Coverage Path Planning And Control For Autonomous Mobile Robots

Mohanakrishnan Balakrishnan
University of Central Florida

COVERAGE PATH PLANNING AND CONTROL FOR AUTONOMOUS MOBILE ROBOTS

By

MOHANAKRISHNAN BALAKRISHNAN
B.S. University of Madras, 2001

A thesis submitted in partial fulfillment of the requirements
For the degree of Master of Science
in the Department of Electrical Engineering
in the College of Engineering & Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2005

ABSTRACT

Coverage control has many applications such as security patrolling, land mine detectors, and automatic vacuum cleaners. This Thesis presents an analytical approach for generation of control inputs for a non-holonomic mobile robot in coverage control. Neural Network approach is used for complete coverage of a given area in the presence of stationary and dynamic obstacles. A complete coverage algorithm is used to determine the sequence of points. Once the sequences of points are determined a smooth trajectory characterized by fifth order polynomial having second order continuity is generated. And the slope of the curve at each point is calculated from which the control inputs are generated analytically. Optimal trajectory is generated using a method given in research literature and a qualitative analysis of the smooth trajectory is done. Cooperative sweeping of multirobots is achieved by dividing the area to be covered into smaller areas depending on the number of robots. Once the area is divided into sub areas, each robot is assigned a sub area for cooperative sweeping.

To my advisor Dr. Yi Guo, parents and brother.

ACKNOWLEDGMENTS

I would like to thank my mentor/graduate advisor Dr.Yi Guo for guiding me with patience through out my research. Dr. Yi Guo always gave me valuable research inputs which were extremely helpful. She also recommended me for the Graduate Teaching Assistantship, which was of great financial help to me. In addition I would like to thank Dr. Zhihua Qu and Dr.Michael Haralambous for agreeing to be my Committee member.   I would also like to thank Jocelyn A. Beyer for helping in formatting this document.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER ONE: INTRODUCTION

Coverage control has many applications and the applications are ever increasing. Some of them are security patrolling, land mine detectors and automatic vacuum cleaning. A great number of different techniques has been and are still developed in order to carry out efficient robot path planning. One of the most popular planning methods is based on the potential function utilization where robot is modeled as a moving particle, inside an artificial potential field that reflects free collision space structure into the robot workspace. There are also many cellular decomposition based approaches to complete coverage. The fundamental concept is to decompose the workspace into a collection of nonoverlapping cells, and then, the robot searches this connectivity graph that represents the adjacency relation among cells. A novel neural network approach was also proposed for complete coverage path planning of a mobile non-holonomic robot. The state space of the topologically organized neural network is the two dimensional Cartesian workspace of a cleaning robot. The proposed neural network model is capable of planning real time complete coverage paths with obstacle avoidance in an unstructured indoor environment. In [2], Yi Guo and Zhihua Qu proposed a method for complete coverage in which minimum number of circles is used to cover a bounded region. The complete coverage takes place by selecting the circle, which has the smallest distance from its center to the boundary. Then feasible trajectories and steering control are then generated so that the robot moves collision free and covers all circles.

In [5], Paolo Fiorini and Zvi Shiller proposed a method computing the time optimal trajectory for a robot among stationary obstacles, subject to robots dynamic constraints. In [6], J-P. Laumond, S. Sekhavat and M. Vaisset proposed optimization using cost function. In [7],H.Delingette proposed optimization based on energy minimization. Aurelio Piazzi “optimal trajectory planning with quintic splines” deals with the generation of optimal paths for the automated steering of autonomous vehicles. The path is parameterized by quintic η- splines, devised to guarantee the overall second order geometric continuity of a composite path interpolating an arbitrary sequence of points. Starting from the closed loop form η- parameterization of the spline an optimization criterion is proposed to design smooth curves. The aim is to plan curves where the curvature variability is kept as small as possible.

1.1 Potential Field Method For Complete Coverage

The artificial potential method is a useful tool in path planning. The main idea is to construct an attractive potential field at the goal, and repulsive potentials on the obstacles. The path is then followed by a weighted sum of potentials. Numerous artificial potential functions have been proposed in the past decade but they all suffer from the problem of local minima. This limits the applicability of artificial potential methods.

Distance Transform Approach Distance transform for planning paths for mobile robot applications was first reported by A.Zelinsky and R.A. Jarvis [8]. This approach consider the task of path planning to find paths from the goal location back to the start location. The path planner propagates a distance wave front through all free space grid cells in the environment from the goal cell. The distance wave front flows around obstacles and eventually through all free space in the environment. For any starting point within the environment representing the initial position of the mobile robot, the shortest path to the goal is traced by walking down hill via the steepest descent path. If there is no downhill, and the start cell is on a plateau then it can be concluded that no path exists from the start cell to the goal cell. To achieve the complete coverage path planning behavior, instead of descending along the path of steepest descent to the goal, the robot follows the path of steepest ascent. In other words the robot moves away from the goal keeping track of the cells it has visited. An advantage of this complete strategy is that the start and goal can be specified. While this strategy does not the guarantee complete coverage path will be an optimum, the complete coverage produces a reasonable path with minimal secondary visits to grid cells.

Path Transform Approach In the path transform approach, instead of propagating a distance from the goal wave front through free space, a new wave front is propagated which was a weighted sum of the distance from the goal together with a measure of the discomfort of moving too close to obstacles. This had the effect of producing a transform which has the desirable properties of potential fields without suffering from the local minima problem. The path transform can be regarded as a numeric potential field. The distance transform is extended to include safety from obstacles information in the following way. Firstly, the distance transform is inverted into an obstacle transform where the obstacle cells become the goals. The resulting transformation yields for each free cell in the data structure is the minimal distance from the center of the free space cell to the boundary of an obstacle cell. Finally a second distance transform is generated through free space from the goal location using a new cost function. This cost function is referred to as the path transform. The path transform for each cell is the minimum propagated path cost to the goal. The path transform forms a better contour path for a robot to implement a path of complete coverage than the contour path generated by the original distance transform. A similar result to path transforms called numeric potential fields was reported by Barraquand and Latombe [9]. The numeric potential is computed in three steps.

Firstly, an obstacle transform is computed of the free space, from which a distance skeleton is extracted. Joining the highest values in the obstacle transform yields a distance skeleton and a distance transform is computed from the goal cell to all members cell of the distance skeleton. Thirdly another distance transform is computed from the distance skeletons to all the remaining free space cells in the environment. This method can guide the robot through narrow free space channels that are close to the goal thus endangering the robot. This problem is countered by removing channels that are narrow but the completeness of solution is lost. The path transform does not suffer this drawback.

1.2 Neural Network Approach For Complete Coverage

Simon X. Yang and Chaomin Luo [10] proposed a novel neural network approach for complete coverage path planning of a single robot and multiple cleaning robots, which is based on a neural network model for conventional real time path planning for a mobile robot. The state space of the topologically organized neural network is the two dimensional Cartesian work space of a cleaning robot. The dynamics of each neuron is characterized by a shunting equation derived from the Hodgkin and Huxley's membrane model for a biological neural system. There are only local lateral connections among neurons. The varying environment is represented by the dynamic activity landscape of the neural network. The proposed model is computationally simple and is capable of planning real time complete coverage paths with obstacle avoidance. By properly defining the external inputs from the varying environment and the internal neural connections, the neural activities of the unclean areas and obstacles are guaranteed to stay at the peak and the valley of the activity landscape of the neural network, respectively. The unclean areas globally attract the robot in the entire state space through neural activity propagation. The two dimensional work space in the proposed approach is discretized into squares as in most other complete coverage models. The diagonal length of each discrete area is equal to the robot sweeping radius that is the size of the effector or foot print. Thus sweeping an area can be achieved by traversing the center of that area represented by a discrete point. Once the robot visits a discreet point, it is assumed that the robot has covered the discrete area of that point. If a cleaning robot covers every discrete point in a workspace, the robot path is then considered as a complete coverage path in the workspace. The proposed neural network model shares some common ideas with the standard artificial potential path planning techniques.

1.3 Generation of Control Inputs of a Nonholonomic mobile Robot in Complete Coverage

It is well known in robotics that, if the number of generalized coordinates equals the number of input commands, one can use a nonlinear static state feedback law in order to transform exactly the nonlinear kinematics and/or dynamics into a linear system. In general, the linearity of the system equations is displayed only after a coordinate transformation in the state space. On the linear side of the problem, it is rather straightforward to complete the synthesis of a stabilizing controller.

Actually two types of exact linearization problems can be considered for a nonlinear system with outputs. Beside the possibility of transforming via feedback the whole set of differential equations into a linear system, one may seek a weaker result in which only the input-output differential map is made linear. Necessary and sufficient conditions exist for the solvability of both problems via static feedback, while only sufficient conditions can be given for the dynamic feedback case.

Consider a generic non-linear system

$$\dot{x} = f(x) + g(x)u \tag{1.1}$$

$$z = h(x) \tag{1.2}$$

Where x is the system state, u is the input, and z is the output to which we wish to assign a desired behavior. Assume the system is square, i.e., the number of inputs equals the number of outputs.

The input-output linearization problem via static feedback is finding a control law of the form

$$u = a(x) + B(x)r, \tag{1.3}$$

With $B(x)$ nonsingular and r an external auxiliary input of the same dimension as u, in such a way that the input-output response of the closed-loop system is linear. In the multi-input case, the solution to this problem automatically yields input-output decoupling, namely, each component of the output z will depend only on a single component of the input r.

In general, a nonlinear internal dynamics, which does not affect the input-output behavior, may be left in the closed-loop system. This internal dynamics reduces to the so-called clamped dynamics when the output z is constrained to follow a desired trajectory. In the absence of internal dynamics, full state linearization is achieved. Conversely, when only input-output linearization is obtained, the boundedness/stability of the internal dynamics should be analyzed in order to guarantee a feasible output tracking.

If static feedback does not allow solving the problem, one can try to obtain the same results by means of a dynamic feedback compensator of the form.

$$u = a(x) + B(x)r, \tag{1.4}$$

$$\dot{\xi} = c(x,\xi) + D(x,\xi)r, \tag{1.5}$$

Where ξ is the compensator state of appropriate dimensions.

There are also distinguished methods for motion planning such as differential geometric and differential algebra techniques, geometric phase, control input parameterization, optimal control approach, etc. The idea behind the differential geometric and differential algebra techniques is to generate motions in the directions of iterated Lie brackets by employing typical inputs [12]. Monaco and Noramnd-Cyrot first proposed to use piece-wise constant inputs in motion planning in [23]. In [11], sinusoids are used as inputs in the motion planning. In [12], a motion-planning algorithm is proposed for nonholonomic systems based on the concept of differential flatness. For differential flat non-linear systems, the motion-planning problem is equivalent to finding the output functions, which satisfy the boundary conditions posed on initial and final states. For nonholonomic Chaplygin systems, various techniques based on the different geometric phase approach. In this thesis, a smooth trajectory is generated using the method presented in [1] and the slope of the curve at each point is calculated for, which the control inputs are generated.

1.4 Optimal analysis of trajectory

H. Delingette and M. Hebert proposed optimization based on energy minimization in [7]. An algorithm based on the deformation of a curve by energy minimization allows solving general geometric constraints. In [6] J-P. Laumond, S. Sekhavat used a cost fuction to obtain the optimum trajectory. In [3] Aurelio Piazzi proposed a method for generation of optimal paths for automated steering of autonomous vehicles using quintic η- splines. The path is parameterized by η- splines, devised to guarantee the overall second order geometric continuity of a composite path interpolating an arbitrary sequence of points. Starting from the closed-form η-parameterization of the spline, an optimization criterion is proposed to design smooth curves. The aim is to plan curves where the curvature variability is kept as small as possible. With good approximation in a flatness based control scheme, this corresponds to minimize the change-rate of the steering control.

CHAPTER TWO: MOBILE ROBOT MODEL AND PROBLEM STATEMENT

In this chapter a qualitative study of non-holonomic system and a kinematic model for the rear wheel drive, front wheel steered robotic car is derived.

2.1 Nonholonomic Constraints

If a system has restrictions in its velocity, but these restrictions do not cause restrictions in its positioning, the system is said to be non-holonomically constrained. Viewed another way the systems local movement is restricted, not its global movement. Mathematically this means that the velocity constraints cannot be integrated to position constraints.

The most familiar example of a non-holonomic system is demonstrated by parallel parking maneuver. When a driver arrives next to the parking space, he cannot simply slide his car sideways into the parking space. The car is not capable of sliding side ways and this is the velocity restriction. However by moving the car backward and forward and turning the wheels the car can be brought in to the space. Ignoring the restrictions caused by external objects, the car can be located at any orientation, despite lack of side ways movement. The non-holonomic constraints of each wheel of the mobile robot are shown in Figure 2.3. The wheel velocity is in the direction of rolling. There is no velocity in the perpendicular direction. This model assumes that there is no wheel slippage.

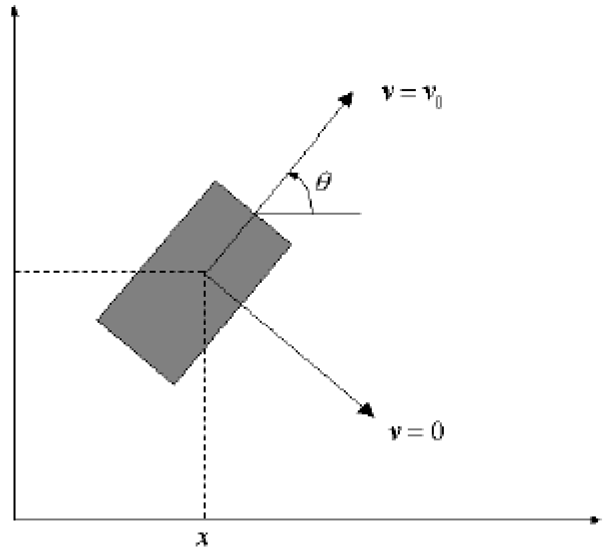

Figure 2.1. The velocity constraint on a rolling wheel with no slippage

With a holonomic system, return to the original internal configuration means return to the original system position. With a non-holonomic system, return to the original internal configuration does not guarantee return to the original system position. More generally the system outcome for a non-holonomic system is path dependent. This has several implications for the implementation of control system.

2.2 Kinematic model of a non-holonomic mobile robot

The exact position and orientation of the car in some global coordinate system can be described by four variables. Fig. 2.4.shows each of the variables. The (x, y) coordinates give the exact position and orientation of the car in some global coordinate system can be described by four variables. Fig. 2.4.shows each of the variables. The (x, y) coordinates give the location of the center of the rear axel. The cars angle with respect to the x-axis is given by θ. The steering wheels angle with respect to the cars longitudinal axis is given by ϕ. Due to the constraints the velocity of the car in the (x, y) directions is given as

$$\dot{x} = u_1 \cos\theta \tag{2.1}$$

$$\dot{y} = u_1 \sin\theta \tag{2.2}$$

where u_1 is the linear velocity of the rear wheels.

The location of the center of the front axle (x_1, y_1) is given by

$$x_1 = x + l\cos\theta \tag{2.3}$$

$$y_1 = y + l\sin\theta \tag{2.4}$$

And the velocity is given by

$$\dot{x}_1 = \dot{x} - l\dot{\theta}\sin\theta \tag{2.5}$$

$$\dot{y}_1 = \dot{y} - l\dot{\theta}\cos\theta \tag{2.6}$$

Applying the no slippage constraint to the front wheels gives

$$\dot{x}_1 \sin(\theta + \phi) = \dot{y}_1 \cos(\theta + \phi) \tag{2.7}$$

Inserting equation 2.5 and 2.6 into equation 2.7 and solving for $\dot{\theta}$ gives

$$\dot{\theta} = \frac{\tan\phi}{l} u_1 \tag{2.8}$$

The complete kinematic model is then given by equation (2.9)

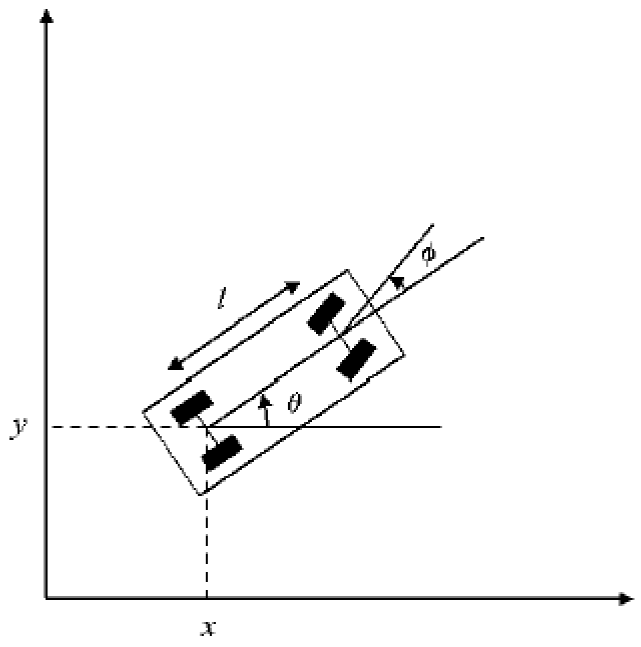

Figure 2.2. A car like robot

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ \tan\phi / l \\ 0 \end{bmatrix} u_1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u_2 \qquad (2.9)$$

Where u_1 is the linear velocity of the rear wheels and u_2 is the angular velocity of the steering wheels.

2.3 Problem Statement

Given an arbitrary region Ω with stationary or moving obstacles, the coverage control is to find a control algorithm such that the mobile robot covers the region Ω considering the presence of obstacles over a finite time. Thus given a sequence of points (x_i, y_i) $(0 \leq i \leq m)$, which the robot has to follow so that it completely covers the given convex region, our first objective is to generate control inputs u_1 and u_2, where u_1 is the linear velocity of the driving wheel and u_2 is the steering velocity of the front wheels. Our next objective is to qualitatively analyze the smooth trajectory so obtained and finally develop an algorithm for cooperative sweeping of multirobots.

CHAPTER THREE: COMPLETE COVERAGE DESIGN METHODS

3.1 Complete Coverage Design

Given an arbitrary region Ω with stationary or moving obstacles, the coverage control is to find a control algorithm such that the mobile robot covers the region Ω considering the presence of obstacles over a finite time. In [2], the authors propose an algorithm to completely cover a region Ω with circles in two-dimensional plane. It is shown that the disk placement described in [2] has a minimum number of disks to cover a rectangle. An algorithm is also proposed to find a complete coverage path to any given convex connected region without obstacles, i.e., a sequence of points (x_i, y_i) $(0 \le i \le m)$.

