PATH PLANNING OF MICRO AIR VEHICLE IN STATIC ENVIRONMENT

M ABUBAKAR MUNIR

2009-NUST-MS-PhD-ME-16

ADVISOR

PROFESSOR DR AFZAAL MALIK

DEPARTMENT OF MECHANICAL ENGINEERING

COLLEGE OF ELECTRICAL AND MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY

2013


بسم الله الرحمن الرحيم

To my Parents
For their Endless Love and Support

To my brothers and my sister
For their never ending encouragement

To my friends
For always being very supportive for me

ACKNOWLEDGEMENTS

I would like to thank the Almighty Allah who gave me strength, courage and resources to do this research work. He has always been very kind to me.

My thesis supervisor Dr. Afzaal Malik deserves special appreciation for his support, help and guidance throughout this work. He kept the perfect balance between making me do things myself so that I can learn the most and guiding me with his experience when i needed.

I would like to thank department of Mechanical Engineering, especially the head of the Department Col Dr. Syed Waheed ul Haq for all the support and facilitation through the course of this work.

I would also like to thank Sir Raja Amer Azim for his guidance and help in choosing the proposed algorithm for my work.

I would also like to thank all the precious technical advices, reviews, suggestions and encouragement of Sir Yasir Jan, Sir Saif Ullah Khalid, Usman Rafique, Sheraz Bhai, Sajjad Sabir, Touqeer Anwar and Awais Hamza.

Special thanks to those who motivated and encouraged me to pursue higher studies.

ABSTRACT

Path planning or motion planning is a fundamental aspect of autonomous vehicle navigation. An important problem in the design of miniature air vehicles (MAVs) is obstacle avoidance. In this research, path planning of micro air vehicle (MAV) in static environment is carried out, keeping in view that MAV has the global knowledge of its environment i.e. terrain map is pre-loaded in it. Path Planning involves the computation of a collision free path from start to goal point without considering robot dynamics and kinematic constraints like max velocity, max turn angle etc. Static environment means that target and obstacles which encountered during flight will be fixed.

We design an optimal and power efficient path planner for path planning of micro air vehicle in static environment having multiple obstacles (circles in 2D & spheres in 3D). Optimal path means a path with minimum path length. Power efficient path means it has minimum number of turn points on its way towards the goal, as with each waypoint actuator is switched on & off which consumes the vehicle battery power. We further refined our path in order to get more optimal results keeping in view that we do not have to compromise on its power efficiency.

Contents Page No

List of Figures

List of Tables

INTRODUCTION

Historical Overview – Unmanned Air Vehicles (UAV)

It has been more than a century, when the piloted aircraft took its first flight in 1903. Since then the progression in the aircraft flight has been amazing. Meanwhile, this amazing development of piloted flight further leads into unmanned aircrafts which provide reconnaissance and discover those places which were not in the access of the humans. These unmanned aircrafts today are named as unmanned aerial vehicles or unmanned air vehicles. It keeps the humans away from those positions which proves risky to their lives.

The concept of unmanned air vehicles generated even before the piloted flights. In 1863, Charles Perley in America intended for taking bombs in a hot air balloon. But actually it wasn't used in a real combat. In 1883, Doughlas Archibald actually made a kite which could take photographs. Later this kite was used by the US army against Spain during a war to take snaps of the critical enemy locations. Nevertheless the US Navy Curtiss trainer aircraft was the first unmanned aircraft and was controlled through the radio signals. It could carry 400 pound bomb and to a range of 90 kilometers. But it wasn't used.

The first UAV which was used in a real combat was German's V-1, which had a warhead of 2000 pounds and had a range of 250 kilometers. It was used against the British during the World War-II. Since then, many countries successfully made UAV's particularly for surveillance purposes. At first, the UAV's were remotely controlled but today they are autonomous as well. Interestingly, they are not now restricted to military uses only but they are also being used for civilian applications like surveillance, inspection, rescue and communications etc. Now, there is a strong desire for Micro air Vehicles (MAV), which is a type of UAV, to provide some of the useful applications in an effective manner.

Fig 1.1 Unmanned Air Vehicle [a]

Micro Air Vehicle (MAV)

A micro air vehicle or micro aerial vehicle (MAV) is a class of unmanned aerial vehicles (UAV) that has a limitation in its size. Modern MAV can be as small as 15 centimeters.

Defense Advanced Research Projects Agency (DARPA) of USA, started a program in 1997 to encourage the design and development of MAVs for their possible use in military and civilian applications [33]. As the name suggests, DARPA limited the weight, dimensions even the payload and mission capabilities. The size cannot exceed 15cm in any dimension, weight cannot be more than 100g and 20% should be designated for the payload. Research is also going on Nano Air Vehicle (NAV) which has the size limitation of 7.5cm, weight limitation of 10g.

In International MAV flight competitions, size and weight limitations for MAV is still not in that range as prescribed by DARPA. In Germany, an international competition on MAV was held in July 2010. The requirements of the competition were:

INDOOR EVENT:

Table 0-1 MAV specification for indoor event

Specification	Max. Size	Max. Weight
Rotary Wing MAV	70cm	1 kg
Flapping Wing MAV	70cm	1 kg
Fixed Wing MAV	80cm	1 kg

OUTDOOR EVENT:

Table 0-2 MAV specification for outdoor event

Specification	Max. Size	Max. Weight
Rotary Wing MAV	100cm	5 kg
Flapping Wing MAV	100cm	5 kg
Fixed Wing MAV	100cm	5 kg

1.2.1 Classification of MAV

Based on the propulsion and configuration methods, the MAV can be classified into three main types, which are as follows:

a) Fixed Wing MAV: Here the thrust is provided by the power plant through propellers etc, whereas the lift is generated by the wings. Neither vertical takeoff nor landing is possible.

