A TWO STAGE, THREE STEP APPROACH TO MOBILE ROBOT PATH PLANNING AND CONTROL

by

JOHN ROBERT KONDERLA JR.

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2013

ACKNOWLEDGEMENTS

I dedicate this work to my wife Babra. Without her sacrifices and encouragement throughout these many years, this thesis would not have been possible. It is as much her accomplishment as mine. I will always be truly grateful. I also thank my sons James and Joseph for all the things we couldn't do together over the years because of school. Dedications such as these are often mentioned in works but only authors and their families truly understand the great sacrifices involved.

I would like to thank all the professors and staff at the University of Texas at Arlington that have helped me since I first started toward my undergraduate degree 14 years ago. Among these, two stand out. Dr. Frank Lewis's teaching style is both informative and engaging. He has the rare combination of ability and experience that allows him to present both detailed theory and application in his courses. This is a combination that is often difficult to find in academia but critical to the training of engineers. Dr. Jason Losh also has these qualities. His knowledge of microprocessors and microcontrollers combined with his challenging class projects have given me the confidence to experiment on my own.

I would also like to recognize my parents, John and Marynell. Among their countless gifts, I could not have started this path without the seed money for my first semester. I cannot imagine a better father-in-law than James Moore. His pride in me has been a great inspiration. My brothers and sisters through blood or marriage have each inspired and motivated me in their own ways.

Finally, my mother-in-law Joyce Moore's love of family has been my greatest aspiration. May we someday meet again when my time here is done. Rest in peace and love.

October 29, 2013

ABSTRACT

A TWO STAGE, THREE STEP APPROACH TO MOBILE

ROBOT PATH PLANNING AND CONTROL

John R. Konderla Jr., M.S.

The University of Texas at Arlington, 2013

Supervising Professor: Frank L. Lewis

The potential field control model provides a powerful, yet mathematically simple method to motivate a mobile robot. It is a local control method based on simple physical principles. This allows the robot to proceed toward a target while avoiding obstacles following a naturally smooth path. It ensures the robot will behave according to Newtonian principles that have been studied for centuries. The potential field model has several weaknesses however. Most of the weaknesses can be lessened through careful modification of the potential field but the success of this approach is limited. For further improvements, global methods must be employed. These methods use previous knowledge of the environment to find a safe and efficient path.

This thesis demonstrates that the strategic addition of a negative velocity feedback to the original potential field model will reduce or eliminate oscillations and eliminate the stall issue. It also shows that the proper approach to the use and placement of “false” obstacles improves operation further with little overhead. From there, it examines particle swarm, genetic algorithm, ant colony global and MAKLINK approaches and their combinations. Finally, it proposes a combined approach to implement on large or resource-sensitive robots.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure Page

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Robot Path Planning

Since the advent of mobile robots, a prime concern has been how to drive the robot from a starting point to a target or goal while avoiding obstacles. The approaches are divided into global approaches or local approaches. A global approach requires fairly detailed prior information on the environment. A local approach, on the other hand, only requires knowledge of the target position relative to its own in advance. The prior knowledge requirement makes global path planning approaches intractable while local approaches fall into the heuristic category as described in [3]. Global approaches generally require extensive calculations to find a suitable path through the environment as mentioned in [6]. As many of these calculations as possible are usually conducted offline to reduce real-time processing requirements. Local path planning relies on the robot's sensor information to determine its immediate environment status. As such, it allows the robot to react to changing environmental conditions as they occur. This is essential in a dynamic environment.

1.2 Technologies for Path Planning

How does one get a robot to reach a target while avoiding obstacles in its environment? This simple problem statement has actually proved very difficult to solve. In fact, a number of technologies have been applied individually and in combination to solve it. Some of the early research employed wall following, visibility graphs and Voronoi diagrams according to [7]. Later methods mentioned by [7] included genetic algorithms, neural networks and fuzzy logic. Some other attempts include exact cell decomposition, approximate cell decomposition, retraction on a network of one dimensional curves, simulated annealing and road mapping [3] [6] [10]. Most of these attempts are global methods. Many have great computational requirements even when

suitable for local use. Computational requirements are acceptable for large, expensive and slow robots in simple environments. Such robots can carry whatever processing equipment is required to perform the computations and the power sources necessary to power it. They can afford the time required to perform advanced calculations. The cost of advanced computers and their power systems are easily absorbed by the project. For small, fast or inexpensive robots, these limitations are major drawbacks. For them, a simple, versatile model that requires little onboard resources is required. Such a model will even free up resources on larger robots for other tasks.

1.3 The Artificial Potential Field

In 1985, Oussama Khatib developed the artificial potential field solution to the general problem of robot movement to achieve a goal while avoiding collisions [3] [4] [6] [7] [9]. His paper [12] showed how a robot manipulator could use simulated physical fields to safely operate a complex robotic arm at a low level using a relatively simple mathematical model. This had previously been considered a high-level path planning problem requiring advanced movement algorithms. The problem was that the high level control took too much time. His intent was not to replace the high level control but to supplement it. Because of its simplicity and elegance, it was applied to the mobile robot problem as well. In general, it is considered a local path planning method.

The various potential field models categorize places and objects in the environment as either targets or obstacles. Targets emit an attractive force. Obstacles emit a repulsive force. A robot's position relative to a force emitter defines its potential level with respect to that point. The artificial forces and their potentials are additive, just as with natural forces. When mapped over the relevant environment, the potentials create a potential field. The force on the robot is the negative gradient of the potential field. In their simplest forms, the attractive fields assume a simple, downward conical shape centered on their point sources as in [1] and [13]. This allows it to direct the robot toward the target wherever the robot is located in the environment. The

repulsive fields are proportional to the inverse square of the distance from their point sources [2] [3] [4] [5] [6] [12] [13]. This creates a localized, exponential effect on the robot to prevent it from hitting the obstacle at its center. These are the models that will be used in this thesis.

1.4 Potential Field Issues

Even though the potential field model solves the robot path planning problem simply and effectively in most situations, it does have some issues. First, there is a tendency for the robot to oscillate in certain instances when it is close to obstacles, close to targets or in narrow passages [1] [7] [13]. The greatest problems though are related to local minima in the potential field. In fact, the problem is significant enough that [1] concluded that it is better to drop the potential field model and adopt a different approach. The most common issue is generally referred to as a trap. It is a result of the repulsive forces and attractive forces in an area canceling each other [1] [7] [10] [11] [13]. The trap typically manifests itself in one of two ways. First, the robot may travel down a blind alley because of the force from a target on the other side. Once it reaches the end, the local force from the wall cancels the global force from the target and the robot stops with no force to drive it elsewhere. The second way is not as obvious. Here, the robot is drawn toward a target and it must pass between two close obstacles or proceed down a narrow passageway to get to the target. Even if the walls or obstacles are far enough apart for the robot to physically fit, the fact they are close together can result in the forces creating local minima. The result is the robot stalls with no apparent reason. Although the trap and stall issues are often treated as the same in the literature, this thesis takes the position that they are actually different and thus can be solved using different modifications to the standard potential field model.

There are two additional local minima issues with the standard model that have been identified in the literature. Unlike the stall and trap, they have been separated and addressed independently. First, there is the GNRON which stands for goals non-reachable with obstacles nearby [7] [8]. This problem manifests itself when an obstacle is sufficiently close to the target

that the local, which is also the global, minima of their combined force, is pushed away from the target. Since the effect of obstacles is local, the displacement is typically not very great but still sufficient to prevent the robot from reaching an acceptable proximity to its goal. The SAROG or symmetrically aligned robot-obstacle-goal problem is caused by a special case alignment of the robot, its target and one or more obstacles in its path [11]. As will be discussed in the next section, a common method to get a robot out of a trap is to create a new force. This is usually placed at the local minima. If an obstacle is directly between the robot and the goal, any force that pushes the robot straight back simply makes it stall further away from the goal where the forces again cancel. The solution presented in this thesis solves both of these scenarios.

1.5 Local Minima Solutions

There have been several methods developed to solve the local minima issue. Several of these involve generating one or more obstacle forces at, or near the local minima [4] [6] [10]. These can be temporary or permanent. In [4], the obstacle takes on the shape of an inverted target and thus makes the effect global. To prevent instability in the robot, the maximum of the force is limited. [6] and [10] place a standard obstacle at the stall point if the robot stops away from the goal. [6] is a fuzzy logic approach. This makes the solution more complex. Caution must be used with these approaches as they can result in large forces on the robot which would cause it to suddenly accelerate. [11] generates a random force to propel the robot away from the trap area.

[3] and [10] redefine the obstacle fields through simulated annealing to reduce the chance of a trap in the first place. This involves an iterative loop that requires a lot of calculation time. While it helps in many situations, the tradeoff of additional complexity does not guaranty success. In such cases, they have to fall back on different search and escape routines. [5] also tries to avoid traps. It does so by considering the robot's distance to target in two different obstacle force equations. This greatly compounds the math involved. [1] changes target location to a reachable one if the target is hidden. It continues to change the location until it can reach

the target. [2] uses a wall following technique to reach the target if a trap is encountered. [7] complicates the potential field model greatly, first by introducing a repulsive potential density field for complex obstacle shapes. It then varies the factors of an already complex formula to avoid U-traps. Finally, it creates an exaggerated target well to eliminate GNRON. All of this takes a lot of processing time. Most of these methods avoid using feedback from the robot except to identify a when a trap exists or an escape has been completed. This creates a system that does not take into account velocity feedback. This thesis will show that the proper use of velocity feedback can remove or reduce oscillations and solve the stall type of trap. It will also show that the proper placement and use of a false obstacle allows a robot to escape a true trap and permanently mark a dangerous area.

1.6 Global Approaches

Even with improvements in the weaknesses of the potential field, there is still no guaranty that the robot will not wander into a trap. The likelihood increases as the complexity of the environment increases. There is also no certainty that the chosen path will be the most efficient one through the environment. Again, the likelihood of an inefficient path increases as the complexity of the environment increases. To solve these issues, some advanced knowledge of the environment is required. This means the use of global path planning in spite of its computational requirements. Three popular methods include particle swarm [25] [36] [41] [49], genetic algorithm [29] [30] [32] [34] [38] [41] and ant colony optimization [26] [34] [35] [38] [39] [45] [48]. Particle swarm methods are based on the swarming and flocking behavior of animals, insects and birds. They start with particles going in different directions following a small set of rules in search of a path to the goal. Once a path is found, the particles continuously adjust themselves based on personal and global best positions to find agreement on the optimal path. Genetic algorithms are based on evolutionary theory. They create a set of paths by heuristically selecting path points in the environment between the start and goal. The number of paths is increased through crossover and mutation operators among others. The viable paths, ones that

do not cross obstacles, are then selected and ranked according to the length of the path and in some cases, the angle changes along the path. The process is repeated through a number of cycles or until the best path remains the same for a selected number of generations. Ant colony methods are modeled after ant behavior as the name suggests. They start with a generation of ants that heuristically search the environment for the goal from a starting point. After reaching the goal, the points in the ants' paths are covered with a pheromone level inversely proportional to the path length of each ant that passed the point. The next generation of ants heuristically seeks new routes from the start to the goal based on the desire of the ant to choose the point with greatest pheromone level verses the desire of the ant to find a new path. Generally, the probability is shifted as the generations progress toward choosing the highest pheromone level. This allows path diversity in the beginning and convergence as time progresses. Between each generation, the pheromone is decreased by a certain percentage before new pheromone is added to prevent saturation and allow the less desirable paths found early on to disappear.

1.7 Heuristic Global Approach Weaknesses

In spite of their power, there are weaknesses with these approaches. As stated before, they require a lot of processing resources. Even though each starts with a map of the environment, they do not know the nature of the environment until they go through an exploration phase in the early generations. The first generation presents the worst case scenario as there is no prior work to consider or optimize. This can lead to many long and winding paths between the start and goal or even invalid paths. They are also heuristic in nature which means that they are good at exploring a great number of paths in the environment and optimizing the best. The best however, is the best path they have found. There is no guaranty that the best path that they have found is indeed the best path through the environment.

Particle swarm methods usually start with a set of lines or curves at regularly spaced intervals where the particles are placed to reduce the optimization calculations. The paths are formed from lines joining particles on one line to those on the lines on each side. If the curves

happened to be placed at unfortunate locations, it can be difficult for valid paths to be established from one line to the next. Also, special precautions must be taken to handle environmental situations where the path must double back. In this case, the particles on an affected curve must be prevented from moving to a position that doesn't allow one or more of the required paths to remain viable.

Genetic algorithms generally require a large number of paths in the initial generations in order to find a set of valid paths. If the environment is complex, the number can be extremely large. This is because of the random selection of points to form a path or chromosome. Since these points, and their order of selection, can lead to paths bouncing all over the environment, there is a good chance that the paths will cross one or more obstacles making them invalid even after crossover and mutation.

Ant colony methods are based on some balance between an ant's desire to find a path toward a new food source and its desire to follow a pheromone trail toward an existing food source. In the beginning generation though, there is no pheromone trail in existence to follow. This means its path is almost completely arbitrary and consequently, long. In some cases, the movement selection may be biased by the known direction of the goal but care must be taken in using this approach. In a sufficiently complex environment, such as a maze, this can actually result in a longer path to a solution or even result in the ants getting caught in traps along the way. This leads to an additional issue. In most versions, all the ants from a generation must reach the goal before the next generation starts. Ants that have been caught in a trap may never reach the goal. Special handling is required in such cases.

1.8 Heuristic Global Approach Solutions

Each global approach also has certain strengths. Particle swarm methods use consensus building between particles to determine their best positions. The crossover and mutation operators in genetic algorithm approaches provide very powerful means of modifying successful paths to see if better paths can be created from them. Ant colony approaches allow

continuous paths to be found through exploration and gradual convergence on the best path discovered. These different strengths have led to many researchers combining aspects of different methods to capitalize on their strengths [26] [29] [35] [39] [41] [45] [50]. This process has led to great improvements in the methods. In the end however, they are still heuristic and thus cannot guaranty that every available route has been explored regardless of the number of entities or generations. To solve this, a more deterministic method must be used. The MAKLINK graph (a kind of vertex graph) is a method that was developed in 1991 [46]. It assumes an environment containing polygonal shaped obstacles as do many of the explored methods. It starts with a map and a list of vertices in the environment. It then divides the environment into convex polygons whose free space borders separate adjoining convex polygons. The mid points of these borders are joined creating a graph that allows a robot to go from any starting point to any goal point in free space by following these free link lines. The only travel apart from the free link lines is the travel from the start to its nearest line and to the goal from its nearest line. All routes between obstacles are found using this process but they are coarse in nature and thus not optimum. To solve this, a combination of methods is required.

1.9 Proposed Method

In order to create a path planning solution that will work with small or large robots, this paper proposes a two stage, three step approach. The first stage takes into account the fact that an environment is usually at least partially known in advance. This stage can therefore be run offline or even off-platform to create an initial path. Once the path is created, it can be loaded into the controller of the robot to guide it through the environment via a series of waypoints. The environment may have changed since the map was formed however, so the second stage must be able to react to the environment through local sensors.

The three steps are formed by combining two of the global path processes and the potential field local process. First the map, along with its vertices, is fed to the global processor which creates a MAKLINK graph. This is fed to an ant colony optimizer in place of its initial

solution using an interpreter based loosely on aspects of the particle swarm method. The output is reduced to a relatively small set of waypoints. These steps take part in the first stage. The final step provides the waypoints to a potential field controller in the second stage to provide motivation to the robot's drive train. This stage also receives processed input from the local sensors to allow the potential field controller to react to changes in the environment that result in the path to one or more of the waypoints being blocked.

This method uses the ability of the ant colony approach in selecting and optimizing discovered paths. It supplements this with the MAKLINK graph's ability to locate all the coarse paths around obstacles as well as its improved speed of finding an initial solution set when compared to the ant colony approach. Next it uses the potential field control's natural suitability to the use of local sensors with limited range and its ability to adapt to unknown portions of the environment. Finally, the combination provides an appropriate point to divide the resource intense global path methods from the resource efficient local path method of the potential field controller.

1.10 Summary

This chapter defined a prime concern of robotics as how to move a robot through its environment toward a target without hitting any obstacles. It mentioned several technologies used to solve the problem including the history and use of artificial potential fields. In spite of the power and simplicity of the potential field model, it was found to contain several weaknesses as a result of local minima. A number of existing approaches to solve these was also explored. Each has strengths and weaknesses.

The chapter went on to describe several global path planning methods that are required for efficient solutions as the complexity of the environment grows. Each of these also has strengths and weaknesses which can be improved upon by carefully combining aspects of the different methods. In the end, it proposes a solution that incorporates aspects of several

methods to create an approach that will capitalize on the strength of each and allow the process to be used on robots ranging in size and complexity.

1.11 Contributions

This thesis makes the following contributions:

- Artificial potential field:
    - It uses velocity feedback in the overall force equations to dampen oscillations due to the increased kinetic energy obtained from the forces in the environment.
    - It distinguishes between a trap due to a local minimum where the path is not physically blocked and one resulting from the path being physically blocked. The former is defined as a stall. The latter is defined as a trap.
    - It uses velocity feedback to solve the stall problem.
    - It uses the proper placement of a false obstacle to allow the robot to escape a trap and mark the dangerous area of the environment.
    - The proposed solutions are effected through minor modifications to the original potential field model and create very little overhead.
    - It defines an efficient method of implementation on resource-sensitive systems.
    - As a minor contribution, it shows how emotion based control can be adapted to fit the model.
- Combined approach:
    - It uses an approach that considers the needs of a small robot with limited resources while maintaining the power of approaches with large resource requirements.
    - It provides a method to convert a MAKLINK graph into the first generation paths of an ant colony algorithm.

- It provides a method to determine an appropriate size of environmental grid for an ant colony algorithm based on MAKLINK graph construction results.
- It provides a method to convert the output of the ant colony algorithm to a reduced set of waypoints for the modified potential field system.
- It uses the combination of these approaches to correct for many of their deficiencies and capitalize on their strengths to create a versatile system.
- The proposed method allows a robot system to start with a map of an environment containing polygonal obstacles and a list of its vertices. It uses this information to develop an optimal path through the environment and then guide the robot from the start to the goal while reacting to changes in the environment.

1.12 Thesis Organization

This thesis takes a developmental approach to solving the problem of moving a robot through its environment toward a goal while avoiding obstacles.

Chapter 2 starts with a breakdown of the problem and develops the basic mathematical model of the potential field approach. It adds velocity feedback to eliminate target oscillation. It then analyzes and solves the stall and improves the trap issues. It continues by showing the modified potential field mathematical model. From there, the model is converted to a form better suited for microcontroller or DSP use. Each of these is demonstrated through simulations using Matlab. The chapter ends with a brief discussion of emotion-based control and its adaptation to the proposed model. This chapter encompasses the meat of the thesis.

Chapter 3 examines several approaches to global path planning including the particle swarm method, genetic algorithms and ant colony optimization. It not only looks at the basic theory of each but also several variations of each method. Finally, it looks at an implementation of the MAKLINK graph.

Chapter 4 develops a step-by-step approach to implementing a path planning method that combines the MAKLINK graph, ant colony optimization and the modified potential field. This method allows the robot to start with a polygonal map of the environment, consider all coarse paths through the environment to develop the optimum one and then guide the robot through the environment while reacting to changes.

Chapter 5 was included to address some of the safety concerns related to mobile robots. Industrial, commercial and private robots can present a danger to people and property if they act in an unexpected manner. This concern is even greater in the case of military robots which have the potential of creating an international incident. Military robots present the additional issue of classified information. Cases are presented that illustrate the great impact that a little information can have on national security.

Chapter 6 summarizes the thesis and presents areas for additional research.

Appendix A shows the mathematical development of several ground platforms that the model can control.

CHAPTER 2

POTENTIAL CONTROL

The first question concerning mobile autonomous systems beyond the platform level is how to drive it toward a goal while avoiding obstacles. This can be a very complex problem indeed. First, the obstacles and even the goal may be fixed or moving. Each can even change suddenly during the mission. More precisely, once the goal and any obstacles are detected, the direction and speed of the robot must be updated continuously to find a quick and safe route to achieve the desired goal or target position.

The first step to solving a complex problem is to break it down into simple parts. The resulting solutions can then be combined to solve the overall problem. From an engineering perspective, the ideal solutions are those that can be described mathematically. Not only does this allow easier implementation on processor based systems, it allows the solutions to be better analyzed for potential issues. It is even better if the solutions are related to simple, well-examined physical systems. Fortunately, there exists such a solution to the question regarding the motivation of an autonomous mobile robot.

With a slight revision to the question above, one can ask how to move the robot toward a goal and away from obstacles. Newton discovered that to move an object to or from a fixed point or velocity, one must apply a force. In fact, his theories regarding force have been well vetted and studied in depth. Therefore, an approach that assumes artificial forces toward the goal and away from obstacles should produce similar results to Newton's observations. This is where potential field control comes into play.

Potential field control borrows some ideas from gravitational field theory or electrostatic field theory. Essentially, the field extending from or toward a charge or mass is infinite. In the case of a charged particle, the field can be attractive or repulsive depending on the polarity of

the charge and that of any other charged particle that happens to be in the field. With gravity, the field may always be attractive but it has the benefit of little interaction with other types of fields even once masses are in motion. This makes its effects easier to visualize. Both have in common the fact that the potential energy of two objects, subject to an attractive force, increases as the objects' distance from one another increases. On the other hand, the forces can be repulsive as in the electrostatic model case where the two charges are the same. In this situation, the potential energy decreases as the distance increases. Whether the fields are attractive or repulsive, the individual potential energies and forces are each additive. This is very important as it requires simple math to determine the effects of multiple objects. An undesired aspect of these natural fields is that they happen to vary based on the square of the distance. While it will be demonstrated that this is desirable for some types of objects, it is certainly not desirable for others. With this in mind, some general characteristics of a potential field model can be defined:

- The forces resulting from the potential field can be attractive or repulsive.
- The force effects should extend throughout the environment under consideration
- The force effects should be additive.
- The potential field should be modeled as the energy level of the position of the robot with respect to positions of other objects in its environment.
- The potential should be lowest at the goal and highest at any obstacles.
- Each type of object in the environment is allowed to have a different shape of distance-based force formula.

The shape of the field should remain as mathematically simple as appropriate.

2.1 Basic Potential Field Model

So far, potential and force have been used almost interchangeably but they are actually different. Force is the amount of push or pull an object in the robot's environment exerts on the robot. It is not the same as the real forces in the environment but as far as the model, and

therefore the robot's control system is concerned, it is real. As it has both magnitude based on the distance to the object and direction based on the objects position relative to the robot, it is a vector. The potential at a given point is a scalar related to the magnitude of the total force at that point. The combined potential at each point forms the potential field whose negative gradient is the force field which, in turn, is a field of the force vectors at each point in the environment. Mathematically, the relationship is

$$\vec{F}(\vec{r}) = \vec{F}_{target}(\vec{r}) + \sum_{i=1}^{n} \vec{F}_{obs_i}(\vec{r}) = -\vec{\nabla} V(\vec{r}) \tag{2.1}$$

where V is the potential. For a 2D environment, the force can be further broken down into its x and y coordinates:

$$F_x(x,y) = -\frac{\partial V(x,y)}{\partial x}$$
$$F_y(x,y) = -\frac{\partial V(x,y)}{\partial y} \tag{2.2}$$

As indicated above, the goal or target should affect the robot throughout the environment. A simple method to accomplish this is to have a constant amplitude force vector field whose angle at any location is toward the goal. The force vectors can also be broken down into their x and y coordinates:

$$F_{Gx}(x,y) = K_G \frac{x_G - x}{r_G}$$

$$F_{Gy}(x,y) = K_G \frac{y_G - y}{r_G}$$

where

$$r_G = \sqrt{(x_G - x)^2 + (y_G - y)^2} \qquad (2.3)$$

This produces a constant slope toward the goal in the potential field. It was also pointed out that obstacles should primarily affect the robot when it is in their immediate vicinity and their influence should rapidly drop with distance. To accomplish this with a simple math formula, it can be subjected it to the inverse square of the distance:

$$F_{ix}(x,y) = -K_i \frac{x_i - x}{r_i^{\,2}}$$

$$F_{iy}(x,y) = -K_i \frac{y_i - y}{r_i^{\,2}} \qquad (2.4)$$

The negative sign in front of the constant K_i indicates that it is a repulsive force. To illustrate a potential field, assume there is a rectangular environment. There is a goal and two obstacles in the environment. Using the force formulas above one can plot the force magnitude at each point in the environment to produce a representation of the potential field as illustrated in figure 2.1.

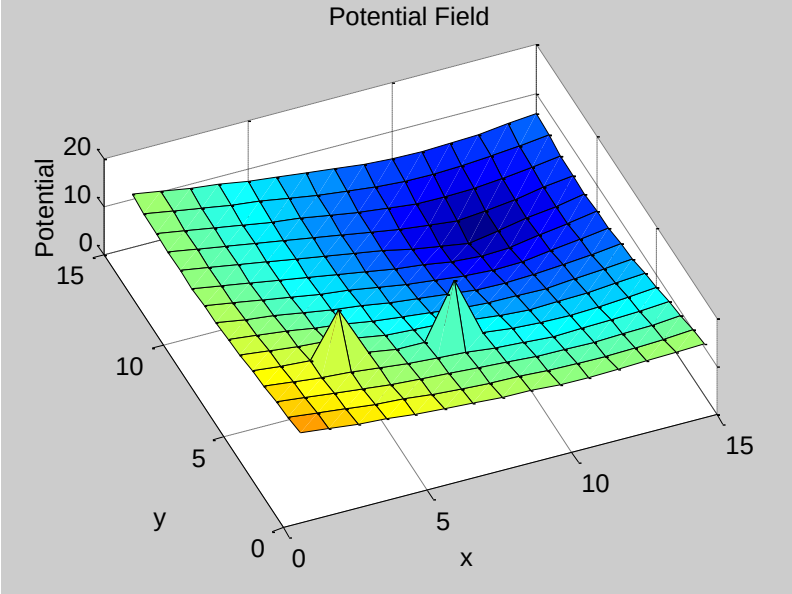


Figure 2.1 Typical Potential Field with Two Obstacles and One Target.

2.2 Target Oscillation Solution

Given a point-source robot starting near the origin and acting solely under the influence of the potential field, one can plot the path of the robot through the environment over a given period of time as illustrated in figure 2.2.