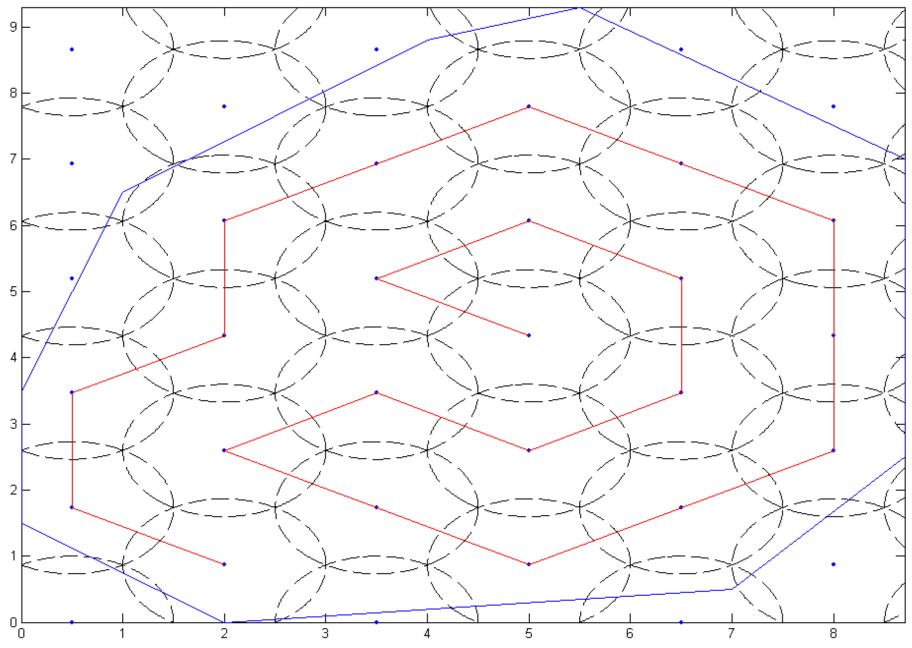

Figure 3.1 Complete coverage design

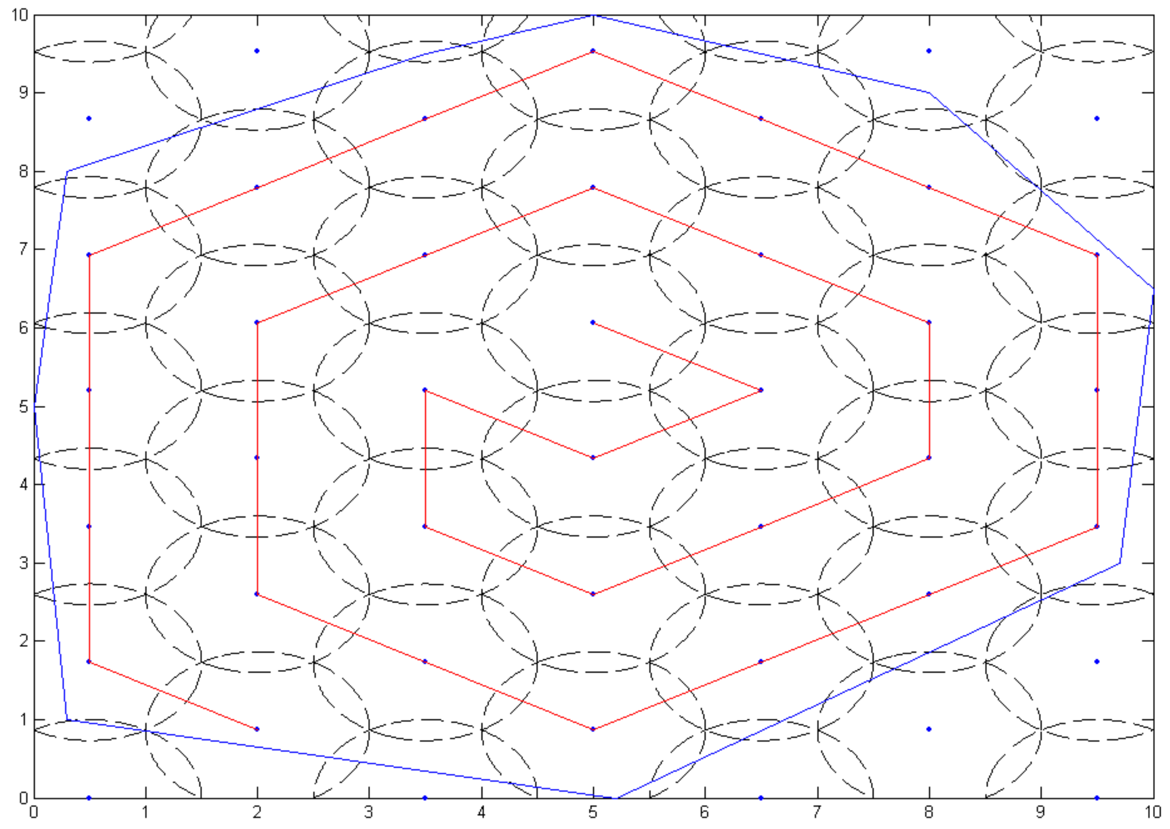

Figure 3.2 Complete coverage design B

3.2 Complete Coverage Using Neural Network Approach

In [10], a novel neural network approach is proposed for complete coverage path planning with obstacle avoidance of cleaning robots in non stationary environments. The dynamics of each neuron in the topologically organized neural network is characterized by a shunting equation derived from Hodgkin and Huxley's membrane equation. The robot path is autonomously generated from the activity landscape of the neural network and the previous location. The model algorithm is computationally simple.

The proposed neural network model is expressed topologically in a discretized workspace Ω of a cleaning robot. The location of the i^{th} neuron in the state space of the neural network, which is denoted by a vector $q_i \in R^2$, uniquely represents an area in Ω. In the proposed model, the excitatory input results from the unclean areas and the lateral neural connections, whereas the inhibitory input results from the obstacles only. The dynamics of the i^{th} neuron in the neural network can be characterized by a shunting equation as

$$\frac{dx_i}{dt} = -Ax_i + (B - x_i)([I_i]^+ + \sum_{j=1}^{k} w_{ij}[x_j]^+) - (D + x_i)[I_i]^- \qquad (3.1)$$

where k is the number of neural connections of the i^{th} neuron to its neighboring neurons within the receptive field. The external input I_i to the i^{th} neuron is defined as

$$\begin{aligned} &I_i = E \quad it \quad is \quad an \quad unclean \quad area \\ &I_i = -E \quad if\, it \quad is \quad an \quad obstacle \quad area \\ &I_i = 0, \quad otherwise \end{aligned} \qquad (3.2)$$

where E >> B is a very large positive constant. The terms $[I_i]^+ + \sum_{j=1}^{n} w_{ij}[x_j]^+$ and $[I_i]^-$ are the excitatory and inhibitory inputs, respectively. Function $[a]^+$ is a linear threshold function defined as $[a]^+$ = max $\{a,0\}$, and the nonlinear function $[a]^-$ is defined as $[a]^-$ = max $\{-a,0\}$. The connection weight w_{ij} between the i^{th} and the j^{th} neurons can be defined as

$$w_{ij} = f\left|q_i - q_j\right| \qquad (3.3)$$

where $\left|q_i - q_j\right|$ represents the Euclidean distance between vectors q_i and q_j in the state space, and $f(a)$ can be any monotonically decreasing function, such as a function defined as

$$f(a) = \frac{\mu}{a}, if \quad 0 < a < r_0 \tag{3.4}$$

$$f(a) = 0, if \quad a \geq r_0 \tag{3.5}$$

where μ and r_0 are positive constants. The neuron has lateral connections only to its 6 neighbors. For a given current location in the state space S denoted by p_c, the next robot location p_n is obtained by

$$x_{pn} = \max\{x_j + cy_j, j = 1,2,...k\} \tag{3.6}$$

where c is a positive constant, and k is the number of neighboring neurons of the p_c neuron. Variable x_j is the neural activity of the j^{th} neuron and y_j is defined as

$$y_j = 1 - \frac{\Delta\theta_j}{\pi} \tag{3.7}$$

where $\Delta\theta_j \in [0, \pi]$ is the absolute angle change between the current and next moving directions.

$$\Delta\theta_j = \left| a\tan(y_{pj} - y_{pc}, x_{pj} - x_{pc}) - a\tan(y_{pc} - y_{pp,} x_{pc} - x_{pp}) \right| \tag{3.8}$$

After the robot reaches its next location, the next location becomes a new current location.

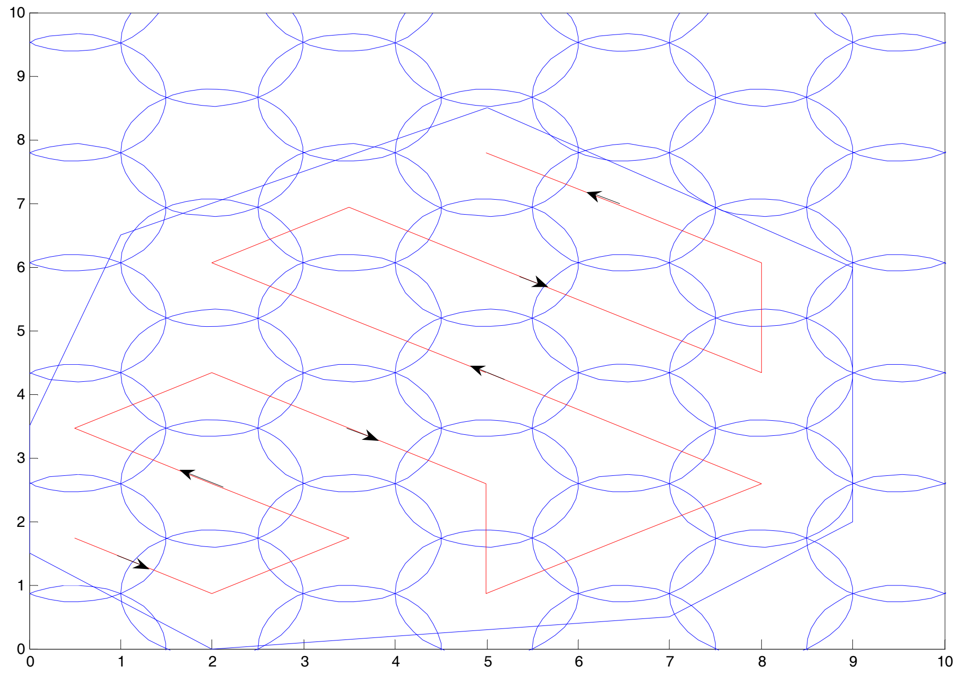

Figure 3.3 Complete coverage design C

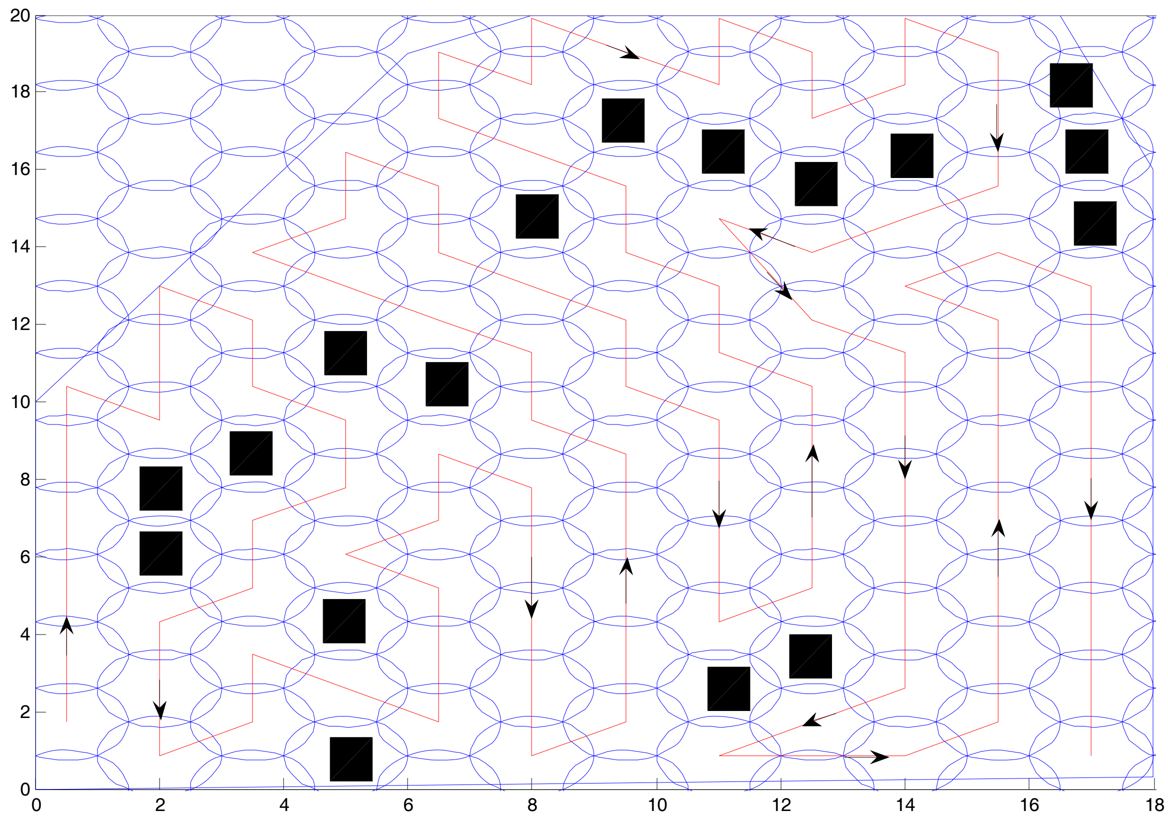

Figure 3.4 Complete coverage with obstacles

3.3 Complete coverage in the presence of dynamic obstacles.

The proposed neural network approach in [24] is capable of generating complete coverage paths in the presence of moving obstacles. In the proposed neural network model, the excitatory input results from the target and its neighboring neurons. The inhibitory input results from the obstacles only. Thus the dynamics of the i^{th} neuron in the neural network is characterized by a shunting equation as

$$\frac{dx_i}{dt} = -Ax_i + (B - x_i)([I_i]^+ + \sum_{j=1}^{k} w_{ij}[x_j]^+) - (D + x_i)[I_i]^- \tag{3.9}$$

Where k is the total number of neurons in the neural network. The terms $[I_i]^+ + \sum_{j=1}^{n} w_{ij}[x_j]^+$ and $[I_i]^-$ are the excitatory and inhibitory inputs, respectively. The external input I_i to the i^{th} neuron is defined as

$$\begin{aligned} I_i &= E \quad it \quad is \quad an \quad unclean \quad area \\ I_i &= -E \quad if\ it \quad is \quad an \quad obstacle \quad area \\ I_i &= 0, \quad otherwise \end{aligned} \tag{3.10}$$

where E >> B is a very large positive constant. The proposed model requires the complete knowledge of the dynamic environment, which can be obtained from the various sensors. The neural connection weight w_{ij} is not only a function of distance but also a function of robot orientation. The activity landscape of the neural network dynamically changes due to the varying external inputs from the target and the obstacles. The robot motion is planned from the dynamic activity landscape by a steepest gradient ascent rule. For a given present robot location in S , denoted by p_c, the next robot location p_n, is obtained by

$$x_{pn} = \max\{x_j + cy_j, j = 1,2,...k\} \tag{3.11}$$

where c is a positive constant, and k is the number of neighboring neurons of the p_c neuron. Variable x_j is the neural activity of the j^{th} neuron and y_j is defined as

$$y_j = 1 - \frac{\Delta\theta_j}{\pi} \tag{3.12}$$

$$\Delta\theta_j = \left| a\tan(y_{pj} - y_{pc}, x_{pj} - x_{pc}) - a\tan(y_{pc} - y_{pp,} x_{pc} - x_{pp}) \right| \tag{3.13}$$