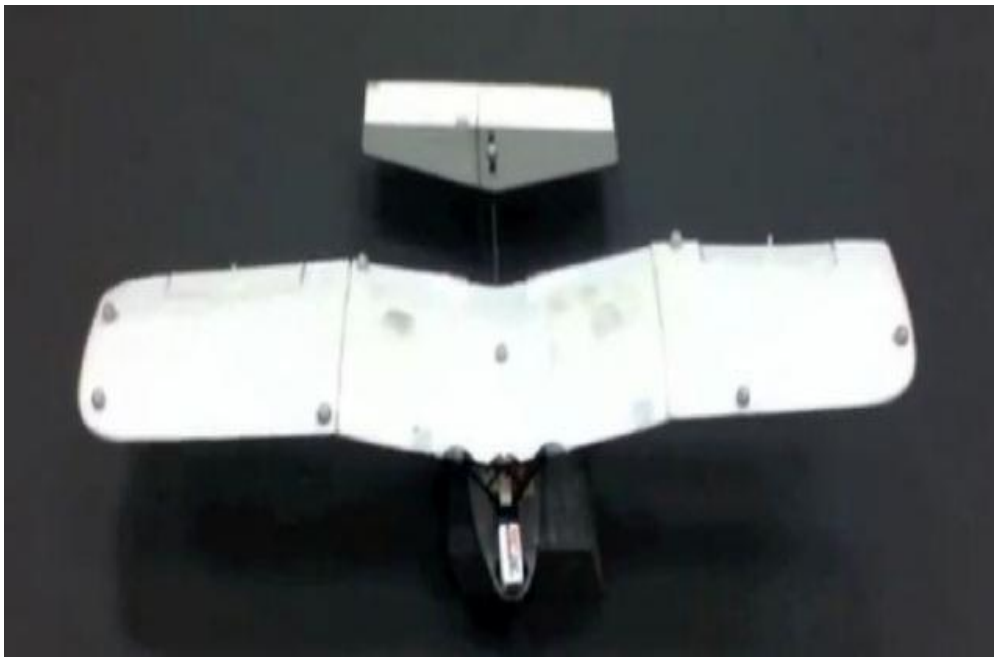

Fig 1.2 Fixed Wing MAV [b]

b) Rotary Wing MAV: Both the lift and thrust are produced by the rotors. Both Vertical takeoff and landing can be done.

Fig 1.3 Rotary Wing MAV [c]

c) Flapping Wing MAV: Both lift and thrust are generated by wings.

Fig 1.4 Flapping Wing MAV [d]

1.2.2 Technical Challenges of MAV

Current challenges for the development of MAV are as follows [33]:

Physical Integration: For MAV, there is no greater challenge than the physical integration of MAV. Putting all the equipments on a single, small MAV body is a tough challenge to ask. Micro Electro Mechanical Systems (MEMS) are developing at a fast pace and are used in the physical integration of MAV. Even then, the individual modules for a specific function would take a lot of volume than actually available.

Flight Control: In flight control, the designer deals with the most number of unknowns. Stabilization of platform and its guidance requires a high speed and self controlling systems.

Power and propulsive systems: It has Poor length to diameter ratios and due to the low Reynolds number, it has low propulsive efficiency as well. Battery power or exotic technologies like fuel cells may be required.

Navigation: Mission constraints do not allow real time interaction of man and machine for guidance and navigation of MAV. In addition, vehicle agility or response to gusts may diminish human operators ability. So advances in small scale navigation systems along with guidance/control systems are essential to overcome this.

Communication: Due to small vehicle size, small antenna size, limits power availability and support the bandwidth requirement (2-4 megabits per second) for image transmission. Image compression may reduce the bandwidth but also increases online processing and leads to higher power requirements.

1.2.3 Applications Of MAV

MAV has numerous applications, mainly due to their ability to operate at low speed with high agility. Few of the applications are given as under:

1) Military

a) Convert imaging in confined areas by infiltration.
b) Biological, chemical and radioactive agent detection.
c) Detection of remote precision mines.
d) Enhanced communication in urban area fighting.
e) Reconnaissance over the hill, woods or river.

2) Civilian

a) Track vehicle with suspected terrorist.
b) Monitoring the abduction situation.
c) Inspection of high rising buildings.
d) Monitoring of traffic.
e) Rescue and fire missions.
f) Power/ communication lines inspection.

INTRODUCTION TO NAVIGATION

Motivation

From last two decades, significant work has been carried out on the navigations and guidance of micro air vehicles (MAV) and unmanned air vehicle (UAV). The research in path planning is divided into two main directions. These are optimal path planning means 'shortest possible path to the goal' and it does not take computational cost into consideration and the other is real time sub optimal path planning in which the computational cost is the major area of focus. One has to decide which direction is to follow, keeping in view the problem statement. In past, people for planning in the static environment, go for the optimal path, irrespective of the number of turn points that the vehicle has to take during its flight. That work well for the most UAV's, but in the case of a MAV, the major area of concern is to decrease the no. of waypoints (turn points) in reaching the goal, as with each waypoint actuator is switched on & off which consume the vehicle battery power and MAV cannot afford to have a larger battery keeping in view its small weight and size. So for the purpose of MAV path planning, researchers try to short down the number of waypoints in its path and in doing so, they might lose optimality.

In this thesis, for the purpose of MAV navigation, we will be focusing on achieving both optimality and at the same time having the minimum number of waypoints in the proposed algorithm so that we don't have to trade off between anyone of these. So for this purpose, we developed a hybrid algorithm that combines Depth First search approach (DFS) and velocity obstacle method (VO) and finds an optimal and at the same time power efficient path for the MAV in an obstacle rich environment. Many hybrid algorithms have been used for the ground robots and UAV's in past and have shown good results.

Navigation

Navigation is a problem of finding a collision free path for a robot system from one configuration to another. The focus is to successfully achieve a goal through a high level language and which is then converted by the robot into low level motion primitives to complete the task.