It can be seen that the robot starts out by accelerating toward the goal. Shortly after starting, it approaches the first obstacle. The obstacle's effect on the field is proportional to the inverse square of the distance. This causes the robot to rapidly decelerate and steer away from the obstacle. It then starts to accelerate again. The second obstacle has a similar, though less pronounced effect as its distance is greater. Due to its position with respect to the robot, the second obstacle pushes the course of the robot back toward the goal. It can be seen by the curve of the track that the goal continues to pull the robot toward it. In the end, it misses the goal due to the simulated energy added to the robot by the force from the obstacles. If a longer time frame is considered, it can be seen that the robot will eventually return toward the goal. This is because the shape of the goal's potential field provides a constant force toward the goal throughout the environment. The longer time frame also reveals a problem. Since the model

contains no dampening mechanism, the robot will continue to oscillate around the goal indefinitely. This is particularly visible in the graph of the x and y components over time shown in figure 2.3.

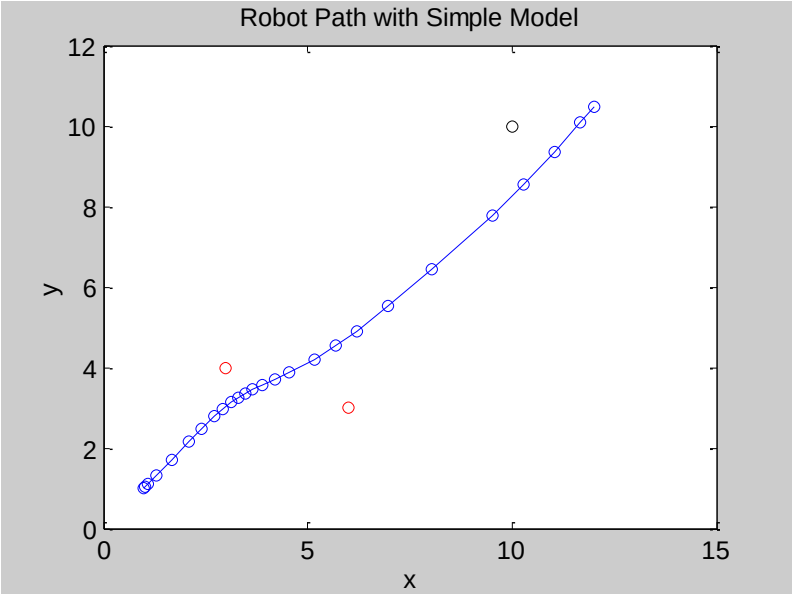


Figure 2.2 Robot Path through a Potential Field.

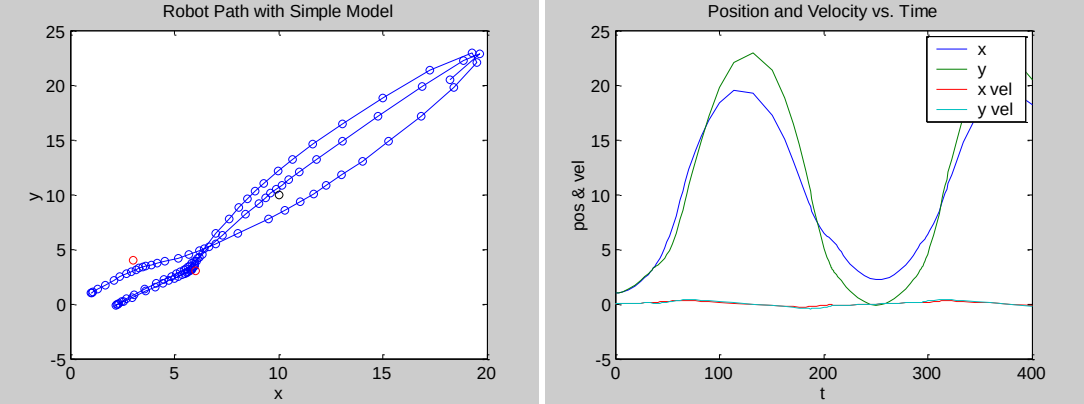


Figure 2.3 Illustration of Un-dampened Path with Position and Velocity Plot.

This can be easily corrected by providing a small negative feedback of the robot's velocity to the system as illustrated in figure 2.4.

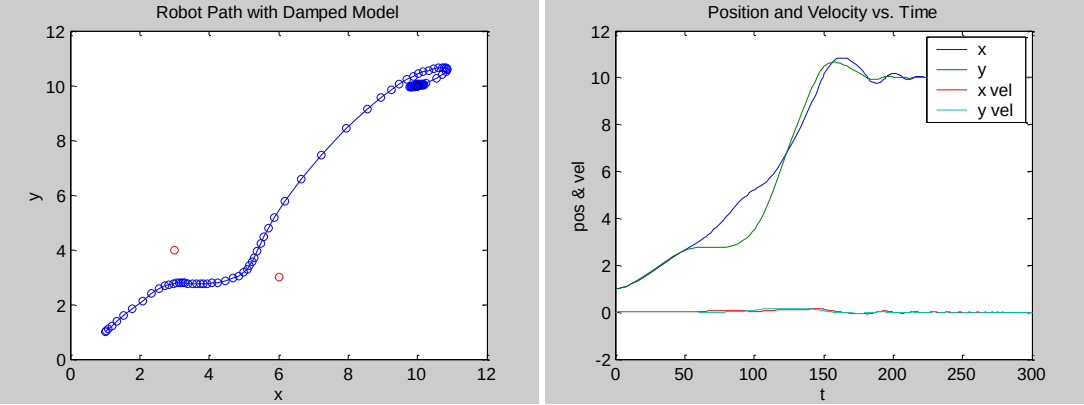


Figure 2.4 Velocity Feedback Removes Oscillation and Allows Robot to Obtain Target.

Mathematically, this can be shown by looking at a simplified system with the robot and the goal. The formula in 2.1 simplifies to:

$$\vec{F}(\vec{r}) = \vec{F}_{target}(\vec{r}) \tag{2.5}$$

The energy of the system is the sum of the kinetic (K) and potential (P) energies of the robot in the environment.

$$L = K + P \tag{2.6}$$

where

$$L = \frac{1}{2}mv^2 + \int Fr\ dr \tag{2.7}$$

In the right side of this equation, Fr indicates that the force operates in a radial direction. Also, this system will continue to transfer energy between potential and kinetic forms indefinitely as shown in figure 2.3. Since dampening is desired, a constant, less than 1, needs to be inserted in the left half of equation (2.7) as (k1) in equation 2.9. This will reduce the transfer of potential energy to kinetic energy in the system. Additionally, the velocity must be found as below:

$$v^2 = 2as = 2\frac{F}{m}s \tag{2.8}$$

where s is the distance traveled. With all this in mind, and solving the right hand side of equation (2.7), L becomes:

$$L = k_1 Fs + \frac{1}{2} Fr^2 \tag{2.9}$$

but

$$s = r_0 - r \tag{2.10}$$

where r_0 is the initial radial distance from the goal. Replacing s in (2.9) gives:

$$\begin{aligned} L &= k_1 F(r_0 - r) + \frac{1}{2} Fr^2 \\ &= \frac{1}{2} Fr^2 - k_1 Fr + k_1 Fr_0 \end{aligned} \tag{2.11}$$

In this, k_1 and r_0 are all constants which can be combined as:

$$k_2 = 2k_1 r_0 \tag{2.12}$$

Now L becomes:

$$\begin{aligned} L &= \frac{1}{2} Fr^2 - k_1 Fr + \frac{1}{2} k_2 F \\ &= \frac{1}{2} F(r^2 - 2k_1 r + k_2) \end{aligned} \tag{2.13}$$

This gives the characteristic equation (c) of L with respect to r:

$$c = r^2 - 2k_1 r + k_2 \tag{2.14}$$

which is the same characteristic equation for the common physical equation:

$$x = at^2 - \beta vt + x_0 \tag{2.15}$$

Other works in control theory have shown that the careful selection of β (k_1 in (2.14)) in such equations will result in a properly damped system. In the final solution in section 2.5, k_1 will be K_{v2}.

2.3 Stall Solution

One issue that arises with potential control results from the presence of local minima on the field. This can be due to the lack of a wide enough space for the robot between two objects or a simple combination of forces in the environment. Either can cause the robot to stall. An example of this is shown in figure 2.5.

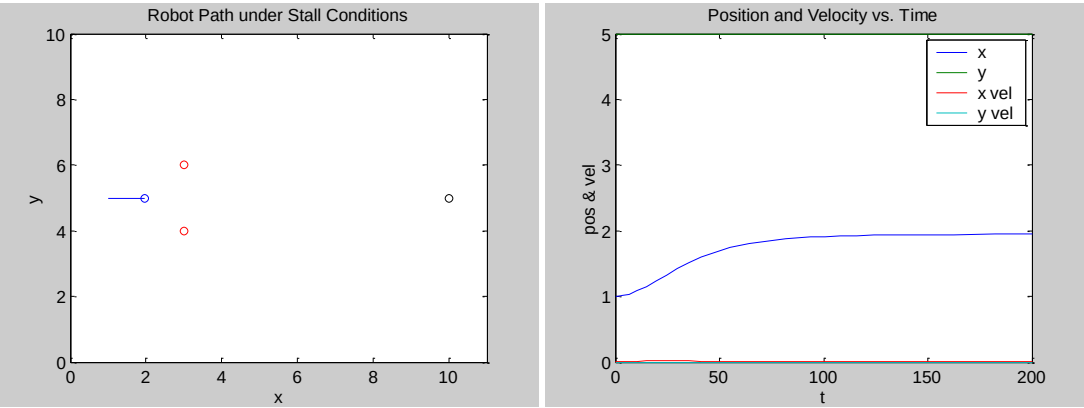


Figure 2.5 Illustration of Stall Problem with Position and Velocity Plot.

Here it is clear that the robot starts out fine but as it approaches the line intersecting the two obstacles, it stalls. The cross section of the force along the robot's path shows a very pronounced local minima followed by a local maxima in the force (figure 2.6).

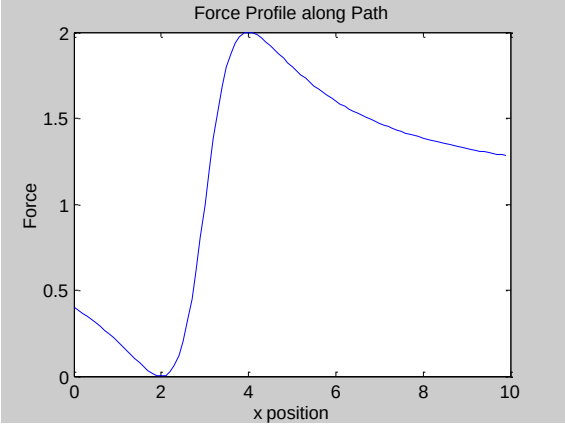
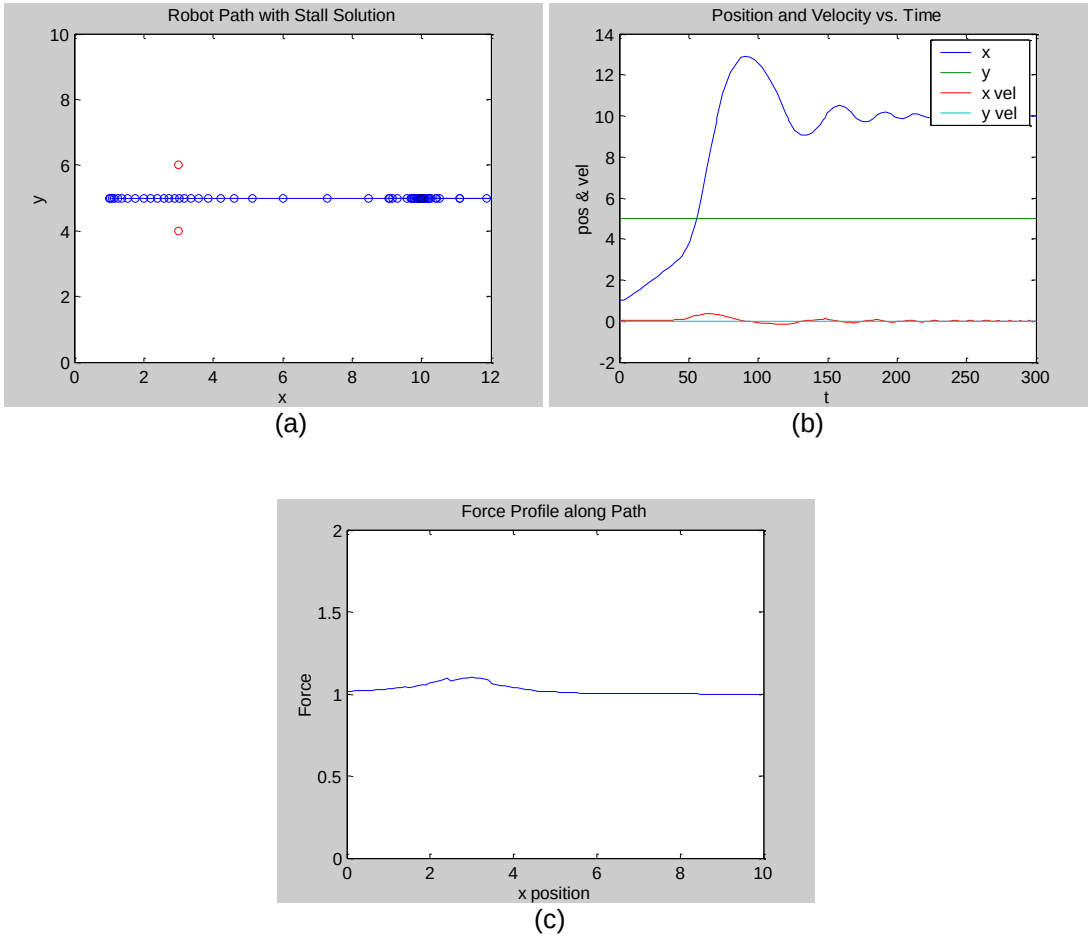


Figure 2.6 Cross Section of Force along Robot's Path.

In order to solve the stall problem, one can again look to a real-life situation. While driving on a multilane freeway with little traffic, a driver can comfortably travel at the posted speed limit. If the freeway suddenly narrows by one or more lanes, the driver will naturally slow down. This phenomenon occurs even with no other vehicles on the road. The reason for this is simple. Under normal conditions, a wide road allows plenty of reaction time to adjust the vehicle's direction and speed. These adjustments are required to counter normal path variances that could result in a collision with surrounding stationary obstacles or changing environmental conditions such as moving obstacles entering the freeway that could intersect the vehicle's path. The moving obstacles can be additional vehicles entering the freeway, animals suddenly running across the freeway or any other obstacle that may need to be considered in choosing velocity vector corrections for the driver's vehicle. When the width of the path shrinks, the reaction time is reduced proportionately to the change in width so long as the velocity remains unchanged. This results in the driver reducing the vehicle's speed so an appropriate reaction time can be maintained. The same situation can be seen in the case of a person pushing a dolly across a room. If the room is empty, he or she can cross relatively quickly. As objects are placed near the path, the person will lower his or her speed to avoid hitting the objects. As the distance between the objects and the path decreases, the person's speed also decreases.

In each of these scenarios one can see that both initial velocity and obstacle distance are considered to maintain sufficient reaction time while target distance is not considered. The potential field model discussed thus far factors in obstacle distance but does not consider robot velocity in adjusting its output velocity. Since the robot's velocity change is a function of the forces on it one can solve the stall problem with the model by simply multiplying the summation of the obstacle forces by K_{v1} times the velocity. Here, K_{v1} is used to determine how much of an impact the velocity should have. At first glance, it would appear that this solution would be problematic as the impact of all obstacles is affected. This is not the case however. While the



(a) (b) (c)

Figure 2.7 Illustration of Stall Solution: (a) Robot Path, (b) Position and Velocity Plot and (c) New Force Cross Section.

impact of all obstacles is affected, only relatively close obstacles impact the robot's path under the current model. This is because the impact to the shape of the potential field due to the obstacles is an inverse square function of the distance. The mathematical proof is therefore similar to the one in section 2.2. The result of this modification to the stall problem scenario is shown in figure 2.7. One can also see the impact of the value of K_{v1} to the original scenario in figure 2.8.

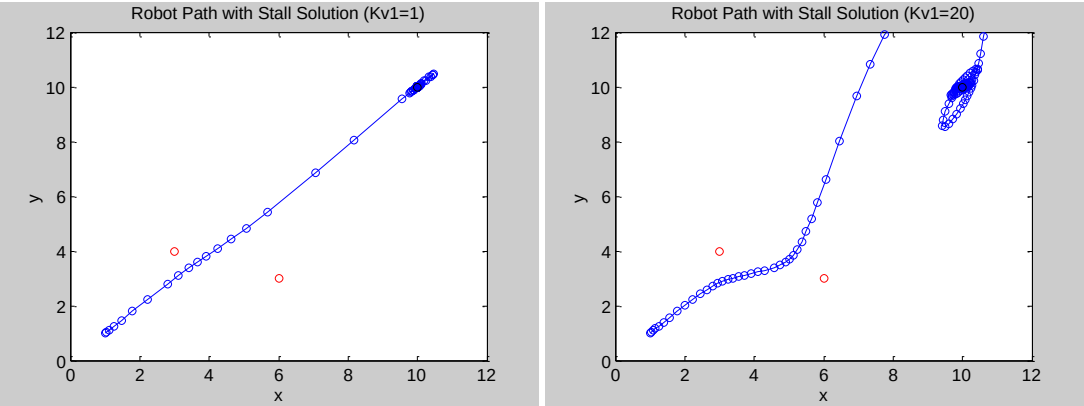


Figure 2.8 Impact of K_{v1} on Robot Path.

It can be seen in the left plot of the robot's course that the lower value results in a more direct path to the target while a moderate value in the right plot results in a safer path around the obstacles. In the extreme cases, if K_{v1} goes to zero, the obstacles will be ignored resulting in collisions while the robot will freeze or even move away from the target if K_{v1} is extremely high.

2.4 Trap Solution

Thus far, the potential field model has assumed a point source robot and obstacles. In reality, this is certainly not the case. Instead each may have a unique complex shape. The typical and mathematically simple approach is to assume a detected obstacle is circular with a radius sufficient to completely enclose it and account for any uncertainty in the robot's obstacle distance measuring equipment. This is similar to how military aircraft, manned or unmanned, handle enemy radar sites. They maintain a minimum distance based on the assumed detection

range of the radar system for the aircraft type. Since the distance is assumed to be omnidirectional, the range forms a circle around the radar site. Mathematically, this represents a slight modification to the x and y obstacle force equations.

$$F_{ix}(x,y) = -K_i \frac{x_i - x}{(r_i - a)^2}$$
$$F_{iy}(x,y) = -K_i \frac{y_i - y}{(r_i - a)^2} \quad (2.16)$$

In this case, *a* is the assumed radius of the obstacle. It effectively moves the inverse distance squared effect out to the edge of the obstacle instead of its center. This solution is very versatile and effective while adding little complexity to the model. There is one potential issue in that *a* either has to be based on the largest obstacle encountered or it must assume different values based on each obstacle's classification. In the latter case, *a* in the above equations should be a_i instead. While this doesn't complicate the model itself, it does mean the robot will need to have the means to distinguish between different obstacles and store the various values of a_i. On the other hand, the purpose of *a* is actually to allow sufficient clearance for the robot so that it will not hit the obstacles. For most robots and vehicles, the required clearance remains fixed. Given this, one can redefine *a* to be the required clearance of the robot or vehicle. The result mathematically as well as functionally is the same but only one value needs to be tracked. To be sure, there may be some instances where a certain safe distance must be maintained as in the enemy radar scenario. For these special situations, *a* is equal to the sum of the robot clearance and the obstacle clearance radii. As with the other two definitions of *a* the result has the same desired effect. In this situation though, it may be preferred to use waypoints instead. The waypoints would consist of temporary target location inputs to the potential field system from a higher level unit such as a mission planner. For the purposes of this paper, it is assumed that the robot clearance (a rather than ai) is the definition used unless otherwise stated.

This improved model of the obstacle force function provides a greater assurance that the robot will not bump into obstacles in the environment but reveals an issue so closely related to the

stall problem that the two are often treated as the same. In the prior section, it was shown that the stall problem could be solved by multiplying the total obstacle force by the robot's velocity and a related impact factor, K_{v1}. When the robot model consisted of a point source, this was sufficient to assure the robot did not get stuck between two or more obstacles. The revised model accounts for the robot's clearance requirements due to its physical size as well as maneuvering requirements. This means it is possible for it to become stuck or trapped between two or more objects that are closer together than the clearance radius requires. The effect is

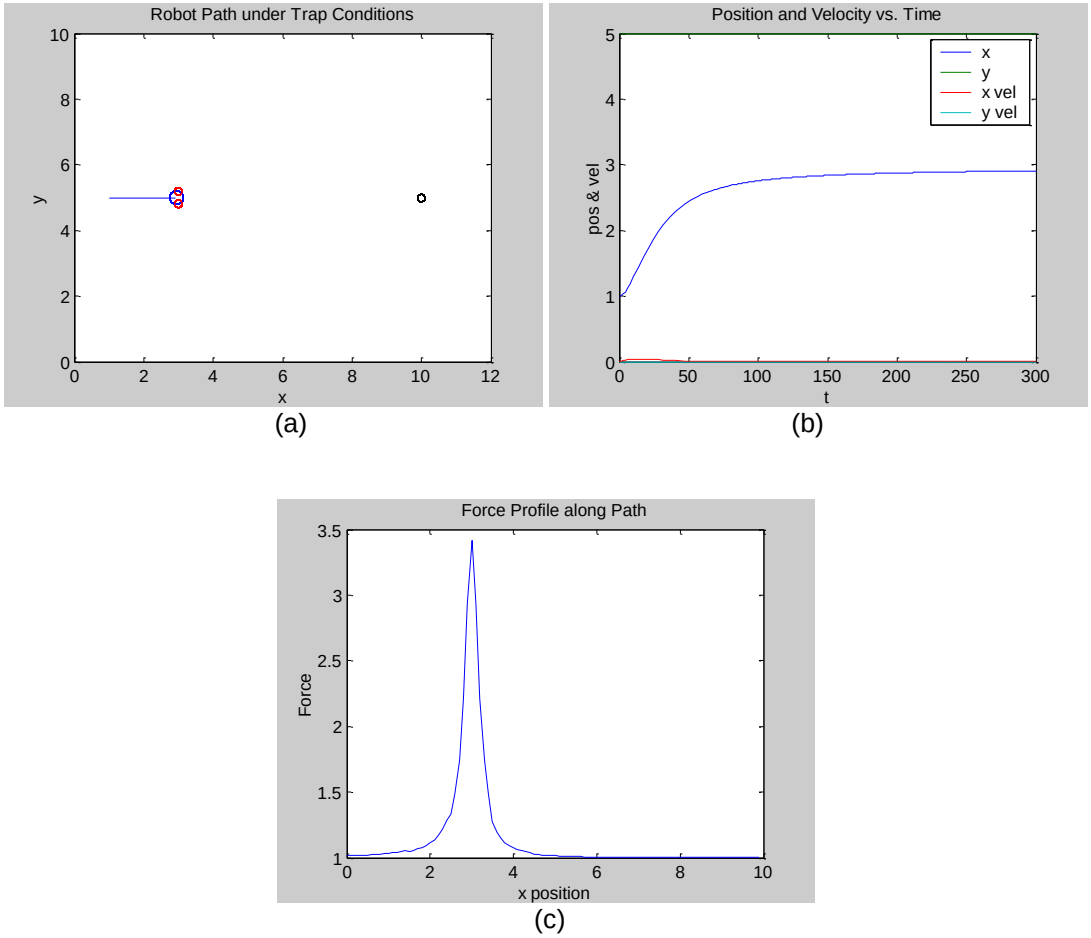


Figure 2.9 Illustration of Trap Problem: (a) Robot Path, (b) Position and Velocity Plot and (c) Force Cross Section.

similar to the stall problem but it is due to environmental limitations rather than an undesired null point in the potential field model. An illustration of the trap problem is shown in figure 2.9.

Here it can be seen that the stall solution causes the robot to continue to decrease the force of the obstacles until the clearance radius of the robot almost touches the obstacles which are at the center of the two obstacle circles. It never reaches the obstacles because the velocity approaches zero as the distance approaches zero. Meanwhile, the forces at the obstacles' positions remain at infinity or a selected limit.

Since the trap problem results from an environmental issue rather than an inadequacy in the model, it presents two problems. First, a way must be found to detect the occurrence of the trap. Second, a solution must be found and implemented to get out of the trap. For the detection of the problem, one cannot simply look at the velocity since the robot will also stop when it reaches the target. To distinguish between a trap and the acquisition of the target, one can turn to Newton. The fact that the velocity of an object will remain unchanged unless acted on by an outside force is the clue. In order for the robot to reach its goal, it must have some velocity or a net force to induce a velocity. This being the case, both must be considered. In a trap, the velocity is decreasing at the same time the net force on the robot is decreasing. This means the system is approaching both a situation of no velocity and no force to change the velocity. This is similar to the stall issue but with the stall issue's solution, the decreasing velocity resulting in the decreasing obstacle force kept the net force on the robot and the velocity from both reaching zero at the same time. Thus to detect a trap situation, the robot must see if the magnitudes of the force and the velocity fall below a minimum at the same time. If they do, either a trap has occurred or the target has been reached. To distinguish between the two, one simply must look at either the distance to the target or the force of the target. If either has fallen below a certain limit, a trap has not occurred otherwise, it has.

Once a trap has been detected, the robot must get out of the situation. Since both the magnitudes of the velocity and force are at or near zero, the simplest way is to introduce a new

force on the robot. Unlike the forces due to obstacles, targets and other items of importance in the environment, this force is completely artificial since it is generated by the robot as a result of a specific condition that should be avoided. While the condition is related to the environment, it is not due to a specific characteristic of the environment. Instead, it is due to the unintended interaction of the robot with the elements in the environment. The force should also be local since the condition is local. Further, it should push the robot away from the area. This makes the shape of the obstacle potential field appropriate. As such, it will be considered a false or fake obstacle. Unlike the other obstacles it should not depend on the robot's velocity for two reasons. First, if it did depend on velocity, it would have as little an effect as the other obstacles once the trap has occurred. Second, the existing obstacles have demonstrated that the area represents a danger that the stall solution cannot solve and thus, it would be better to avoid the area in the future. This means that the model should handle false obstacles like real obstacles but separately since they will not be subject to velocity considerations.

Now that the determination has been made that a new force should be used and the nature of that force has been determined, a position must be selected for it. To do this, the situation leading to the trap needs to be examined again in light of the current model. First, the robot is trapped because the force toward the target is cancelled by the combined force of the obstacles. Second, this could not be solved by the stall solution because the way is physically blocked by obstacles. This means the most logical direction to travel is backwards or away from the target hence the false force should be placed between the robot and the target. This will cause the robot to move away from the trap but it will soon find itself in another situation where the forces cancel, another trap. This is known as the SAROG (Symmetrically Aligned Robot-Obstacle-Goal) problem. It can be solved if there is at least some force perpendicular to the line between the robot and the target. The perpendicular force, even if it is small, will cause the robot to not only move away from the trap but also to steer around it as the combined force will

no longer be completely cancelled. So, instead of the false target being directly in line with the target, the angle with respect to the robot should be toward the target plus some small angle ε_a.