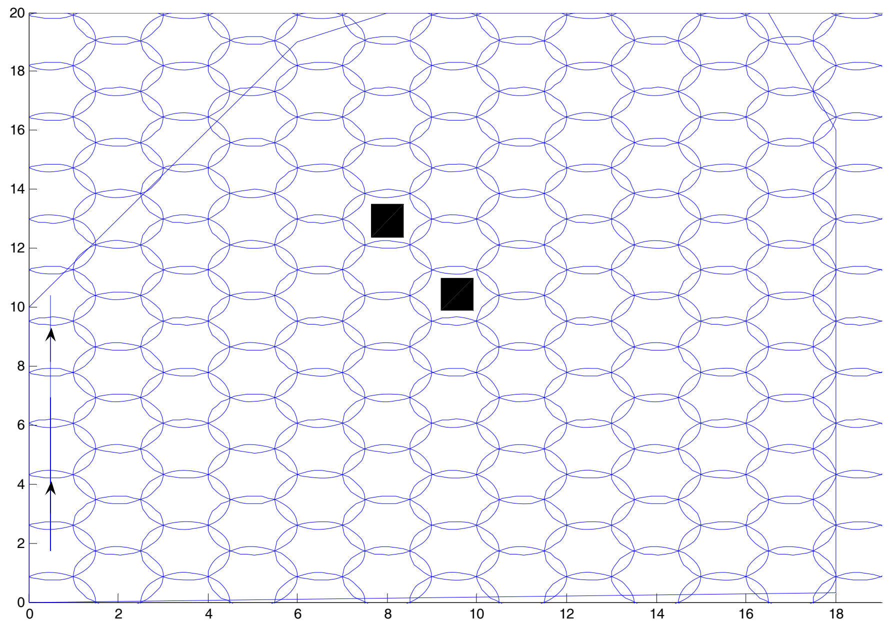

Figure 3.5 Robot path in the presence of dynamic obstacles

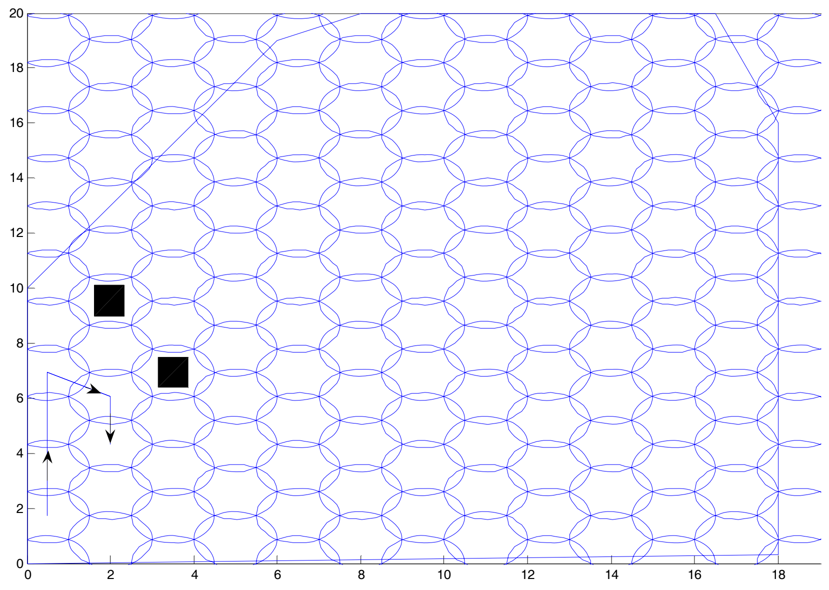

Figure 3.6 Robot path in the presence of dynamic obstacles B

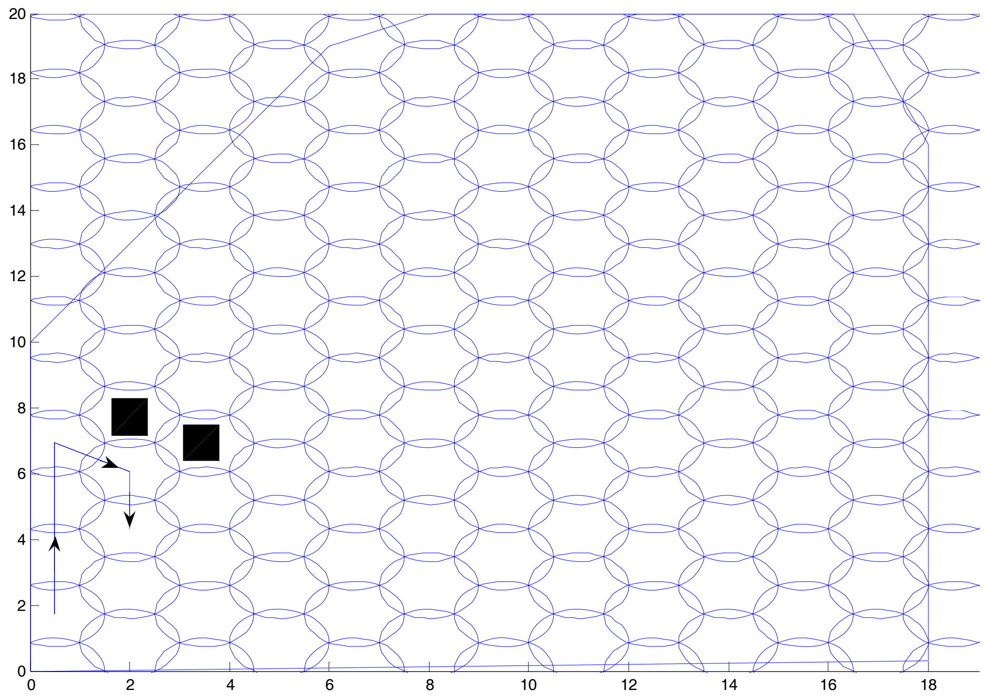

Figure 3.7 Robot path in the presence of dynamic obstacles C

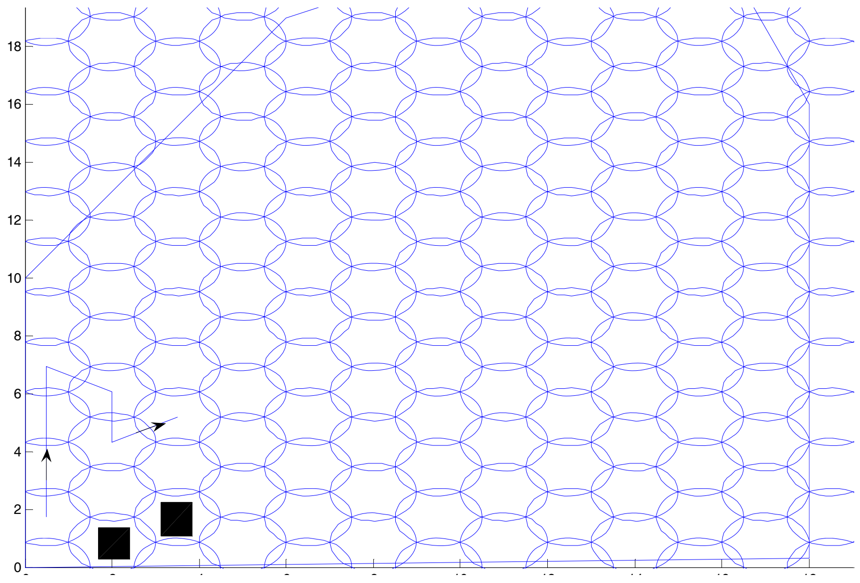

Figure 3.8 Robot path in the presence of dynamic obstacles D

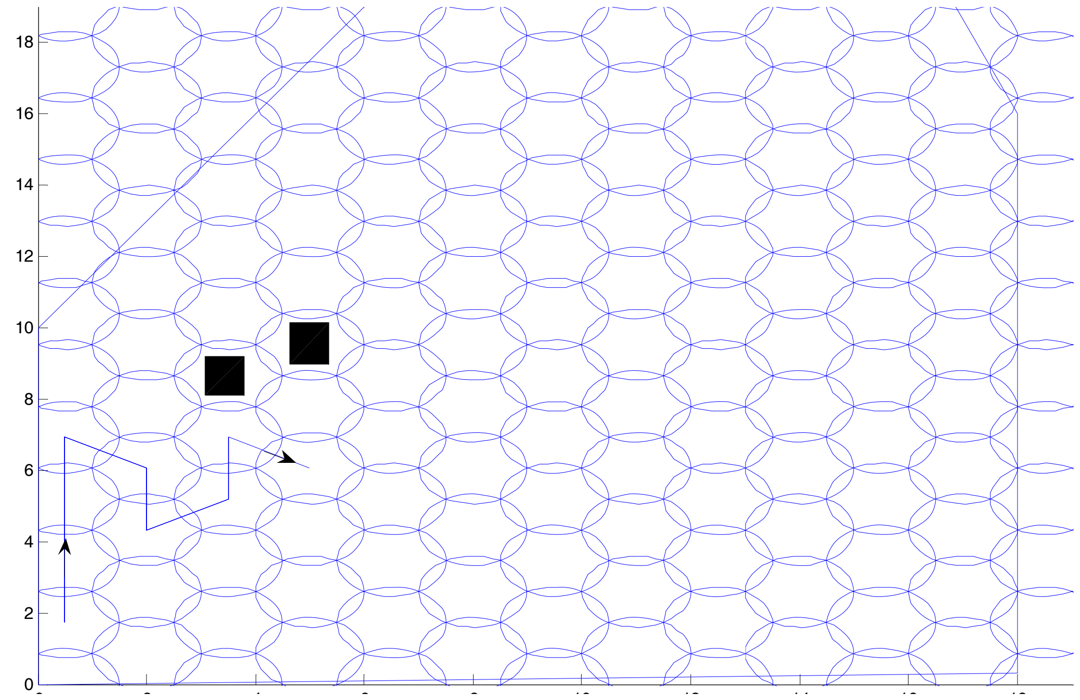

Figure 3.9 Robot path in the presence of dynamic obstacles E

3.4 Cooperative Sweeping of MultiRobots

It has long been known that for any polygon there is a collection of smaller polygonal pieces that can be arranged to form different polygons. This collection of smaller pieces is known as dissection. For Example a square may be partitioned into four polygonal pieces which can be rearranged to form equilateral triangle of the same areas. In [25], geometric dissection is used to cut geometric figures into smaller pieces, which can be rearranged to form other figures. The basic idea of dividing a polygon into triangles can be extended to dividing a polygon based on circles. In [2], the authors propose an algorithm to completely cover a region Ω with circles in two-dimensional plane. It is shown that the disk placement described in [2] has a minimum number of disks to cover a rectangle. If N circles are used to cover a region Ω. For n number of robots to patrol the region, the number of circles that each robot has to travel is calculated as

$f(N/n) \quad for \;\; n-1 \;\; robots,$

$where \;\; f(N/n) \;\; represents \;\; the \;\; result \;\; rounded \;\; off \;\; to \;\; nearest \;\; whole \;\; number$

The n^{th} robot has to travel $N-(f(N/n)(n-1))$. The areas are divided as

```
N is the total number of circles covering the full region
n is the number of robots
$m = N - \mathrm{int}(N/n)$, $x = \mathrm{int}(N/n)$
select the start point
set selected point = start point
While (total number of points selected less than m)
while( total number of points selected less than x)
find neighbors of points selected and add to list
set neighbors selected as point selected
count total number of points
end
update the total number of points selected
store the selected points in a seperate list
```

```
    find the point which is closest to a uncovered vertex and also
    to the last points selected
    set the point obtained in above step as selected point
end
The remaining points form a last list .
```

Once the actual area is divided into sub areas, each robot is assigned a particular sub area to get a complete coverage of the actual area.

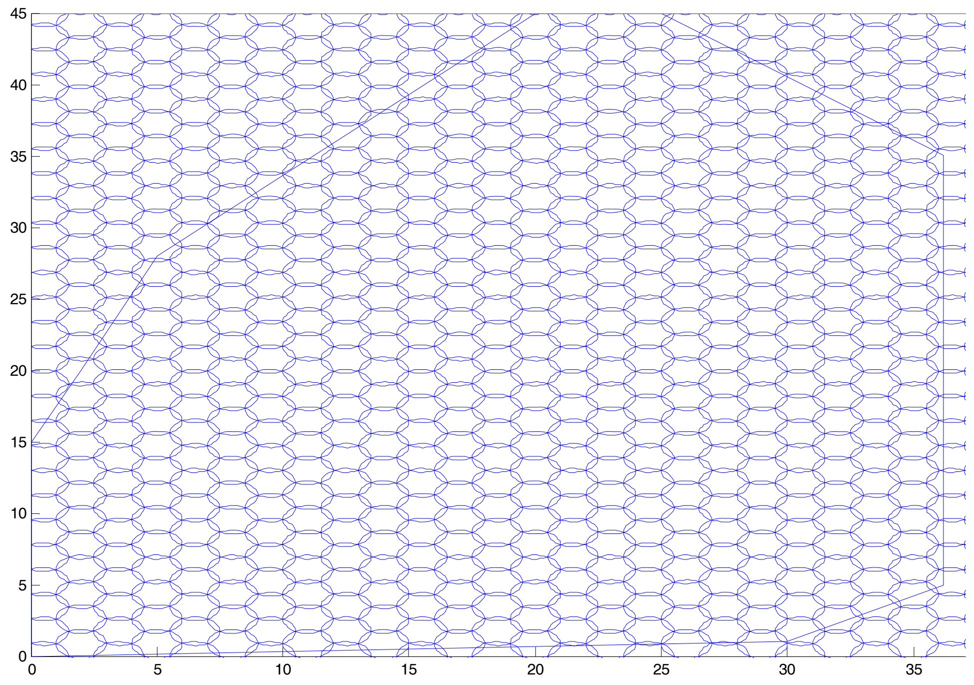

Figure 3.10 Polygon to be dissected

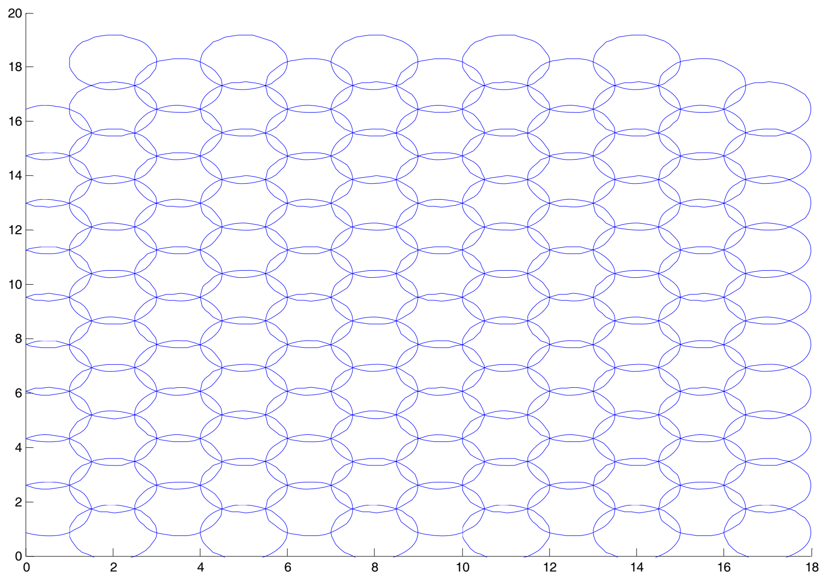

Figure 3.11 Dissection part A

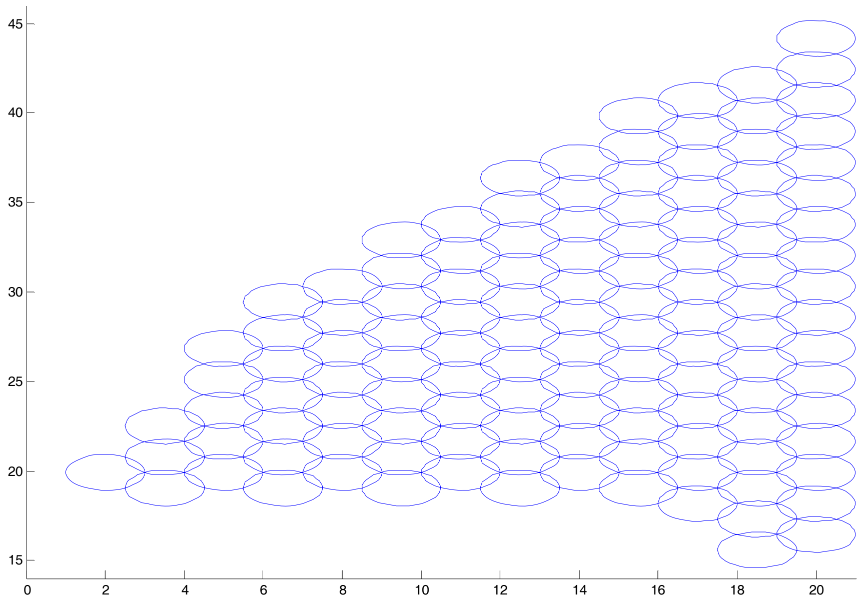

Figure 3.12 Dissection part B

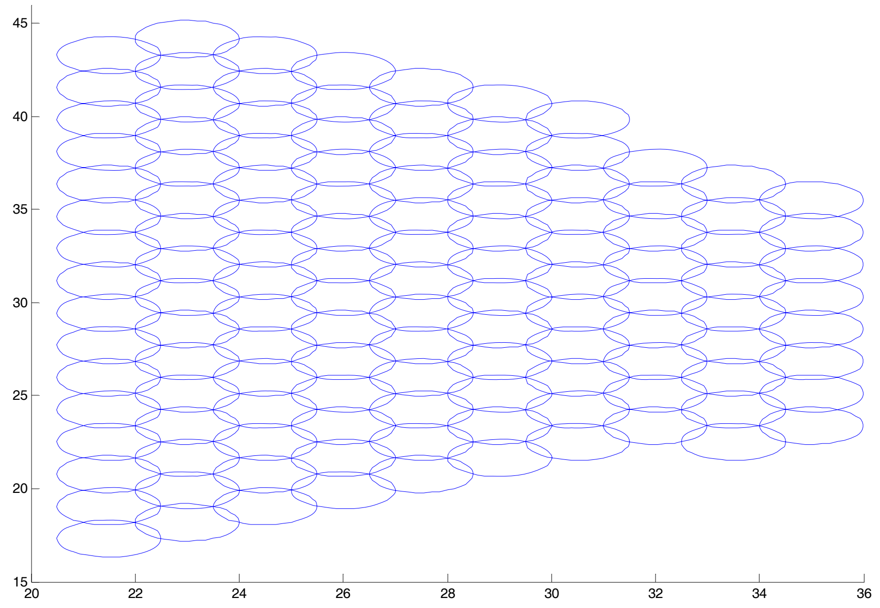

Figure 3.13 Dissection part C

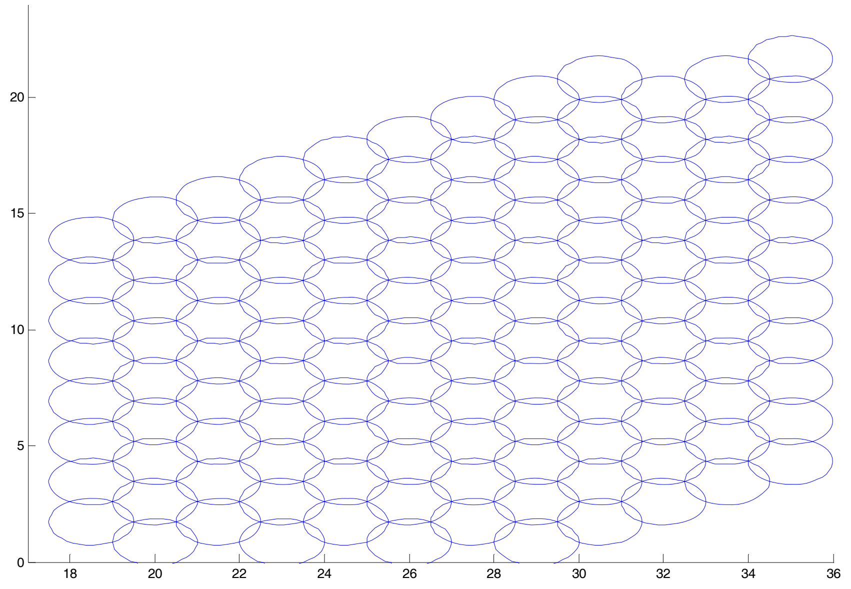

Figure 3.14 Dissection part D

CHAPTER FOUR: SMOOTH CURVES AND OPTIMAL ANALYSIS

4.1 Generation of Smooth Trajectory

Trajectory generation for the first two sequence of points is slightly different from the trajectory generation for the rest of the points in the sequence because in any path planning approach the robot will have an initial configuration which has to be taken into taken into account. The initial angle made by the centre of the wheel axle with the X-axis is used as the initial dy/dx and the initial $\frac{d^2y}{dx^2}$ is an input data from which the trajectory is generated. For the rest of the sequence, the slope and the rate of change of slope are calculated.