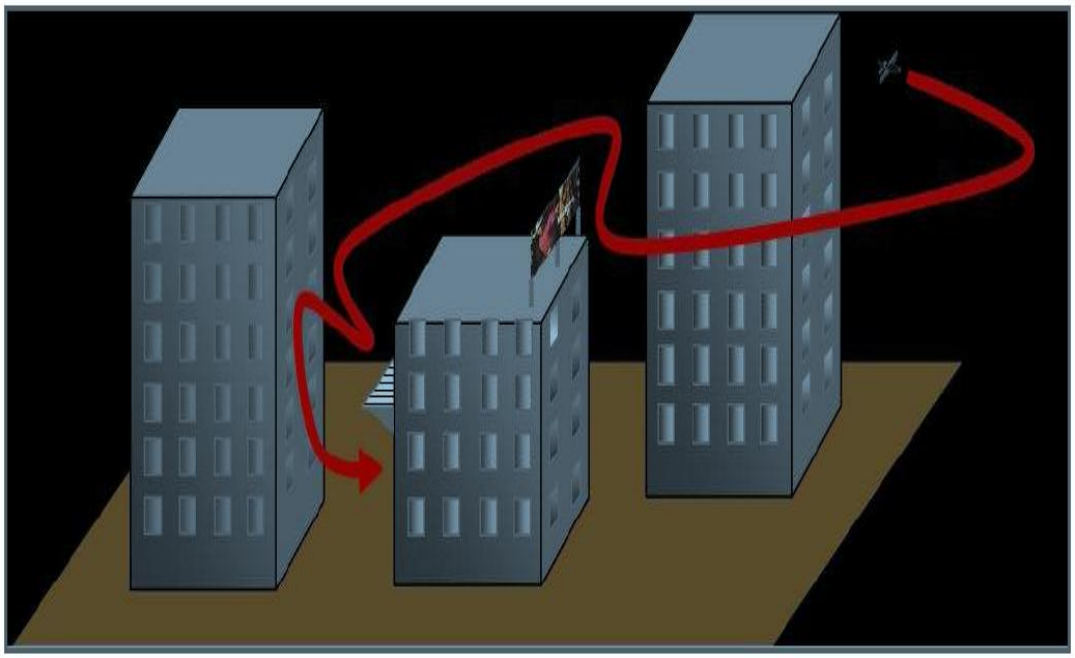

Fig 2.1 Micro air vehicle navigating in an urban environment [2].

1.2.1 Types of Navigation

Navigation is further divided into two categories: Path Planning and Trajectory Planning [18].

a) Path Planning: It does not consider robot dynamics and kinematic constraints like max velocity, max acceleration, max turn angle etc.
b) Trajectory Planning: It does consider system's dynamics and kinematic constraints.

1.2.2 Types of Obstacles

There are two types of obstacles that will be encountered during the flight.

a) Priori Known Obstacle: These are the fixed obstacles which the robot deals with during the flight. Offline path planning can be done if there are fixed obstacles in the environment. Offline path planning means trajectory can be developed before the flight and the robot does not need to readjust its preplanned path during the flight. The

environment which has only fixed obstacles is called static environment. Off course, in this case the computational expense is low.

b) Pop up Obstacles: The moving obstacles in the environment is termed as the pop up obstacles. The environment is then called the dynamic environment. Online path planning needs to be done for this kind of scenario as the moving obstacles continuously change their position. Calculations and path needs to be adjusted online by the robot. . It then leads to an offset path. Offset path is that MAV path which it follows after changing its original path due to the moving obstacles. The distance between the original path and the offset path is termed as the offset of the original path. Here the computational cost is very high as compared to the offline path planning problem.

1.2.3 Difference between Navigation and Guidance

a. Navigation: It is the process of finding a collision free path when the target is fixed.

b. Guidance: It is the process of finding a collision free path when the target is moving e.g. guided missiles.

1.2.4 Fundamentals of Navigation

Whatever technique may be used for the navigations of MAV, there are some key fundamentals which must be kept in mind. These are:

a) Maximum turn angle limitation of MAV should not exceed.
b) It must choose an optimal path.
c) Path plan from starting point to end point should be given.

1.2.5 Principles of Navigation

Regardless of the technique used for the purpose of planning a path, there are four basic principles which must be followed. These are:

a) It guarantee obstacle avoidance.
b) It stays within the complex constraints of flight dynamics.

c) It computes new trajectory immediately after detection of a pop up obstacle in current trajectory.
d) It arrives at the goal in minimum time or shortest distance.

1.2.6 Non-Holonomic Motion

If the robot is having some constraints like velocity constraints in its motion e.g. a car does not move sideways, then both the constraint and vehicle are called non-holonomic. MAV also has a non-holonomic motion.

1.2.7 Omni Directional Motion

If the vehicle can move in any direction, then it will be termed as Omni Directional Motion.

1.2.8 Configuration Space

Configuration space and configuration of a robot are one of the most critical concepts of motion planning. A configuration describes the state of the robot, and the space which describes set of all possible configurations is called configuration space C [18]. For example:

If the robot is considered as a single point which is translating in a 2-dimensional plane, then configuration workspace C is a plane, and it requires two parameters (x, y) for its representation.

If the robot is considered as a single point which is translating in a 3-dimensional space, then configuration workspace C represents a space, and it requires three parameters (x, y, z) for its representation.

If the robot is taken as a 2-dimensional shape, having the ability of both translation and rotation, then the configuration space is still 2-dimensional and it requires three parameters (x, y, θ) for its representation.

If the robot is taken as a 3-dimensional shape, having the ability of both translation and rotation, then the configuration space is 3-dimensional and it requires six parameters (x, y, θ, α, β, γ) for its representation.

2.2.8.1 Configuration Space Obstacle

It is a set of configuration points for which the robot intersects any obstacle in its workspace. It is represented by C_{obs}.

2.2.8.2 Free Configuration Space

It is a set of configuration points for which the robot does not intersect any obstacle in its workspace. Also if we take the complement of C_{obs} in C, then we will be having free configuration space and vice versa. It is represented by C_{free}.

2.2.9 Complete Path Planner

If a path planner either finds a solution for the given problem or gives the output that there is no path that exists for the given problem, then it is said to be a complete path planner.

The research in path planning is divided into two main directions. These are optimal path planning which do not take computational cost into consideration and the other is real time sub optimal path planning. An optimal path means 'shortest possible path'. The difference in a robot and a MAV path planning is that a MAV has to maintain a certain velocity to minimize its time of span, which suggests that there might be some difficulties in following a path with sharp turn or vertices. This turn is basically the smallest size circular turn that the vehicle is capable of making. Many common path planning algorithms are inefficient when applied to micro air vehicle due to their turn radius and airspeed constraints.