With the angle determined, the distance must be selected. It has been determined that false obstacles are to be treated mathematically like real obstacles by the model. Therefore, their force is dependent on the distance between the obstacle and the clearance radius of the robot. This means it makes sense to place the false obstacle at or beyond the clearance radius. Since placing it at the clearance radius will cause a divide by zero problem, the obstacle must be placed beyond the clearance radius. Too far beyond will make its effect negligible, too close will cause the immediate force to be huge. Considering these limits, the distance of the false obstacle from the robot should be sufficiently greater than the clearance radius to produce the maximum desired acceleration from the resulting force. The following graph (figure 2.10) shows the effect of adding the trap solution to the potential field model.

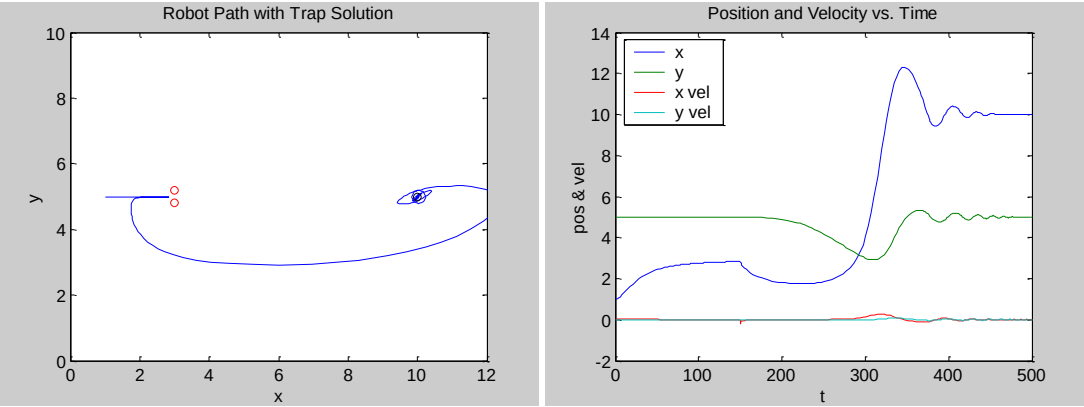


Figure 2.10 Illustration of Trap Solution with Position and Velocity Plot.

To avoid the GNRON (Goals Non-Reachable with Obstacles Nearby) issue, the same check of the force, velocity and proximity to the target is used. In this case though, if the distance is less than the minimum K_{v3} is zero. Otherwise it is one.

2.5 Improved Mathematical Model

The focus of this section so far has been the platform independent motivation of an autonomous mobile robot. It started by referring to Newton's discoveries to develop a simple mathematical model that would determine the desired direction and velocity based on simulated physical properties. This potential field model consisted of the following set of equations:

$$\vec{F}(\vec{r}) = \vec{F}_{target}(\vec{r}) + \sum_{i=1}^{n} \vec{F}_{obs_i}(\vec{r}) = -\vec{\nabla} V(\vec{r}) \tag{2.1}$$

where:

$$F_{Gx}(x,y) = K_G \frac{x_G - x}{r_G}$$

$$F_{Gy}(x,y) = K_G \frac{y_G - y}{r_G} \tag{2.3}$$

$$F_{ix}(x,y) = -K_i \frac{x_i - x}{r_i^{\,2}}$$

$$F_{iy}(x,y) = -K_i \frac{y_i - y}{r_i^{\,2}} \tag{2.4}$$

and the r_G and r_i in these equations are:

$$r_k = \sqrt{(x_k - x)^2 + (y_k - y)^2}$$

In these equations *x* and *y* are the current position of the robot. This model proved to be very good at guiding the robot toward the goal or target while avoiding obstacles in the environment.

Upon further examination, it was found that the model, while good, was lacking in certain areas. First, it was discovered that there was no dampening force in the model so the robot tended to oscillate around the target. This was corrected by including a small negative velocity feedback in the dynamics. It was also discovered that the robot could stall due to a local minimum in the potential field. The solution to this was to make the force of the obstacles

dependent upon the velocity of the robot and a velocity factor. The solution also allowed the model to capitalize on the tradeoff between a safer path and a shorter path to the target. Even with these improvements, the model was still based on a point source robot with no size or clearance requirements. The clearance requirements proved simple to implement by adjusting the force equations. The solution however, revealed a problem seen in real-life systems and closely related to the stall problem. The robot can become trapped by obstacles that are too close together and block its path to the target. It was revealed that this isn't a property inherent in the model but a property of the local environment that the model can solve if the condition can be detected. To detect the condition, the velocity of and force on the robot are monitored and when both fall below a minimum level, the robot creates a new imaginary or "false obstacle" that is not subject to the velocity constraints of the real obstacles. The obstacle is placed slightly off axis to the line between the robot and the target and beyond the clearance radius of the robot. This caused the robot to move away from the trap and at an angle thus allowing the robot to go around the problem area and head toward the target or goal once again. These solutions resulted in the following comprehensive mathematical model equations:

$$\vec{F}(\vec{r}) = \vec{F}_{target}(\vec{r}) + K_{v1} \cdot \vec{v} \cdot \sum_{i=1}^{n} \vec{F}_{obs_i}(\vec{r}_i) + K_{v3} \cdot \sum_{j=1}^{n} \vec{F}_{fobs_j}(\vec{r}_j) - K_{v2} \cdot \vec{v} = -\vec{\nabla} V(\vec{r}) \tag{2.17}$$

where:

$$F_{Gx}(x,y) = K_G \frac{x_G - x}{r_G}$$
$$F_{Gy}(x,y) = K_G \frac{y_G - y}{r_G} \tag{2.18}$$

$$F_{ix}(x,y) = -K_i \frac{x_i - x}{(r_i - a)^2}$$
$$F_{iy}(x,y) = -K_i \frac{y_i - y}{(r_i - a)^2} \tag{2.19}$$

and the r_G and r_i in these equations is:

$$r_k = \sqrt{(x_k - x)^2 + (y_k - y)^2}$$

As a note, the robot clearance radius is not required for the target or goal force calculation since it is desired that the robot end up centered on the target as opposed to touching it.

The comprehensive potential field model that has been developed accounts for the major issues encountered in the motivation of a robot toward a goal while avoiding obstacles. Since this is based on a force calculation being fed to the robot's platform, unseen forces in the environment can cancel the calculated force and result in an environmental stall. For instance, a hill or valley in the environment will cause forces on the robot due to gravity. A soft or rough surface such as sand or rocks can produce other forces including drag. The environmental stall has similarities to both the stall and trap problems solved earlier in this paper. Like the stall problem, the combination of calculated and environmental forces results in a local null. Like the trap problem, the source is environmental. The solution can be easily achieved by redefining the output of the potential field model. Force is mass times acceleration. If the mass is assumed to be one, then the force is the same as the acceleration. Substituting acceleration for each force in the equations changes the output of the model to acceleration. This can be fed to the platform instead. The platform then is responsible for achieving the desired acceleration instead of force. This results in the drive system of the platform overcoming these forces automatically. As for the platform, force sensors will need to be replaced by sensors capable of determining acceleration directly or indirectly.

2.6 Reduced Processing Requirements Solution

When it comes to implementation of a mathematical model, one should consider platform characteristics. While it may be possible to implement this model using discrete circuitry, the far more likely scenario will involve one or more processors. Many systems today are built around general purpose computers containing the same microprocessors found in personal computers. As general purpose computers, they possess a great many capabilities,

are very versatile and have the additional benefit of being familiar to those who will program them. On the negative side they can be bulky, use a lot of power, are not optimized for repetitive operations and may possess a number of peripherals and other resources that a robot doesn't need. This becomes increasingly problematic as the size of the robot shrinks. On the other hand, a DSP circuit is optimized for repetitive operations such as those required for sensor signal processing or the continuous calculations required by the potential field model that has been developed. A DSP also contains peripheral hardware onboard that general purpose computers have spread across multiple components. Additionally, microcontrollers are designed specifically to contain a great deal of hardware onboard required to control a wide array of systems. Some of this hardware includes pulse modulators, various communication circuits to allow the circuit to interface with other system components and the outside world, analog to digital converters, an onboard clock, timers, sleep and wake functions, onboard ram and program memory. Many have even more functions available. They are almost an entire system on a chip short of drivers, interfacing circuits and a few discrete components. This makes microcontrollers the best choice for small robots and extremely useful for larger robots.

Since a microcontroller is the best choice for a small robot, one should look at the potential field model that has been developed in light of a typical controller's capabilities. For this, the Microchip PIC24FJ64GA004 family will be used as an example [16]. In particular, the number of clock cycles required for different mathematical functions is of prime importance. The data sheet shows that addition, subtraction and multiplication each take one cycle. Division, on the other hand, takes 18 cycles. A square root function is even worse as it requires a subroutine. One such subroutine can be found in Texas Instruments' User's Guide for the TMS320C3x DSP [17]. It is an iterative function that requires 4 multiplications, 1 subtraction and 1 division per iteration plus 1 division when the iterations are complete. Since each of the force calculations require a division and a radius which in turn requires a square root, it would be very beneficial to find some way of eliminating these repetitive calculations. Alternatively, a faster

processor can be used but increased memory is typically far less expensive than increased processing speed. If somehow processor speed requirements could be exchanged for memory requirements, the system could handle more targets, obstacles, etc. for a given processor speed. To help with this, the hardware assets of the microcontroller can be examined. To start with, the family has between 5.5k and 22k words or 11k to 44k bytes of instruction memory that can be used to store permanent data. If more memory is required, a parallel port can be implemented to access external memory. So, memory is available within certain limits.

The first clue toward finding a solution comes from the recognition that the shapes of the forces remain constant for a specific type of object regardless of its location in the environment. Second, the additive nature of the forces means that one can consider the effect of each force on the robot individually based and the robot's location relative to the source of the force. In the case of both real and false obstacles, the force rapidly falls toward zero as the distance increases. This shape was specifically chosen to keep the effect of obstacles local. As such, one can simply assume the force is zero if it falls below a point considered significant. For instance, if one maps a 21X21 obstacle potential field centered on an obstacle's location into memory, it will take 441 bytes or words of memory. The potential at ten points away from the center will be one percent the value at one point from the center. A 51X51 field requires 2,601 bytes or words and the value at 25 points from the center is .16 percent the value at one point away. This reveals another tradeoff with this approach. More resolution or precision costs more memory. Once the required resolution is determined, this shape can be mapped into memory offline and permanently stored. With this approach, the inverse distance squared function simply becomes a mapping problem. One determines the rounded distances of the robot less its clearance radius in the x and y directions from a selected obstacle. This identifies the memory location where the potential is store. If x or y goes beyond the defined size of the map, the potential is zero. Since the force in a direction is the differential of the potential with respect to the direction and can be approximated by the difference between two adjacent points in that

direction, one takes the differences between the potential at the original point and the adjacent point in each direction times $-K_i$ to get Fx and Fy on the robot from the obstacle.

The target force can be handled essentially the same way as the obstacle force. A similar size potential field is mapped into memory and the robot's position in the field determines the potential at its location and the adjacent potentials in the x and y directions. The difference occurs near the edge of the stored field. Since the potential field is continuous throughout the environment in order to produce a constant force, a position value cannot simply assume a fixed value beyond its edge. Instead, one must expand the field or rather, reduce the robot's x and y position components so that the larger of the two lies within the border of the field. Since a square field is assumed, the larger value of x or y is checked to determine if it lies past one position from the edge the field. If so, the ratio of the x or y value to the size of the stored field is applied to both the x and y positions to reposition the robot two spaces within the border of the stored field. To avoid confusion, note that this is for calculation purposes only and does not actually change the robot's position. The selected position is two spaces within the border so that the adjacent x and y positions exist in the mapped area no matter where the robot position is moved. Since this process is iterative, the accuracy of the calculations increases as the robot approaches the target. The maximum accuracy occurs once the robot's position is within the map without scaling. As with the obstacle map, more memory means more precision. If the size of the environment is fixed and the precision is acceptable, the ratio can be determined in advance. Figure 2.11 illustrates robot operation using this method.

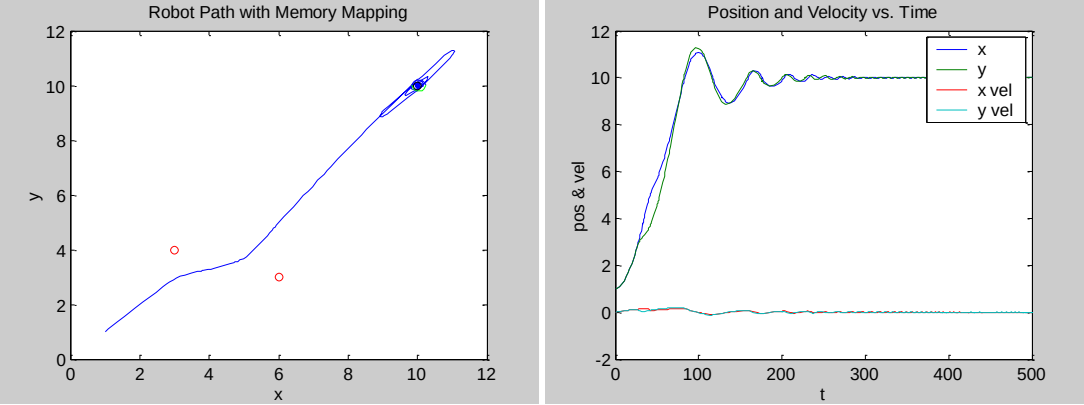


Figure 2.11 Illustration of Reduced Processing Solution with Position and Velocity Plot.

2.7 Emotion-based Control

A truly autonomous robot must keep track of a wide range of variables in order to safely accomplish its goals beyond the location of obstacles and targets. These can be environmental or internal, beneficial or detrimental, even fixed or mobile. The effect and thus reaction time may also be immediate or cumulative. The importance of each can also vary greatly. Many of these may have unknown relationships with the environment and require different solutions. [14] presents an approach that can be adapted to help with these issues. They propose that emotions are an important component of autonomous controls. They start with the idea that people can describe their overall emotional state on a given scale such as -10 to +10. This current emotional state factors in and scales a great number of variables that can affect how the person feels. As such, it also provides a simplified result to describe the impact a large number of factors. The impact of the individual factors can vary greatly and nonlinearly. For instance, temperature may have no effect unless it exceeds or falls below a certain value. Then the impact may be cumulative and increasing as the temperature continues to increase or decrease.

The paper proposes that the patterns formed by these parameters can be tracked so that the robot can learn and recognize the appropriate response given the developing pattern. Since the importance of time also varies with the parameter and each calculation takes a finite

amount of time, the order in which the calculations takes place should be considered. If the general emotional state is good, there is plenty of time to determine the best action based on all the parameters and learned patterns. On the other hand, if the state becomes bad, the response time is more critical so only the most important parameters and patterns should be examined to determine the appropriate course of action. The work was based on a comparison to the nervous system of living organisms.

Even though it is not directly stated in the paper, neural networks can assist the memorization of patterns formed by the various parameters and the use of those patterns to motivate appropriate actions by the robot in spite of great complexity and not readily apparent relationships. Clearly these functions are beyond the scope of the potential field model. They would therefore be accomplished by an emotional component of the robot's mission manager. To the extent that the required action involves the movement of the robot toward or away from an area, the mission manager would convert the required movement to an obstacle or target input to the potential control system.

CHAPTER 3

GLOBAL PATH PLANNING

As shown, local path planning using modified potential field control is very versatile in traversing an unknown environment. Its relative simplicity and low resource requirement works well for small, fast moving robots where resources are at a premium. This being said, the nature of unknown environments can result in many scenarios where the robot can become stuck or otherwise unable to reach the target. Although several of the most common problems have been addressed, many more remain. This model also does not ensure an efficient route. In fact, a complex environment, with several traps between the robot and its goal, will result in a very inefficient route. Fortunately, in most situations, the environment is at least partially known in advance. In these cases, a global path planner can be run offline to find the most efficient route. The route can be fed to the potential field model as a series of goals or waypoints. This approach capitalizes on the strengths of both approaches. It uses the detailed analysis of a resource-intensive global planner while maintaining the simplicity and resource-friendly qualities of the local planner.

A large number of approaches have been used in global path planning including cell decomposition, skeleton, fuzzy logic, neural networks, evolutionary algorithms [41] and others. Three of the most common approaches currently being explored are variations of Particle Swarm Optimization, Genetic Algorithm and Ant Colony. This section explores each of these three approaches.

3.1 Particle Swarm Optimization

Particle Swarm Optimization, originally proposed in 1995 by Eberhart and Kennedy, is a population based optimization method inspired by bird flocking [36] [25]. It is a heuristic system wherein each particle changes its velocity toward a combination of its best prior position

as well as the best previous position of the swarm. Paths are formed by the linking of particles from the start to the goal. Only paths that do not cross obstacles are considered valid. The best position of the swarm is the valid path with the smallest overall length. The process is iterative so both of these will change over time until convergence is reached. This is typically determined by a certain number of iterations resulting in little or no change in the best path or optimization/length functions. The best position of the particle is its location when its best path was shortest.

The movement of the particles is determined by the formulas [36]:

$$v_{ij}^{k+1} = \omega v_{ij}^{k} + c_1 rand_1^k (p_{ij}^k - x_{ij}^k) + c_2 rand_2^k (g_j^k - x_{ij}^k) \tag{3.1}$$

$$x_{ij}^{k+1} = x_{ij}^{k} + v_{ij}^{k+1} \tag{3.2}$$

where i is the particle under consideration, j is the dimension of the particle and k is the iteration. The learning rates, c_1 and c_2, are generally equal to 2. They act as acceleration constants. The best personal position is p, g is the best global position, and $rand_1$ and $rand_2$ are random numbers between 0 and 1. The random numbers allow for the model to be heuristic. The inertial weight, ω, dampens the velocity or decelerates the particle over time. Together, these equations allow the particle to find its best position.

In the case where the particle is at the best personal and global position, the velocity equation reduces to:

$$v_{ij}^{k+1} = \omega v_{ij}^{k} \tag{3.3}$$

This can lead to premature convergence away from the local minimum. To correct this, guaranteed convergence particle swarm optimization (GCPSO) changes the equations of the global best particle to:

$$v_{ij}^{k+1} = x_{ij}^{k} + g_j^k + \omega v_{ij}^{k} + \rho^k (1 - 2 \times rand^k) \tag{3.4}$$

$$x_{ij}^{k+1} = g_{ij}^{k} + \omega v_{ij}^{k} + \rho^k (1 - 2 \times rand^k) \tag{3.5}$$

where

$$\rho^0 = 1, \rho^{k+1} = \begin{cases} 2\rho^k, s > 15 \\ .5\rho^k, f > 5 \\ \rho^k, otherwise \end{cases} \quad (3.6)$$

Here, f is the number of generations where the optimization function values remain the same and s is the number of generations where the optimization values consecutively decrease.

3.1.1 Example 1

One method of implementing the particle swarm model [25] starts with defining the environment in terms of a polar coordinate system. First, a radial line is defined from the start to the goal. The line is divided into segments of equal length. Each segment terminates in a circle whose center is the start of the line. The circles, or dimensions, are where the particles will reside. Since obstacles can be of any shape, they are simplified by creating wrapping circles around them that take into account their dimensions along with their velocities as illustrated in figure 3.1:

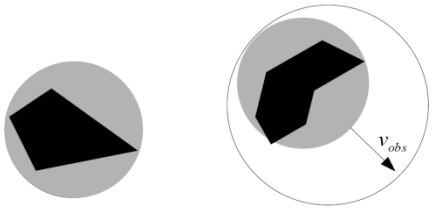


Figure 3.1 Obstacle Wrapping Circles. [25]

The path segments terminate on the start, goal and circles in between the start and goal. The combined result is shown in figure 3.2.

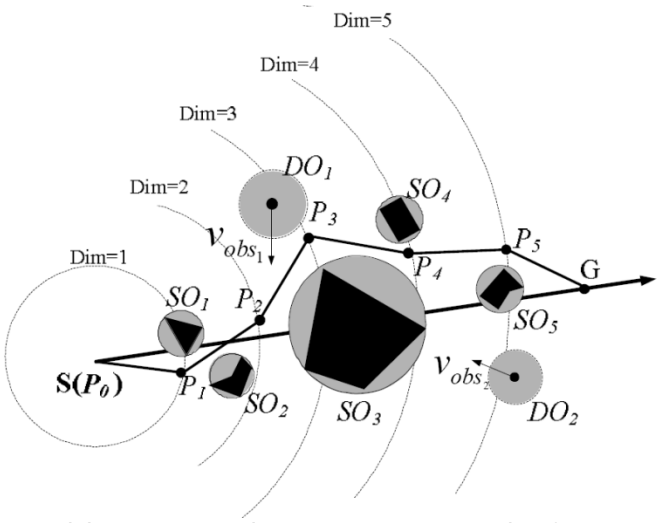


Figure 3.2 Particle Swarm Environment Representation for Example 1. [25]

With this model, the path can be described as the turning angle at each point, P0-P5. The allowable turning angles are those where the particle can go from its current location to the next circle towards the goal without entering a wrapping circle. In the basic particle flow model, this angle is chosen completely randomly. The model is improved in [25] by selecting a normal distribution random number generator, centered at zero for the initial turning angles and disallowing the choices that violate a wrapping circle boundary. The best path is determined by the cost function:

$$f(X) = \sum_{i=0}^{Dim-1} \Delta l_i \tag{3.7}$$

where Δl_i is the length of the ith segment between wrapping circles and Dim is the number of line segments. As is typical, the path with the lowest cost is the best.

3.1.2 Example 2

In [36], the X axis is aligned with the line going from the start to the goal by the formula:

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} X' \\ Y' \end{bmatrix} \tag{3.8}$$

and the circles are replaced with lines perpendicular to the new X axis at the end of the line segments:

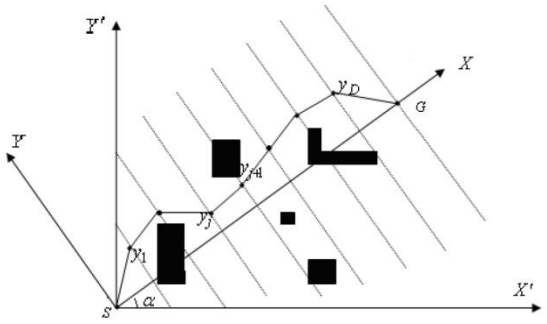


Figure 3.3 Particle Swarm Environment Representation for Example 2. [36]

This approach changes the cost function to:

$$f(Y_1) = \sum_{j=0}^{D-1} \sqrt{\left(\frac{L_{SG}}{D+1}\right)^2 + (y_{j+1} - y_j)^2} \tag{3.9}$$

where D is the same as Dim in the last implementation and L_{SG} is the length of the line from start to goal. Since these values remain constant, the cost function is dependent upon how far the particles vary from the start-to-goal line from one segment to the next. This is the reason the points are labeled y_x. The function has the effect of selecting the paths with the smallest turning angles as before since larger turning angles would result in a larger difference in y from one segment to the next. This in turn, would result in a larger summation, thus a greater cost.

It is inefficient to allow a greater area of the environment than necessary to be processed in finding a path. To correct this situation, [36] introduces a process to define “active regions” for the particles. An active region is a rectangle whose opposite ends are placed on the start and goal. The sides are adjusted sufficiently away from the start to goal line to provide an unobstructed path around any obstacles from the start to the goal. Figure 3.4 shows the result.

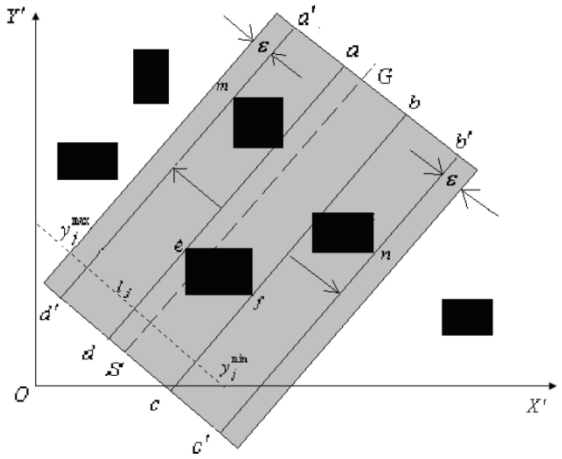


Figure 3.4 Active Region Formation. [36]

The process to find the active region follows:

1. Locate the obstacles residing on the line from start to goal (SG).
2. Locate the two vertices of the obstacles furthest on each side of the SG line (nodes e and f in figure 3.4).
3. Use e, f, S and G to form the new rectangle boundary abcd, parallel to the SG line.
4. Locate the obstacles residing on the new boundary lines (ad and bc).
5. Select the vertices of the obstacles farthest from the SG line (nodes m and n).
6. Replace the last rectangle boundary (abcd) with a new boundary (a'b'c'd') as in figure 3.4 using m, n, S and G.
7. Repeat step 2 until a boundary is formed that doesn't intersect any obstacles.
8. Expand the boundary slightly to allow room for the robot to pass (ϵ in the figure).
9. If the boundary is still free from obstacles, the area inside is the active region.
10. If the boundary intersects obstacles, go to step 3.

Once the active region is found, the initial position of the particles is given by:

$$y_j = y_j^{\min} + r \times (y_j^{\max} - y_j^{\min}), (j = 1,2,\cdots,D) \tag{3.10}$$

Where j is the dimension of the i^{th} particle and r is a random number between 0 and 1. y(j, min) and y(j, max) are determined by the boundary lines or the limit of the environment of the j^{th} dimension line. Line l_j in figure 3.4 illustrates the results. This version of the model uses a velocity limit for the particles:

$$v_{j\max} = 0.1 \times (y_j^{\max} - y_j^{\min}) \tag{3.11}$$

It allows the velocity to be up to 10% of the position range. The initial velocity is selected by the formula:

$$v^0{}_j = (2r - 1) \times v_{j\max} \tag{3.12}$$

In this model, the inertia (ω in equations (3.1), (3.3), (3.4) and (3.5)) is varied over time as well:

$$\omega = \omega_{\max} - k \times \frac{\omega_{\max} - \omega_{\min}}{k_{\max}} \tag{3.13}$$

Here, k is the iteration so k_{max} is the maximum iteration. The maximum and minimum inertias are ω_{max} and ω_{min} respectively. They were set to .9 and .4 in the paper [36]. This iteration-based approach gives the particles a lot of inertia in the beginning to prevent premature convergence. As the iterations increase though, the inertia falls. This makes the current velocity less significant and the global and best positions more significant in determining the next velocity. This allows the particle to settle on the best position as the iterations increase. The acceleration coefficients in equation (3.1) are also varied over time. At the start, c_1 is large and c_2 is small. This causes the personal best position to have priority in the beginning and the global best position to have a greater effect as the iterations increase. This prevents premature convergence on the global best position while the inertia prevents premature convergence on the personal best position. The formulas for c_1 and c_2 are:

$$c_1 = (c_{1i} - c_{1f}) \left(\frac{k_{\max} - k}{k_{\max}} \right) + c_{1f} \tag{3.14}$$

$$c_2 = (c_{2i} - c_{2f})\left(\frac{k_{\max} - k}{k_{\max}}\right) + c_{2f} \tag{3.15}$$

where c_{1i} and c_{2i} are the initial values for c_1 and c_2. Likewise, c_{1f} and c_{2f} are the final values. The paper uses default values of c_{1i}=2.5, c_{2i}=.5, c_{1f}=.5 and c_{2f}=2.5.