In this section [1] is used to calculate the slope of a curve at its initial and final points. The method considers a local subset of six points to define sequentially a local polynomial approximation to the curve between points 3 and 4 of the local subset. Define a cumulative polygon approximation to arc length, or Euclidean distance, as:

$$\left.\begin{aligned} & s_1 = 0 \\ & s_k = s_{k-1} + \sqrt{(x_k - x_{k-1})^2 - (y_k - y_{k-1})^2};\, k = 2,3,\ldots,6 \end{aligned}\right\} \tag{4.1}$$

by choosing a power series of the form and we have

$$x(s) = \sum_{i=1}^{n} A_i s^{i-1} \tag{4.2}$$

$$y(s) = \sum_{i=1}^{n} B_i s^{i-1} \tag{4.3}$$

The procedure is to use three overlapping cubic fits of x and y as functions of pseudo-arc length. We divide the six data points into 3 parts: left, middle and right. They are in the form of

$$x = A_1 + A_2 s + A_3 s^2 + A_4 s^3 \tag{4.4}$$

$$y = B_1 + B_2 s + B_3 s^2 + B_4 s^3 \tag{4.5}$$

where s in the equations are known and the coefficients of them can be calculated as

$$A_L = S_L{}^{-1} X_L \qquad B_L = S_L{}^{-1} Y_L \tag{4.6}$$

$$A_M = S_M{}^{-1} X_M \qquad B_M = S_M{}^{-1} Y_M \tag{4.7}$$

$$A_R = S_R{}^{-1} X_R \qquad B_R = S_R{}^{-1} Y_R \tag{4.8}$$

where the left middle and right coefficients are defined as a 4 x 1 matrix and denoted as

$$A_L = \begin{bmatrix} A_{1L} \\ A_{2L} \\ A_{3L} \\ A_{4L} \end{bmatrix}, \; B_L = \begin{bmatrix} B_{1L} \\ B_{2L} \\ B_{3L} \\ B_{4L} \end{bmatrix}, \; L \to M, R. \tag{4.9}$$

And the left, middle and right data matrices are defined as

$$X_L{}^T = [x_1 \quad x_2 \quad x_3 \quad x_4]; x \to y \tag{4.10}$$

$$X_M{}^T = [x_1 \quad x_2 \quad x_3 \quad x_4]; x \to y \tag{4.11}$$

$$X_R{}^T = [x_1 \quad x_2 \quad x_3 \quad x_4]; x \to y \tag{4.12}$$

and S to be inverted in (4.6), (4.7),(4.8) are

$$S_L = \begin{bmatrix} 1 & s_1^2 & s_1^3 & s_1^4 \\ 1 & s_2^2 & s_2^3 & s_2^4 \\ 1 & s_3^2 & s_3^3 & s_3^4 \\ 1 & s_4^2 & s_4^3 & s_4^4 \end{bmatrix} \tag{4.13}$$

$$S_M = \begin{bmatrix} 1 & s_2^2 & s_2^3 & s_2^4 \\ 1 & s_3^2 & s_3^3 & s_3^4 \\ 1 & s_4^2 & s_4^3 & s_4^4 \\ 1 & s_5^2 & s_5^3 & s_5^4 \end{bmatrix} \tag{4.14}$$

$$S_R = \begin{bmatrix} 1 & s_3^2 & s_3^3 & s_3^4 \\ 1 & s_4^2 & s_4^3 & s_4^4 \\ 1 & s_5^2 & s_5^3 & s_5^4 \\ 1 & s_6^2 & s_6^3 & s_6^4 \end{bmatrix} \tag{4.15}$$

The initial slope of the curve is obtained as

$$\frac{dx}{ds}\Big| s_3 = \frac{1}{2}\left[\frac{dx_L}{ds}\Big| s_3 + \frac{dx_M}{ds}\Big| s_3\right] \tag{4.16}$$

$$\frac{dy}{ds}\Big| s_3 = \frac{1}{2}\left[\frac{dy_L}{ds}\Big| s_3 + \frac{dy_M}{ds}\Big| s_3\right] \tag{4.17}$$

The final slope of the curve is obtained as

$$\frac{dx}{ds}\Big| s_4 = \frac{1}{2}\left[\frac{dx_M}{ds}\Big| s_4 + \frac{dx_R}{ds}\Big| s_4\right] \tag{4.18}$$

$$\frac{dy}{ds}\Big| s_4 = \frac{1}{2}\left[\frac{dy_M}{ds}\Big| s_4 + \frac{dy_R}{ds}\Big| s_4\right] \tag{4.19}$$

The initial rate of change of the slope is obtained as

$$\frac{d^2x}{ds^2}\Big| s_3 = \frac{1}{2}\left[\frac{d^2x_L}{ds^2}\Big| s_3 + \frac{d^2x_M}{ds^2}\Big| s_3\right] \tag{4.20}$$

$$\frac{d^2y}{ds^2}\Big| s_3 = \frac{1}{2}\left[\frac{d^2y_L}{ds^2}\Big| s_3 + \frac{d^2y_M}{ds^2}\Big| s_3\right] \tag{4.21}$$

The final rate of change of the slope is obtained as

$$\frac{d^2x}{ds^2}|s_4 = \frac{1}{2}\left[\frac{d^2x_L}{ds^2}|s_4 + \frac{d^2x_M}{ds^2}|s_4\right] \tag{4.22}$$

$$\frac{d^2y}{ds^2}|s_4 = \frac{1}{2}\left[\frac{d^2y_L}{ds^2}|s_4 + \frac{d^2y_M}{ds^2}|s_4\right] \tag{4.23}$$

The above initial and final conditions become the boundary conditions.

Consider an equation of the form

$$x = A(1) + A(2)s + A(3)s^2 + A(4)s^3 + A(5)s^4 + A(6)s^5 \tag{4.20}$$

$$y = B(1) + B(2)s + B(3)s^2 + B(4)s^3 + B(5)s^4 + B(6)s^5 \tag{4.21}$$

The coefficients of the above equations can be obtained from the boundary conditions. For a non holonomic mobile robot the boundary conditions are as follows

$$x_i = A(1) + A(2)s_i + A(3)s_i^2 + A(4)s_i^3 + A(5)s_i^4 + A(6)s_i^5 \ ; x \to y \tag{4.21}$$

$$x_{i+1} = A(1) + A(2)s_{i+1} + A(3)s^2{}_{i+1} + A(4)s^3{}_{i+1} + A(5)s^4{}_{i+1} + A(6)s^5{}_{i+1} \ ; x \to y \tag{4.22}$$

$$\frac{dx_i}{ds} = A(2) + 2A(3)s_i + 3A(4)s_i^2 + 4A(5)s_i^3 + 5A(6)s_i^4 \ ; x \to y \tag{4.23}$$

$$\frac{dx_{i+1}}{ds} = A(2) + 2A(3)s_{i+1} + 3A(4)s^2{}_{i+1} + 4A(5)s^3{}_{i+1} + 5A(6)s^4{}_{i+1} \ ; x \to y \tag{4.24}$$

$$\frac{d^2x_i}{ds} = 2A(3) + 6A(4)s_i + 12A(5)s_i^2 + 20A(6)s_i^3; \quad x \to y \tag{4.25}$$

$$\frac{d^2x_{i+1}}{ds^2} = 2A(3) + 6A(4)s_{i+1} + 12A(5)s_{i+1}^2 + 20A(6)s_{i+1}^3; \quad x \to y \tag{4.26}$$

where (x_i, y_i) and (x_{i+1}, y_{i+1}) represent the initial and final (x, y) coordinates. Rewriting the above equation in matrix from.

$$S \quad = \quad \begin{bmatrix} 1 & s_i & s_i^{\,2} & s_i^{\,3} & s_i^{\,4} & s_i^{\,5} \\ 1 & s_{i+1} & s^2{}_{i+1} & s^3{}_{i+1} & s^4{}_{i+1} & s^5{}_{i+1} \\ 0 & 1 & 2s_i & 3s_i^{\,2} & 4s_i^{\,3} & 5s_i^{\,4} \\ 0 & 1 & 2s_{i+1} & 3s^2{}_{i+1} & 4s^3{}_{i+1} & 5s_i^{\,4} \\ 0 & 0 & 2 & 6s_i & 12s_i^{\,2} & 20s_i^{\,3} \\ 0 & 0 & 2 & 6s_{i+1} & 12s^2{}_{i+1} & 20s^3{}_{i+1} \end{bmatrix} \tag{4.27}$$

$$\mathrm{A} \;=\begin{bmatrix} A(1) & A(2) & A(3) & A(4) & A(5) & A(6) \end{bmatrix}^T ;\; A \to B \tag{4.28}$$

$$X \;= \quad \begin{bmatrix} x_i \\ x_{i+1} \\ \dfrac{dx_i}{ds} \\ \dfrac{dx_{i+1}}{ds} \\ \dfrac{d^2 x_i}{ds} \\ \dfrac{d^2 x_{i+1}}{ds} \end{bmatrix} \tag{4.29}$$

The coefficient matrix A can be found from

$$A = S^{-1} X \tag{4.30}$$

The coefficient matrix B can be found from

$$B = S^{-1}Y$$

Therefore

$$x = A(1) + A(2)s + A(3)s^2 + A(4)s^3 + A(5)s^4 + A(6)s^5 \tag{4.31}$$

$$y = B(1) + B(2)s + B(3)s^2 + B(4)s^3 + B(5)s^4 + B(6)s^5$$

Represents a smooth trajectory between given set of points.

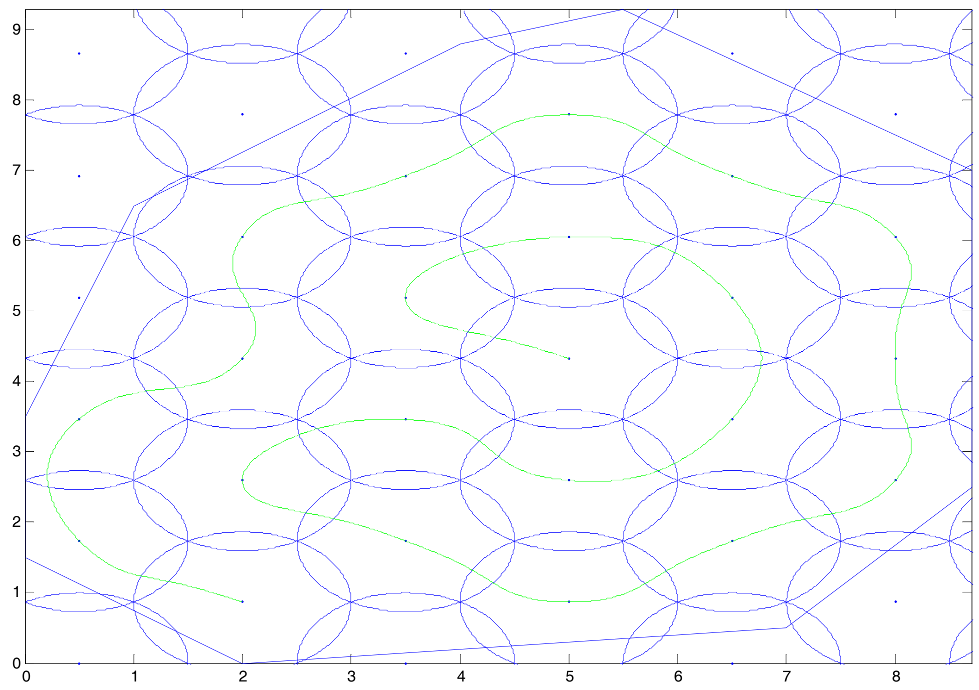

Figure 4.1 Smooth trajectory

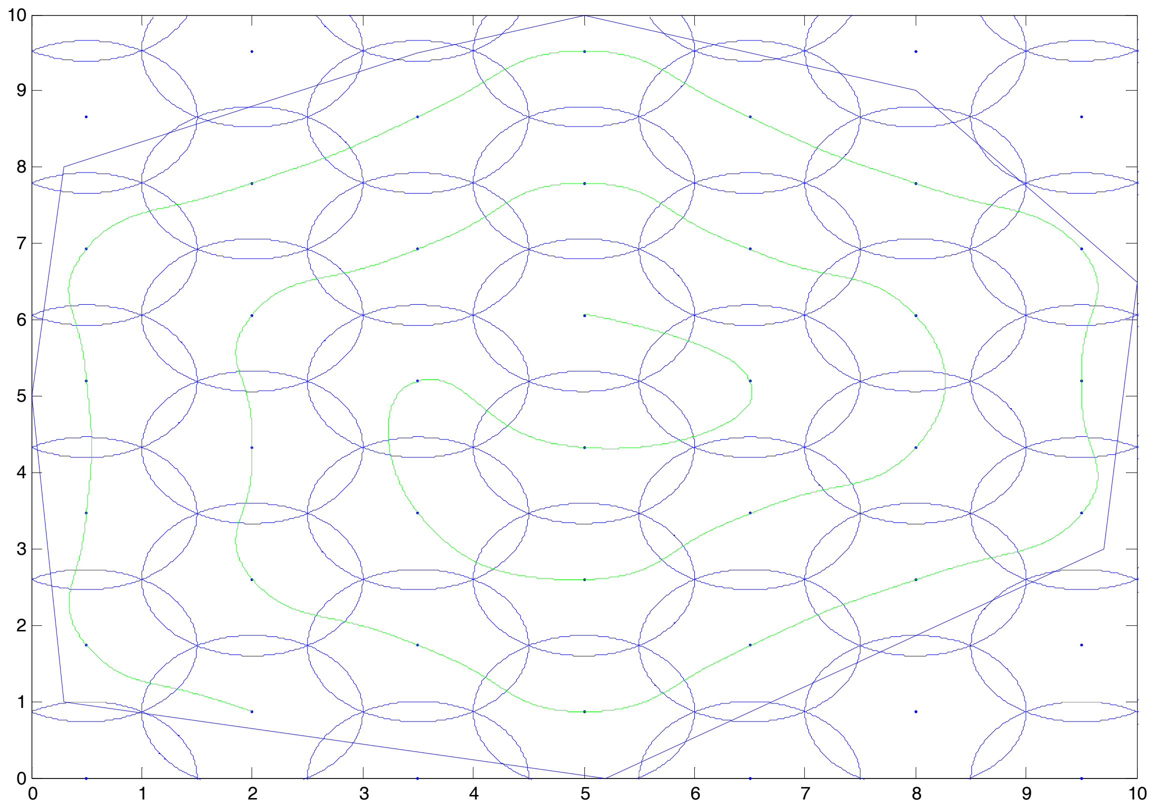

Figure 4.2 Smooth trajectory B

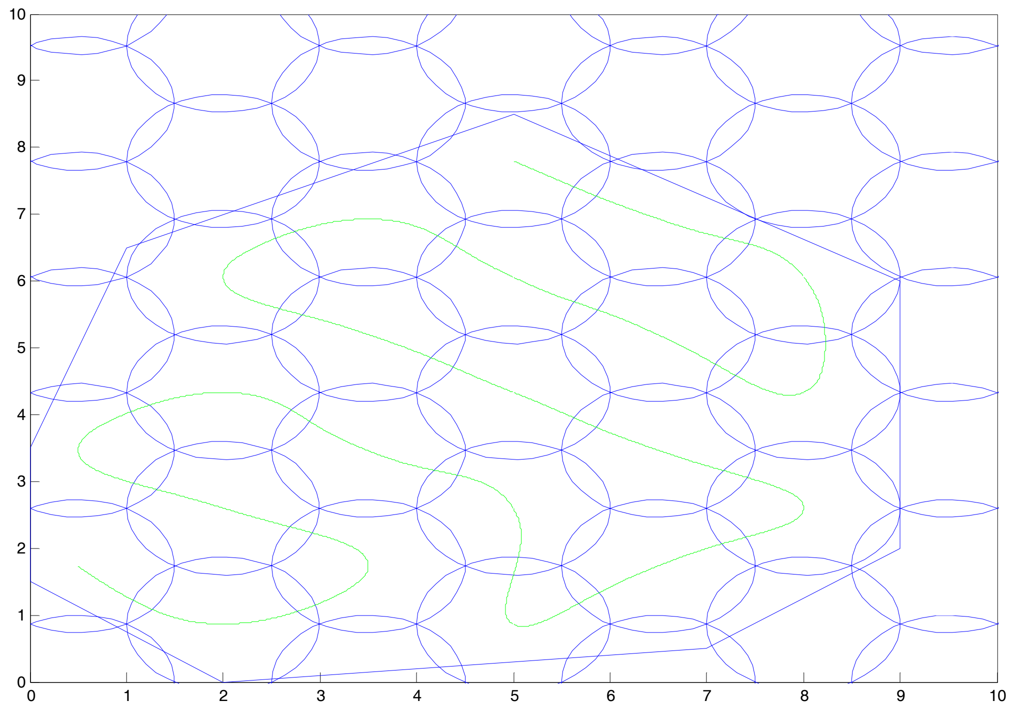

Figure 4.3 Smooth trajectory C

4.2 Optimal analysis of trajectory

An optimal trajectory is obtained by minimizing the change-rate of the curvature. The path is parameterized by a quintic η-spline, devised to guarantee the overall second order geometric continuity of a composite path interpolating an arbitrary sequence of points. Starting from the closed form η- parameterization of the spline, an optimization criterion is proposed to design smooth curves. The aim is to plan curves where the curvature variability is kept as small as possible. Closed from expressions of the η- spline can be presented as follows

$$p(u) = \begin{bmatrix} x(u) \\ y(u) \end{bmatrix} \tag{4.32}$$

$$x(u) = x_0 + x_1 u + x_2 u^2 + x_3 u^3 + x_4 u^4 + x_5 u^5 \qquad (4.33)$$

$$y(u) = y_0 + y_1 u + y_2 u^2 + y_3 u^3 + y_4 u^4 + y_5 u^5 \qquad (4.34)$$

Where

$$x_0 = x_A \qquad (4.35)$$

$$x_1 = \eta_1 \cos\theta_A \qquad (4.36)$$

$$x_2 = \frac{1}{2}(\eta_3 \cos\theta_A - \eta_1{}^2 \kappa_A \sin\theta_A) \qquad (4.37)$$

$$x_3 = 10(x_B - x_A) - (6\eta_1 + \frac{3}{2}\eta_3)\cos\theta_A - (4\eta_2 - \frac{1}{2}\eta_4)\cos\theta_B + \frac{3}{2}\eta_1{}^2\kappa_A \sin\theta_A - \frac{1}{2}\eta_2{}^2\kappa_B \sin\theta_B \quad (4.38)$$

$$x_4 = -15(x_B - x_A) + (8\eta_1 + \frac{3}{2}\eta_3)\cos\theta_A + (7\eta_2 - \eta_4)\cos\theta_B - \frac{3}{2}\eta_1{}^2\kappa_A \sin\theta_A + \eta_2{}^2\kappa_B \sin\theta_B \quad (4.39)$$

$$x_5 = 6(x_B - x_A) - (3\eta_1 + \frac{1}{2}\eta_3)\cos\theta_A - (3\eta_2 - \frac{1}{2}\eta_4)\cos\theta_B + \frac{1}{2}\eta_1{}^2\kappa_A \sin\theta_A - \frac{1}{2}\eta^2{}_2\kappa_B \sin\theta_B \quad (4.40)$$

$$y_0 = y_A \qquad (4.41)$$

$$y_1 = \eta_1 \sin\theta_A \qquad (4.42)$$

$$y_2 = \frac{1}{2}(\eta_3 \sin\theta_A - \eta_1{}^2 \kappa_A \cos\theta_A) \qquad (4.43)$$

$$y_3 = 10(y_B - y_A) - (6\eta_1 + \frac{3}{2}\eta_3)\sin\theta_A - (4\eta_2 - \frac{1}{2}\eta_4)\sin\theta_B + \frac{3}{2}\eta_1{}^2\kappa_A \cos\theta_A - \frac{1}{2}\eta_2{}^2\kappa_B \cos\theta_B \quad (4.44)$$

$$y_4 = -15(y_B - y_A) + (8\eta_1 + \frac{3}{2}\eta_3)\sin\theta_A + (7\eta_2 - \eta_4)\sin\theta_B - \frac{3}{2}\eta_1{}^2\kappa_A \cos\theta_A + \eta_2{}^2\kappa_B \cos\theta_B \quad (4.45)$$

$$y_5 = 6(y_B - y_A) - (3\eta_1 + \frac{1}{2}\eta_3)\sin\theta_A - (3\eta_2 - \frac{1}{2}\eta_4)\sin\theta_B + \frac{1}{2}\eta_1{}^2\kappa_A \cos\theta_A - \frac{1}{2}\eta^2{}_2\kappa_B \cos\theta_B \quad (4.46)$$

Subscripts A and B indicate the assigned interpolating conditions relative to the spline endpoints

while $\eta = [\eta_1 \quad \eta_2 \quad \eta_3 \quad \eta_4]^T \in \mathrm{H} = (-\infty, +\infty)$ is a vector.