An effective path planner is the one that has the power to deal directly with the combination of the expected motion of the obstacles, the uncertainties in its location and how these uncertainty changes with time. So as to achieve the desired path for the MAV, several steps must be taken into account. Without these directions, MAV path planning will be in some unknown directions. These important steps are:

a) MAV kinematics
b) terrain information
c) threat information

These are the factors which actually form the flight constraints that must be handled in motion planning procedure.

2.3 Literature Review:

Previous work in micro air vehicle motion planning includes Rapidly-exploring Random Tree (RRT) algorithm used by S. M. LaValle [7] and Peng Cheng [10] which randomly generates waypoints to span a configuration space. A sequence of waypoints is connected to the goal by avoiding obstacles that come on its way. Control inputs are applied to the states existing in the tree. However, it ends up in an open-loop solution. The modified RRT algorithm developed by Stephen Griffiths [6] differs from the basic RRT planner in a way that here the output states are being searched rather the inputs and generates as many waypoints on its path. For each waypoint path, it follows the desired MAV trajectory and make it sure that are only feasible paths in the search tree. Each branch of the tree is checked to ensure that the MAV has the collision free path throughout the configuration space. Another improved RRT algorithm that has been implemented by Jeffery B. Saunders [4] differs from the basic algorithm. The basic RRT algorithm which is an open loop path planner produces control inputs that are time-parameterized, in moving from initial position to the goal position. So in this paper, extended RRTs were used in closed-loop path planning where MAV detected and avoided the obstacles successfully. Girish Chowdhary and John Ottander [1] discusses methods, algorithm and results for navigation and control of micro air vehicle in GPS denied areas by using guidance, Navigation & Control (GNC) laws which takes information from available range sensors. The beauty of this method lies in the fact that for complex problem of indoor navigations, simple GNC laws proves to be efficient and robust. However, it has some limitations particularly with the design of those PID controllers that are dependent on range measurements only and these measurements are not as effective as the human pilots. Yohannes Ketema and Yiyuan J. Zhao [3] investigated that how wind patterns may affect the design of MAV trajectories. The effect of wind on MAV trajectory is done from the point of view of controllability and reachability sets for a given point. Controllability set is the set of points which helps in reaching at a given point within a given amount of time. Reachability set is the set of points in space where MAV can reach within a given amount of time and from a given location. The proposed problem formulation results in trajectories that arrive at the target state if possible, and to a nearby state, if

not. But they increase in the computation expense by catering too much waypoints on its way to the goal. For MAV navigation, Sayan Ghosh and Abhishek Halder [8] proposes a fuzzy quadtree based path planner. Results show that these planners are more effective than the quadtree planner. This algorithm can significantly reduce both space and time complexity and still yield an optimal and safe path for the vehicle. Keeping in view the MAV's kinematics and dimensions, the formulation done here shows how to take the minimum quad length. Here pixel level computation does not considered and also as compared to the conventional planner, execution time is less. Thus this proposed method results in a faster, cheaper and better solution for MAV path planning problem. Gargantini [12] proposes a computationally efficient method by using quadtree framework method. Here initially an image of the workspace is taken. Then that image is broken into quadtrees. Nodes or quads having obstacles are colored black and rest of the free passing nodes are colored white. Tree generated by connecting free nodes. It leads to a complete path but lacks optimality. Sayan Ghosh and Abhishek Halder [8] proposes a fuzzy quadtree based path planner. Keeping in view the MAV's kinematics and dimensions, the formulation done here shows how to take the minimum quad length. Results show that these planners are more effective than the original quadtree planner. This algorithm significantly reduced both space and time complexity and still yielded an optimal and safe path as compared to the original quadtree planner. Here pixel level computation is not considered and also as compared to the conventional planner, execution time is less. Jeffery Saunders and Randal Beard [9] proposes a method for reactive obstacle avoidance for multiple obstacles with a guidance strategy based on observations, made in the image plane. Trajectory is generated in the form of tiers. The first tier reacts instantaneously to detect obstacles using a reactive guidance strategy based on a feedback control law. The second tier plans a waypoint path around obstacles locally in the body frame of the MAV. The third tier creates a global path from the current state till the end. Integration of these tiers into the fully function obstacle avoidance system will accomplish all goals of a trajectory generator but leads to a very high computation time. Rong Zhu and Zhaoying [11] proposed a path planning method by generating Delaunay weighted tree that expects to be an optimal and safe path for a MAV navigating in an urban terrain. In this method, polygonal obstacles are considered which shows good results for the urban terrain which results in less computation time while achieving closer to an optimized path if not the optimal. Ryan Donovan Hurley [2] proposes anew solution for MAV navigations in which vehicle's path is

generated using two parts. In the first part, they combine 3-D motion primitives by using rapidly exploring random trees (RRT). This part travels along the obstacles. The second portion of the part uses 3-D Dubins Path to travel from the end of a tree branch to the goal or end point. The beauty of this work is that with no obstacle, the first portion i.e. RRTs can be excluded from the planning algorithm, thus making it computationally very efficient path planner. However Dubins approach showed one drawback that it could not be used to compute motion in which vehicle could move in the reverse direction. Paolo Fiorini and Zvi Shiller [13] proposes a new method in order to decrease the computational inefficiency of the algorithm by using velocity obstacle method. In this method, spherical obstacles are considered. A collision cone is drawn from the vehicle to the spherical obstacle and a velocity is set from the initial point in the direction of the goal. If that velocity lies inside the collision cone, then the new direction of the velocity is set in the direction of the closest tangent. The geometric nature of velocity obstacle method makes it suitable for planning of multiple mobile robots and such applications. Zhuoning Dong, Zongji Chen [17] proposes a hybrid algorithm that combines virtual force and A* search algorithm. A* was used to avoid the local minima problem that comes in the virtual force method. The solution provides an sub-optimal path but with good computational efficiency.

2.4 Research Objective

The objective of this research is to develop an optimal, fast and at the same time an inexpensive algorithm for the purpose of micro air vehicle navigation in static environment. Inexpensive algorithm means having minimum number of waypoints along the path. The method will be robust and will work for different combinations of obstacles position, initial and goal positions of the robot.