There is another issue to consider with particle swarm models. Since the path is simply a line from a particle on one dimension to one on an adjacent position, it is possible for the particles to move to a position where the path is blocked by an obstacle. This causes the particles to become invalid. Rather than spending time reinitializing the particles and checking their path for validity, [36] proposes replacing some particles by the global best position and others by the best positions of neighbor particles as determined by:

$$L_i \in \{P_{i-l}, P_{i-l+1}, \cdots, P_{i-1}, P_i, P_{i+1}, \cdots, P_{i+1-1}, P_{i+l}\} \tag{3.16}$$

$$f(L_i) = \min\{f(P_{i-l}), f(P_{i-l+1}), \cdots, f(P_{i-1}), f(P_i), f(P_{i+1}), \cdots, f(P_{i+1-1}), f(P_{i+l})\} \tag{3.17}$$

The general process for this version of this particle flow model is:

1. Initialize M particles.
2. Initialize the i^{th} particle position using (3.10).
3. Randomly generate the $(j+1)^{th}$ dimension of particle i in the active region until the line from the node in the $(j+1)^{th}$ region to the node in the j^{th} region does not cross any obstacles.
4. Initialize the velocity of all the particles using equations (3.11) and (3.12).
5. Calculate fitness values of all particles. The position of the particle with the lowest fitness value is the global best position. The current position of each particle is its personal best position.
6. Update the velocity of the global best particle by (3.3) and the other particles by (3.1) within the limits:

   If $V_{ij} > V_{jmax}$ then $V_{ij} = V_{jmax}$.

If $V_{ij}<V_{jmin}$ then $V_{ij}=V_{jmin}$

7. Update positions of all obstacles using (3.2) limited by boundary and obstacle constraints.
8. Calculate fitness values of each particle and update personal best and global best positions.
9. Go to step 2 until maximum iterations have been reached or the fitness value of the global best particle remains unchanged for 20 consecutive generations.

The model was simulated using simple and complex environmental models. The simple environment consisted of a 100X100 work space, a dimension equal to 4 and a population size of 10. The results are in figures 3.5 and 3.6.

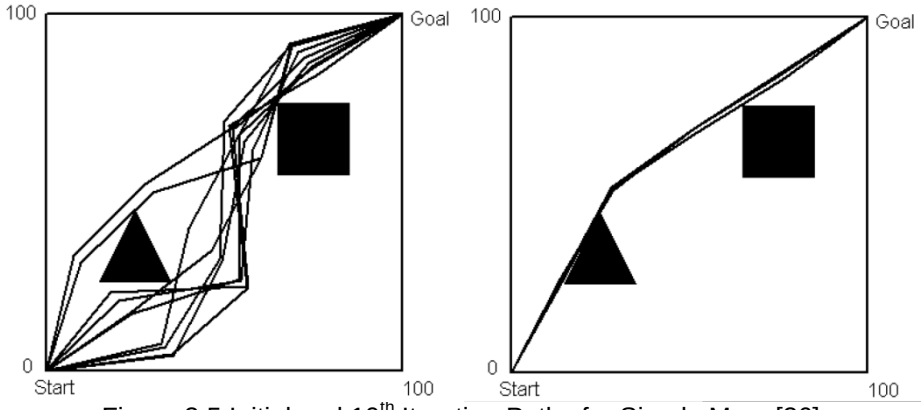


Figure 3.5 Initial and 10th Iteration Paths for Simple Map. [36]

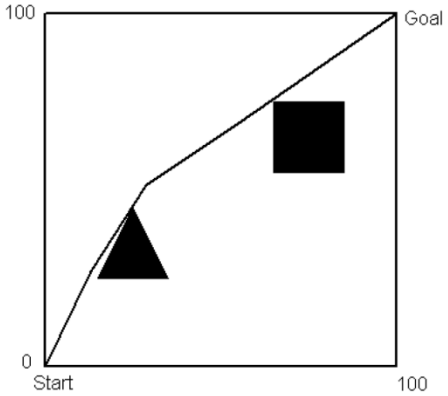


Figure 3.6 30th Iteration Path for Simple Map. [36]

The complex environment had more obstacles, a population size of 10, particle dimension equal to 8 and a maximum iteration equal to 60. The results are shown in figures 3.7 and 3.8.

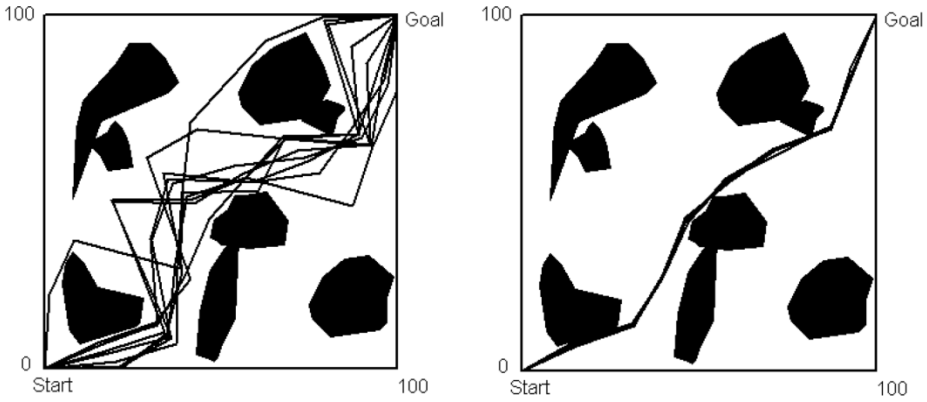


Figure 3.7 Initial and 20th Iteration Paths for Complex Map. [36]

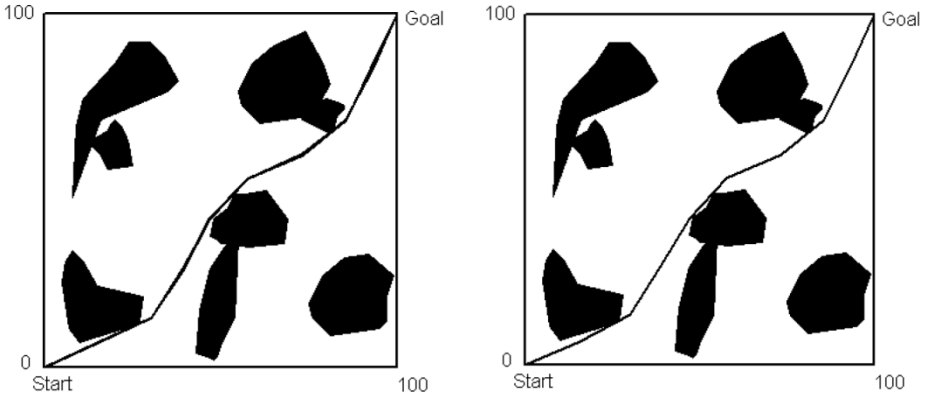


Figure 3.8 40th Iteration and Optimal Paths for Complex Map. [36]

3.2 Genetic Algorithm Optimization

The genetic algorithm model is based off of evolutionary processes observed in natural biological systems [26]. It first appeared in 1975 [34]. The environment is grid-based. Its solutions are called individuals whose chromosomes comprise a list of grid positions from the start to the goal. It is a self-adapting, heuristic, global search algorithm that uses three primary operators to modify the solutions. The operators are reproduction, crossover and mutation.

The reproduction operator copies selected individuals from one iteration to the next. The probability of an individual being selected is proportional to its adaptive value. This allows the best chromosomes to continue to the next generation while eliminating the weaker ones. This process increases the overall fitness of the population.

The crossover operator uses a crossover probability (Pc) to select pairs of individuals for processing. Once chosen, the two individuals exchange a certain portion of their chromosomes. Since the chromosomes represent a path through the environment, crossover will sometimes result in a shorter (better) valid path than either parent. This serves to help convergence on a global solution.

The mutation operator uses a small mutation probability (Pm) to select individuals for the mutation process. The selected individuals will have portions of their chromosomes changed stochastically. This introduces random routes to the path that increases the diversity of the population so it doesn't converge prematurely.

3.2.1 Modified Genetic Algorithm

The [29] version of the genetic algorithm uses a model with an additional operator and some modifications to the crossover and mutation operators. The new operator is the deletion operator. It attempts to remove sharp corners to shorten the path. The mutation operator is changed to avoid premature convergence. Changes to it and the crossover operator also allow them to self-adjust. Both will be described in more detail in their respective sections.

3.2.1.1 Environment Description and Population Initialization

As mentioned above, the genetic algorithm uses a grid-based environment. Figure 3.9 illustrates this approach. The line represents the robot's path and the shaded blocks are

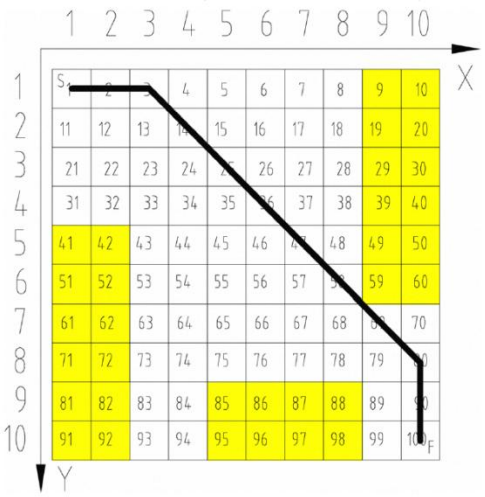


Figure 3.9 Grid Based Environment. [29]

obstacles. Numbering the blocks in this manner allows a simple chromosome model. In this case, the path code of the chromosome would be (1, 2, 3, 14, 25, 36, 47, 58, 69, 80, 90, 100). The formula for the path code is based off the X and Y coordinates and the size of the grid in the X direction:

$$code = X + (Y - 1) \times m_x \tag{3.18}$$

In most genetic algorithms, the initial population is generated randomly. As with the particle swarm methods, this potentially results in a large number of invalid paths. This can result in a lot of time being used to evaluate the paths and find a valid initial population. To eliminate the chance of generating invalid initial paths, this model borrows concepts from the ant colony approaches. The process for generating the initial population follows these steps:

1. Acquire the current node A coordinates. The first “current” node will be S.
2. Determine the direction from node A to the finish node F. Divide the eight neighbors of node A into three priority groups based on the direction to F.
    a. The three neighbors closest to the direction of F is the highest priority group.
    b. The two neighbors parallel to the direction of F is the second priority group.
    c. The three neighbors furthest from the direction of F is the lowest priority group.

3. Each node is associated with a counter which indicates how many times the node has been chosen. This is borrowed from the ant colony pheromone system.
    a. Assign a probability to each neighbor in the highest priority group based on the number of times it has been chosen.
    b. The highest probability is assigned to the neighbor with the lowest count. This encourages population diversity.
    c. Randomly choose a neighbor of the highest priority group in accordance with the assigned probability. B is set to the chosen node.
    d. If the neighbor is not available, choose another member of the group as in 3.c.
    e. If no nodes are available in the highest priority group, repeat the process for the next highest priority group.
    f. If no nodes are available in the next priority group, repeat the process for the lowest priority group.
    g. If no neighbors are available, set current node A to node S and go to step 1
4. If node B is available:
    a. Add node B to the path.
    b. Let node A equal node B.
    c. Increment the counter for node B.
5. If node B equals node F:
    a. A collision-free path has been found. Go to step 6.
    b. If node B is not equal to node F, go to step 1 to find the next node in the path.
6. If the number of individuals equals the population size, the initial population has been generated. If not, set node A to node S and go to step 1 to generate the next individual.

3.2.1.2 Fitness Operator

As with particle swarm models, a fitness function is critical for determining which solution is the best. In this implementation, the following function is used:

$$f = \frac{1}{\sum_{i=1}^{n} d_i + \frac{1}{2} nn} \tag{3.19}$$

Here, d_i is the length between two consecutive nodes and n is the number of nodes. The number of turns in the path is nn. Diverting a robot from a straight path often takes more energy than allowing it to continue on a straight path. This formula allows the model to discriminate against a route that requires an excessive number of turns in favor a more direct path of similar length. The best path gets the highest fitness value. Normally, each initial individual must be evaluated to determine validity. This is not necessary in this version of the model since the initialization steps in the prior section guaranty a valid path.

3.2.1.3 Selection Operator

After the fitness has been determined for each individual, there must be a determination of which ones to use in the next generation. This formula determines the quantity of each to use in the next generation:

$$quantity_A = \frac{f(A)}{\sum_{i=1}^{popsize} f_i} \times popsize \tag{3.20}$$

In this function, f(A) is the individual's fitness and popsize is the number of individuals in the population. The fractional part of the equation normalizes the individual's fitness to the total fitness of the population. The remainder of the equation keeps the population size the same for each generation while ensuring the best individuals have the most offspring in the next generation.

The individuals selected for the new generation do not remain stagnant. If they did, there would simply be a repeat performance from one generation to the next of the copies of the

best individual. This would cause the copies to represent an increasing percentage of each subsequent generation due to the selection operator. The result would be for the model to rapidly converge on the best performing initial individual with no improvement of performance. Instead, the remaining operators randomly modify the individuals of the next generation created by the selection operator. The selection operator ensures that the modifications are done on copies of the most-fit individuals and that there are plenty of copies of these individuals to modify in different ways.

3.2.1.4 Crossover Operator

The first modification to the next generation is a result of the crossover operation. This process is analogous to the transfer of genetic information from two parents to their offspring. The offspring ends up with partial genetic information from each parent. This creates a unique individual that may prove to be stronger or weaker than either parent with respect to its environment.

To implement this process, candidate pairs are selected from the current generation (next generation in the selection operator discussion above) if they have at least one common point aside from the start and finish. The pairs are chosen randomly using the crossover probability (Pc). The [29] model uses a one-point crossover operation. In it, one of the common nodes, aside from the start and finish, is selected for the crossover point. All the nodes following this point in each individual are swapped between the two individuals. Since both paths are valid and the crossover occurs at a common point, the resulting path is valid as well. The example given starts with the chromosomes of a pair of individuals that have been selected using Pc:

- (I , 2, 3, 14, 25, 36, 47, 58, 69, 80, 90, 100)
- (I, 12, 23, 34, 45, 56, 67, 68, 69, 79, 89, 90, 100)

The common nodes, 69 and 90, have been underlined. Next, 69 is selected randomly from the two choices. Now the crossover operation is performed which produces the new sequences:

- (1, 2, 3, 14, 25, 36, 47, 58, 69, 79, 89, 90, 100)
- (1, 12, 23, 34, 45, 56, 67, 68, 69, 80, 90, 100)

Figure 3.10 illustrates the before and after results of this operation:

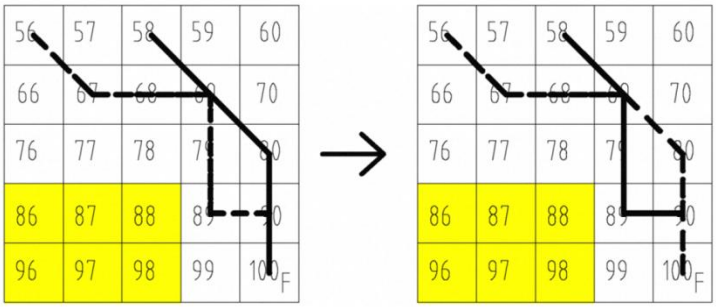


Figure 3.10 Results of Crossover Operation. [29]

3.2.1.5 Mutation Operator

As its name suggests, the mutation operator provides random changes to the chromosomes of selected individuals. This “mutation” process is found in nature and is believed to be at least partially responsible for the diversity of life found on earth. When the effects are significant, it usually proves detrimental for the individual that acquires the mutated gene. In some cases though, the mutation allows the individual to be better suited to its environment and thus more fit. As this mutation is passed from generation to generation, the species becomes stronger or a new species branches off from those without the mutation.

Individuals that are to undergo the mutation process are randomly chosen using the mutation probability (Pm). The basic implementation of the genetic algorithm model follows the natural process where a mutation can result in a node in the chromosome being changed to a completely random location in the environment. This may lead to a valid path but often results in an invalid one. A check must be done between the changed node and its two neighbors in the

chromosome to determine which result occurred. Even when the result is valid, the individual has a high probability of being less fit than before the mutation. The [29] model uses a more limited approach. Instead of selecting any node to change, this version only selects nodes that form a corner. The corner node is changed to one of its neighboring nodes based on their counters from the initial population routine. The neighbor is chosen randomly where the node with the lowest count has the highest probability. After the new node is selected, the initial population routine is run from the corner node prior to the one being changed in the chromosome through this one (the changed one) and on to the next corner node. If there is no corner node, before or after the changed node, then the start or finish node is used as appropriate. Figure 3.11 shows the results of the mutation operation:

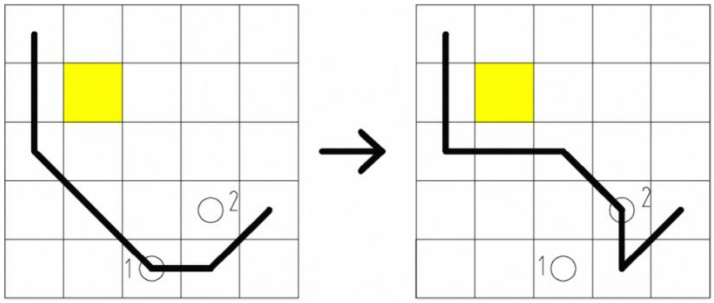


Figure 3.11 Results of Mutation Operation. [29]

3.2.1.6 Deletion Operator

The deletion operator is based off the principle that two sides of a triangle are longer than the third side alone. The deletion operator looks at the nodes of a chromosome to determine if any form an acute or right angle. If so, it deletes the node and attaches the two nodes to which it was connected. If the new line is valid, it keeps the change. If the line is not valid, it keeps the original node configuration. As an example, the left portion of figure 3.12 from [29] shows, node 3 forms angle 234 and node 7 forms 678. Deleting these nodes removes 45 degrees from each angle and makes the path shorter as shown in the right portion of figure 3.12.

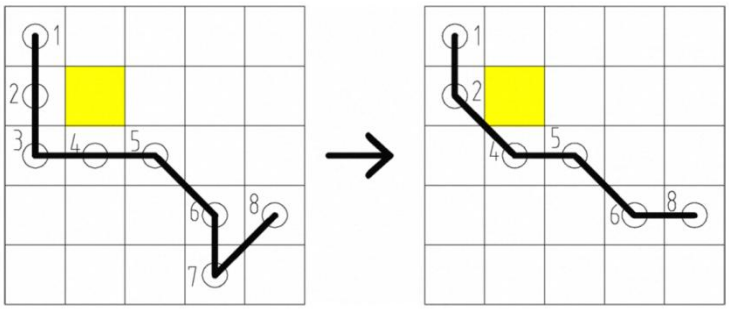


Figure 3.12 Results of Deletion Operation. [29]

3.2.1.7 Probability Selections

As described above, the genetic algorithm uses two key probabilities to modify the chromosome copies from the prior generation. The crossover operation depends on Pc and the mutation operation is dependent on Pm. The purpose of the crossover operation is to provide a means of combining paths to perhaps create a shorter path than either original. This helps the model converge on a more optimal path. The mutation operation is designed to create greater diversity by deviating from the more optimal paths. The initial paths are developed independently so the initial population is fairly diverse. This being the case, it is logical to start by optimizing the paths. As optimization increases, diversity decreases. Since there is no guaranty that the original population was diverse enough to ensure the best path was included in the original set, it makes sense to provide opportunities for new paths to be found by increasing the diversity. The need to increase diversity grows as the path choices are reduced through the optimization process. To balance these time variant requirements, this version of the genetic algorithm changes Pc and Pm as conditions change. This is the process for determining their values:

1. The model is initialized with a relatively high Pc and a low Pm. This allows the crossover function to be dominant which focuses on optimization.
2. Run the next generation.

3. Compare the optimum individual from the generation with the global optimum individual. Keep the best of the two as the new global optimum individual. This is elitism.
4. If the global optimum individual remains the same for three consecutive generations, Pc is decreased and Pm is increased toward their limits. This allows the mutation operation to create more divergence.
5. If the limits of Pm and Pc have not both been reached, go to step 2. If they have been reached, the process ends.

The model was applied to simple and complex environment examples to verify its ability to find a solution. The simple environment consisted of a 20X20 grid. The population size was set to 10. The Pc range was 0.65 to 0.3 and the Pm range was 0.01 to 0.08. The maximum number of generations was 50. Its final optimal path was found in 13 generations. The path is shown in the left half of figure 3.13. The complex model was run on a 30X30 environment with randomly mapped obstacles. To verify similar results were acquired, the process was run 10 times on the same map. It took 11 to 14 generations to find the optimal paths. The average solution took 12.3 generations with a length of 45.6768. The difference between the minimum and maximum solution lengths was 2.929. The right half of figure 3.13 shows the final path.

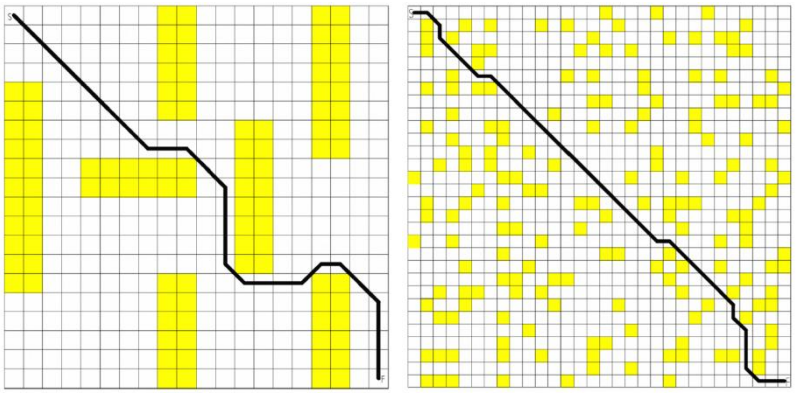


Figure 3.13 Results of Modified Genetic Algorithm on Simple and Complex Maps. [29]

3.2.2 Parallel Elite Genetic Algorithm

It has been shown that one of the problems with the genetic algorithm is premature convergence. This can cause the model to optimize the best path that has been discovered rather than the best path available through the environment. The mutation operator is designed to help with this issue but it still relies on slightly modifying a small set of existing chromosomes. The selection operation complicates this more by making sure the majority of the next generation is based on the best performer of the prior generation. In [32], an attempt is made to improve several aspects of the model to improve the chance of finding the absolute best path through the environment.

3.2.2.1 Migration Operator

The first significant contribution of [32] is the unusual application of parallel processing to solve the path planning problem. Typically, parallel processing involves running the same program on multiple processors and comparing the results, sharing information continuously between identical processes run on multiple processors or running separate portions of a process on multiple processors. This instance of the model uses a slightly different approach. Most of the time, it runs independent genetic algorithms on each processor. Periodically though, the processors exchange their best performers. This is performed by a new, migration operator. The migration operation allows the populations to evolve in isolation for a time to produce independent, strong groups of individuals. Then, before the population converges too much, new individuals are inserted to add diversity. The difference is this diversity is not produced by arbitrary chromosomes. It is a result of an influx of chromosomes that have proven successful in the environment instead.

3.2.2.2 Mutation Sub-operators

The mutation operator produces diversity but at the cost of a high probability that the new path will be invalid. In [29], the solution was to restrict the mutation to a small group of

positions surrounding the node selected for mutation. In [32], the solution is to create two sub-operators to help a resulting invalid path to become a valid path.

When the path between two nodes crosses an obstacle the normal solution is to declare the mutated chromosome invalid. This version of the genetic algorithm uses an addition sub-operator instead to attempt to salvage the mutated path. It functions by randomly inserting a new node between two, randomly selected nodes. If the old path went through an obstacle, this provides the chance that the new path between the original nodes (through the new node) creates a valid path. The tradeoff is that the new path, if successful, is longer than the original but valid. Figure 3.14 illustrates the use of the addition operator.

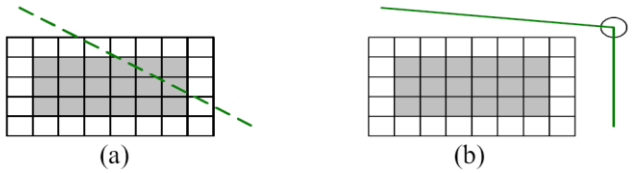


Figure 3.14 Results of Addition Operator. [32]

The [32] version uses the deletion operator in a different way than the [29] version. Here it is a sub-operator to the mutation function. Its job is to randomly delete a node. This results in a shorter path as long as the node is not directly in line with the nodes it is between. Whether it is or not, it will result in a shorter chromosome. The negative consequence is that it may cause a path that goes around an obstacle to go through it instead. Figure 3.15 shows the intended result of this type of deletion sub-operator.

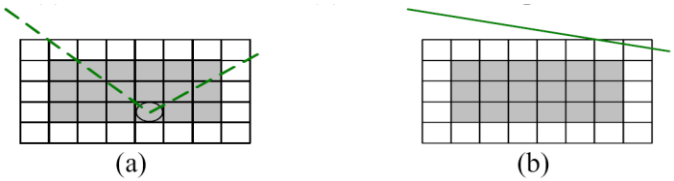


Figure 3.15 Results of Modified Deletion Operator. [32]

3.2.2.3 Improved Reproduction Strategy

The prior model of the genetic algorithm, described in the last section, used a very different reproduction strategy than the [32] model. It kept an “elite” individual which represented the best performing chromosome of all the generation. This represented the global best individual and was used as a reference for comparison. A certain number of copies were created in the next generation based on performance compared to the other members of the generation. From there, they were treated the same as all the other individuals in the new generation. First, the diversity in the new generation starts with a subset of the diversity of the last generation. It is increased with the mutation operator but that only provides adjustments to existing chromosomes which may not contain the best path through the environment. Also, the mutation and crossover operations may produce valid children that are not as fit as their parents. In the early generations this is critical as there are less copies of each successful individual so the best candidate in the long term may see its lineage die out due to an early, unfortunate modification. The result is a global result that is not the best global candidate. Even if the best path is eventually found, it can take longer to do so.