It is clear that with a proper selection of η, it is possible to obtain a wide number of shapes for the path, all of then satisfying the interpolating conditions at the curve. This suggests choosing the four parameters according to some sort of optimality criterion. η is selected by minimizing the change–rate of the curvature. By minimizing $\left|d\kappa/ds\right|$, where κ represents the curvature and is given by

$$\kappa = \frac{\dot{x}\ddot{y} - \ddot{x}\dot{y}}{((\dot{x})^2 + (\dot{y})^2)^{3/2}} \tag{4.47}$$

$$\frac{d\kappa}{ds} = \frac{d\kappa/du}{\|\dot{p}\|} \tag{4.47}$$

The variability of the steering angle can be indirectly reduced by minimizing the maximum of $|d\kappa/ds|$ along the whole path. Thus the optimization problem can be formulated as

$$\min \quad \max\left|\frac{d\kappa}{ds}\right| \quad , \eta \in \mathrm{H} \quad \textit{and} \quad u \in [0,1] \tag{4.48}$$

Subject to

$$\|\dot{p}(u)\| \quad > \quad 0 \tag{4.49}$$

The minimax problem can be converted into a semi infinite problem by adding to vector η an auxiliary variable η_5. The optimization then becomes

$$\min \eta_5 \tag{4.50}$$

Subject to

$$\left|\frac{d\kappa}{ds}\right| \le \eta_5 \qquad\qquad \forall \quad u \in [0,1]; \tag{4.51}$$

$$\|\dot{p}\| > 0 \qquad \forall \quad u \in [0,1]. \qquad (4.52)$$

The interpolating conditions are obtained from the curve for which qualitative analysis has to be done. The curves are then compared graphically to obtain an overall qualitative analysis. The results obtained infer that the smooth trajectory obtained from [3] and the optimal curve so obtained are qualitatively the same.

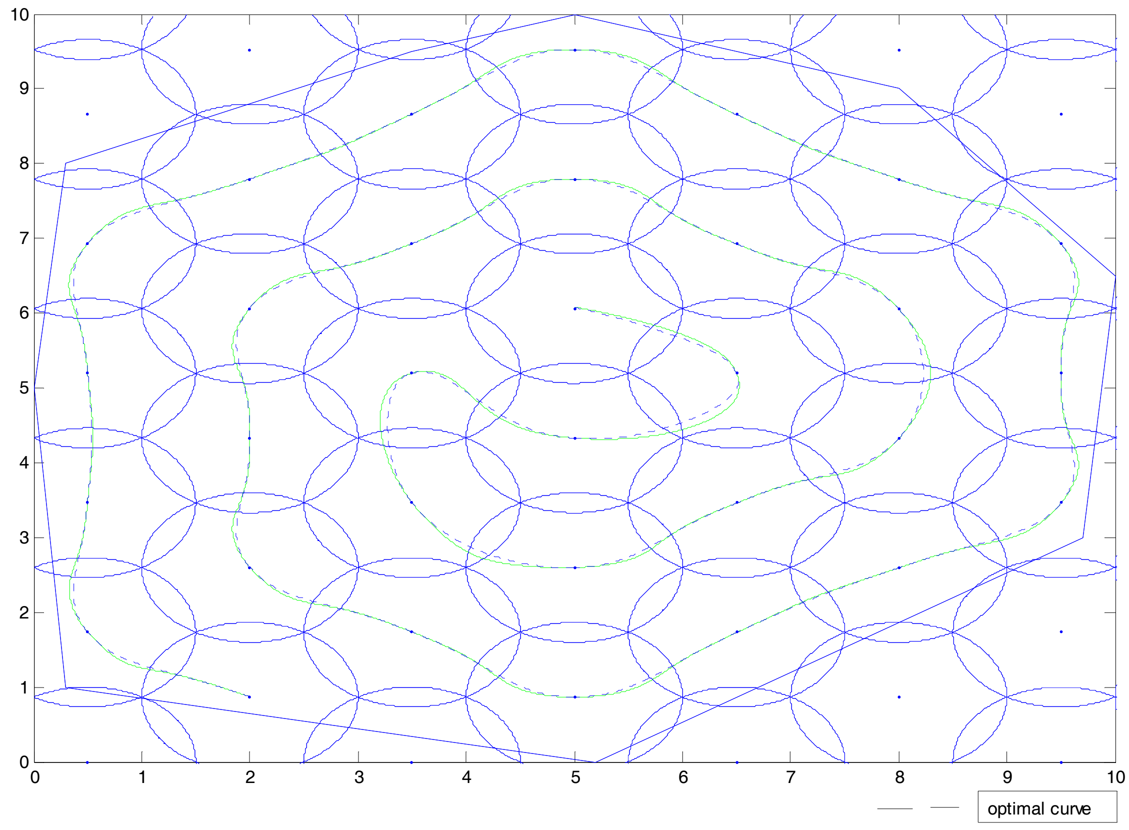

Figure 4.4 Optimal curve

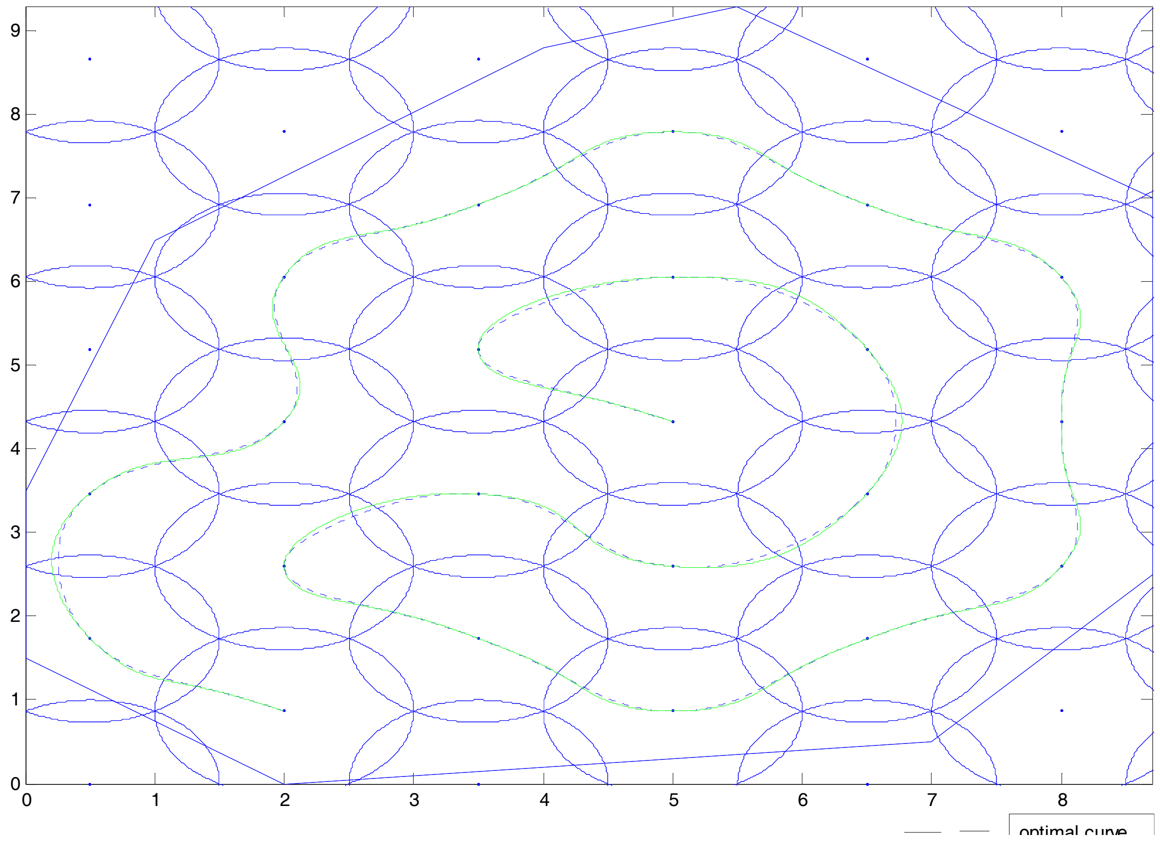

Figure 4.5 Optimal curve B

CHAPTER FIVE: GENERATION OF CONTROL INPUTS

This Chapter deals with the generation of control inputs for the mobile autonomous robot. From the smooth trajectory, the slope is calculated and by mathematical operations u_1 and u_2 are obtained where u_1 is the linear velocity of the driving wheels and u_2 is the steering velocity.

$$x = A_1 + A_2 s + A_3 s^2 + A_4 s^3 + A_5 s^4 + A_6 s^5 \tag{5.1}$$

$$y = B_1 + B_2 s + B_3 s^2 + B_4 s^3 + B_5 s^4 + B_6 s^5 \tag{5.2}$$

the equations (5.1) and (5.2) represent segment of a curve for which the control inputs are to be generated. From the Robot dynamic equations control input u_1 is given by (5.3)

$$u_1 = \frac{dx}{\cos\theta} \tag{5.3}$$

$$\theta = \tan^{-1}(dy / dx) \tag{5.4}$$

$$dx = A_2 + 2A_3 s + 3A_4 s^2 + 4A_5 s^3 + 5A_6 s^4 \tag{5.5}$$

$$dy = B_2 + 2B_3 s + 3B_4 s^2 + 4B_5 s^3 + 5B_6 s^4 \tag{5.6}$$

$$d^2 y = 2B_3 + 6B_4 s + 12B_5 s^2 + 20B_6 s^3 \tag{5.7}$$

$$d^2 x = 2A_3 + 6A_4 s + 12A_5 s^2 + 20A_6 s^3 \tag{5.8}$$

$$d^3 y = 6B_4 + 24B_5 s + 60B_6 s^2 \tag{5.9}$$

$$d^3 x = 6A_4 + 24A_5 s + 60A_6 s^2 \tag{5.10}$$

$$d\theta = \frac{(dx)^2}{(dx)^2 + (dy)^2}\left[\frac{dxd^2 y - dyd^2 x}{(dx)^2}\right] \tag{5.11}$$

the control input u_2 is given by

$$\begin{aligned} u_2 &= d\phi \\ \phi &= \tan^{-1}(ld\theta / u_1) \end{aligned} \qquad (5.12)$$

We get $$u_2 = u_{21} + u_{22} + u_{23} \qquad (5.13)$$

$$u_{21} = \frac{1 \cdot \left(dx^3 \cdot d^2 \cdot y - dx^2 \cdot dy \cdot d^2 \cdot x\right) \cdot \left(\frac{2 \cdot dy \cdot d^2 \cdot y}{dx^2} - \frac{2 \cdot dy^2 \cdot d^2 \cdot x}{dx^3}\right)}{2 \cdot \left(1 + \frac{dy^2}{dx^2}\right)^{\frac{3}{2}} \cdot \left(dx^5 + dx^3 \cdot dy^2\right)}$$

$$u_{22} = \frac{(+1) \cdot \left(2 \cdot dx^2 \cdot d^2 \cdot y \cdot d^2 \cdot x + dx^3 \cdot d^3 \cdot y - 2 \cdot dx \cdot dy \cdot \left(d^2 \cdot x\right)^2 - dx^2 \cdot dy \cdot d^3 \cdot x\right)}{\left(1 + \frac{dy^2}{dx^2}\right)^{\frac{1}{2}} \cdot \left(dx^5 + dx^3 \cdot dy^2\right)}$$

$$u_{23} = \frac{(-1) \cdot \left(dx^3 \cdot d^2 \cdot y - dx^2 \cdot dy \cdot d^2 \cdot x\right) \cdot \left(5 \cdot dx^4 \cdot d^2 \cdot x + 3 \cdot dx^2 \cdot dy^2 \cdot d^2 \cdot x + 2 \cdot dx^3 \cdot dy \cdot d^2 \cdot y\right)}{\left(1 + \frac{dy^2}{dx^2}\right)^{\frac{1}{2}} \cdot \left(dx^5 + dx^3 \cdot dy^2\right)^2}$$