2.5 Contribution

Micro air vehicle is a very emerging area in research nowadays because of its applications in civil, military and rescue missions. Due to the state of the art technology and resources, remarkable work and achievements have been achieved but a lot of research has to be done yet. Research is being carried out from the last 12 years about the navigation of MAV. MAV path planning needs to provide an optimal and power efficient path.

In this report, an efficient path planner for the purpose of MAV path planning in static environment has been proposed. Results have shown that it proves to be an optimal and power efficient path planner. The robustness of this algorithm also ensures it performs well in different scenarios.

3. AN OVERVIEW OF NAVIGATION ALGORITHMS

3.1 Obstacle Avoidance

A key problem in the process of path planning is the obstacle avoidance. There are many algorithms which have their own worth when it comes to avoiding an obstacle. All of these algorithms try to avoid the obstacle with minimum path deviations and proving out to be a complete path planner. If a path planner either finds a solution for the given problem or gives the output that there is no path that exists for the given problem, then it is said to be a complete path planner.

In this work, we have considered the obstacles to be static i.e. not moving. A brief discussion various obstacle avoidance algorithms are given below. We know there are many conventional methods for obstacle avoidance used in motion planning techniques for similar scenario i.e. fixed target point and static obstacles. A few of these methods are:

3.1.1 Bug Algorithm

Bug algorithms are the earliest and simplest sensor based path planners. These algorithms consider a robot as a point moving in the plane having contact sensors or a zero range for checking of an obstacle in the workspace. It does not use the range sensors. These algorithms are very straight forward and easy to implement. Results have proven that their success is guaranteed whenever it finds a path [18]. These algorithms follow two main principles. First is to move on a straight line and second is to follow a certain boundary. It follows the idea of navigating towards the goal and going around the obstacles. It can only detect an obstacle when it touches the obstacle. During navigation, the robot moves along a certain line towards the goal until it reaches the goal or obstructed by an obstacle. If it hits an obstacle, it then circumnavigates that obstacle. It then finds the nearest point to the goal, on going through the obstacle perimeter and traverses back to the original straight line path that would have been followed if there was no obstacle.

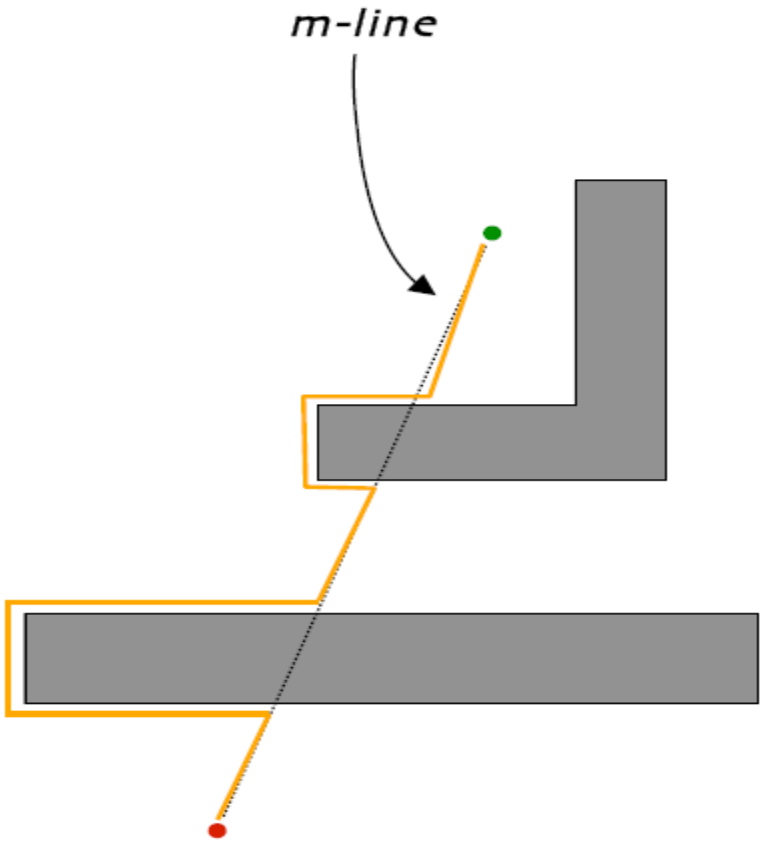

Fig 3.1: Path follows by a basic bug Algorithm [18]

3.1.2 Tangent Bug Algorithm

It is very much simpler to the basic bug algorithm but unlike the basic bug algorithms, which use tactile sensing, this method uses range sensing. This method requires only local knowledge of the environment, and of course can be useful even of whole environment is known [18]. This method governs when and how much to follow the boundary of obstacle and when to move towards the goal. It computes tangents on the boundary of an obstacle at any point, and moves the robot in that direction. These algorithms are restricted to two-directional configuration spaces and cannot work in the three-dimensional workspace.