To improve diversity while maintaining a larger elite pool, this version creates the new generation using a different approach. First, the current generation is ranked according to fitness. The top 20% of the individuals are directly reproduced in next generation without any processing. This maintains a larger pool of elite individuals and makes sure that each generation performs at least as well as the prior one. An additional 20% of the new generation is created randomly using the initial population method. This takes more time but maintains significantly greater diversity from one generation to the next. The remaining 60% of the new population is created performing crossover and mutation operations on randomly selected members of the current generation. This allows for possible improvements to the prior generation and eventual convergence to the best path.

This model also doesn't automatically discard invalid paths. It is possible that an invalid path could be the best candidate with a simple modification. It can also take considerable time to create an initial population that only consists of valid paths. Rather than search for specific solutions or continuously generating random paths to find valid ones, the model uses a modified fitness function that takes into account the intersection of an obstacle:

$$F = \sum_{i=1}^{M_{\max}} (d_i + \alpha_i T) \tag{3.21}$$

It can be seen at first glance that the function primarily depends on the distance of the path since M_{max} is the number of line segments in the path and d_i is the length of the ith segment. The part of the summation to the right of the plus sign is the part that allows invalid paths to be treated properly without deletion. T is a constant that determines how much higher a fitness value (lower degree of fitness) an invalid path will get compared to a valid path. The lower the value of T, the more likely a short, invalid path will have preference over a long, valid path. α_i determines how much an invalid path will be penalized:

$$\alpha_i = \begin{cases} 0, & if\ the\ ith\ line\ segment\ is\ feasible \\ \sum_{k=1}^{N} k, & if\ the\ ith\ line\ segment\ \mathrm{intersects\ obstacle}(s) \end{cases} \tag{3.22}$$

This allows the penalty to increase rapidly as the number of line segments grows in an invalid path and eliminates the penalty if the path is valid. There is an additional benefit for the strategy of this model that is not mentioned in the paper. The maintenance of invalid paths combined with the random use of the addition and deletion operators allows for chromosomes that can adjust their size according to environmental requirements. With this model, the chromosome length can be minimized as well as the path length. The [41] model uses this fitness function as well.

3.3 Ant Colony Optimization

As the name suggests, Ant Colony Optimization (ACO) is based on the foraging behavior of ants. It was first introduced in 1992 by Marco Dorigo [34]. While ants are looking for food, they travel a random path. As they travel, they leave behind a pheromone trail that dissipates over time. Other ants can sense the pheromone levels and therefore determine the path that other ants have taken. Once an ant finds food, it travels back to its mound. This leaves more pheromone along the path. As it travels back and forth, the pheromone accumulates. Other ants pick this higher concentration up as they cross the path. The increased levels make the ants more likely to follow the path to the food source. As they travel back and forth, they occasionally deviate slightly from the path in an effort to make the route shorter. If the deviation is successful, the path will contain a little more pheromone per ant. This is because an ant taking the route will take less time than an ant taking the original route. The newly deposited pheromone will have less time to dissipate than that deposited on the original path. This makes the new path more likely to be followed by other ants. As the process continues, the path becomes optimal. This positive feedback type of course selection is called Autocatalytic behavior [26]. It is a common mechanism for reinforced learning.

In practice, a grid based environment is used. A certain number of ants begin at the starting point. Each one randomly chooses an adjacent free space. The choice probability is based on the amount of pheromone in the adjacent space (exploitation) and the chance of a little used route (exploration). Exploitation is responsible for convergence to an optimal path. Exploration is responsible for preventing premature convergence on a sub-optimal path. Once a space is chosen by each ant, it moves to the new space. This process continues until each ant has, or at least a certain portion of them have, found the goal. Once the ants arrive at the goal, the pheromone at each location in the environment is reduced by a certain percentage. This represents the partial evaporation of old pheromone. Next, each space along the path of the successful ants has new pheromone added to it. The amount added to each space is equal to a

set amount divided by the length of the path. This allows shorter, more used paths to receive more pheromone. This completes one generation. A new generation of ants is placed back at the start and the process is repeated for the required number of generations.

3.3.1 Ant System Algorithm (AS)

The algorithm in [34] is based on Dorigo's model with a new heuristic rule for determining the probability for choosing the next space:

$$Probability\ ij = heuristic * pheromone = [(1/distance\ between\ vector\ start\ to\ subpath\ and\ start\ to\ perpendicular\ subpath\ with\ reference\ goal)^{\beta} * (trail / \sum trail)^{\alpha}] \quad (3.23)$$

where

ß=heuristic coefficient, α=pheromone trail coefficient

This formula has a couple of effects. First, it allows the probability to be based in part on a pseudo-random location and in part on the pheromone level of any trail that my go through surrounding locations. This allows for the balance between exploration and exploitation. Second, a space that has both a shorter distance calculation and a higher pheromone value will have a higher probability to be chosen.

Rather than waiting until all ants have reached the goal as mentioned in the prior section, pheromone evaporation is evaluated locally during the search process in this version. The new value is given by:

$$T_{ij}(new\ trail) \leftarrow (1-\rho) * \tau_{ij}(old\ trail) \quad (3.24)$$

where

ρ=evaporation rate

Once the ants have all reached the goal, the deposited pheromone is determined. This is consistent with the basic model. The new value for each space is:

$$t_{ij} \leftarrow t_{ij} + \sum t_{ij}^{k} \quad (3.25)$$

$\sum t_{ij}^{k}$ is the amount of pheromone that ant k deposits on the space. If the ant has traveled to the space, the space gets additional pheromone otherwise, it doesn't. The amount of pheromone each space gets is given by:

$$\Delta t_{ij}^{k} = \begin{cases} Q/C^{k}, & if\ \ arc(i,j)\ belongs\ to\ \ path\ P^{k} \\ 0, & otherwise \end{cases} \tag{3.26}$$

Here, Q is the total number of nodes and C^{k} is the length of the path P^{k} travelled by ant k. The nodes, or spaces, that have been visited by the most ants from successive generations will attract more ants from later generations. This increases the chance that the ants find the optimal path over time.

3.3.2 Improved Augment Ant Colony

In [26], operators are borrowed from the genetic algorithm approach to improve the ant colony system. It also introduces a targeted heuristic algorithm to improve the initial population efficiency. Finally, it splits the ants into two groups which also increase the initial population efficiency.

One of the problems with Ant Colony Optimization (ACO) is premature convergence. As described above, the shortest path gets the most pheromone at each location along it. This attracts more ants by increasing the probability that the ant chooses the path. The increased number of ants also causes more pheromone to accumulate along the path which further increases the chance of the path being chosen by more ants. If this process happens too quickly, the system will converge on the best current or local path rather than the best global path. If the pheromone level increases rapidly enough, it becomes possible that the best local path becomes difficult to optimize. This is because the probability of choosing a location that is not on the path becomes very small. Even if a better location is chosen, the amount of additional pheromone deposited may be swamped by the pheromone deposited by all the ants that travel the established path. This makes the heuristic function critical in order to prevent this problem. In [26], a certain level of diversity in the environment is better maintained by borrowing

the mutation operator from the genetic algorithm theory. This keeps paths available that have had little exploration early in the process while many unexplored paths remain and the pheromone levels remain low. To reduce the swamping effects of near-optimal paths from preventing improvements to the paths, the genetic algorithm crossover operator is used to continually seek shorter paths than individual ants have found on their own. This can help spread the pheromone over a wider area than the ants' original paths early on thus allowing optimization of the local path within the area.

Another issue with ACO methods is the time it takes to find valid paths with the initial population. When ants leave the mound to search for food, they do not know where the food may be. They have two choices. They can choose a random path until they happen upon a food source or follow a pheromone trail left by other ants to a food source. The initial population does not have the benefit of any existing pheromone trails so a random path must be constructed. In nature, the location of the target is not known but in robotic situations, the location of the goal is usually known even if the best path isn't. This gives rise to the use of a heuristic formula that considers the distance of each viable adjacent position from the target as well as the deviation angle between the line to the target and the line to the next location being considered:

$$\varphi_{ij} = \begin{cases} (m\frac{\theta}{\pi})^{\lambda}\left(\frac{(n(d_{\max}-d_j)+\mu)^{\omega}}{\sum_{j\in\Theta}(n(d_{\max}-d_j)+\mu)^{\omega}}\right), & j\in\Theta \\ 0, & others \end{cases} \tag{3.27}$$

In this equation, i is the present location of the ant and j is the potential next position that is under evaluation. The maximum distance from the start point to the target is d_{max} and d_j is the distance from j to the target. θ is the absolute angle between the line from i to the goal and the line from i to j. Θ includes the feasible region around i. The remaining variables are constants.

Assume the environment consists of a 100 X 100 space grid. The start is in one space and the goal is in another. In the usual ACO system, all the ants begin at the start location and

seek the goal location. During the early generations, there is little pheromone to guide any of them to the goal. This problem is particularly bad during the first generation when there is no pheromone from prior generations. At this stage, each ant's path is completely random or at best guided by a heuristic function. In either case, they are each independently looking for the 1 space out of 10,000 available spaces that contains the goal. In the best case scenario, the ants travel in a straight path from the start to the goal in L turns where L represents the length from the start to the goal. In the worst case scenario, the ants wander around in a much longer path until they happen upon the goal. Obviously, this can take a considerable amount of time.

Alternatively, assume that the n ants are divided into two groups of n/2. One group begins at the start and seeks the goal as usual. The second group begins at the goal and seeks the start. Further assume that each group can recognize the paths of the opposite group. Now the worst case scenario would occur if each of the ants finds its respective target without encountering any paths from the ants in the opposite group. Since more spaces are occupied by ants from the opposite group as time goes by, the upper boundary with respect to time is less than the normal scenario. Even if the ants from one group wander back and forth between their beginning point and an adjacent space, the other ants need to find either of 2 out of the 10,000 spaces. This is because either the path of the other ants or the target represents a valid solution. In the best case scenario, the ants from each side head directly toward their respective targets. By L/2 + 1 the ants from each group meet the ants from the opposite group. The ants from one group have established a valid path from the goal and the ants from the other group have established a valid path from the start. The combination of the paths represents valid paths from the start to the goal. Since the ants can recognize the paths of the opposite group, the information for complete, valid paths is available and thus they have accomplished their objective once the paths cross.

Aside from these limiting situations, the two-group scenario still receives benefits over the normal method of beginning all the ants from the same point. To illustrate this, consider the

ability of the ants to move to a space that fails to result in a solution. As time continues, each group of ants covers more of the environment. This leaves fewer spaces in the environment available for the ants to move without intercepting the path of an ant from the opposite group or its target and thus completing its mission. This can rapidly increase the chance of forcing a solution as it effectively reduces the size of the available environment as time continues. There is no such limiting factor in the one-group scenario. This method is used in [26].

In [35], a bi-directional approach is used with a different heuristic system. It uses a set of precedence rules rather than a probability formula to determine the next position for the ant:

1. Don't choose the point the ant passed.
2. Choose the point with the highest pheromone first.
3. Choose the path with the same direction as its aim point first.
4. Choose the point that is closest to the line between its beginning and aim point first.
5. Admit some faults in the process of ant colony selection.

It calls this the heuristic bidirectional ant colony algorithm. In its results, it compared several algorithms based on the time it took for each to find a solution and the number of obstacles in the environment. The results are shown in figure 3.16.

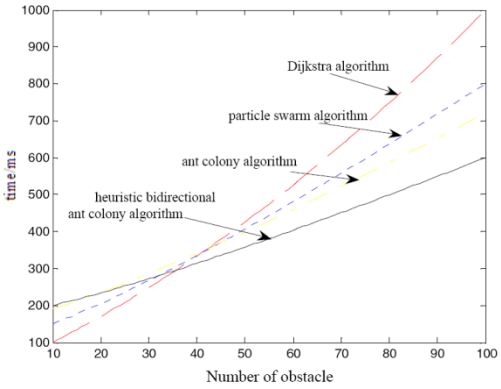


Figure 3.16 Results from Four Global Methods vs. Number of Obstacles. [35]

3.3.3 Multi Colony Ant Algorithm Applied to Aircraft

In [45] the ant colony algorithm is applied to aircraft routing. It assumes the aircraft is allowed to fly in certain areas and must avoid others. This means altitude can be ignored which reduces the problem to two dimensional space as has been the case so far. The environment and the obstacles both have a polygonal shape. Unlike the other ant colony systems though, the environment does not consist of a grid. Instead, it is reduced to a MAKLINK graph. In essence, a MAKLINK graph is produced by strategically linking some of the corners of the obstacles in the environment and some of the lines perpendicular to the edges of the environment. When designed properly, an obstacle-free path can be found between any two points in the obstacle-free space of the environment by travelling between the centers of the lines and terminating on the start and end points. This will be discussed in more detail later but for now, figure 3.17 is an example of a MAKLINK graph.

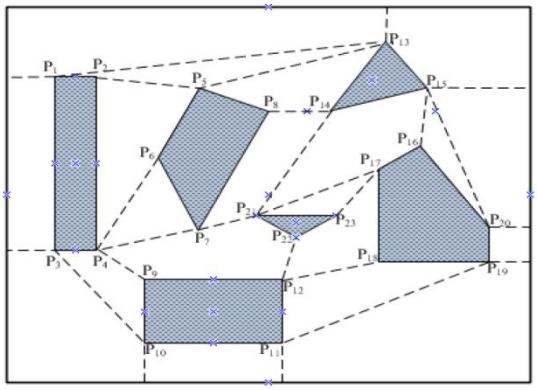


Figure 3.17 A Sample MAKLINK Graph. [45]

Assuming an ant(k) is on one of these lines, it can move to any other line visible to it. Each of these lines can be divided into x number of segments which give x+1 points it can move to on each line. The probability of ant(k) choosing a point j to move to from its present point i is determined by:

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha} \times [\eta_{ij}(t)]^{\beta}}{\sum_{s \subset allowed_k} [\tau_{ij}(t)]^{\alpha} \times [\eta_{ij}(t)]^{\beta}}, & j \in allowed_k \\ 0, & otherwise \end{cases} \tag{3.28}$$

$Allowed_k$ is the set of points visible to the ant, $\tau_{ij}(t)$ is the amount of pheromone on the path from i to j and $\eta_{ij}(t) = 1/d_{ij}$ is a heuristic value. For $\eta_{ij}(t)$, d_{ij} is the distance from i to j. α and ß are constants that determine the impact of the pheromone and heuristic values on the ant's decision.

Only a certain number of the best ants M_{best} of M total ants are allowed to apply pheromone to the path. The amount is determined by:

$$\tau_{ij}(t+n) = \rho \times \tau_{ij}(t) + \Delta\tau_{ij}(t) \tag{3.29}$$

and:

$$\Delta\tau_{ij}(t) = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t) \tag{3.30}$$

In these equations, ρ is the amount of pheromone to be added to the path between i and j. Δτ is given by:

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \dfrac{Q}{L_k}, & if\ k^{th}\ ant\ uses\ path\ in\ tour \\ 0, & otherwise \end{cases} \tag{3.31}$$

Here, Q is a constant and L_k is the length of the entire path from start to goal that was found by ant(k).

In the multi-colony approach used in [45], there are N independent colonies of M ants each. These colonies operate in parallel for NC generations. After each generation, the local best paths are exchanged between the colonies. This allows the colonies to operate independently during each generation with each of the best paths marked from the prior generation of the colonies. It helps balance the divergence resulting from independent colonies and the need for convergence of the colonies on a globally best solution.

To determine the best ants for depositing the pheromone trail, a cost function common to military aircraft is implemented:

$$\min J = \sum_i [kJ'_{fuel} + (1-k)J'_{threat}] \tag{3.32}$$

J'_{threat} is used by military aircraft based on the radar profile of the aircraft along the path. It is ignored in this version of the ant colony approach. J'_{fuel} is related to the fuel consumption required for maneuvering the aircraft. At constant speed on a straight path, it is directly proportional to the length of the path. K is a constant between 0 and 1.

The complete process flow of this method is:

1. Establish a MAKLINK model of the environment's free space and build visibility relation table of each MAKLINK line from the start to the goal.
2. Create N colonies of M ants each. Specify the α, β, ρ and τ_0 parameters. Set iteration counter t = 1. Define the maximum number of iterations NC. Place all ants at the start point.
3. For a node on h(i) for ant(k), choose the next node based on p(i, j) and the visibility relation table. Repeat for each ant(k).
4. Go to 3 until the paths are completed.
5. Compute J_k and L_k for the path of each ant(k).
6. Update the pheromone concentration of each node per the above method. Repeat from step 3 for each colony. After each colony has run, exchange the pheromone information between colonies.
7. Increment t.
    a. If t < NC and all ants do not make the same path, place all ants to the start and go to step 3.
    b. If all ants have the same path or t = NC then output the best path determined by J_k and end.

Figure 3.18 shows the results of this version of ant colony optimization (Path 3) as compared to the path found by genetic algorithm (Path 2) and a geometrical algorithm (Path 1).

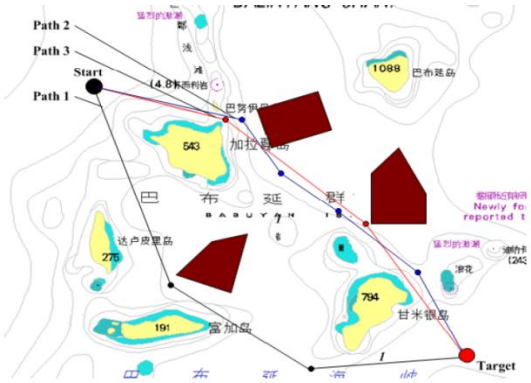


Figure 3.18 Results of Ant Colony Applied to Aircraft. [45]

3.3.4 Heterogeneous Ant Colony

The heterogeneous ant colony optimization (HACO) approach in [38] is the most complex of the ACO approaches. To begin with, it uses a modified Transition Probability Function (TPF) and Pheromone Update Rule (PUR). It also performs a path crossover function from genetic algorithms as part of the PUR process. Next, it introduces heterogeneous ants which include different species of ants with different capabilities. It uses attractive and repulsive visibility functions borrowed from potential field control. It even handles the transition from one generation to the next differently which results in a varying number of ants in each generation or "epoch".

The most important aspect of this version of the ant colony approach is the use of five different species of ants. The species are separated by their speed and sight distance characteristics. The speed is determined by the distance they can move in each turn. The sight distance is determined by the number of spaces between the ant and the closest obstacle and maximum distance the species can see. The differences in these characteristics results in species with different strengths as shown in figure 3.19.

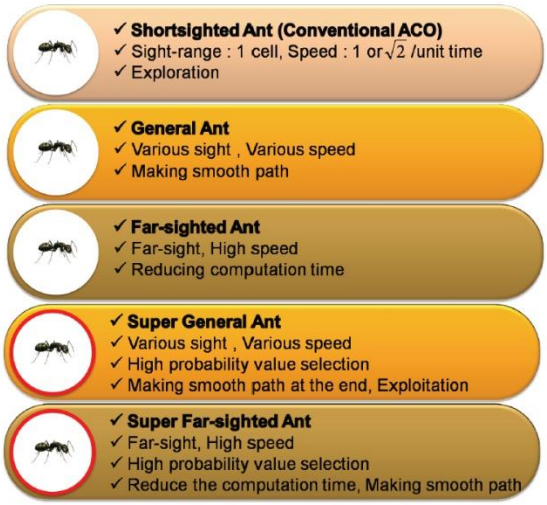


Figure 3.19 Species of HACO Ants. [38]

The sight and speed characteristics are determined by the ant species' sight window. The sight window for the conventional ant used in the prior ant colony approaches is one space in each direction as illustrated in figure 3.20. The ant can move to any one of the eight orange squares surrounding it. As indicated in figure 3.20 below, the shortsighted ant's speed is limited to a distance of 1 in the horizontal or vertical direction and 1.414 in the diagonal direction. To get the window for the general ant, the far-sighted ant and their super cousins, the principle of

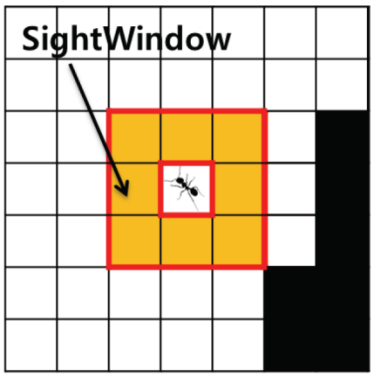


Figure 3.20 The HACO Sight Window. [38]

sight expansion is employed. Expanding the sight window starts with establishing the shortsighted ant's sight window. The borders of this window are moved by one cell in each

direction until one or more cells contain an obstacle as in figure 3.21. The general ant and super general ant can each choose any of the covered spaces that do not contain an obstacle. The right half of figure 3.21 shows the 23 allowable spaces for this example in orange. This gives them various speed and various sight capabilities. They also allow for a smoother path than the other species since more angles are represented in the additional cell choices. The far-sighted and super far-sighted ants are limited to the 16 border cells of the expanded sight window as shown in the left half of figure 3.21. This gives them far sight and high speed. The additional speed increases the chance that they will find the goal faster than the other species which reduces the computation time.

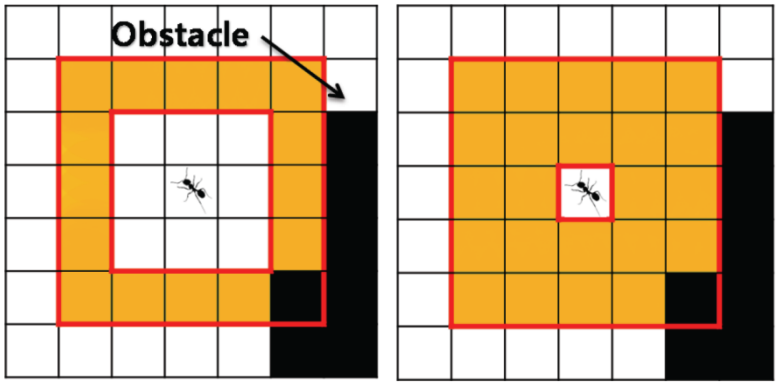


Figure 3.21 Expanded HACO Sight Windows. [38]

Using a sight window in this manner, leads to an interesting issue for pheromone allotment. If a far-sighted or various sighted ant picks a position along the outer boundary that is directly right, left, above, below or diagonal to its present position for its next move, it is easy to see what cells are in its path. The path goes through the center of each. This makes pheromone distribution simple. It is applied to each cell along the path. But where should the pheromone be applied if the path doesn't go directly along one of these paths? In this case, the path goes between some cells and not through their centers as in figure 3.22. The authors of [38] propose using interpolation as a solution. To accomplish this, a one dimensional function is proposed. The path is carefully followed from the original cell to the final cell in order to determine the

order that the intermediate cells are crossed. This allows the pheromone to be placed in the appropriate cells while providing an apparently continuous path. The resulting apparent route from the current cell to the next would be up, left and up as in the right half of figure 3.22. It is important to note that the actual route is directly from the current cell to the final cell. This is what the pheromone length and angle should be based on to prevent less optimum paths being preferred.

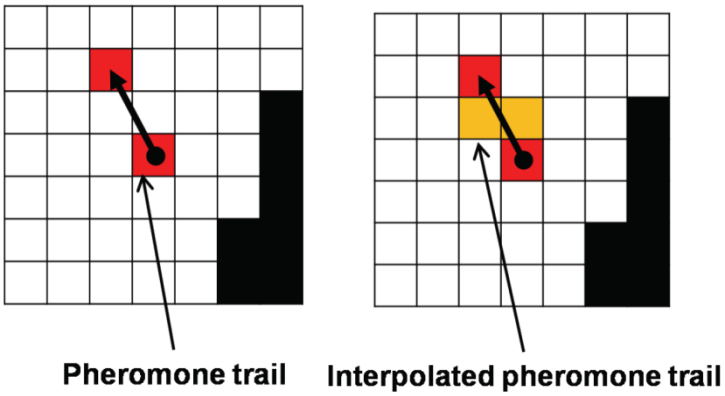


Figure 3.22 Interpolation of Pheromone Trail. [38]

The authors developed a modified transition probability function (MTPF) to pick the next cell for the ant:

$$p_{ij}^{k}(t) = \frac{[\tilde{\tau}_{ij}(t)]^{\alpha}[\tilde{V}_{jg}(j)]^{\beta}\,\tilde{\xi}_{ij}(j)}{[\sum_{k\in allowed_k}\tilde{\tau}_{ik}(t)]^{\alpha}[\tilde{V}_{kg}(k)]^{\beta}\,\tilde{\xi}_{ik}(k)}, \quad if \;\; j \in allowed_k \tag{3.33}$$

In this, $allowed_k$ is the set of k cells in the sight window. The current cell is i. The next cell is j. α and β are user selected constants that determine the relative weight of the pheromone and visibility functions. These two functions produce normalized values so α and β provide more consistent results in different environments. ξ is the angle factor given by:

$$\tilde{\xi}_{ij}(j) = \begin{cases} \dfrac{\theta}{\pi} & if \;\; \dfrac{\pi}{2} \le \theta \le \pi \\ 0.00001 & if \;\; 0 \le \theta \le \dfrac{\pi}{2} \end{cases} \tag{3.34}$$

The lower value is used for escape from a dead-end. Θ is the angle between the current direction of travel and the direction needed to reach the next cell.

τ is the pheromone level of the cell. Since there are no clearly defined generations and the number of ants in the environment can vary, the authors limit the pheromone level to [0.01, 1]. Within this range, the level changes according to:

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{L_{elite}(t+1)}{L_k}, & if\ \ kth\ ant\ uses\ edge(i,j)\ in\ its\ tour \\ 0, & otherwise \end{cases} \tag{3.35}$$

Here, t+1 is used to denote the time at which an ant arrives at the goal. This change from the formulas used in the prior ant colony methods takes into account the fact that, as stated before, there are no defined generations. $L_{elite}(t+1)$ is the length of the shortest path as of t+1 which includes the new path of the ant that reached the goal. The function also serves to normalize the pheromone change to the best path and reduce the effect of longer paths.