$$dx = A_2 + 2A_3 s + 3A_4 s^2 + 4A_5 s^3 + 5A_6 s^4 \qquad (5.14)$$

$$dy = B_2 + 2B_3 s + 3B_4 s^2 + 4B_5 s^3 + 5B_6 s^4 \qquad (5.15)$$

$$d^2 y = 2B_3 + 6B_4 s + 12B_5 s^2 + 20B_6 s^3 \qquad (5.16)$$

$$d^3 y = 6B_4 + 24B_5 s + 60B_6 s^2 \qquad (5.17)$$

$$d^2 x = 2A_3 + 6A_4 s + 12A_5 s^2 + 20A_6 s^3 \qquad (5.18)$$

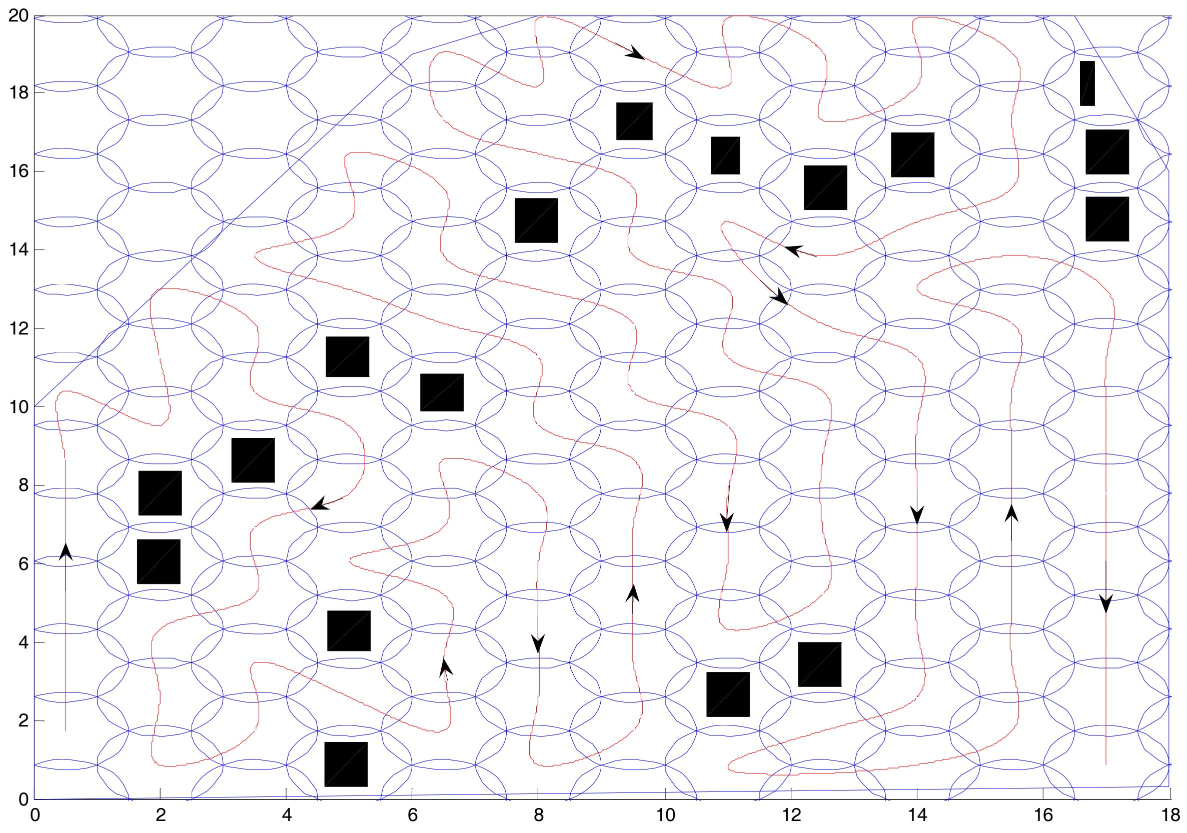

Figure 5.1 Robot trajectory

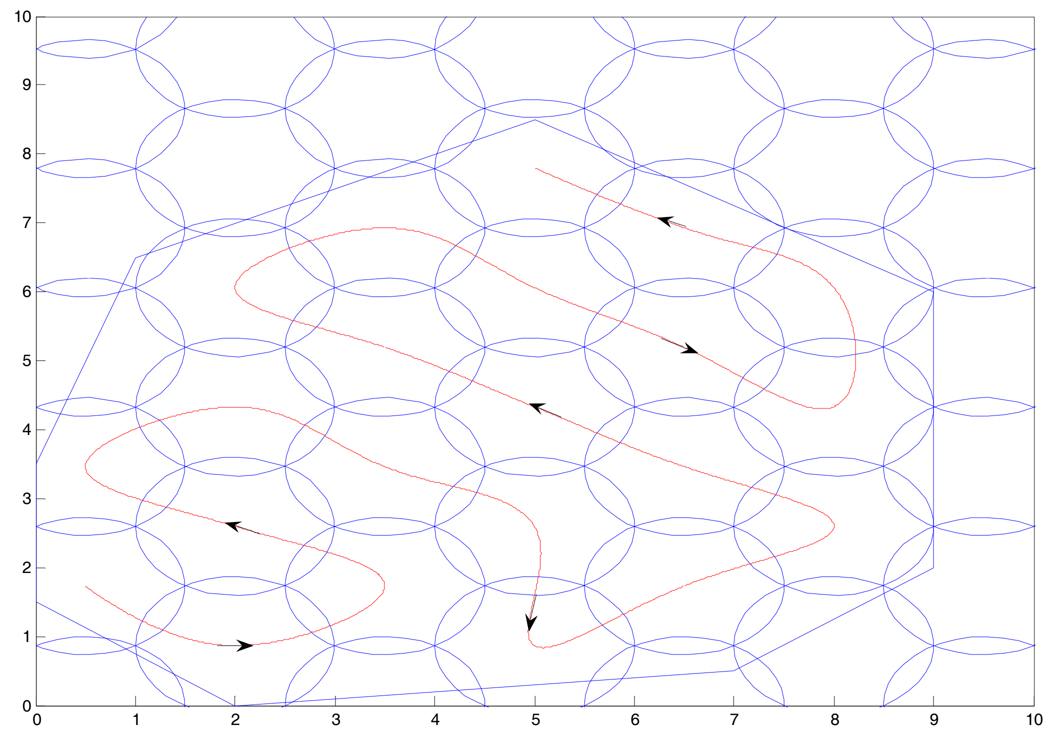

Figure 5.2 Robot trajectory

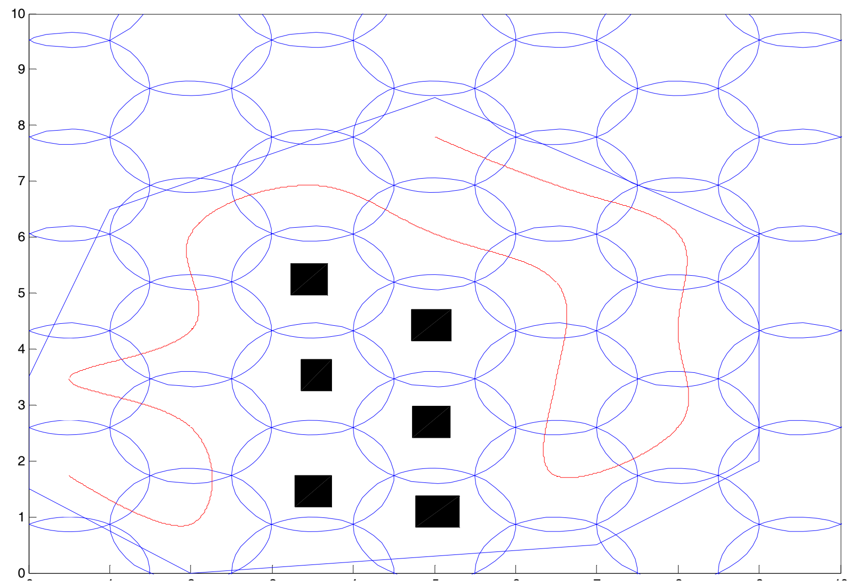

Figure 5.3 Robot trajectory

CHAPTER SIX: SIMULATION RESULTS

6.1 Complete Coverage Design

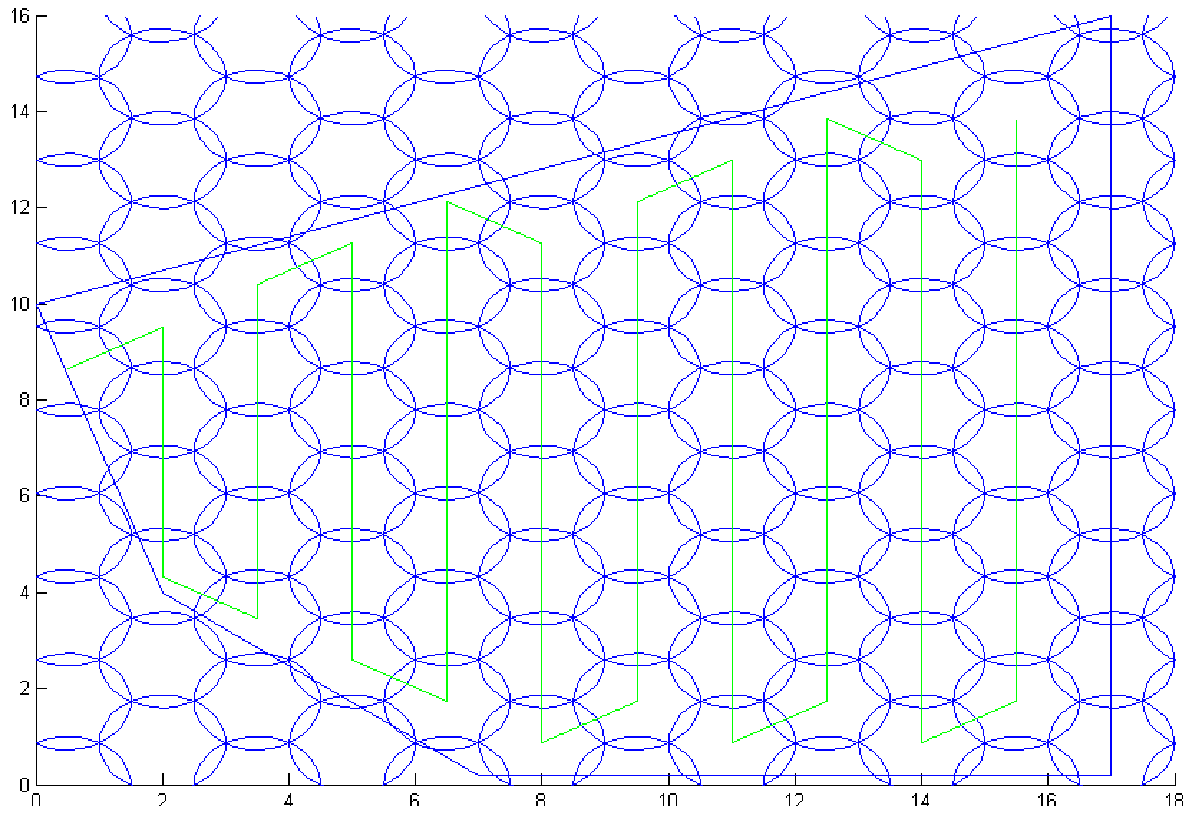

Figure 6.1 Complete coverage design

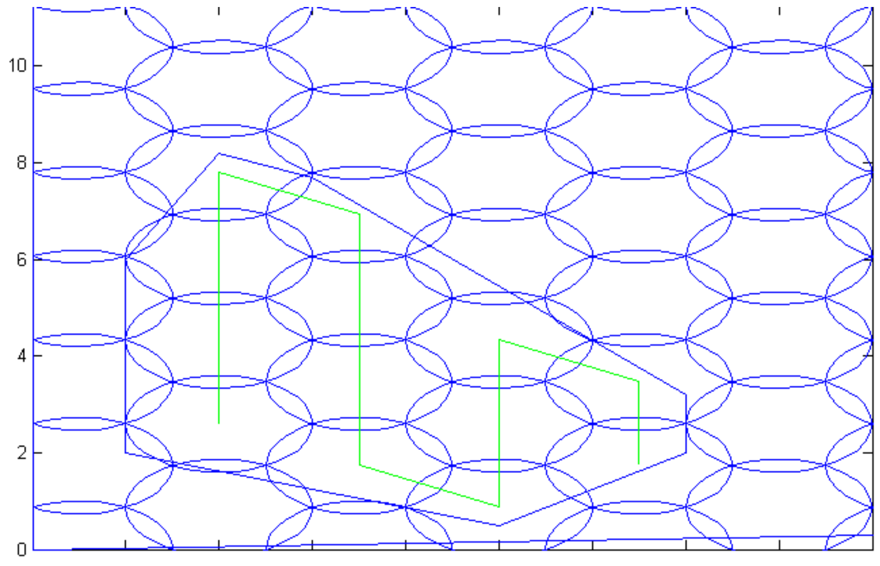

Figure 6.2 Complete coverage design B

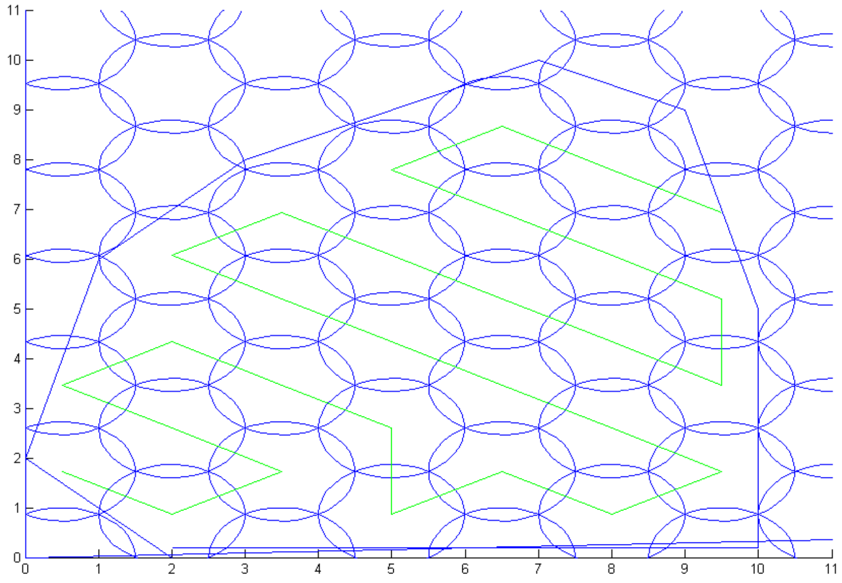

Figure 6.3 Complete coverage design C

6.2 Generation of Smooth Curves

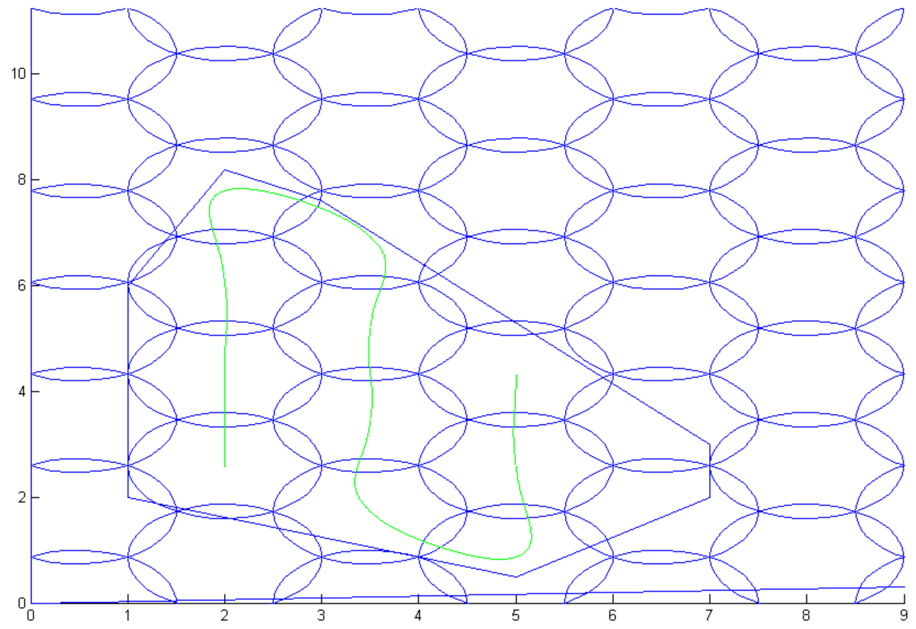

Figure 6.4 Smooth trajectory

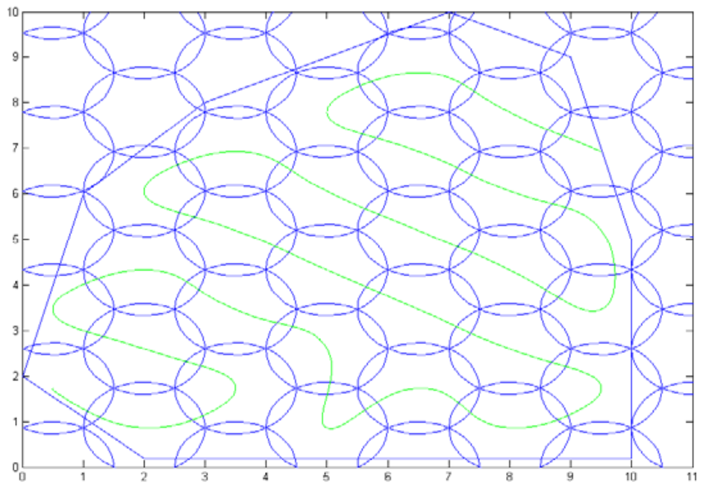

Figure 6.5 Smooth trajectory B

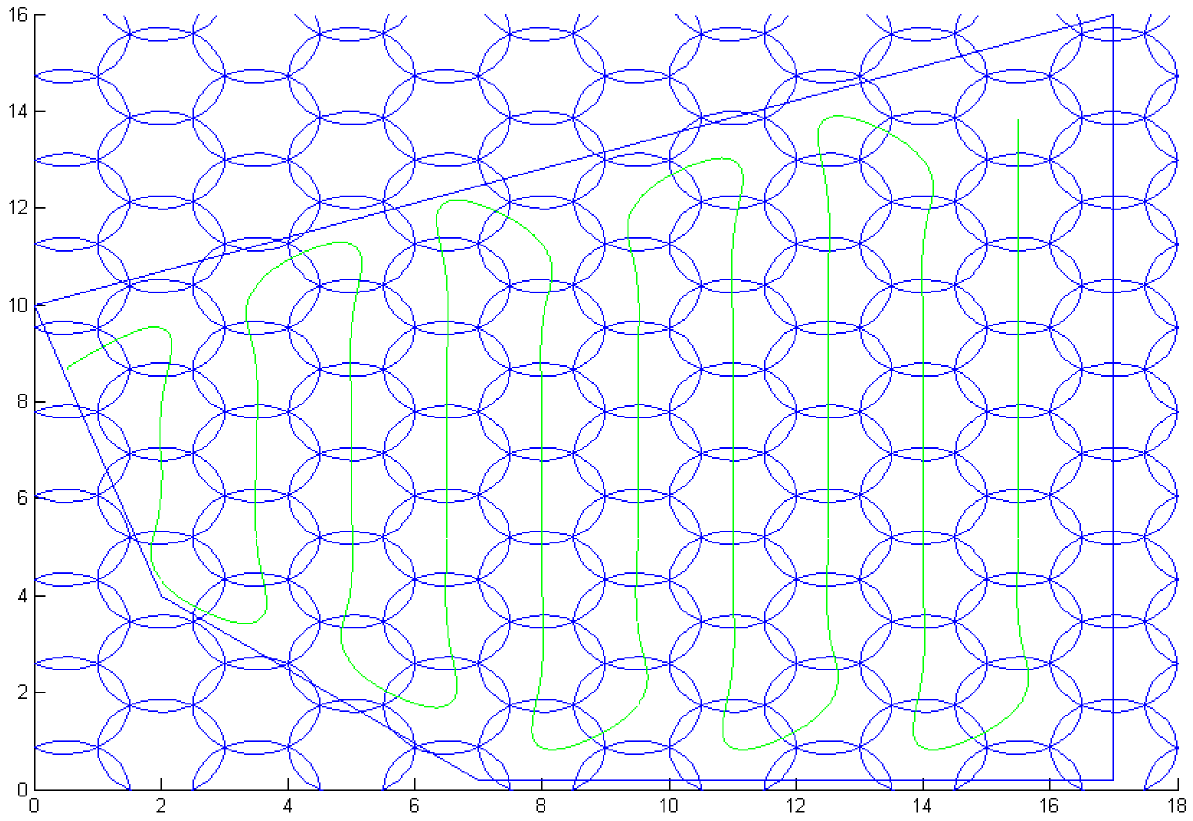

Figure 6.6 Smooth trajectory C

6.3 Qualitative Analysis

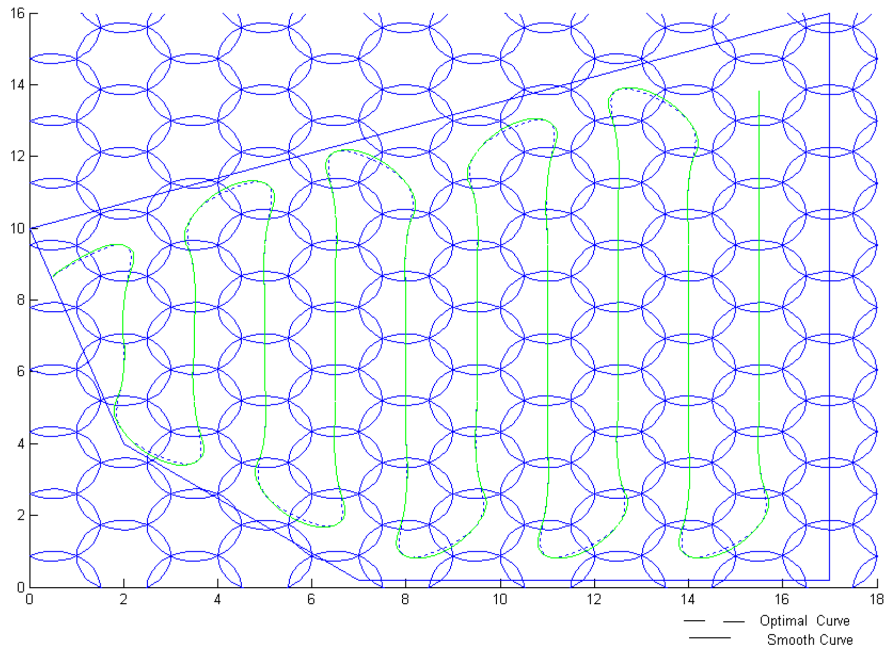

Figure 6.7 Qualitative analysis

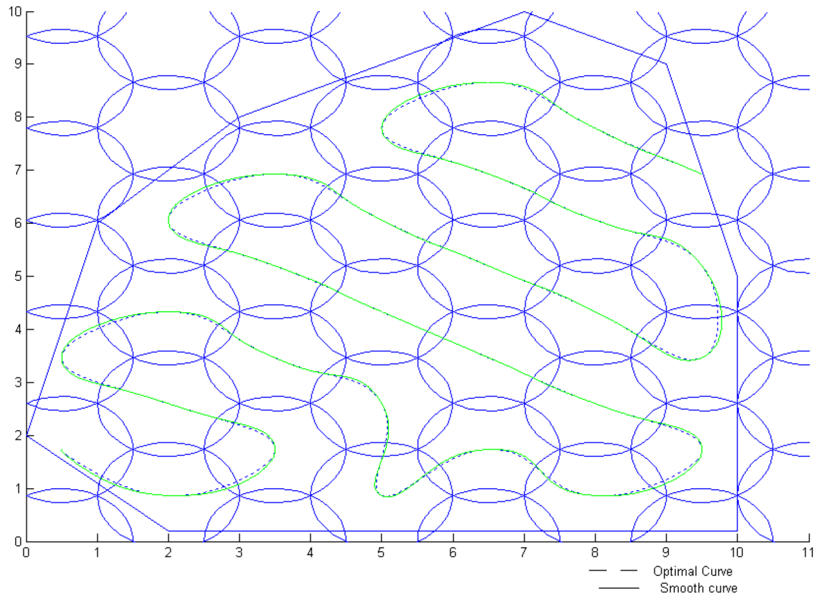

Figure 6.8 Qualitative analysis B

6.4 Dynamic Obstacle Avoidance

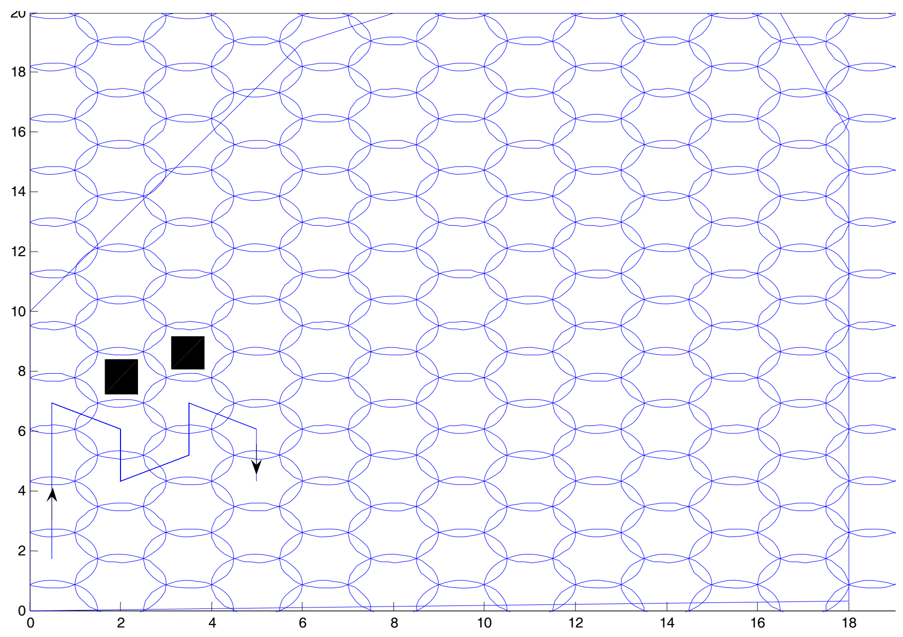