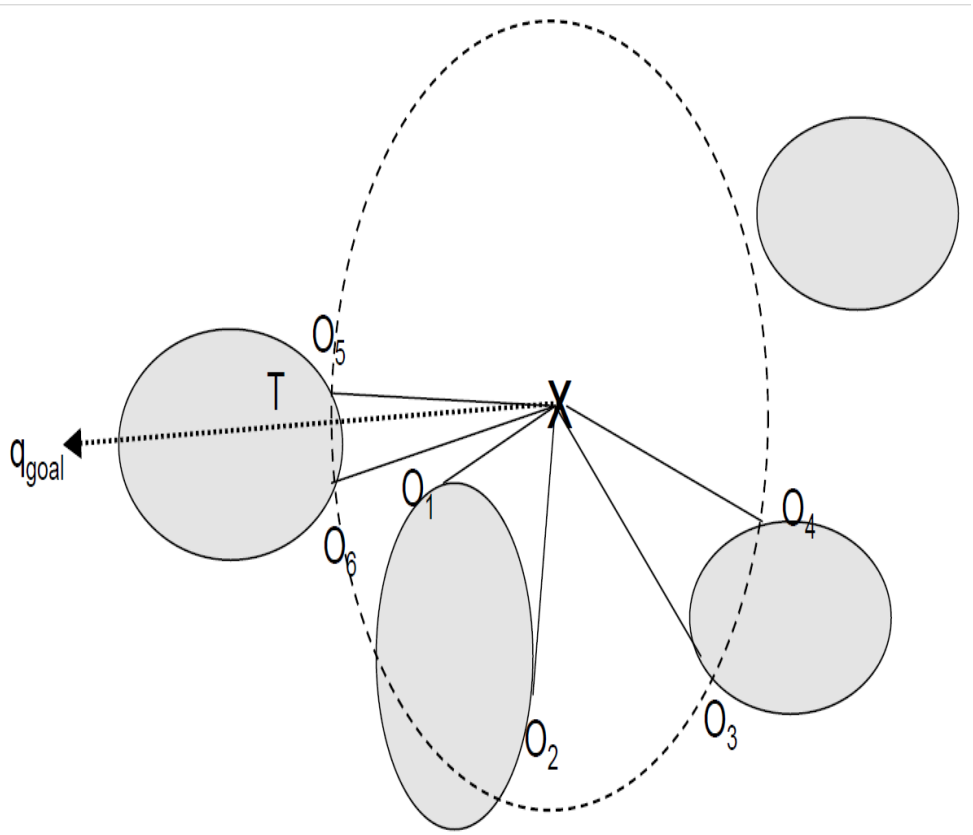

Fig 3.2: Path follows by a tangent Bug Algorithm [18]

3.1.3 Artificial Potential Fields

In this method, we need global knowledge of the environment. It defines a gradient vector field for each point in the workspace. As the name suggests, it gives each point in workspace a gradient of some function. The goal or target generates an artificial attractive field and the obstacles generate repulsive field. Based on these two types of field, a resultant field is computed which helps navigation. The robot or vehicle moves in the direction of minimum resultant field i.e. the net attractive force acting on it. The most common problem of artificial potential field is that it ends up with local minima [18]. It is the case which happens when the total force acting on all the points in the workspace is zero and the robot has not reached its destination yet and it has no other point in the space to proceed which has some force value. The potential function of a point is constructed by

$U(n) = U_{attractive}(n) + U_{repulsive}(n)$ …….. (3.1)

Where

U(n) = total potential force of a point

$U_{attractive}(n)$ = attractive force that a node gets from the goal node.

$U_{repulsive}(n)$ = repulsive force that a node gets from the obstacle.

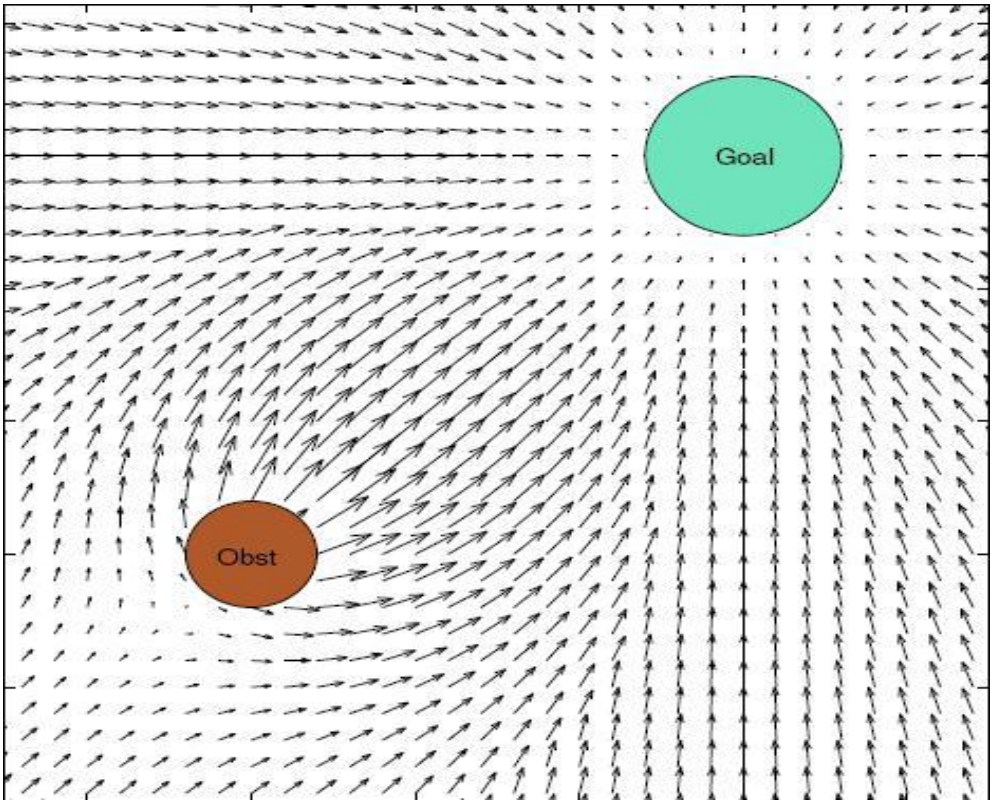

Fig 3.3: Combination of attractive and repulsive linear potential fields [18]

3.1.4 Navigation Potential Field

The artificial potential fields method discussed above, has resultant field based on an attractive and some repulsive fields (equal to number of obstacles). This resultant field should ideally have only 1 minimum i.e. the goal. But in fact, the potential field method suffers from

problem of local minima, which are points that are not goal but have locally minimum resultant field and the robot or vehicle gets stuck in it because it cannot move to a direction of minimum field anymore.

To avoid this problem, a method namely Navigation Field is used. This method also generates an artificial field, governed by its analytical relations by using knowledge of goal and obstacles. This method ensures to have only 1 global minimum and that is goal [18]. A function will be called a navigation function if:

a) It is smooth.
b) It has only one minimum in the configuration space and that minima must be the goal point ideally.
c) It is uniformly maximal on free space boundary.

3.1.5 Cell Decomposition

In cell decomposition method (which has further specialized versions Exact Cell Decomposition and Approximate Cell Decomposition) is another method that requires global knowledge of the environment. In this method the environment is broken down to smaller regions called cells [18]. The boundaries of the cells which are shared often tell a physical meaning. Those cells which share the common boundary are adjacent cells. An adjacency graph depicts the adjacency relationship of these decomposed cells. When the cells decomposition is done, then path planning with these cells is done in two steps;

a) First, the path planner determines those cells which occupy the start and goal nodes.
b) Second, it searches for the obstacle free path by joining these decomposed cells in the adjacency graph.