$V_{jg}(j)$ is the modified visibility between the next position j and the goal g. It is based off potential field theory. It is determined by:

$$\tilde{V}_{qg}(q) = \begin{cases} k_{mix} \cdot \tilde{V}_{att}(q) + (1-k_{mix}) \cdot \tilde{V}_{rep}(q), & if\ \ q\ is\ a\ non-obstacle\ cell \\ 0, & if\ \ q\ is\ an\ obstacle\ cell \end{cases} \tag{3.36}$$

In this formula, k_{mix} is a constant set to 0.5 by the authors. $V_{att}(q)$ is the normalized attractive visibility function with respect to the goal:

$$\tilde{V}_{att}(q) = 1 - \frac{d_{goal}(q)}{\max\limits_{i \in Field} \left| d_{goal}(i) \right|} \tag{3.37}$$

where $d_{goal}(q)$ is the Euclidean distance to the goal. $V_{rep}(q)$ is the normalized repulsive visibility function with respect to obstacles:

$$\tilde{V}_{rep}(q) = 1 - \frac{R(q)}{\max\limits_{i \in Field} \left| R(i) \right|} \tag{3.38}$$

where R(q) is given by:

$$R(q) = \begin{cases} \left(\frac{1}{d(q)} - \frac{1}{d_0} \right) \frac{1}{d^2(q)}, & if \ d(q) < d_0 \\ 1, & if \ d(q) \geq d_0 \end{cases} \quad (3.39)$$

In this equation, d(q) is the distance to an obstacle from point q and d_0 is the obstacle's distance of influence and can be adjusted to account for the robot's size.

Combined, these formulas form a very elaborate method of picking the next cell for the ant. On the positive side, they combine to form a method that takes into account the various items that impact the detection of the best path. This method also normalizes the items to make the method more universal with minimum adjustment.

As a final means of optimization, the heterogeneous ant colony method uses path crossover (PC) from genetic algorithms. When an ant arrives at the goal, its path is compared to the paths of other successful paths (those that reach the goal). Every path will have two common points at the start and goal. Any that have three or more points in common with the new path are candidates for PC. In this version, each possible crossover is performed and stored according to its final length along with other paths in the rank list. This helps to improve convergence speed to the optimal path. Zig-zag patterns are also reduced since they represent longer paths between two points.

The process of updating the pheromone level and rank list is performed in the following order:

1. An ant arrives at the goal.
2. PC is performed between the path of the newly arrived ant and a selected number of paths from the rank list.
3. The rank list is updated with the path of the newly arrived ant, the resulting PC paths and the paths of the prior rank list. The ranking is determined by:

$$f_{eval} = k_{length} \times (path\ length) + k_{\Delta angle} \times \sum \Delta angle \quad (3.38)$$

where Δangle is equal to π-θ and k_{length} and k_{angle} are positive integers. The rank list is arranged in order of f_{eval} where the path with the lowest f_{eval} has the highest rank. The length of the rank list is cut to a user selected value.

4. The pheromone field is updated.

The authors compared the results of their method with the results obtained from the use of a genetic algorithm using the two maps shown in figure 3.23. Tables 3.1 and 3.2 show

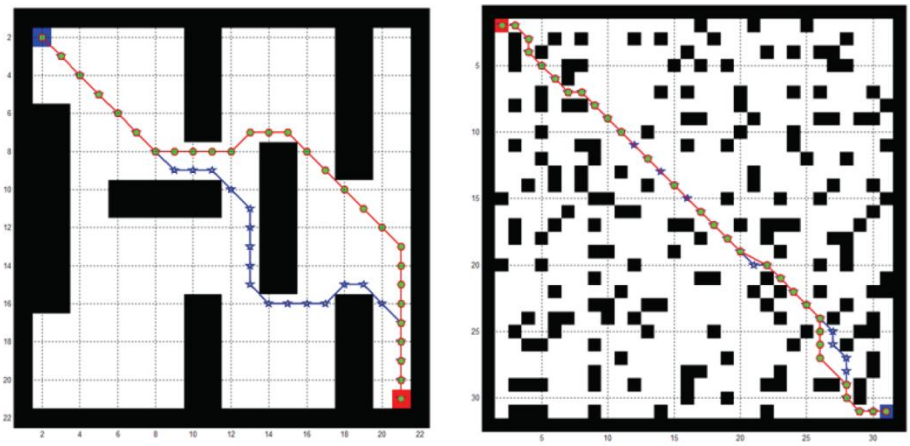


Figure 3.23 Results of GA (blue) and HACO (red) with Simple and Complex Maps. [38]

the relative performance of each approach. It can be easily seen that the two approaches achieve similar results with the HACO approach exhibiting slightly better performance.

Table 3.1 Results of Simple Map. [38]

	Computation time	Length of path
HACO	3.0195	32.3848
GA	4.062	32.5494

Table 3.2 Results of Complex Map. [38]

	Max	Min	Max-Min	Average
HACO	47.4979	43.763	3.7349	45.3697
GA	46.8701	43.9411	2.929	45.6768

A more subtle improvement using the HACO system is found in the number and size of turns between each of the solutions. The HACO has fewer turns with smaller angles. This results in a more efficient robot path since turns represent a greater loss of energy than a straight path.

3.4 MAKLINK Graph

The three forms of global path planning discussed so far have a certain degree of intelligence. They start by creating multiple entities that randomly explore the environment according to certain rules. These entities find one or more paths through the environment to the goal. As time progresses, the entities share their information in different ways to gradually agree on the best path. The more advanced versions of each form incorporate elements of the others to improve their performance. The problem is they each remain pseudo-random processes. While this helps them adapt to complex environments, it also means they can easily miss small paths between obstacles that are shorter than the ones that have been found. This means steps must be taken to avoid the resulting premature convergence. Unfortunately, many of these steps require more entities, generations or other solutions that increase processing time. Even so, the steps serve to increase the probability of successfully locating all paths but not ensure it. Establishing the initial paths in the first generation typically takes the greatest amount of time since the entities have no prior knowledge of successful paths to the goal. In order to foster diversity during the first generations, the path choices contain large, random elements for each of the entities. This means the resulting paths may contain portions that are counterproductive or even invalidate the entire path. The inefficiency and probability issues can be avoided if pathways that allowed access to every point in the environment were established first. This is where the MAKLINK graph comes in.

3.4.1 The Importance of the Convex Polygon

In order for a robot to travel from one position to another without collision, it is vital for the path to be clear of obstacles. If “free space” is defined as an area that contains no obstacles, then this is the same as stating that the path of the robot must remain in free space.

A first approximation of a robot's path is a series of straight lines. In order for the path to remain in free space, every point of each line must remain in free space. The challenge is to determine a method to identify the free space in a given area.

An ideal free space map would be one in which any point would be reachable from any other point with a straight line. This would make a circle the ideal shape. To generalize this a little, the line forming the circle's perimeter can be approximated to a series of lines forming a polygon as in the left half of figure 3.24. As with the circle, it can be seen that any line whose

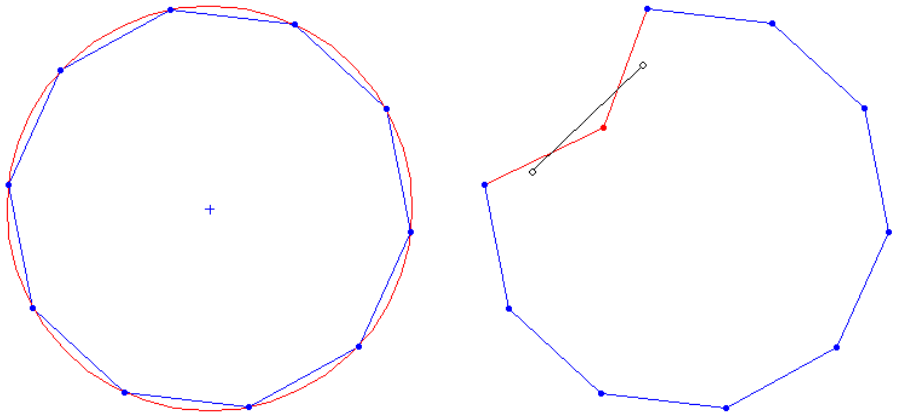


Figure 3.24 Approximation of a Circle (left) and Unacceptable Polygon (right).

end points fall within the polygon will have all of its points in the free space. This is not true of all polygons that fall inside the circle however. For example, if one of the vertices is moved toward the central region of the polygon as in the right half of figure 3.24, a line can be drawn whose points do not all fall within the polygon. This means not every point can reach every other point in the region by means of a straight line while remaining in the free space of the polygon. In other words, there exist some paths that will result in a collision with the boundaries of the free space. These boundaries can result from the shape selected for the free space itself or an obstacle intruding into the free space.

It was shown above that moving a vertex of an acceptable polygon can lead to an unacceptable polygon. A closer examination of this is in order. Assume there is a line segment

terminating in a vertex. There extends from the vertex a selection of line segments at various inside angles as with the black lines in figure 3.25. The free space (FS) is the area above or on the first segment extending around to the chosen second segment. There is a point along the first segment from which lines can be drawn to any point in the free space. The figure demonstrates that the chosen segment must have an angle less than or equal to 180 degrees. Any line segment with an angle greater than this will require the robot to exit the free space to reach the destination as the red line indicates. This and the prior discussion define the properties of a convex polygon.

1. Its interior is a convex set.
2. Every line segment between two vertices remains inside or on the boundary of the polygon.
3. Every internal angle is less than or equal to 180 degrees.

Any polygon that is not convex is concave. The important thing for this discussion is that a robot in a convex polygon can travel to any other point in the polygon without collision. In a concave polygon, this is not assured. This is why so many of the methods start with defining the obstacles as polygons. It is so that the entire free space may be more easily divided into adjoining convex polygons. This, in turn, allows a robot to reach any point in a contiguous free space from any other point in the free space without collision.

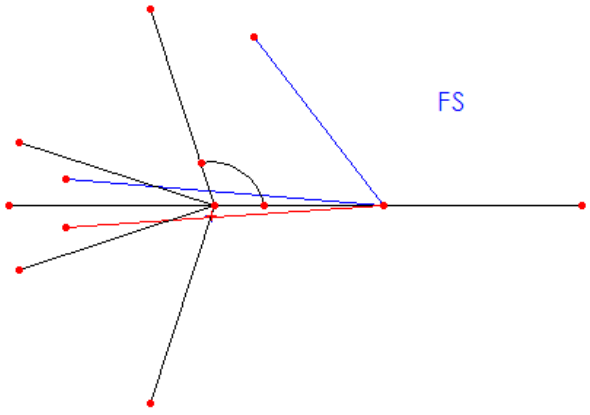


Figure 3.25 Angular Limits to Free Space Borders.

One approach to handling a complex free space map is to break it down into a collection of convex polygons. For instance, assume there is an object protruding into a convex free space as indicated by the black object in figure 3.26. It is clear that this makes the space concave. Further, the free space area can be divided in two if a line is drawn from the vertex of the object to the end of the original free space area. The challenge is to divide the space in a way that will result in the original area being subdivided so that only free space areas remain. Given the prior discussion regarding the angular limit of free space borders; it becomes clear that the line must reside in the gray area. This limit will allow the internal angles of both spaces to be less than or equal to 180 degrees thus dividing the space into lower and upper free space areas. In fact, there can be any number of these lines drawn at the same time creating any number of free space areas. This would be unnecessary though. In a complex environment, the challenge is to create a minimum number of convex free space areas. The first step in creating a MAKLINK graph is to use these concepts to create free links that define convex free space areas.

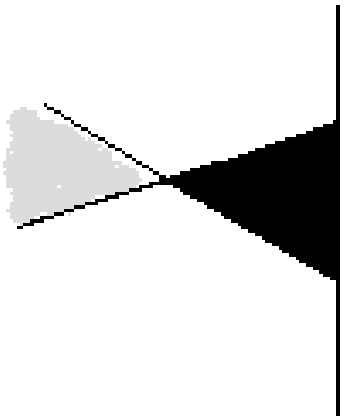


Figure 3.26 Dividing Free Space Area to Avoid Concavity.

3.4.2 Creating a MAKLINK Graph

A very good method for constructing a MAKLINK graph can be found in [46]. As with other methods, the authors simplify the environmental mapping with the following assumptions:

1. The environment is two dimensional and the obstacles are enclosed by polygons of sufficient size and shape that the 3D height is irrelevant. The border of the environment, or Working Space Boundary (WSB), is also represented by a polygon.
2. The robot is assumed to be a point. The borders of the obstacles are expanded to account for the real dimensions of the robot, its sensor characteristics and its maneuverability.

The environment is further simplified by representing the obstacles and the WSB by a collection of vertices:

$$O(i) = \{x(1)\ y(1),\ x(2)\ y(2),\ \ldots\ldots,\ x(n)\ y(n)\}$$

where O(i) is the set of vertices for obstacle i. The environment, or world, is the union of the obstacles and the Working Space Boundary which, as with the obstacles, is described by the set of its vertices:

$$W = \{WSB\ O(1)\ O(2)\ \ldots\ldots\ O(m)\}$$

The free space in this "world" is converted to free convex areas using a free link approach where the free links have the following characteristics:

1. A free link is a line between two vertices or between one vertex and a point on the working space boundary wall. The vertices can be on the same or different obstacles.
2. A free link marks the edge between two adjacent convex free space areas.
3. A free link must not intersect any edge of any obstacle in the environment. In other words, it cannot join two vertices that require the line to go through an obstacle.
4. The boundary of each convex free area must have at least two free links among its edges.

This, along with the discussion in the last section, describes the characteristics of free links and provides an idea of how they are used in converting the free space into adjoining convex free areas. Now what is needed is an actual process to create and select the free links. Again, [46] provides a detailed method:

1. Select a vertex from one of the obstacles. Find the lines that go from it to each of the other vertices on all the obstacles including the selected one. Also find the lines from the vertex perpendicular to each of the working space edges.
2. Sort all the lines from step 1 by length from shortest to longest.
3. Select the first line from the list.
4. Check the line to see if it intersects any of the obstacles. If it does intersect an obstacle, discard it as invalid and select the next line. Repeat this process until a valid line is found and proceed to the next step.
5. Check the angles on each side of the line to the edges of the obstacle connected to the selected vertex.
    a. If both angles are less than 180 degrees, it is selected as the best free link. All other free links from this vertex are ignored. Go to step 8.
    b. If one of angles is greater than 180 degrees, it is not the best. Add it to the list of free links for the selected vertex.
6. Draw the free links for the selected vertex from the list in 5. Measure the angle from one side of the obstacle (extending from the selected vertex) to the first line encountered. Measure the angle from that line to the next and continue until the second side of the obstacle is reached as in figure 3.27. If any angle is greater than 180 degrees, select the next line on the list from step 2 and go to step 4. Otherwise go to step 7.
7. Check to see if any of the free links for the vertex are redundant. If so, delete them.
8. Repeat steps 1 through 7 for each vertex of each obstacle.

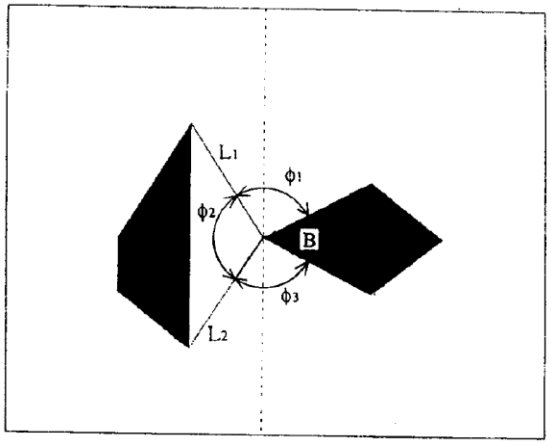


Figure 3.27 Determining Free Links around a Vertex. [46]

At this point, the map consists of obstacles and free convex areas marked by closed loop paths consisting of the free links, obstacle edges and the environment edges.

9. Mark the midpoints of each of the remaining free links as in figure 3.28.

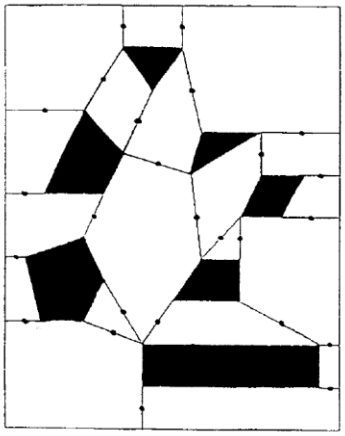


Figure 3.28 Selected Free Links with Midpoints Marked. [46]

10. Add the start and goal points to the graph.
11. Check to see if a straight line can be drawn between the start and goal points without intersecting any obstacles. If it can, the final path is the straight line. Otherwise, continue to the next step.

12. Replace the free link lines belonging to the border of the start point's free convex area with paths from the start point to the midpoint of each line. Repeat the process for the goal point.
13. Since each midpoint lies on the border of an adjacent free convex area. Repeat step 12 for each mid point's second free convex area. Continue until each midpoint of the graph has been processed. Ignore the fact that the free link lines of the final step will have already been replaced. In other words, the last step will just involve adding lines between points and not deleting lines.

The left half of figure 3.29 shows the MAKLINK graph that has been created less the start and goal points along with their lines. It can be seen that the start and goal points can reach each other from any position in the free space by travelling between midpoints. The last steps find the best collision-free midpoint path for the robot as shown in the right half of figure 3.29.

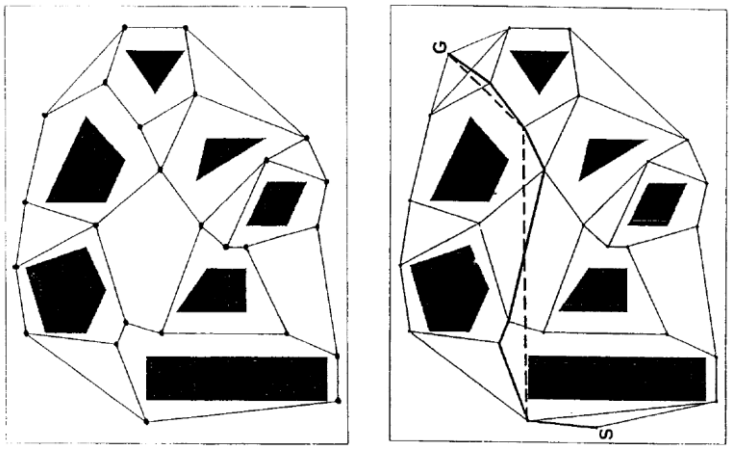


Figure 3.29 MAKLINK Graph Alone (left) and with Start, Goal and Path (right). [46]

14. Search the graph for paths between the start and goal points. The paths will be expressed as a sequence of midpoints and orientations.
15. Check to see if the paths can be reduced. For each path, remove each midpoint one at a time and see if the preceding and succeeding points can be connected without

intersecting an obstacle. If so, remove the point from the path. The shortest resulting path is selected as the best.

CHAPTER 4

A TWO STAGE, THREE STEP APPROACH

In the first section, potential field control was discussed. It was found that it is very well suited for local path planning where only a small portion of the environment is known. Its operation is consistent with the limited information acquired from sensors onboard the robot. It is also mathematically simple so it requires relatively few resources to implement. In spite of its simplicity, it allows a robot to avoid obstacles very well while it seeks its positional goal. There was even some discussion of minor modifications that can be implemented to help the robot avoid stall and trap situations. Those improvements do not make it foolproof though and the more complex the environment is, the more likely the robot is to get into trouble.

At the start of the second section, it was pointed out that robots are usually used in areas where the environment is largely known. Global path planning is more suitable in these situations. The four methods that were examined can be further divided into two types. First there are the heuristic methods. These methods start with pseudorandom processes to explore the environment and learn various suitable routes from the start to the goal. They then try to optimize the best path or paths found to select the optimum route. The problem here is that there is no guaranty that the globally optimum path will be found. The reason is that there is no guaranty that every path between every obstacle is explored. The only thing they can do with any certainty is optimize the best path that they discover. On the good side, they can handle much more complex environments than the potential field model. Since they rely on large numbers of entities in several generations, they are far more likely to find a valid path. Once they find a set of good paths, they are very good at optimizing them within the limits of the resolution of the environment representation. On the bad side, they require a large amount of processing. The greatest amount of the processing comes from finding an initial set of paths to

optimize. Particle swarm, genetic algorithms and ant colony optimization fall into this subcategory. Even though they take very different approaches with different strengths and weaknesses, it was seen that the most successful methods use elements of two or more of these systems.

The second type of global path planner is deterministic. The MAKLINK graph was examined for this section. This method redefines the free space as a set of adjacent convex polygons. As such, it is able to identify paths, around and between every obstacle in the environment that are wide enough to accommodate the robot. The paths allow travel between any two points in the contiguous free space of the environment by only adding lines from the established paths to the two points. It then coarsely optimizes the shortest of these available paths. While it identifies every general route in the environment with less processing requirements than the heuristic methods, it does not provide very good optimization.

When choosing a path planning method, the platform or hardware limitations of the robot must be considered as well. As discussed above, the different methods require different levels of resources. The complex models found in the heuristic, global set are the most sophisticated and generally provide the most optimum results given a sufficiently large set of entities and generations. They do so at the expense of processing, memory and time resources. This is not a problem for large, slow robots where weight, power and size constraints are not an issue. For small, quick robots, the situation is very different. A more powerful processor with external memory and accessories results in greater power consumption and weight. These may not be issues on a platform with large powerful motors but on a small system, the power source can represent a major portion of the weight. If the robot is also capable of flight, the limitations are even more severe. For these reasons, it is preferable to use as small a control system as possible. Ideally, this would consist of a single integrated circuit that requires very little path planning time.

4.1 Two Stages

As stipulated before, the environment is largely known in most cases. As such, the global path can be processed off line, while the robot is idle or even totally off board from the robot. In most variations of the heuristic models, the environment is expressed as a grid. The optimized output is a series of points on the grid. The number of points can be reduced by eliminating redundant points along any lines. This produces a compact path, or set of waypoints, that can be loaded into the mission manager of the robot. On a complex robot, the mission manager may be a separate processor or even a different on-board computer system. For a small robot, this can be as simple as a set of memory locations and a subroutine on the main control processor.

Whether the mission manager of a robot consists of a separate, complex processing system for continuously updating the best path or a simple routine to access stored points, its basic function is to feed new waypoints to the robot's control system at appropriate times. The control system must then drive the robot to the goal while avoiding any local obstacles that were not originally known to be part of the environment. Regardless of the mission manager's complexity, it must be onboard the robot along with the control system. For these reasons, it is proposed that the robot's mission be considered in two stages. The global planning stage can be run at any time prior to or during the mission to determine the best path through the known environment. This is the first stage. The mission manager is run during the mission to feed information to the control system. The control system directs the robot toward the goal while reacting to the information received from the local sensors. It uses local path planning for this. These two are the second stage. Figure 4.1 illustrates this configuration.

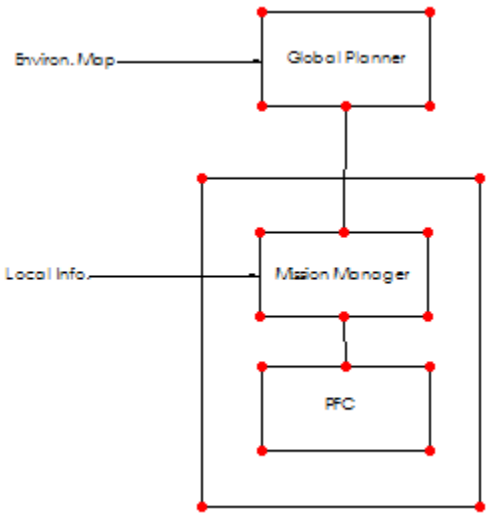


Figure 4.1 Two Stage Approach to Path Planning.

4.2 Three Steps

It was mentioned above that each of the path planning methods have strengths and weaknesses. Much research has been done to find ways to use multiple methods in a cooperative manner to capitalize on the strengths of each and minimize the weaknesses. This paper proposes a similar approach. To start with, the MAKLINK graph of the environment will be created to identify each of the coarse routes through the environment. The second step will use the Heterogeneous Ant Colony approach to optimize the path. This is similar to the method used in [45] but it is used in a completely different manner. In [45] the authors use the ant colony to find the best MAKLINK node (midpoint) order to the goal. In this paper, the ant colony is used to find the optimal path given a set of initial ant colony paths based on the MAKLINK graph. To form the transition from the MAKLINK graph to the initial ant colony path, a modified particle flow method is proposed. With the completion of the second step, the optimal global path is identified. This provides the data for the first stage of the preceding section. This path is used as a series of waypoints that is fed into the potential field control system described in the first section. This step allows the robot to proceed toward the goal while avoiding unexpected

obstacles in its path. The unexpected obstacles are a result of an imperfect map of the environment. This can be caused by initial inaccuracies of the map or changes in the environment since the creation of the map. The potential field system is the third step and the heart of the second stage described in the preceding section.

4.2.1 Coarse Global Routes

The proposed control method begins with certain basic assumptions:

1. The environment is polygonal.
2. The environment contains polygonal obstacles that can be sufficiently described in two dimensions.
3. The borders of the obstacles have been expanded to account for the size of the robot and its handling characteristics.
4. The global path planner is provided with a copy of the map.
5. The global path planner is provided with a list of the obstacle vertices.

The MAKLINK graph is formed using the method in [46] described above. The method is repeated here for reference:

1. Select a vertex from one of the obstacles. Find the lines that go from it to each of the other vertices on all the obstacles including the selected one. Also find the lines from the vertex perpendicular to each of the working space edges.
2. Sort all the lines from step 1 by length from shortest to longest.
3. Select the first line from the list.
4. Check the line to see if it intersects any of the obstacles. If it does intersect an obstacle, discard it as invalid and select the next line. Repeat this process until a valid line is found and proceed to the next step.
5. Check the angles on each side of the line to the edges of the obstacle connected to the selected vertex.

a. If both angles are less than 180 degrees, it is selected as the best free link. All other free links from this vertex are ignored. Go to step 8.
b. If one of angles is greater than 180 degrees, it is not the best. Add it to the list of free links for the selected vertex.

6. Draw the free links for the selected vertex from the list in 5. Measure the angle from one side of the obstacle (extending from the selected vertex) to the first line encountered. Measure the angle from that line to the next and continue until the second side of the obstacle is reached. If any angle is greater than 180 degrees, select the next line on the list from step 2 and go to step 4. Otherwise go to step 7.
7. Check to see if any of the free links for the vertex are redundant. If so, delete them.
8. Repeat steps 1 through 7 for each vertex of each obstacle.
9. Mark the midpoints of each of the remaining free links.
10. Add the start and goal points to the graph.
11. Check to see if a straight line can be drawn between the start and goal points without intersecting any obstacles. If it can, the final path is the straight line. Otherwise, continue to the next step.
12. Replace the free link lines belonging to the border of the start point's free convex area with paths from the start point to the midpoint of each line. Repeat the process for the goal point.
13. Since each midpoint lies on the border of an adjacent free convex area. Repeat step 12 for each mid point's second free convex area. Continue until each midpoint of the graph has been processed. Ignore the fact that the free link lines of the final step will have already been replaced. In other words, the last step will just involve adding lines between points and not deleting lines.