Figure 6.9 Robot path in the presence of dynamic obstacles

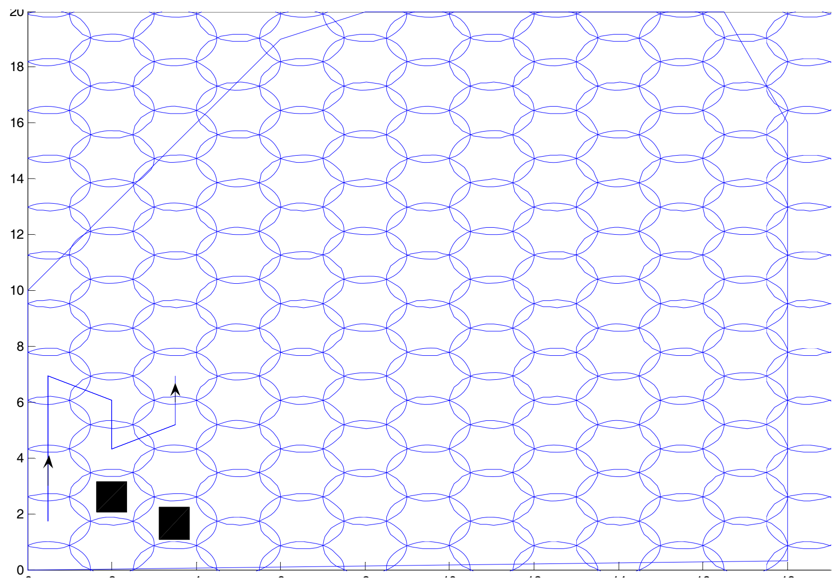

Figure 6.10 Robot path in the presence of dynamic obsatcles B

6.5 Generation of Robot Trajectory

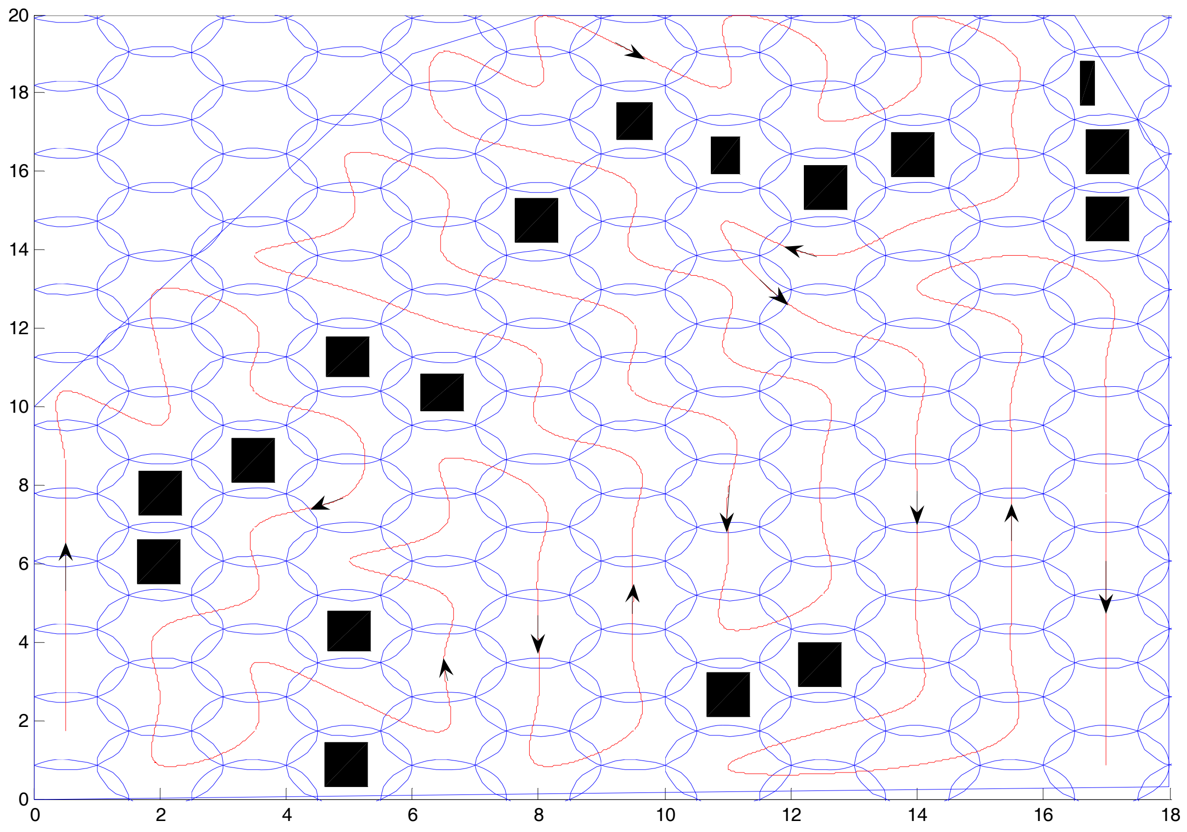

Figure 6.11 Robot path trajectory

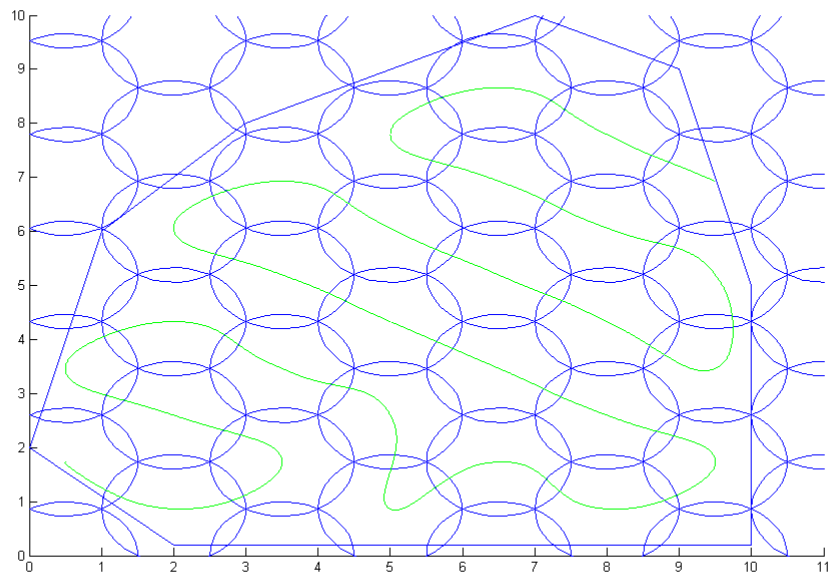

Figure 6.12 Robot trajectory B

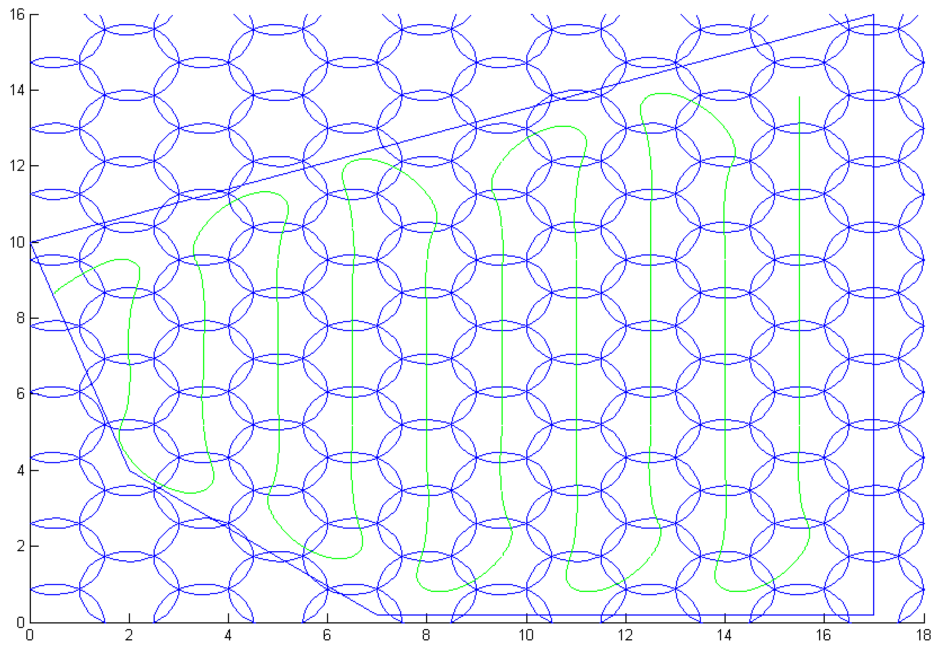

Figure 6.13 Robot path trajectory C

6.6 Cooperative Sweeping of Multi Robots

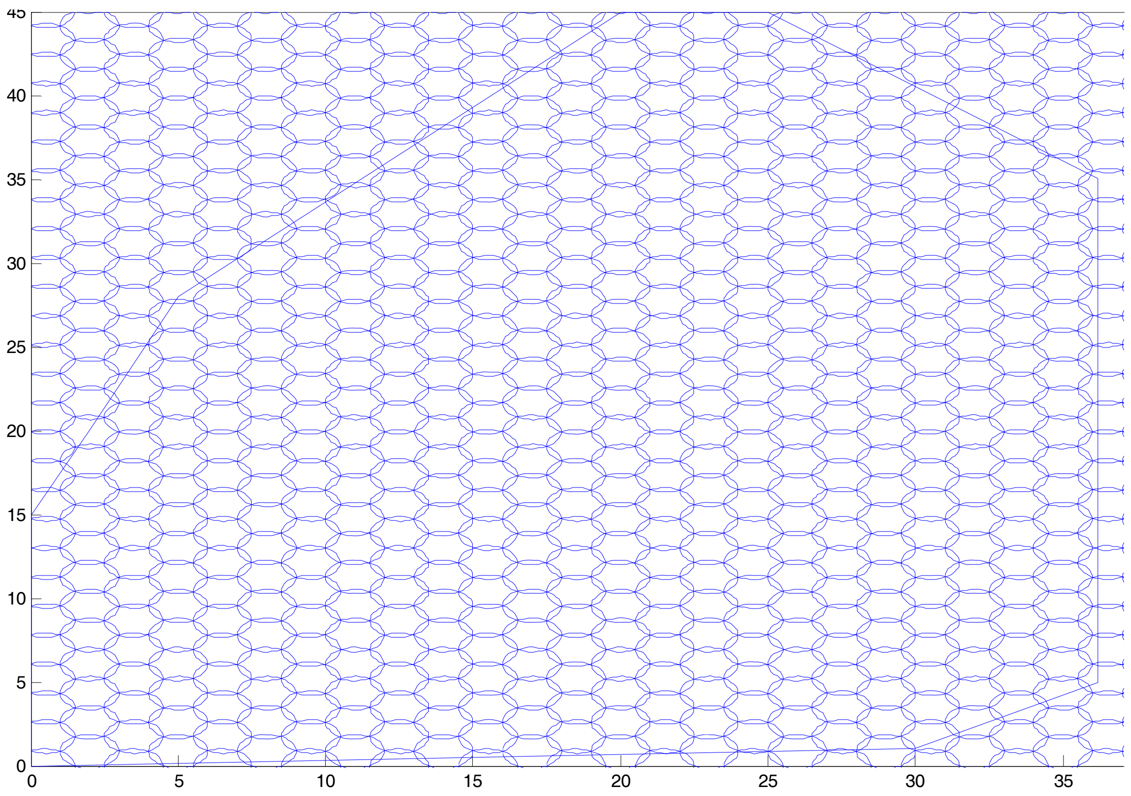

Figure 6.14 Area to be swept

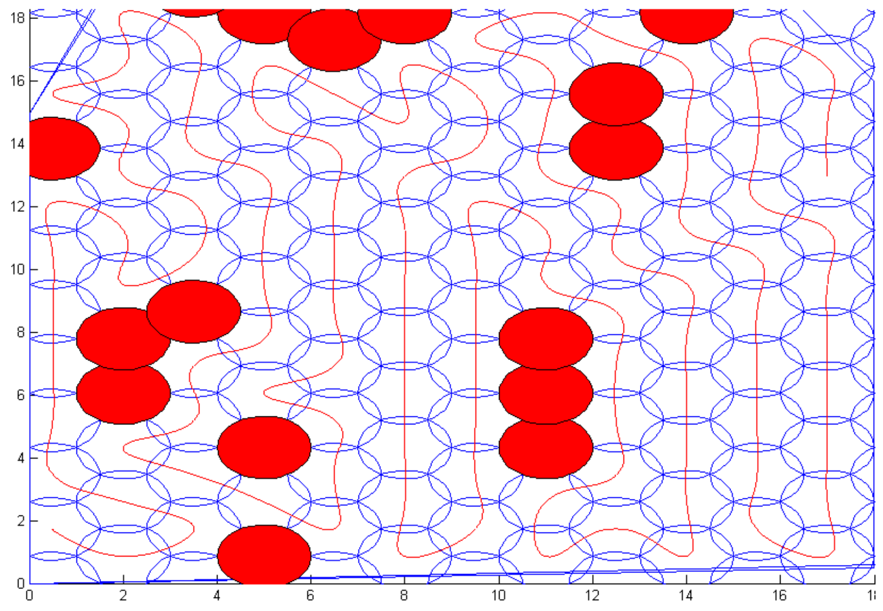

Figure 6.15 Cooperative sweeping

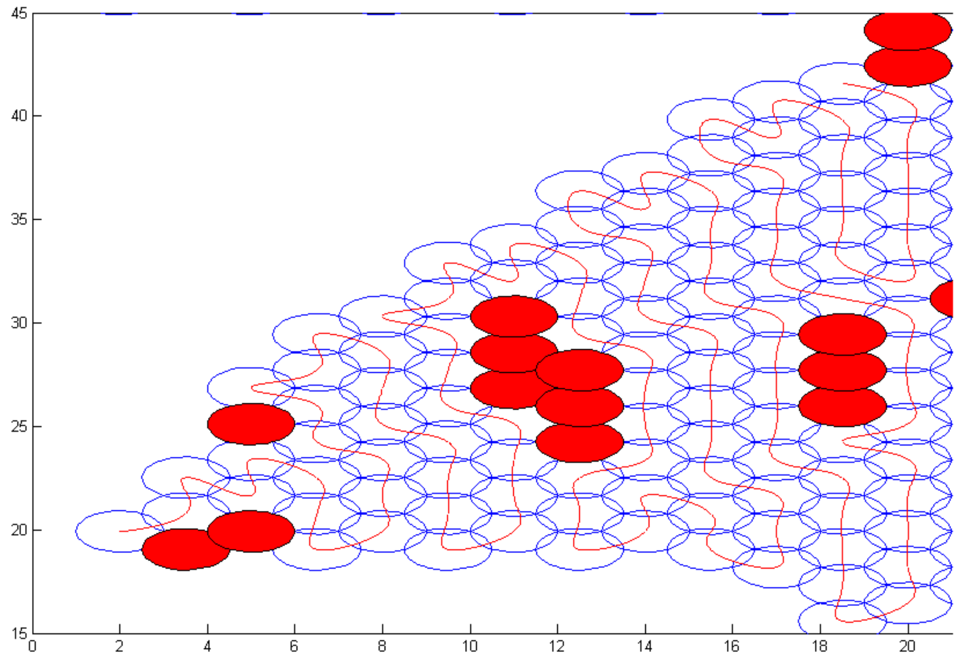

Figure 6.16 Cooperative sweeping B

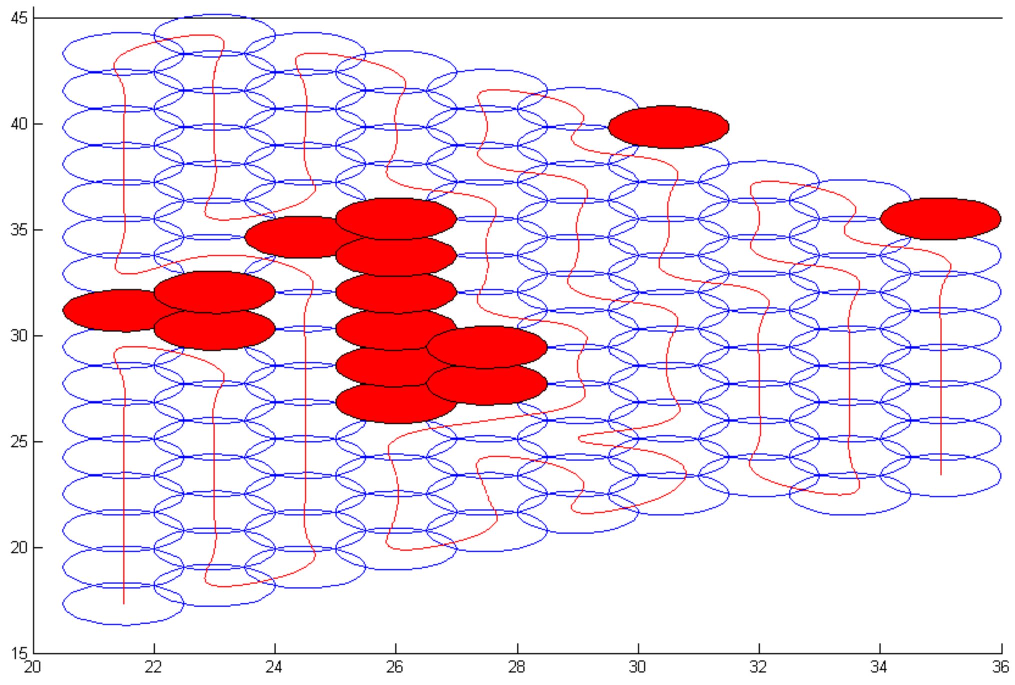

Figure 6.17 Cooperative sweeping C

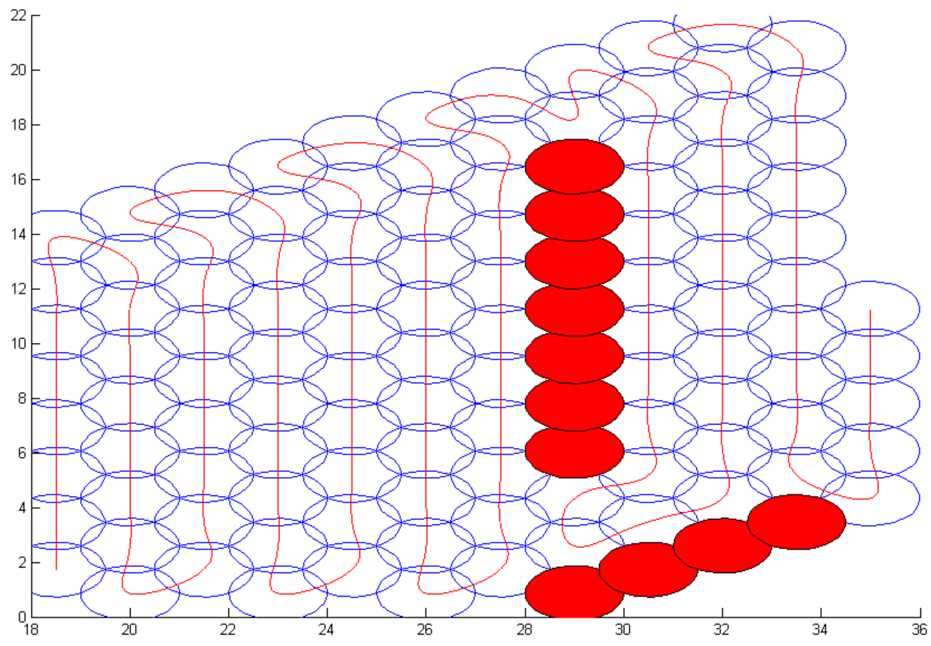

Figure 6.18 Cooperative sweeping D

CHAPTER SEVEN: CONCLUSION

7.1 Contribution

In this Thesis an algorithm for complete coverage of a given area in the presence of stationary and dynamic obstacles has been presented. A smooth trajectory is then constructed from which the control inputs are generated analytically. Also a qualitative analysis of the smooth trajectory is done. Finally for cooperative sweeping of multi robots, the given area is divided into sub areas depending on the number of robots and each robot is assigned a particular area for sweeping.

