot itself keep updating at every coordinate and takes another optimal path to the destination. It keeps on updating with the shortest path to destination to each point it travels
9. In map the top-left cell has coordinates (0, 0), moving down increases x-value, moving right increases y-value.

## 9 Conclusion

In this project, we provide a comprehensive overview of the path planning problem, including the various categories of the problem, methodologies used to address it, its complexity, and so on. Many sophisticated algorithms have been presented to address this problem efficiently. These algorithms include a wide range of approaches. Because of this variability, it is difficult for a robotic application designer to select a certain technique for a certain problem. Typically, researchers design and apply a specific algorithm, even though other methods may be more productive. Thereby it can be concluded that the modified A* algorithm is better than the actual A* algorithm in terms of processing time on a little cost of path length and can be applicable for fast processing applications

## References

[1] D. Nakhaeinia, S. Tang, S.M. Noor, O. Motlagh "A review of control architectures for autonomous navigation of mobile robots," Inter-national Journal of Physical Sciences, 6 (2) (2011), pp. 169-174W.-K. Chen, *Linear Networks and Systems.* Belmont, Calif.: Wadsworth, pp. 123-135, 1993. (Book style)

[2] O. Hachour "Path planning of autonomous mobile robot,"
International journal of systems applications, engineering & development, 2 (4) (2008), pp. 178-190K. Elissa, "An Overview of Decision Theory," unpublished. (Unpublished manuscript)

[3] R. Chatila, J.-P. Laumond "Position referencing and consistent world modeling for mobile robots," Robotics and Automation. Proceed-ings. 1985 IEEE International Conference on, 2 (1985), pp. 138-145

[4] Sankaranarayanan, A. and Masuda, 1. (1992). "A New Algorithm for Robot Curve-Following amidst Unknown Obstacles, and a generalization of Maze-Searching," Proceedings of the IEEE International Conference on Robotics and Automation, Nice, France, pp. 2487-2494.

[5] Lumelsky, V. J. (1987). "Algorithmic and complexity issues of car in an uncertain environment," Journal of Complexity, 3:146-182

[6] Kangari, R. (1990). "Automation in construction," Robotics and Autonomous Systems, vol. 6. pp. 327-335

[7] R. C. Arkin, "Navigational path planning for a vision-based mobile robot", *Robotica*, vol. 7, pp. 49-63, 1989.

[8] T. Asano, T. Asano, L. Guibas, J. Hershberger and H. Imai, "Visibility-polygon search and euclidean shortest paths", *Proc. 26th IEEE Symp. Foundations of Comput. Sci.*, pp. 155-164, 1985.

[9] J. L. Crowley, "Navigation for an intelligent mobile robot", *IEEE J. Robotics Automat.*, vol. RA-1, pp. 31-41, 1985.

[10] Cousineau, L and Miura, N. (1998). Construction Robots: The Search for New Building Technology in Japan, ASCE PRESS, Reston, Virginia.

[11] Kamon, 1. and Rivlin, E. (1997). "Sensory-Based Motion Planning with Global Proofs," IEEE Transactions on Robotics and Automation, 13(6):814-822.

[12] E. W. Dijkstra, "A note on two problems in connection with graphs", *Numer. Math.*, vol. 1, pp. 269-271, 1959.

[13] K. Kant and S. W. Zucker, "Toward efficient trajectory planning: The path-velocity decomposition", *Int. J. Robotics Res.*, vol. 5, no. 3, pp. 72-89, 1986.

[14] J. S. Singh and M. D. Wagh, "Robot path planning using intersecting convex shapes: Analysis and simulation", *IEEE J. Robotics Automat.*, vol. RA-3, pp. 101-108, 1987.

[15] E. Welzl, "Constructing the visibility graph for n line segments in o(n2) time", *Inform. Process. Lett.*, vol. 20, pp. 167-171, 1985.

[16] K. Fujimura and H. Samet, "A hierarchical strategy for path planning among moving obstacles", *IEEE Trans. Robotics Automat.*, vol. 5, pp. 61-69, 1989.

[17] S. K. Dambhampati and L. S. Davis, "Multiresolution path planning for mobile robots", *IEEE Trans. Robotics Automat.*, vol. RA-2, pp. 135-145, 1986.