Two of the most common cell decomposition methods are:

a) Trapezoidal Decomposition: It decomposes the cells into number of polygons for planner configuration space.
b) Morse Decomposition: It shows more versatility with the representations of nonpolygonal and nonplanar spaces.

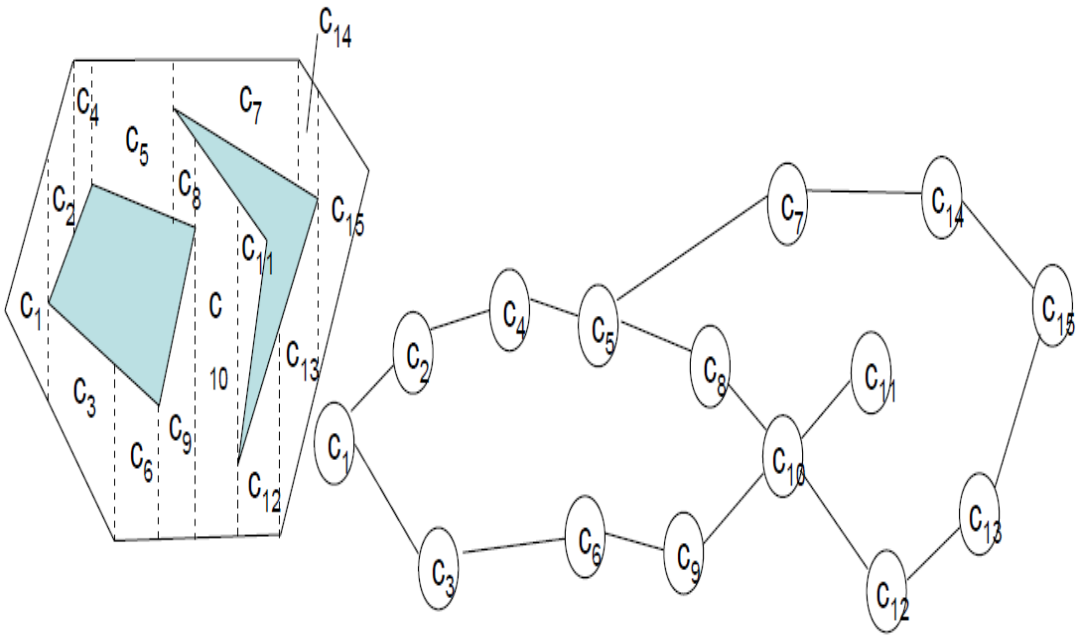

Fig 3.4: Trapezoidal Decomposition [18]

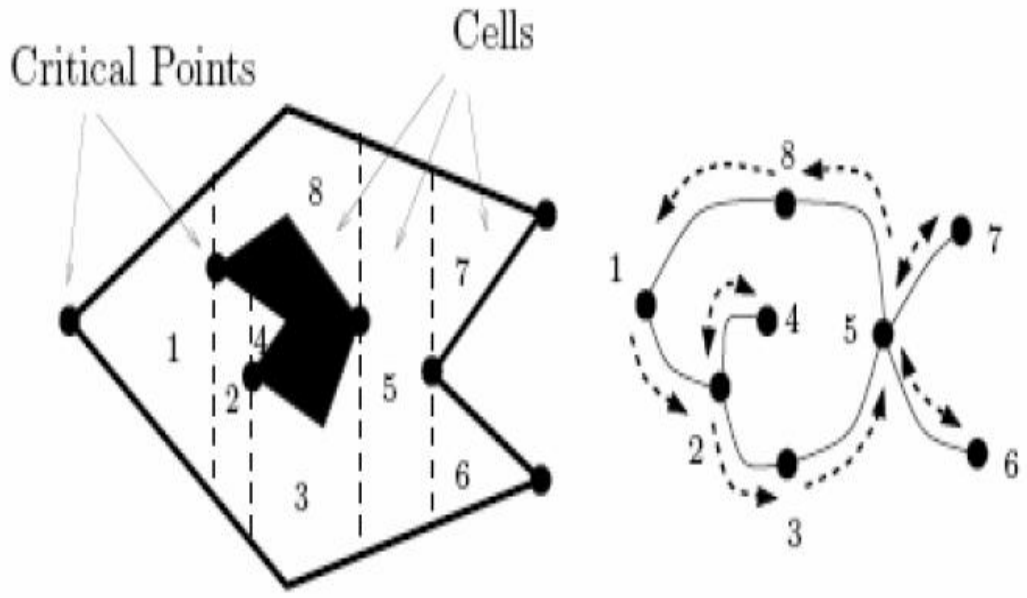

Fig 3.5: Morse Decomposition [18]

3.1.6 Roadmaps

Methods like Cell Decomposition use knowledge of that particular scenario to compute a feasible path from start to goal point. For different scenarios, the whole process is repeated every time which is not efficient if a vehicle has to plan its path frequently in a given environment. To address this problem, Roadmaps method is used. In this method, we build a roadmap of the given environment, which is similar to the concept of roads and highways used by humans [18]. Once a roadmap is built for an environment, then for every path planning scenario, we connect the vehicle start position with roadmap. The path of vehicle is planned on the roadmap till the point which is close and can be connected to the goal position. The most common roadmap is the visibility map. It applies to configuration spaces having polygonal obstacles. All different types of roadmaps has their own graph representation.