Figure 4.2 shows the resulting MAKLINK graph with the start and goal points. Ignore the dashed line as it is not relevant in this section.

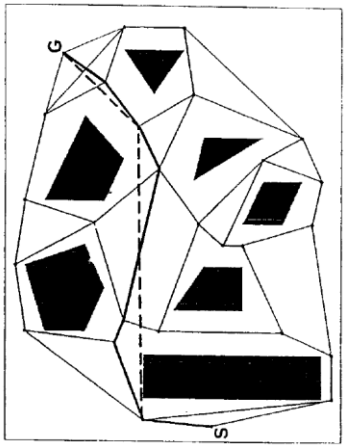


Figure 4.2 MAKLINK Graph with Start, Goal and Path. [46]

In [46], additional steps found the best coarse path. In the proposed method, ant colony optimization will be used to find the best optimal path. Like [46], the path must be changed to a series of midpoint locations in order to be useful to the ant colonies. This is where the proposed transition method starts.

The transition borrows from the particle flow methods but confines the particles to the paths established by the MAKLINK graph. To start the process, the environment is converted to a form suitable for the ant colony approach by overlaying a grid over the map. In order to ensure that the grid is sufficient to allow each path to be explored, the cells must be no larger than the shortest vertex line from step 2 above. To avoid difficulty in grid alignment, the cell size should be around one half to one third the length of the shortest line. Once the grid has been established, each vertex and midpoint is assigned to the proper grid location. Now the transition process can proceed as follows:

1. Select the start node and create one particle for each free link line. Record the start node position as the first path point for each particle.
2. Allow each particle to proceed to the next midpoint. Record the free link line that was followed, the angle of the path from the prior point to the midpoint and the midpoint that was reached in each particle's path.

3. For each particle that has not reached the goal, create new particles for each free link line except the one just used by the arriving particle. Create a path for each new particle consisting of the arriving particle's path. Then eliminate the arriving particle to avoid duplication.
4. If the goal has been reached by a particle, record the free link line that was followed, the angle to reach the goal from the last midpoint and the goal node in the particle's path. Record the nodes, angles and midpoints from the path as a successful route. Record any link lines from the path to the "included links" list that are not already on the list. Eliminate the particle.
5. If there are any link lines that are still not on the "included links" list, go to step 2.
6. If all link lines have been explored, as indicated by a complete "included links" list, send the "successful routes" list to the ant colony optimizer.

4.2.2 Optimized Global Routes

The ant colony optimizer takes the "successful routes" list, the grid and the environment map as inputs. It uses the Heterogeneous Ant Colony Optimization (HACO) system from [38] to find the globally optimal path. It can do this since the routes list contains all the coarse routes around the obstacles that the robot can successfully navigate as determined by the MAKLINK graph process. For the reasons mentioned before, HACO cannot ensure that all available paths are found on its own. The optimizer performs its function using the following process. More details can be found in the HACO section above.

1. Initialization:
    a. All grid cells are assigned a minimum pheromone level of .01.
    b. Initialize t.
    c. The length of each path from the "successful routes" list is determined. The shortest length is assigned to $L_{elite}(t+1)$.

d. The rank list is formed with the path of each successful route. The ranking of each is determined by:

$$f_{eval} = k_{length} \times (path\ length) + k_{\Delta angle} \times \sum \Delta angle \tag{3.38}$$

where delta angle is equal to π-θ and k_{length} and k_{angle} are positive integers. The rank list is arranged in order of f_{eval} where the path with the lowest f_{eval} has the highest rank. The length of the rank list is set to the number of successful routes. This ensures that all of the paths are considered as candidates.

e. The pheromone field is updated (using interpolation as required) per:

$$\Delta \tau_{ij}^{k} = \begin{cases} \dfrac{L_{elite}(t+1)}{L_k}, & if\ \ kth\ ant\ uses\ edge(i,j)\ in\ its\ tour \\ 0, & otherwise \end{cases} \tag{3.35}$$

f. Place the desired number of the various types of ants at the start point.

g. Clear ant to goal flag.

h. Clear iterations.

i. Clear generations.

j. Clear stable result count.

2. Optimization:

a. Select the next ant(k).

b. Fill $allowed_k$ with the cells available for the ant's next move based on the ant's type and sight window expansion where appropriate.

c. Randomly choose the next location for the ant from $allowed_k$ based on the probability function:

$$p_{ij}^{k}(t) = \frac{[\tilde{\tau}_{ij}(t)]^{\alpha}[\tilde{V}_{jg}(j)]^{\beta}\,\tilde{\xi}_{ij}(j)}{[\sum_{k \in allowed_k} \tilde{\tau}_{ik}(t)]^{\alpha}[\tilde{V}_{kg}(k)]^{\beta}\,\tilde{\xi}_{ik}(k)}, \quad if\ \ j \in allowed_k \tag{3.33}$$

where

$$\tilde{\xi}_{ij}(j) = \begin{cases} \dfrac{\theta}{\pi} & if\ \dfrac{\pi}{2} \le \theta \le \pi \\ 0.00001 & if\ 0 \le \theta \le \dfrac{\pi}{2} \end{cases} \tag{3.34}$$

$$\tilde{V}_{qg}(q) = \begin{cases} k_{mix} \cdot \tilde{V}_{att}(q) + (1 - k_{mix}) \cdot \tilde{V}_{rep}(q), & if\ q\ is\ a\ non-obstacle\ cell \\ 0, & if\ q\ is\ an\ obstacle\ cell \end{cases} \tag{3.36}$$

$$\tilde{V}_{att}(q) = 1 - \frac{d_{goal}(q)}{\max\limits_{i \in Field} |d_{goal}(i)|} \tag{3.37}$$

$$\tilde{V}_{rep}(q) = 1 - \frac{R(q)}{\max\limits_{i \in Field} |R(i)|} \tag{3.38}$$

$$R(q) = \begin{cases} \left(\dfrac{1}{d(q)} - \dfrac{1}{d_0} \right) \dfrac{1}{d^2(q)}, & if\ d(q) < d_0 \\ 1, & if\ d(q) \ge d_0 \end{cases} \tag{3.39}$$

d. Add the ant's current position (before update) and the angle to the next position to the ant's path.

e. Move the ant to its next location.

f. If the ant has reached the goal:

   i. Set ant-to-goal flag.

   ii. PC is performed between the path of the newly arrived ant and a selected number of paths from the rank list.

   iii. Copy the rank list to prior rank list. Copy L_{elite} to prior L_{elite}

   iv. The rank list is updated with the path of the newly arrived ant, the resulting PC paths, the mutated paths and the paths of the prior rank list. The ranking is determined by:

$$f_{eval} = k_{length} \times (path\ length) + k_{\Delta angle} \times \sum \Delta angle \tag{3.38}$$

where Δangle is equal to the angle to the prior point – the angle to the new point and k_{length} and k_{angle} are positive integers. The rank list is arranged in order of f_{eval} where the path with the lowest f_{eval} has the highest rank. The length of the rank list is cut to a user selected value.

v. The pheromone field is updated for each point on the path per:

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{L_{elite}(t+1)}{L_k}, & if\ \ kth\ ant\ uses\ edge(i,j)\ in\ its\ tour \\ 0, & otherwise \end{cases} \tag{3.35}$$

vi. Delete the current ant(k) and place a new ant(k) at the start.

g. Go to 2.a. until all ants have been processed.

h. Reset k and increase iteration.

i. If max iterations is reached, all ants are lost, trapped, etc. Path with lowest f_{eval} is best. Go to 2.l.

j. If the ant to goal flag is cleared, no ants have found the goal yet. Go to 2.a.

k. If the ant to goal flag is set, one or more ants reached the goal:

   i. Clear the ant to goal flag.

   ii. Increment generations.

   iii. If generations = maximum number of generations, the ants have found the best path they can in the designated time. The path with the lowest f_{eval} is the best. Go to 2.l.

   iv. If |prior L_{elite}-L_{elite}| < the minimum significant change and the prior rank list is the same as the rank list, increment the stable result count.

   v. If stable result count is > maximum number of stable generations, the ants have converged on the best path. The path with the lowest f_{eval} is the best. Go to 2.l.

   vi. If the prior three conditions have not been satisfied, go to 2.a.

l. Check the best path to remove consecutive points on a straight line:
    i. Pick first angle from path.
    ii. If next angle = selected angle, delete the selected angle and the point following it in the list.
    iii. If point prior to next angle in the path is the goal, straight lines have been eliminated. Output path to mission manager. End process.
    iv. If prior condition is not satisfied, set next angle from path as selected angle. Go to 2.l.ii.

4.2.3 Local Control

At a minimum, the robot platform control system consists of the mission manager, controller and hardware systems to operate the robot. The focus of this paper is the intelligent motivational control of the robot. As such, it is sufficient to assert that the hardware of the control system acts on an input indicating direction and magnitude of the desired force from the controller. It is further assumed that the sensor system provides obstacle direction and distance information to the mission manager which is aware of the maximum range of the sensors. For the system being described, the mission manager stores the waypoints developed by the optimization system above. It also monitors the control system location, velocity and force components to determine if a false obstacle should be created. It provides the waypoints, the real obstacle information and false obstacle information to the potential field controller (PFC) according to the following process:

1. Provide the initial waypoint beyond the starting point of the robot to the potential field controller (PFC) as its goal.
2. Provide the obstacle location information to the PFC from the short range sensors.
3. Receive robot velocity and force information from the PFC.
    a. If both drop below the trap detection thresholds and the robot goal force (without obstacles) is above the user selected value, a trap has occurred.

i. If a trap has been detected, create a new false obstacle in the direction of the PFC goal plus a user selected small angle ε_a and a user selected distance beyond the clearance radius of the robot.

ii. Provide the false obstacle to the PFC.

b. If both drop below the trap detection threshold and the robot goal force (without obstacles) is below the user selected value, the robot is near the goal but the false obstacles may be offsetting the local force minimum at the PFC goal.

i. Clear the PFC false obstacle information.

4. If the PFC goal has not been reached, go to step 2.
5. If the PFC goal has been reached and it is not the global goal, select the next waypoint (or global goal if it is next) as the PFC goal. Provide the new PFC goal to the PFC. Clear the PFC false obstacle information. Go to step 2.
6. If the PFC goal and next waypoint (global goal) are within the range of the local sensors and there are no obstacles in the direction of the next waypoint (global goal) and closer than the next waypoint (global goal), select the next waypoint (global goal) as the PFC goal. Provide the new PFC goal to the PFC. Clear the PFC false obstacle information. Go to step 2. This step can provide additional smoothing of the robot path.
7. If the PFC goal has been reached and it is the global goal, the robot has reached its targeted destination. Stop the robot.

While the mission manager process is running, the PFC will convert the goal and obstacle information to force outputs to the control system hardware using the following equations from the potential field discussion:

$$\vec{F}(\vec{r}) = \vec{F}_{target}(\vec{r}) + K_{v1} \cdot \vec{v} \cdot \sum_{i=1}^{n} \vec{F}_{obs_i}(\vec{r}_i) + K_{v3} \cdot \sum_{j=1}^{n} \vec{F}_{fobs_j}(\vec{r}_j) - K_{v2} \cdot \vec{v} = -\vec{\nabla} V(\vec{r}) \tag{2.17}$$

where:

$$F_{Gx}(x,y) = K_G \frac{x_G - x}{r_G}$$

$$F_{Gy}(x,y) = K_G \frac{y_G - y}{r_G} \qquad (2.18)$$

$$F_{ix}(x,y) = -K_i \frac{x_i - x}{(r_i - a)^2}$$

$$F_{iy}(x,y) = -K_i \frac{y_i - y}{(r_i - a)^2} \qquad (2.19)$$

the r_G and r_i in these equations is:

$$r_k = \sqrt{(x_k - x)^2 + (y_k - y)^2}$$

The variables in these equations are defined in detail in chapter 2. As suggested in the potential field discussion, one may handle uneven terrain conditions by assuming a certain robot mass and designing the control hardware to treat the force output as an acceleration control input rather than a force input.

4.2.4 Conclusion

This system starts with a polygonal map of the environment and its set of vertices. It locates all the coarse paths between start, goal, obstacles and edges of the environment that the robot can follow. It then identifies and optimizes the best of these global paths. This process can be done on or offline. Once the global path is identified, the points on the path are fed to a local control system comprised of a mission manager and potential field controller (PFC). The mission manager oversees the operation and feeds waypoints and the global goal to the PFC as its progress dictates. It also introduces false obstacles to the PFC if it gets trapped without reaching a waypoint or the global goal and removes them as each PFC goal is reached. Finally, it determines when the mission is accomplished and stops the robot. The PFC converts the obstacles and goals into force signals based on their positions and the robot's velocity to allow it to react to unknown conditions in the environment dynamically. It combines these forces to produce an output to the control hardware that determines the magnitude and direction of the

robot's acceleration. The fact that the complex part can be handled offline allows a robot with limited resources to travel through its environment in much the same way as its much more sophisticated relatives.

CHAPTER 5

SAFETY AND SECURITY CONCERNS WITH AUTONOMOUS ROBOTS

5.1 Civilian Applications

There are a tremendous number of applications for a comprehensive autonomous vehicle guidance system presently and in the future. Some current systems include autonomous vacuum cleaners, mowers, industrial logistics robots, water quality monitoring submersible robots and various military vehicles. In each case, engineers must remain aware that there are potential costs associated with their operation that can far exceed the value of the robot or vehicle. Industrial systems must be designed to ensure that employees remain free from danger. This may require that they halt operation when people enter the area or they may require advanced sensors to detect when people are in their way and stop at a safe distance or find a safe path around them. In the later case, the default must be to assume an unidentified object is a person and avoid hitting it. If a safe path is chosen, it must be selected with a person's unpredictability in mind. When a robot is used in homes, the safety question is of highest importance. In these situations, young children and pets may see it as a toy an attempt to play with it. This can be a particular challenge to the engineer as he or she must consider every possible danger mode. Even unlikely scenarios must be examined.

5.2 Military Applications

As terrible as an injured child would be, the stakes surrounding military equipment are much higher. A failure in a foreign country can create an international incident that could lead to war. In an existing war, it can potentially affect the support of allies or draw other nations to the side of the enemy. Perhaps worse than this are the long term implications of allowing advanced technology to fall into an enemy's hands. This is especially important when the technology is so

sensitive that it is not even shared with a nation's allies. Some examples of the effect of technology falling into a potential adversary's hands illustrate these long term impacts.

In 1976, a Soviet pilot defected to Japan with his MIG-25. Upon seeing that the systems were still vacuum tube based, the Japanese and American experts assumed the Soviets must have been further behind technologically than they had thought. The pilot brought up some points that the others had not considered together. It was well understood that an EMP pulse will destroy silicon based components. On the other hand, vacuum tube systems experience a momentary interruption as the pulse occurs and then return to normal operation. Whereas Japan, America and their allies expected a nuclear exchange to end a war, the Soviets were prepared to fight through it. This information resulted in the United States developing a host of tests, requirements and methods to ensure that an EMP pulse would not destroy their military capability in spite of silicon based circuitry.

Until the 1980's the United States was able to easily track Soviet submarines while U.S. submarines were difficult to track. Then, over a relatively short period, the Soviet submarines became quiet. In 1985, the reason became apparent after the arrest of U.S. Navy Chief Warrant Officer John Walker Jr. He had been spying for the Soviet Union since 1967. Among the many secrets he had provided the Soviet Union were the fact that the U.S. was able to track their submarines by the cavitation of their propellers and the specifics of the design the U.S. was using to avoid the same problem. The Soviets were able to copy the design so their submarines became quiet and the U.S. tightened its information security requirements.

More recently, an American remote controlled RC-170 Sentinel UAV was flying near the Iranian border in Afghanistan in December of 2011. It was captured intact by Iran. There has been much speculation about how it happened. Iranian sources report that they had blocked the command signals of other Sentinels and saw that they attempted to return to base. On this flight, they claim to have also fed it false GPS signals to route it to a location they desired. U.S. sources claim that military UAV aircraft typically use inertial navigation as a primary system to

return home if they lose command communication. Other stories report that the Sentinel is programmed to land safely if it experiences a critical failure and this is likely what happened. Regardless, the end result is the same. An unfriendly nation captured a critical piece of technology largely intact. It is still unknown whether critical hardware and software remained intact. Clearly, it would have been better if were programmed to crash in a field if it could not make it back to base or burst into flames upon landing if the inertial navigation and external location signals disagreed by hundreds of miles unless a “safe” signal was received by an independent radio receiver. The long term impact of this is yet to be determined but it emphasizes the point that as amazing as autonomous and semi-autonomous vehicles are, their potential damage can far exceed their cost on many fronts. The safety of non-combatants and the security of the craft’s information must always receive top priority even over the safety and cost of the vehicle. At the same time, the designer must also take into account that an adversary’s engineers will seek to defeat or capture the system. While the methods used to address these concerns are beyond the scope of this paper, it is important to note them in a work that deals with the guidance of such systems.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

This thesis started with a study of existing literature on the use of artificial potential fields for robot control. It examined several issues related to them and various attempts to solve those issues. Some of the solutions proved to be rather calculation intensive. Others abandoned the model entirely during the implementation of the solution only to return after escaping the problem area.

The paper continues by developing the potential field model to enable autonomous machines, or robots, to successfully navigate through their environment toward a target while avoiding obstacles. Some of the potential problems of the standard model were examined in greater detail. First, the model allowed the robot to oscillate about the target or goal. Second, the robot could stall due to local minima in the field. Finally, the robot could become trapped by obstacles that are too close together for the robot to pass. The solution of these resulted in an improved potential field model with very little additional overhead.

After solving the primary issues associated with the potential field model, the paper described an alternate approach to improve the system performance for use on digital platforms such as microprocessors, microcontrollers and DSP circuits. From there, the use of emotion-based control was examined for issues that may be related to the environment but not detected in advance such as excessively hot or other dangerous areas. It was shown that the results of an emotion system could be put in terms of obstacle or target inputs to the potential field controller.

Even though the solutions that were used in the thesis represent an improvement over existing methods, they could still lead to problems if not handled with care. This led to a detailed

discussion of several variations of three popular heuristic global path planning methods along with their strengths and weaknesses. It was shown that significant improvements have been realized by combining aspects of different methods to capitalize on the strengths of each and correct for their weaknesses. It was pointed out that in spite of these gains, combining heuristic methods may improve performance but could not guaranty the best path is found. With this in mind, the deterministic MAKLINK graph was introduced. It was shown that even though it fails to provide an optimum path from the start to the goal, it is capable of finding every viable, coarse path from any free space point in the environment to any other free space point in the environment.

Once the various methods of solving the problem of path planning were explored, the original question remained, “how to find a safe path through the environment in a way that can be employed on a robot with limited resources?” To answer this, a three step method was developed to combine the positive aspects of the MAKLINK graph, ant colony optimization and the potential field approach. These were combined in a manner that allowed a convenient and reasonable division between the global path planning, which can be done off line in advance, and local robot motivation which must be done in real time.

The work concluded with a cautionary note on the concerns surrounding the use of autonomous and semi-autonomous vehicles in real world scenarios. While the solutions are beyond the scope of the paper, their impact is within the scope of applications that incorporate the material. Finally, the paper included an appendix which described several two dimensional platforms the model could be used to control.

6.2 Future Work

Future work in this area can include the examination of different shaped obstacle and target fields that could be useful for different scenarios. A couple of these include zonal coverage in sport or military applications and achieving orbit around a target instead of merging with it. Group or fleet control can be examined using individual robots with potential field

controllers. This can be expanded to examine scenarios where one or more leaders exist and the remaining robots follow their lead. The leaders may have the same or different targets. Cooperative and adversarial relationships between robots with different, interrelated missions can be examined as well. Each of these instances will likely introduce new issues to be examined as well. Also, the model can be adapted with a little effort to a three dimensional environment and the typical platforms used in such environments may be discussed.

Many of the solution's parts contain customized variables that may be set at the user's discretion. It would prove very beneficial to explore methods to automate or optimize their settings based on the nature of the robot and its environment. Perhaps the settings may be adjusted according to environmental patterns that can be discovered in the MAKLINK stage. Perhaps the ant colony algorithm can be further optimized based on the fact that its initial conditions include all possible coarse routes through the environment. It may also prove useful to take a closer look at the mission manager. It may be useful to allow it to decide between several optimized routes based on environmental conditions encountered during real time operation. Finally, it is important to point out that an ideal method will allow a robot to start with a partially correct, actual map of the environment, analyze the map, develop a set of alternate paths and sub paths to the goal and guide the robot to the goal while correcting for unanticipated changes in the environment. This means that additional work may also include exploration into how a human readable map, that includes non-polygonal objects, may be converted into a MAKLINK or alternative style of map. This will allow a generalized method for the robot to plan and execute a movement through an environment from start to goal with minimal additional processing or outside intervention even when the robot is relatively small and fast.

APPENDIX A

VEHICLE MODELS

The potential field model used in the last part of the proposed system has proven very versatile in the simulation of a robot motivational system's actions in a complex environment. In order for it to be useful though, it must be connected to a real platform. The model ultimately outputs a force or acceleration in the x and y directions. With these values, the angle and magnitude of the force output can also be found. Most platforms, on the other hand, are wheeled systems that require drive and steering torques or drive and steering velocities. As such, there must be an interface between the potential field system and the mechanical components of the platform. This is the function of the drive system. The various analog and digital approaches to the implementation of a drive system are beyond the scope of this paper but some characteristics of the conversion process can be noted. Torque is defined as force times distance about an axis. If the distance is assumed to be constant, then the drive force can be seen as a constant gain (the assumed distance) applied to the magnitude of the potential field system force. The angle of the potential field system output is the desired angle for the platform. The difference between the desired angle and the platform angle is the error. The torque is the force required to drive the platform's steering system to minimize this error. A PID system may be used to ensure a smooth operation of both platform parameters.

Alternatively, some platforms require a drive velocity and steering velocity as inputs. Since force is a function of acceleration when the mass is fixed and velocity is the integral of acceleration over time, a PID can provide this conversion. Steering velocity is the angular velocity of the platform. Angular velocity is the derivative of the angular position with respect to time. Again, a PID can provide this conversion. It should be noted that there are other, more optimal systems that can be used but the PID remains a very common type of controller in industry. Whether a PID or some other analog or digital system is used, it must also take into account the platform's limitations so as not to overdrive the system, introduce integration

windup issues, etc. There are many different types of platforms with different characteristics. Some of the most common are differential drive vehicles, car-like vehicles and tractor-trailer systems. [15] presents a very good description of each. Their characteristics, as detailed in the book, are discussed in the following sections.

A.1 Differential Drive Vehicle

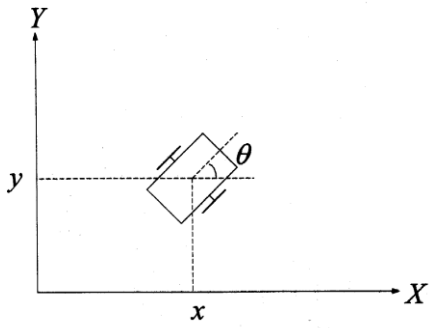


Figure A.1 Differential drive vehicle.