7.2 Future work

In this thesis only convex boundary is considered hence future work lies in including all types of boundaries for complete coverage. This algorithm works on the fact that the map of the area is known and hence work can be extended to areas for which map is not available. The controller design does not include feedback, future work lies in including feedback of the controlled variables in the controller design. Future direction will also include implementation on a real mobile robot system. The obstacle avoidance or dynamic obstacle avoidance depends on the fact that the sensor data gives a accurate information on the position of the dynamic obstacles. Hence future work lies in minimizing dynamic obstacle avoidance based on the accuracy of the sensor data.

APPENDIX: MATLAB CODE

```
% this is the main function for complete coverage

dummysum = 1;

[xa ,  ya] =  neural_pointsinboundary(dummysum )

[xb ,  yb] =  neural_pointsinboundaryb(dummysum )

[xc ,  yc] =  neural_pointsinboundaryc(dummysum )

[xd ,  yd] =  neural_pointsinboundaryd(dummysum )

hold on;

obs_plot(dummysum )

trajectoryabcd(xa,ya,xb,yb,xc,yc,xd,yd)


% this is the neural selection  function

clc;

%axis([ 0 20 0 20 ] );

axis([ 0 36 0 45 ] );

% Statement to draw a rectangle

rectangle('position',[0,0,45,45]);

hold on;

% xw  x axis value of rectangle is 10

% yw y axis value of retangle is 10

xw = 45;

yw = 45;

% value of radius of the circle /disk
```

```
rc = 1;

%calculate number of disk needed in each column and row

column = yw / (sqrt(3) * rc);

i = 0;

while i < (column - 1);

i = i + 1;

end

if ( column - i) <= .5

actualcolumn = i + 1;

else

actualcolumn = i + 2;

end

%debugging ok till this point

row = xw/( 1.5 * rc);

i1 = 0;

while  i1 < (row-1);

i1 = i1 + 1;

end

if (row - i1)<= (2/3)

actualrow = i1 + 1;

else

actualrow = i1 + 2;

end
```

```
% debugging ok till this point

% generating circles most important part

% variable p is indexing variable for xc, yc.

p = 0;

for h = 1 : actualcolumn

for g = 1 : actualrow

if rem(h,2) == 0

xc = .5 + (h - 1) * (3/2) * rc;

yc =  (sqrt(3)/2)*rc + (g -1) * (sqrt(3) * rc);

p = p + 1;

xmatrix(p) = xc;

ymatrix(p) = yc;

theta = linspace(0,2*pi,40);

x = xc - (cos(theta));

y = yc - (sin(theta));

plot(x,y)

else

xc = .5 + (h - 1) * (3/2) * rc ;

yc =  (g - 1) * (sqrt(3)*rc);

p = p + 1;

xmatrix(p) = xc;

ymatrix(p) = yc;

theta = linspace(0,2*pi,40);
```

```
x = xc - (cos(theta));

y = yc - (sin(theta));

plot(x,y)

end

end

end

% debugging ok till this point

% till this point debug sucessfull

% boundary points and storeing them.

hold on;

boundaryx = [ 0   30  36.2  36.2    25   20   5    2    0   0 ];

boundaryy = [ 0   1   5       35    45   45   28   20   15   0 ];

%plot(boundaryx,boundaryy)

hold on;

boundaryx = [0 , 18,18,16,8,6,1.7,0,0 ];

boundaryy = [0 ,.5,16,19,19,19,19,15,0 ];

%plot(boundaryx,boundaryy)

 % to find points which are inside the polygonal area.

IN = inpolygon(xmatrix,ymatrix,boundaryx,boundaryy)

[IN ON] = inpolygon(xmatrix,ymatrix,boundaryx,boundaryy)

dummy = actualrow * actualcolumn

noofpoints = 1;

for ix = 1:dummy
```

```
if IN(ix) == 0

dumv = 0;

else IN(ix) == 1

importantx(noofpoints) = xmatrix(ix)

importanty(noofpoints) = ymatrix(ix)

importantpf(noofpoints) = 0;

noofpoints = noofpoints + 1;

end

end


% assign initila neural volt.

% total number of points is 37

points_covered  = 1;

[neuralvolt , clean, unclean ]  = initialneural(22)

 startx = .5000;

 starty = 1.7321;

 clean(1) = 1;

 unclean(1) = 0;

 stepx = 0;

 stepy = 0;

 var = 0

 sx = 0;

 sy = 0;
 p = 2;
 while(points_covered < 109)
%while(points_covered < 46 )
 [clean,unclean ] = moveing_obstacles(clean , unclean,points_covered)
 %while(points_covered < 6 )
  var = var + 1;
[ neighborx , neighbory ,neighbor_neuralvolt, no_neighbor,neighbor_clean,
neighbor_unclean,startx,starty ] = neighborxy( startx , starty , importantx , importanty ,clean
,unclean, neuralvolt,stepx,stepy,points_covered )
  [ neighborx, neighbory,neighbor_neuralvolt] = neuralvoltage( neighborx , neighbory ,
neighbor_neuralvolt , no_neighbor, neighbor_clean, neighbor_unclean,startx, starty,sx,sy,p )
[ next_x ,  next_y,sequence_pointx,sequence_pointy ] =
nextnode(neighborx,neighbory,neighbor_neuralvolt,no_neighbor,var);
sequen_pointx(var) = sequence_pointx;
sequen_pointy(var) = sequence_pointy;
X = [ startx , next_x ];
Y = [ starty , next_y ];
%plot(X, Y)
p = points_covered + 1;
stepx(p) = next_x;
stepy(p) = next_y;
sx = stepx;
sy = stepy;
```

```
points_covered = points_covered + 1;

[startx,starty,next_x,next_y,clean,unclean] =  update(

importantx,importanty,neighbor_neuralvolt,startx,starty,next_x,next_y,clean, unclean);

[neuralvolt]  = updateneuralvolt(importantx,importanty,neighborx, neighbory,

neighbor_neuralvolt, no_neighbor,neuralvolt);

end

stepx(1) =0.5000;

stepy(1) = 1.7321;

xa = stepx;

ya = stepy;

%trajectory(x,y)

% This main function for optimal anlaysis

e0 = [ 1 ; 1 ;1;1;1];

[e,fval] = fseminf(@myfun,e0,2,@mycon);

eta = [ e(1) e(2) e(3) e(4) ]

function  [ X_plot , Y_plot ,X2, Y2 , ka, kb, theta_a , theta_b,e1,e2, e3, e4] = xy_coor(dummy)


hold on;

hold on;

a =    [0.0000   12.0417  -11.8958    4.6458   -0.8542    0.0625];

b =   [12.4130  -34.2321   33.6427  -15.0231    3.1634   -0.2526];

% the following comments are for understanding statements.

% a and b are the coefficients for the equation
% x = a(1) + a(2)*s + a(3)*s^2 + a(4)*s^3 + a(5)*s^4 + a(6)*s^5

% y = b(1) + b(2)*s + b(3)*s^2 + b(4)*s^3 + b(5)*s^4 + b(6)*s^5

% the above equation is obtained from smooth curves.

% the above coefficients are for first curve generated.

% so for each curve we have different set of a,b values.

% value of s3 = 2 and value of s4 = 3;

syms s;

a_3db = (( a(2) + 2*a(3)*s + 3*a(4)*s*s + 4*a(5)*s^3 +5*a(6)*s^4 )^3*(2*b(3) + 6*b(4)*s +

12*b(5)*s*s + 20*b(6)*s^3) );

a_2bda = (( a(2) + 2*a(3)*s + 3*a(4)*s*s + 4*a(5)*s^3 +5*a(6)*s^4 )^2*( b(2) + 2*b(3)*s +

3*b(4)*s*s + 4*b(5)*s^3 +5*b(6)*s^4 )*(2*a(3) + 6*a(4)*s + 12*a(5)*s*s + 20*a(6)*s^3));

a_5 = ( a(2) + 2*a(3)*s + 3*a(4)*s*s + 4*a(5)*s^3 +5*a(6)*s^4 )^5;

b_2 = ( b(2) + 2*b(3)*s + 3*b(4)*s*s + 4*b(5)*s^3 +5*b(6)*s^4 )^2;

a_3 = ( a(2) + 2*a(3)*s + 3*a(4)*s*s + 4*a(5)*s^3 +5*a(6)*s^4 )^3;

theta = atan(( b(2) + 2*b(3)*s + 3*b(4)*s*s + 4*b(5)*s^3 +5*b(6)*s^4 )/( a(2) + 2*a(3)*s +

3*a(4)*s*s + 4*a(5)*s^3 +5*a(6)*s^4 ));

cos_teta = cos(atan( b(2) + 2*b(3)*s + 3*b(4)*s*s + 4*b(5)*s^3 +5*b(6)*s^4 )/( a(2) + 2*a(3)*s

+ 3*a(4)*s*s + 4*a(5)*s^3 +5*a(6)*s^4 ));

sin_teta = sin(atan( b(2) + 2*b(3)*s + 3*b(4)*s*s + 4*b(5)*s^3 +5*b(6)*s^4 )/( a(2) + 2*a(3)*s

+ 3*a(4)*s*s + 4*a(5)*s^3 +5*a(6)*s^4 ));

num = cos_teta*(a_3db - a_2bda)

den = a_5 + a_3*b_2

input_u2 = diff(atan(num/den) , s)
```

```
in_u1 =(a(2)+2*a(3)*s+3*a(4)*s^2+4*a(5)*s^3+5*a(6)*s^4)/cos(atan( b(2) + 2*b(3)*s +
3*b(4)*s*s + 4*b(5)*s^3 +5*b(6)*s^4 )/( a(2) + 2*a(3)*s + 3*a(4)*s*s + 4*a(5)*s^3
+5*a(6)*s^4 ))
inu1 =(b(2)+2*b(3)*s+3*b(4)*s^2+4*b(5)*s^3+5*b(6)*s^4)/sin(atan( b(2) + 2*b(3)*s +
3*b(4)*s*s + 4*b(5)*s^3 +5*b(6)*s^4 )/( a(2) + 2*a(3)*s + 3*a(4)*s*s + 4*a(5)*s^3
+5*a(6)*s^4 ))
inx =  cos_teta*in_u1
iny =  sin_teta*inu1
p = 2 : .001 : 3;
% input u2 is the steering rate
in_u2 = subs(input_u2, s, p)
plot(p , in_u2,'red')
hold on;
syms s;
% this is used for generation of trjecory
function [ x_tra ,  y_tra ] =generate(in_u1,in_u2,theta,phi)


% Finding the optimal curve useing the quintic splines curve method.
% the initial conditions are found from smooth curves method.
syms a1;
syms a2;
syms a3;
syms a4;
syms a5;

syms a6;

syms b1;

syms b2;

syms b3;

syms b4;

syms b5;

syms b6;

syms s;

x = a1 + a2*s + a3*s^2 + a4*s^3 + a5*s^4 + a6*s^5;

y = b1 + b2*s + b3*s^2 + b4*s^3 + b5*s^4 + b6*s^5;

dx = diff(x , s);

dy = diff(y,s);

ddx = diff(dx , s);

ddy = diff(dy , s);

ka  = (dx*ddy - ddx*dy)/( (dx)^2 + (dy)^2 )^(3/2)

%a = [ -14.5000   55.3750  -56.9792   26.1875   -5.5208    0.4375 ];

%b = [   3.4641    7.1447   -6.7237    2.6342   -0.4932    0.0361 ];

a1 = -14.5000  ; a2 = 55.3750   ; a3 = -56.9792    ; a4 =    26.1875   ; a5 =   -5.5208  ; a6 =

0.4375  ;

b1 = 3.4641  ; b2 =  7.1447  ; b3 = -6.7237  ; b4 =      2.6342       ; b5 =    -0.4932   ; b6 =  0.0361

;

s =  2;
```

```
s =  3;

theta = atan(( b2 + 2*b3*s + 3*b4*s*s + 4*b5*s^3 +5*b6*s^4 )/( a2 + 2*a3*s + 3*a4*s*s +

4*a5*s^3 +5*a6*s^4 ))

ka

% find value of ka and theta.
```

LIST OF REFERENCES

[1] John L. Junkins and James R. Jancaitis, “Smooth Irregular Curves,” Mathematical Terrain Analysis” pp. 565 - 573, 1971.

[2] Y.Guo, Z.Qu, “Coverage Control for a Mobile Robot Patrolling a Dynamic and Uncertain Environment”.

[3] A. Piazzi, C. Bianco, M. Bertozi, A. Fascoli, A. Broggi, “Quintic G^2 - Splines for the Iterative Steering of Vision-based Autonomous Vehicles”. IEEE transactions on Intelligent Transportation Systems, Vol. 3, No.2, March 2002.

[4] S. La Valle, J. Kuffner, Jr., “Randomized Kinodynamic Planning”. The International Journal of Robotics Research, Vol. 20, May 2001, pp. 378 – 400.

[5] Paolo Fiorini and Zvi shiller “Time Optimal Trajectory Planning in Dynamic Environments”. IEEE transactions on Intelligent Transporation Systems, Vol. 3, No.3, March 2000.

[6] J-P. Laumond, S. Sekhavat and M. Vaisset “Collision Free Motion Planning For a Non-Holonomic Mobile Robot with Trailers” International Symposium on Intelligent Robotics. India, pp. 1507 – 1512.

[7] H.Delingette, M.Hebert and K.Ikeuchi “Trajectory Generation with Curvature based on Energy Minimization”, IROS 91, Osaka, Japan.

[8] A.Zelinsky, R.A. Jarvis, J.C.Byrne “Planning Paths of Complete Coverage of an Unstructured environment by a Mobile Robot”, Proceedings of the 3rd International Conference on Intelligent Autonomous systems, Feb, 1993, Pittsburgh, USA.

[9] J.Barraquand and J.C. Latombe, “Robot Motion Planning: a distributed Representation approach”. International Journal of Robotics Research.

[10] Simon X. Yang and Chaomin Luo "A Neural Network Approach to Complete Coverage Path Planning". IEEE Transactions on systems. Man and cybernetics – Part B: Cybernetics. Vol.34, No.1.February 2004.

[11] R. Murray and S.Sastry, "Nonholonomic Motion Planning: Steering Using Sinusoids", IEEE Transactions on Automatic Control, Vol.38, pp 700-716, 1993.

[12] P.Rouchon, M.Fliess, J.Levin, and P.Martin, "Flatness and Motion Planning: The car with n Trailers," Proceedings of the European Control Conference, pp 1518 – 1522, Netherlands 1993.

[13] A.Piazzi and C. Bianco "Optimal Trajectory Planning with Quintic -splines" Proceedings of the IEEE Intelligent Vehicles Symposium 2000, pp 620 -625.

[14] Anthony Stentz "Optimal and Efficient Path Planning for Partially-Known Environments"

[15] Allan Willms, Simon X. Yang "An efficient Dynamic System for Real-Time Robot Path Planning" Robot Autonomous Systems., vol. 13, n0.2 ,pp 143 – 148, 2000.

[16] Shingo Shimoda, Yoji Kuroda and Karl Iagnemma "Potential Field Navigation of High Speed Unmanned Ground Vehicles on Uneven Terrain" Proceedings of International Symposium on Experimental Robotics, June 25 -27, 1991, France.

[17] Birgit Graf and Christopher "Flexible path Planning of Non Holonomic Mobile robots" Proceedings of the Australian Conference on Artificial Intelligence, November 1986.

[18] F. Antritter, B.Miller and J. Deutscher "Tracking Control for Nonlinear Flat Systems by Linear dynamic Output Feedback"

[19] S.S. Ge and Y.J.Cui "Dynamic Motion Planning for Mobile Robots using Potential Filed Method." IEEE transactions on Robotics and Automation, Vol 14 No.3 June 1998.

[20] Kikuo Fujimura and Hanan Samet "A Hierarchical Strategy for Path Planning Among

Obstacles" IEEE international conference on Robotics Automation, Belgium, May 1998, pp3588 – 3593.

21] J. Borenstein and Y.Koren "Real-time Obstacle Avoidance for Fast Mobile Robots" IEEE transactions on Systems, Man, and Cybernetics, Vol. 19, oct/1989. pp 1179 – 1187.

[22] Timothy D Barfoot, Christopher M Clark "Kinematic Path – Planning for Formations of Mobile Robots with a Nonholonomic Constraint" IROS 2002, Lausanne, Switzerland. September-30, 2002.

[23] S. Monaco and D. Normand-Cyrot, "An introduction to motion planning under Multirate Digital control" IEEE transactions on automatic Control, Sep 2000, pp 1586 – 1589.

[24] S.X.Yang and Meng, "An Efficient Neural Network based approach to Robot Dynamic Motion Planning". IEEE transactions on Neural Networks, Vol.14, No.6, November 2003.

[25] Greg Frederickson, "Dissections: Plane and Fancy", Published by Cambridge University Press, ISBN 0-521-52582-9.