In its simplest form, a differential drive system has two wheels, one on each side. This can be extended to vehicles with more than one wheel on each side or even tracked vehicles such as a military tank so long as each wheel is the same size as the other wheels on that side of the vehicle and its angular velocity is the same as the others on that side. The center of the vehicle is the reference point and is assumed to be the center of mass. Its relevant coordinates are:

$$q = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{A.1}$$

Here, θ is the heading angle of the vehicle relative to the x axis. For the sake of simplifying the vehicle's model, it is assumed that it does not slide sideways. Mathematically, this gives the constraint:

$$\dot{x}\cdot\sin(\theta)-\dot{y}\cdot\cos(\theta)=0$$

or

$$0=[\sin(\theta)\quad -\cos(\theta)\quad 0]\cdot\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix}\overset{\Delta}{=}A(q)\,\dot{q}$$

Since

$$A(q)\cdot G(q)=0$$

Then

$$G(q)=\begin{bmatrix}\cos(\theta) & 0\\ \sin(\theta) & 0\\ 0 & 1\end{bmatrix}=B(q) \tag{A.2}$$

Given a driving velocity (u_1) and a steering velocity (u_2), the kinematic model is:

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix}=\begin{bmatrix}\cos(\theta) & 0\\ \sin(\theta) & 0\\ 0 & 1\end{bmatrix}\cdot\begin{bmatrix}u_1\\ u_2\end{bmatrix} \tag{A.3}$$

The driving velocity (u_1) and steering velocity (u_2) are actually functions of the left and right wheel angular velocities (ω_l and ω_r) and the radius of the wheels (ρ):

$$u_1=\frac{\rho}{2}\cdot(\omega_r+\omega_l)$$
$$u_2=\frac{\rho}{2}\cdot(\omega_r-\omega_l) \tag{A.4}$$

The best way to get the dynamic model is to first look at the energy of the system. Since there is no potential energy in the system, this breaks down to the kinetic energy of the vehicle. This is

the sum of the linear kinetic energy and the rotational kinetic energy. The linear kinetic energy is a function of the mass and linear velocity of the vehicle. The rotational kinetic energy is a function of the angular velocity and the inertia with respect to the vertical axis running through the center of the vehicle. This gives the Lagrangian:

$$L = \frac{1}{2} m \cdot (\dot{x}^2 + \dot{y}^2) + \frac{1}{2} J \cdot \dot{\theta}^2$$

or

$$M \ddot{q} = A^T(q)\lambda + B(q)\tau \tag{A.5}$$

This gives

$$\begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & J \end{bmatrix} \cdot \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} \sin(\theta) \\ -\cos(\theta) \\ 0 \end{bmatrix} \cdot \lambda + \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \tag{A.6}$$

In these equations, M is the inertia matrix, λ is the Lagrangian multiplier, τ_1 is the driving force and τ_2 is the steering torque. To further simplify this, the reduced-order dynamic model is used. This model is given by the set of equations:

$$\dot{q} = G(q) \cdot u$$

$$M'(q) \cdot \dot{u} + N'(q,u) = G^T(q) \cdot B(q) \cdot \tau$$

Where

$$M'(q) = G^T(q) \cdot M(q) \cdot G(q)$$

$$N'(q,u) = G^T(q) \cdot M(q) \cdot \dot{G}(q) \cdot u + G^T(q) \cdot N(q, G(q) \cdot u) \tag{A.7}$$

While this looks complicated, solving the equations by parts quickly reduces the system:

$$G^T(q)\cdot B(q) = I_2$$

$$G^T(q)\cdot M(q)\cdot \dot{G}(q) = 0$$

$$G^T(q)\cdot M(q)\cdot G(q) = \begin{bmatrix} m & 0 \\ 0 & J \end{bmatrix}$$

Thus

$$\begin{bmatrix} m & 0 \\ 0 & J \end{bmatrix}\cdot \dot{u} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\cdot \tau \qquad \text{(A.8)}$$

This gives a final model consisting of the following simple equations:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix}\cdot \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

and

$$\begin{bmatrix} m & 0 \\ 0 & J \end{bmatrix}\cdot \begin{bmatrix} \dot{u}_1 \\ \dot{u}_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\cdot \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \qquad \text{(A.9)}$$

A.2 Car-like Vehicle

A car-like vehicle has two sets of wheels. The rear wheels can rotate but otherwise remain fixed with respect to the vehicle. They are mounted across from each other and the line connecting their centers remains perpendicular to the heading of the vehicle. The front wheels are mounted similarly but may be turned about a perpendicular axis. They are responsible for changing the heading of the vehicle. It should be noted that in practice, the steering axis is not always vertical but this assumption simplifies the model. Typically, each set of wheels reside on

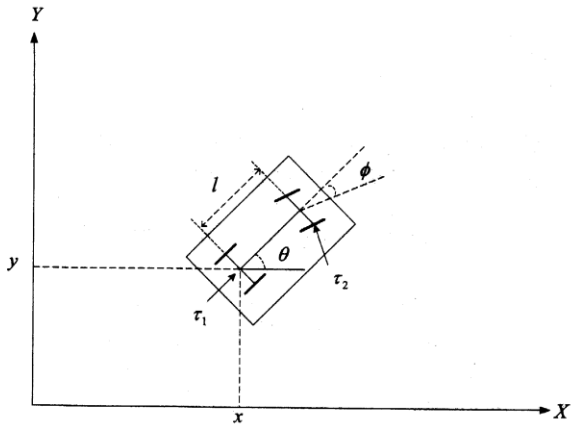


Figure A.2 Car-like Vehicle.

an axle. The distance between the two axles is l. The center of the rear axle is the reference point. Its relevant coordinates are:

$$q = \begin{bmatrix} x \\ y \\ \theta \\ \phi \end{bmatrix} \tag{A.10}$$

Here, θ is the heading angle of the vehicle relative to the x axis and ϕ is the steering angle. As with the differential drive vehicle, it is assumed the wheels do not slide. Since two sets of wheels are involved, this results in the following two constraints:

$$\begin{aligned} v_x^f \cdot \sin(\theta + \phi) - v_y^f \cdot \cos(\theta + \phi) = 0 \\ v_x^b \cdot \sin(\theta) - v_y^b \cdot \cos(\theta) = 0 \end{aligned} \tag{A.11}$$

Since the reference is the center of the rear axle, its position is (x,y). Since the heading of the vehicle is along the line perpendicular to the rear axle and that such a line going through the reference point will intersect the front axle at its center, it is natural to select this as a secondary reference for calculations involving the front wheels. The midpoint for the front axle is (x+lcos(θ), y+lsin(θ)). This gives front and rear axle velocities:

$$\begin{aligned} v_x^b &= \dot{x} \\ v_y^b &= \dot{y} \\ v_x^f &= \dot{x} - l \cdot \dot{\theta} \cdot \sin(\theta) \\ v_y^f &= \dot{y} - l \cdot \dot{\theta} \cdot \cos(\theta) \end{aligned} \tag{A.12}$$

These, in turn, give the constraints in matrix form:

$$0 = \begin{bmatrix} \sin(\theta+\phi) & -\cos(\theta+\phi) & -l \cdot \cos(\phi) & 0 \\ \sin(\theta) & -\cos(\theta) & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} \stackrel{\Delta}{=} A(q)\dot{q} \tag{A.13}$$

From this, G(q) can be found which is used in the generalized kinematic model:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ \frac{1}{l} \cdot \tan(\phi) & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

Where

$$u_1 = \sqrt{\dot{x}^2 + \dot{y}^2}$$
$$u_2 = \dot{\phi} \qquad \text{(A.14)}$$

This makes u_1 the driving velocity and u_2 steering velocity. As the wheels do the work for the vehicle, u_1 and u_2 need to be converted so that they represent the angular velocities of the wheels. This will result in a specific model that differs between a rear wheel drive and a front wheel drive vehicle. For a rear wheel drive vehicle where ω_1 is the angular velocity of the back wheels, ω_2 is the steering rate of the front wheels and ρ is the radius of the back wheels, u_1, u_2 and the model become:

$$u_1 = \rho \cdot \omega_1$$
$$u_2 = \omega_2$$

So

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \rho \cdot \cos(\theta) & 0 \\ \rho \cdot \sin(\theta) & 0 \\ \frac{\rho}{l} \cdot \tan(\phi) & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \omega_1 \\ \omega_2 \end{bmatrix} \quad \text{(A.15)}$$

Since tan(+/-π/2) is undefined as the steering wheels are perpendicular to the force from the back wheels, ϕ is limited to falling between -π/2 and +π/2 (not inclusive). For a front wheel drive vehicle, ω_1 is the angular velocity of the front wheels, ω_2 is the steering rate of the front wheels and ρ is the radius of the front wheels. This gives the u_1, u_2 and model below:

$$u_1 = \rho \cdot \omega_1 \cdot \cos(\phi)$$
$$u_2 = \omega_2$$

So

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \rho \cdot \cos(\theta) \cdot \cos(\phi) & 0 \\ \rho \cdot \sin(\theta) \cdot \cos(\phi) & 0 \\ \frac{\rho}{l} \cdot \sin(\phi) & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \omega_1 \\ \omega_2 \end{bmatrix} \quad \text{(A.16)}$$

Solving for the total kinetic energy in order to find the dynamic model gives:

$$L = \frac{1}{2} m \cdot (\dot{x}^2 + \dot{y}^2) + \frac{1}{2} (J_b \cdot \dot{\theta}^2 + J_s \cdot \dot{\phi}^2) \tag{A.17}$$

Where J_b is the vehicle's rotational inertia about its reference point and J_s is the steering mechanism inertia. As with the differential drive vehicle, the dynamic equation is:

$$M \ddot{q} = A^T(q)\lambda + B(q)\tau$$

Where

$$M = \begin{bmatrix} m & 0 & 0 & 0 \\ 0 & m & 0 & 0 \\ 0 & 0 & J_b & 0 \\ 0 & 0 & 0 & J_s \end{bmatrix}$$

$$B(q) = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \tag{A.18}$$

Even though G(q) is different for the two types of car-like vehicles, the reduced order dynamic model comes out the same:

$$(m + \frac{J_b}{l^2} \cdot \tan^2(\phi)) \cdot u_1 = \frac{J_b}{l^2} \cdot u_1 \cdot u_2 \cdot \tan(\phi) \cdot \sec^2(\phi) + \tau_1$$

and

$$J_s \cdot \dot{u}_2 = \tau_2 \tag{A.19}$$

A.3 Tractor-Trailer Vehicle

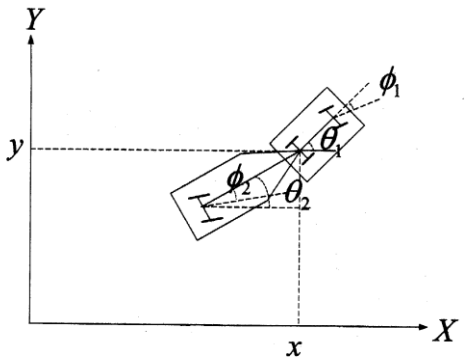


Figure A.3 Tractor-trailer Vehicle.

Typically, a tractor-trailer system consists of a car-like tractor vehicle with a trailer attached behind the rear axle. For simplicity, it will be assumed that the trailer is attached at the center point of the tractor's rear axle. The tractor may also have multiple rear axles but it is safe to assume one using the same reasoning as with the differential drive system. The trailer usually has an axle that is fixed. There are some cases though, where the rear axle is steerable as with an excessively long fire ladder truck. This is required for the vehicle to successfully navigate city streets. A tractor with farming implements attached can have this requirement as well. Given that non-steerable wheels on a trailer are equivalent to steerable wheels that remain aligned to the heading of the trailer, a generalized model should include a steerable trailer system. The reference point will remain as the center of the rear axle like in the car model. The relevant coordinates are:

$$q = \begin{bmatrix} x \\ y \\ \phi_1 \\ \theta_1 \\ \phi_2 \\ \theta_2 \end{bmatrix} \tag{A.20}$$

With these coordinates, ϕ_1 is the steering angle of the tractor's front wheels with respect to its heading, θ_1 is the heading of the tractor with respect to the x axis, ϕ_2 is the steering angle of the trailer's wheels with respect to the trailer's heading and θ_2 is the heading of the trailer with respect to the x axis. Now the location of the midpoint of each axle (tractor front (f), tractor rear (m) and trailer (b)) can be determined:

$$\begin{aligned} x_f &= x + l_f \cdot \cos(\theta_1) \\ y_f &= y + l_f \cdot \sin(\theta_1) \\ x_m &= x \\ y_m &= y \\ x_b &= x - l_b \cdot \cos(\theta_2) \\ y_b &= y - l_b \cdot \sin(\theta_2) \end{aligned} \tag{A.21}$$

In these equations, l_f is the distance between the midpoints of the trailer axles and l_b is the distance between midpoints of the rear tractor and the trailer axles. Continuing the theme of no side slippage for any of the wheels gives the constraints:

$$\begin{aligned} \dot{x}_f \cdot \sin(\theta_1 + \phi_1) - \dot{y}_f \cdot \cos(\theta_1 + \phi_1) &= 0 \\ \dot{x}_m \cdot \sin(\theta_1) - \dot{y}_m \cdot \cos(\theta_1) &= 0 \\ \dot{x}_b \cdot \sin(\theta_2 + \phi_2) - \dot{y}_b \cdot \cos(\theta_2 + \phi_2) &= 0 \end{aligned} \tag{A.22}$$

These combine to produce the matrix equation:

$$0 = \begin{bmatrix} \sin(\theta_1+\phi_1) & -\cos(\theta_1+\phi_1) & 0 & -l_f \cdot \cos(\phi_1) & 0 & 0 \\ \sin(\theta_1) & -\cos(\theta_1) & 0 & 0 & 0 & 0 \\ \sin(\theta_2+\phi_2) & -\cos(\theta_2+\phi_2) & 0 & 0 & 0 & l_b \cdot \cos(\phi_2) \end{bmatrix} \cdot \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi}_1 \\ \dot{\theta}_1 \\ \dot{\phi}_2 \\ \dot{\theta}_2 \end{bmatrix} \stackrel{\Delta}{=} A(q)\dot{q} \tag{A.23}$$

Using this equation, the generalized kinematic model can be derived:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi}_1 \\ \dot{\theta}_1 \\ \dot{\phi}_2 \\ \dot{\theta}_2 \end{bmatrix} = \begin{bmatrix} \cos(\theta_1) & 0 & 0 \\ \sin(\theta_1) & 0 & 0 \\ 0 & 1 & 0 \\ \frac{1}{l_f} \cdot \tan(\phi_1) & 0 & 0 \\ 0 & 0 & 1 \\ -\frac{1}{l_b} \cdot \sec(\phi_2) \cdot \sin(\phi_2 - \theta_1 + \theta_2) & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \tag{A.24}$$

For the model, u_1 is the linear velocity of the tractor, u_2 is the steering rate of the tractor's front wheels and u_3 is the steering rate of the trailer's wheels.

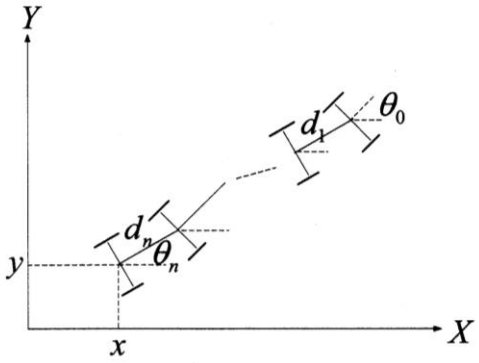


Figure A.4 N-trailer Vehicle.

A slight modification to the tractor-trailer model gives the n-trailer system that looks and behaves similarly to a snake with wheels. To get this system, the car-like tractor is replaced with trailer zero, a differential drive vehicle. The reference point is the midpoint of the last trailer's axle unlike the above examples. Going through the same processes as before yields the kinematic model equations:

$$\dot{x} = v_n \cdot \cos(\theta_n)$$

$$\dot{y} = v_n \cdot \sin(\theta_n)$$

$$\dot{\theta}_n = \frac{1}{d_n} \cdot v_{n-1} \cdot \sin(\theta_{n-1} - \theta_n)$$

$$\bullet$$

$$\bullet$$

$$\dot{\theta}_i = \frac{1}{d_i} \cdot v_{i-1} \cdot \sin(\theta_{i-1} - \theta_i), \qquad i = n-1, \cdots, 2$$

$$\bullet$$

$$\bullet$$

$$\dot{\theta}_1 = \frac{1}{d_1} \cdot u_1 \cdot \sin(\theta_0 - \theta_1)$$

$$\dot{\theta}_0 = u_2 \qquad \text{(A.25)}$$

In these equations, u_1 is the tangential velocity of the tractor and u_2 is its steering rate. θ_j is the heading of the j^{th} trailer with respect to the x axis and d_j is the distance between the j^{th} and $(j-1)^{th}$ trailer. Finally, v_i is the tangential velocity of the i^{th} trailer given by:

$$v_i = \prod_{k=1}^{i} \cos(\theta_{k-1} - \theta_k) \cdot u_1, \qquad i = 1, \cdots, n \tag{A.26}$$

The dynamic models for these two systems can be derived in the same manner as the previous systems. They also result in a very similar reduced form so the process will not be duplicated here.

REFERENCES

[1] Y. Koren and J. Borenstein, "Potential Field Methods and Their Inherent Limitations Mobile Robot Navigation," in *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1398-1404, April 1991.

[2] M.D. Adaim and P. J. Probert, "Towards a Real-Time Navigation Strategy for a Mobile Robot," in *IEEE International Workshop on Intelligent Robots and Systems*, vol. 2, pp. 743-748, August 2002.

[3] M. Park, J. Jeon and M. Lee, "Obstacle Avoidance for Mobile Robots Using Artificial Potential Field Approach with Simulated Annealing," in *IEEE International Symposium on Industrial Electronics*, vol. 3, pp. 1530-1535, August 2002.

[4] M. Park, and M. Lee, "Artificial Potential Field Based Path Planning for Mobile Robots Using a Virtual Obstacle Concept," in *Proceedings of the 2003 IEEWASME International Conference on Advanced Intelligent Mechatronics (AIM 2003)*, vol. 2, pp. 735-740, September 2003.

[5] E. Shi, T. Cai, C. He and J. Guo, "Study of the New Method for Improving Artifical Potential Field in Mobile Robot Obstacle Avoidance," in *Proceedings of the IEEE International Conference on Automation and Logistics*, pp. 282-286, October 2007.

[6] R. Irajit, and M. Manzuri-Shalmanit, "A New Fuzzy-Based Spatial Model for Robot Navigation among Dynamic Obstacles," in *2007 IEEE International Conference on Control and Automation*, pp. 1323-1328, May-June 2007.

[7] Q. Jia and X. Wang, "Path Planning for Mobile Robots Based on a Modified Potential Model," in *Proceedings ofthe 2009 IEEE International Conference on Mechatronics and Automation*, pp. 4946-4951, August 2009.

[8] P. Shi and Y. Zhao, "Global Path Planning for Mobile Robot Based on Improved Artificial Potential Function," in *Proceedings of the IEEE International Conference on Automation and Logistics*, vol. 91, pp. 1900-1904, August 2009.

[9] Q. Jia and X. Wang, "An Improved Potential Field Method for Path Planning," in *Control and Decision Conference (CCDC)*, pp. 2265-2270, May 2010.

[10] A. Synodinos, and N.A. Aspragathos, "Path Planning of a Mobile Robot using Solid Modeling Techniques on Potential Fields," in *2010 IEEE/ASME International Conference on Mechatronics and Embedded Systems and Applications (MESA)*, pp. 549-553, July 2010.

[11] J. Lee, Y. Nam, and S. Hong, "Random Force based Algorithm for Local Minima Escape of Potential Field Method," in *2010 11th Int. Conf. Control, Automation, Robotics and Vision*, pp. 827-832, December 2010.

[12] O. Khatib, "Real-time Obstacle Avoidance for Manipulators and Mobile Robots," in *Proceedings 1985 IEEE International Conference on Robotics and Automation*, vol. 2, pp. 500-505, March 1985.

[13] P. Stingu and F. Lewis, "Motion Path Planning for Mobile Robots," University of Texas at Arlington, Automation and Robotics Research Institute.

[14] A. Zhdanov and Alexei N. Vinokurov, "Emotions Simulation in Methodology of Autonomous Adaptive Control," in *Proceedings of the 1999 IEEE International Symposium on Intelligent Control/Intelligent Systems and Semiotics*, pp. 73-78, September 1999.

[15] Z. Qu, *Cooperative Control of Dynamical Systems*, pp. 11-25, Springer-Verlag London Limited, 2009.

[16] "PIC24FJ64GA004 Family Data Sheet," Microchip Technology Incorporated, pp. 215-219, 2008.

[17] "TMS320C3x User's Guide," Texas Instruments Incorporated, rev. J, pp. 11-34 – 11-37, October 1994.

[18] M. Pachter and P. R. Chandler, "Challenges of Autonomous Control," in *IEEE Control Systems*, vol. 2, issue 4, pp. 92-97, August 1998.

[19] H. Chen, X. Wang and Y. Li, "A Survey of Autonomous Control for UAV," in *International Conference on Artificial Intelligence and Computational Intelligence*, vol. 2, pp. 267-271, November 2009.

[20] W. Wang, G. Song, K. Nonami, M. Hirata and O. Miyazawa, "Autonomous Control for Micro-Flying Robot and Small Wireless Helicopter X.R.B," in *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2906-2911, October 2006.

[21] R. Eaton, J. Katupitiya, K. Siew and B. Howarth, "Autonomous Farming: Modeling and Control of Agricultural Machinery in a Unified Framework," in *15th International conference on Mechatronics and Machine Vision in Practice (M2VIP08),* pp. 499-504, December 2008.

[22] T. Matsumaru, K. Hagiwara, and T. Ito, "Incorporation of Autonomous Control Elements in Combination Control of Remote Operation and Autonomous Control," in *IEEE 2002 28th Annual Conference of the Industrial Electronics Society IECON 02,* vol. 3, pp. 2311-2316, November 2002.

[23] Sandor M. Veres, "Principles, Architectures and Trends in Autonomous Control," in *The IEE Seminar on Autonomous Agents in Control,* pp. 1-9, May 2005.

[24] T. Johnson, H. Sutherland, S. Bush, W. Yan and C. Eaker, "The TRAC Mission Manager Autonomous Control Executive," in *IEEE Proceedings Aerospace Conference, 2001,* vol. 2, pp. 2/639-2/646, 2001.

[25] Y. Hao, W. Zu, S and Y. Zhao, "Real-Time Obstacle Avoidance Method based on Polar Coordination Particle Swarm Optimization in Dynamic Environment," in *2nd IEEE Conference on Industrial Electronics and Applications, 2007. ICIEA 2007,* pp. 1612-1617, May 2007.

[26] M. Gao, J. Xu, and J. Tian, "Mobile Robot Global Path Planning Based on Improved Augment Ant Colony Algorithm," in *Second International Conference on Genetic and Evolutionary Computing,* pp. 273-276, September 2008.

[27] D. Bodhale, N. Afzulpurkar, and N. Thanh, "Path Planning for a Mobile Robot in a Dynamic Environment," in *IEEE International Conference on Robotics and Biomimetics, 2008. ROBIO 2008,* pp. 2115-2120, February 2009.

[28] Y. Gao, S. Sun, and D. He, "Global Path Planning of Mobile Robot base on Particle Filter," in *2009 World Congress on Computer Science and Information Engineering,* vol. 3, pp. 205-209, March-April 2009.

[29] J. Zhao, L. Zhu, G. Liu, G. Liu and Z. Han, "A Modified Genetic Algorithm for Global Path Planning of Searching Robot in Mine Disasters," in *Proceedings of the 2009 IEEE International Conference on Mechatronics and Automation,* pp. 4936-4940, August 2009.

[30] W. Haiying, X. Rui, B. Mingyou and J. Jinlin, "Optimization Design of Mobile Robot Based on Genetic Algorithm," in *Proceedings of the 2009 IEEE International Conference on Mechatronics and Automation,* pp. 767-771, August 2009.

[31] J. Guo, L. Liu, Q. Liu and Y. Qu, "An Improvement of D* algorithm for Mobile Robot Path Planning in Partial Unknown Environment," in *2009 Second International Conference on Intelligent Computation Technology and Automation,* vol. 3, pp. 394-397, October 2009.

[32] H. Huang, C. Tsai and S. Lin, "SoPC-Based Parallel Elite Genetic Algorithm for Global Path Planning of an Autonomous Omnidirectional Mobile Robot," in *Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics,* pp. 1959-1964, October 2009.

[33] Anand TM, "Reactive Deformation of Path for Navigation Among Dynamic Obstacles," in *Proceedings of the 2009 IEEE International Conference on Robotics and Biomimetics,* pp. 544-549, December 2009.

[34] Nohaidda B. Sariff and Norlida Buniyamin, "Comparative Study of Genetic Algorithm and Ant Colony Optimization Algorithm Performances for Robot Path Planning in Global Static Environments of Different Complexities," in *2009 IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA),* pp. 132-137, December 2009.

[35] Q. Zhang, M. Li and X. Wang, "Global Path Planning Method of Mobile Robot in Uncertain Environment," in *2010 Chinese Control and Decision Conference (CCDC),* pp. 4320-4324, May 2010.

[36] Q. Li, Y. Tang, L. Wang, C. Zhang and Y. Yin, "A Specialized Particle Swarm Optimization for Global Path Planning of Mobile Robots," in *2010 Third International Workshop on Advanced Computational Intelligence,* pp. 271-276, August 2010.

[37] Sylvia Horn and Klaus Janschek, "A Set-Based Global Dynamic Window Algorithm for Robust and Safe Mobile Robot Path Planning," in *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK),* pp. 1-7, June 2010.

[38] J. Lee, B. Choi, K. Park and J. Lee, "Comparison between Heterogeneous Ant Colony Optimization Algorithm and Genetic Algorithm for Global Path Planning of Mobile Robot," in

2011 IEEE International Symposium on Industrial Electronics (ISIE), pp. 881-886, June 2011.

[39] W. Dong-Shu and Y. Hua-Fang, "Path Planning of Mobile Robot in Dynamic Environments," in *2011 2nd International Conference on Intelligent Control and Information Processing (ICICIP),* vol. 2, pp. 691-696, July 2011.

[40] S. Asadi, V. Azimirad, A. Eslami and A. Ghanbari, "A Novel Global Optimal Path Planning and Trajectory Method Based on Adaptive Dijkstra-Immune Approach for Mobile Robot," in *2011 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM2011),* pp. 1093-1098, July 2011.

[41] Hsu-Chih Huang and Ching-Chih Tsai, "Global Path Planning for Autonomous Robot Navigation Using Hybrid Metaheuristic GA-PSO Algorithm," in *2011 Proceedings of SICE Annual Conference (SICE),* pp. 1338-1343, September 2011.

[42] J. Seok, C. Oh, J. Lee and H. Lee, "Integrated Path Planning for a Partially Unknown Outdoor Environment," in *2011 IEEE/SICE International Symposium on System Integration (SII),* pp. 643-648, December 2011.

[43] Kurt D. Rueb and Andrew K. C. Wong, "Structuring Free Space as a Hypergraph for Roving Robot Path Planning and Navigation," in *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. PAMI-9, issue 2, pp. 263-273, March 1987.

[44] T. Guan-Zheng, H. Huan and S. Aaron, "Ant Colony System Algorithm for Real-Time Globally Optimal Path Planning of Mobile Robots," in *Acta Automatica Sinica,* vol. 33, issue 3, pp. 279-285, March 2007.

[45] Y. Hao, Z. Shen and Y. Zhao, "Path Planning for Aircraft Based on MAKLINK Graph Theory and Multi Colony Ant Algorithm," in *2009 International Joint Conference on Computational Sciences and Optimization,* vol. 2, pp. 232-235, April 2009.

[46] Maki K. Habib and Hajime Asama, "Efficient Method to Generate Collision Free Paths for an Autonomous Mobile Robot Based on New Free Space Structuring Approach," in *IEEE/RSJ International Workshop on Intelligent Robots and Systems '91. 'Intelligence for Mechanical Systems, Proceedings IROS '91,* vol. 2, pp. 563-567, November 1991.

[47] Bernard Chazelle and David P. Dobkin, "Optimal Convex Decompositions," in *Toussaint, G.T., Computational Geometry,* pp. 63-133, Elsevier, 1985.

[48] C. Beer, T. Hendtlass and J. Montgomery, "Improving Exploration in Ant Colony Optimisation with Antennation," in *2012 IEEE Congress Evolutionary Computation (CEC),* pp. 1-8, June 2012.

[49] Wang Yu-qin and Yu Xiao-peng, "Research for the Robot Path Planning Control Strategy Based on the Immune Particle Swarm Optimization Algorithm," in *2012 International Conference on Intelligent System Design and Engineering Application,* pp. 724-727, January 2012.

[50] I. Châari, A. Koubâa and H. Bennaceur, "smartPATH: A hybrid ACO-GA Algorithm for Robot Path Planning," in *WCCI 2012 IEEE World Congress on Computational Intelligence,* pp. 1-8, June 2012.

BIOGRAPHICAL INFORMATION

John R. Konderla Jr. received his Bachelor of Science in Electrical Engineering degree from the University of Texas at Arlington in 2007. He started work on his Masters of Science in Electrical Engineering one year later. He has been working as an Electrical Engineer in R&D and Testing while pursuing his Masters degree.

Prior to obtaining his undergraduate degree, Mr. Konderla had amassed 21 years of experience in high tech fields as a technician. His experience includes 6.5 years in the U.S. Navy as an avionics technician where he received extensive avionics training covering radar, communication and navigation systems. He worked as a bench-level radar technician for the E2C Hawkeye. After the Navy, he continued in the technology sector. His maintenance and manufacturing experience includes industrial robotics, cryogenic systems, laser marking systems, automated test equipment, HVAC, office equipment, televisions, stereos, and city-wide automated water plant systems for one of the 50 largest cities in the U.S. He also holds an FCC General Radiotelephone (repair) license with radar endorsement.