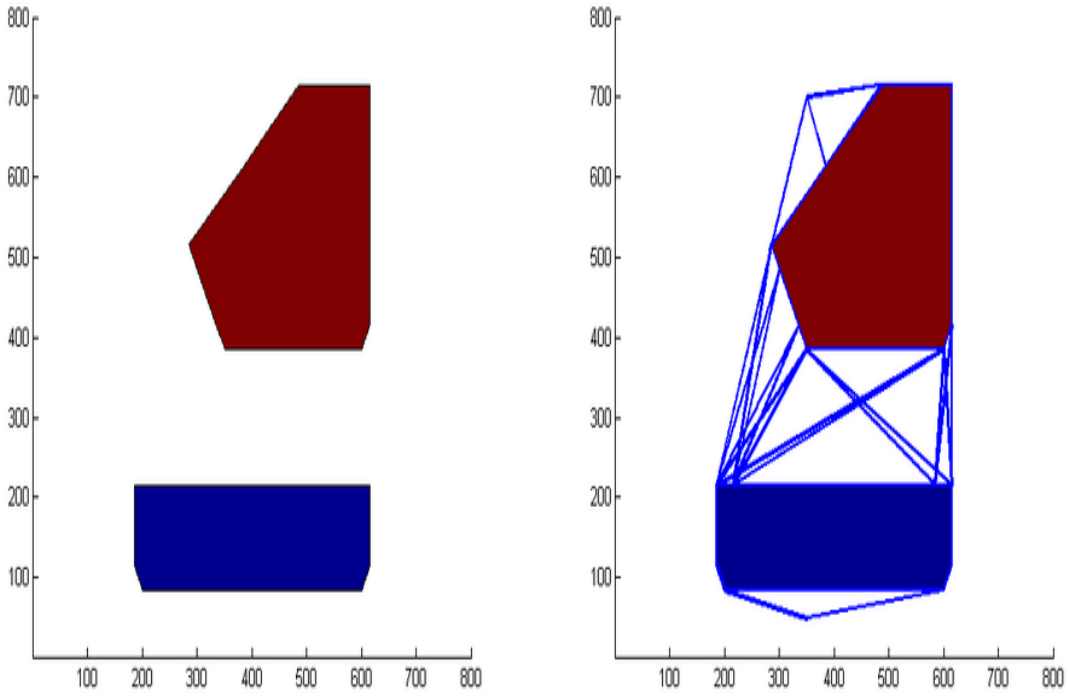

Fig 3.6: Visibility map representation [18]

3.1.7 Probabilistic Roadmaps

In probabilistic roadmaps, we do not use a fixed computational method to generate a roadmap. Rather we generate random possible samples or configurations or positions of the vehicle. The points which are found to be inside an obstacle are removed. We keep adding random points or samples till a path from start to goal is found by joining these points [18].

The probabilistic roadmap planner works in two phases. These are

a) Construction Phase: In this phase, a roadmap (graph) is made, approximating different paths that can be followed in the workspace.
b) Query Phase: In this phase, the start and goal nodes are connected to the roadmap, and the final path is obtained by a search algorithm

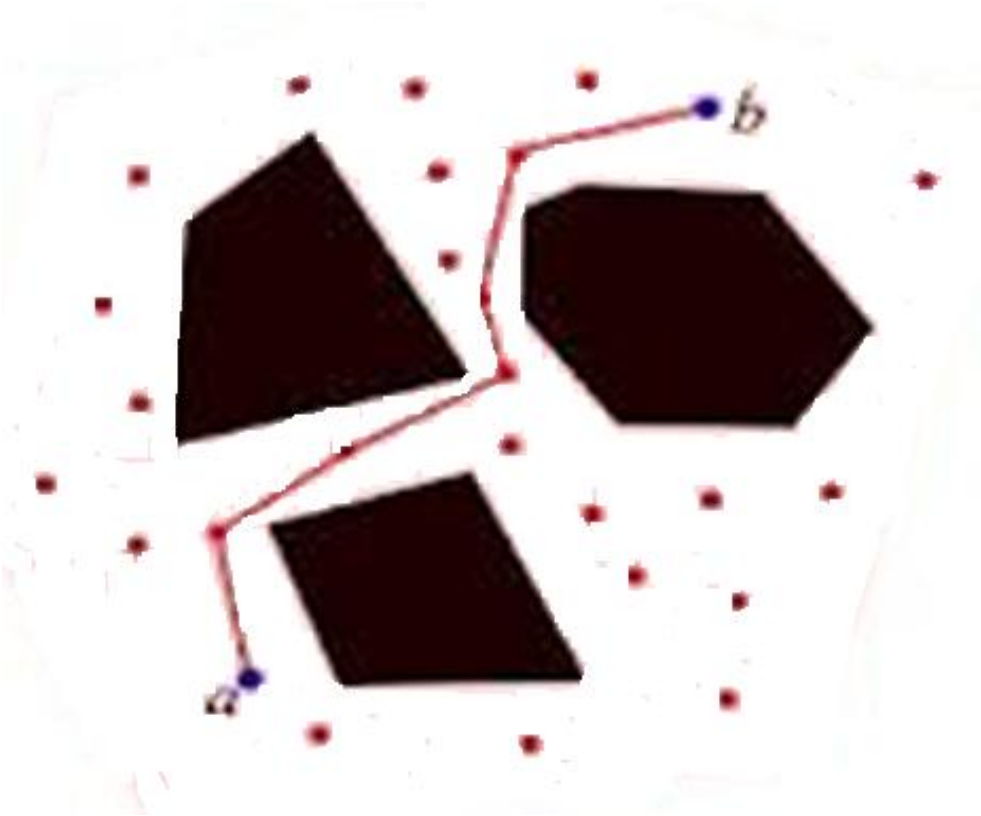

Fig 3.7: Probabilistic Roadmaps representation [18]

3.1.8 Velocity Obstacle Method

The velocity obstacle method takes an input the current position of the vehicle, its velocity and information of obstacles and returns a feasible velocity which ensures that there is no collision with obstacles. It takes only circular obstacles in 2D and spherical obstacles in 3D. It converts vehicle into point vehicle. Radius of obstacles is increased as much the vehicles dimensions are reduced. Here, the length of vehicle is larger than its width (which is true for most of car-like nonholonomic vehicles). Also it increases the obstacle radius equal to the length of vehicle, because that is the larger dimension and it will give some safety margin as well. If Ro is the original radius of the obstacle, new radius of obstacle in C-Space will be 'Ro + Ln' where Ln is the length of the vehicle. Next, a collision cone is made. It is the cone defined by the tangents, drawn on new c-space obstacle from the point vehicle. If the original velocity of the vehicle lies inside the collision cone, then new velocity is generated in the direction of closest tangent [22].

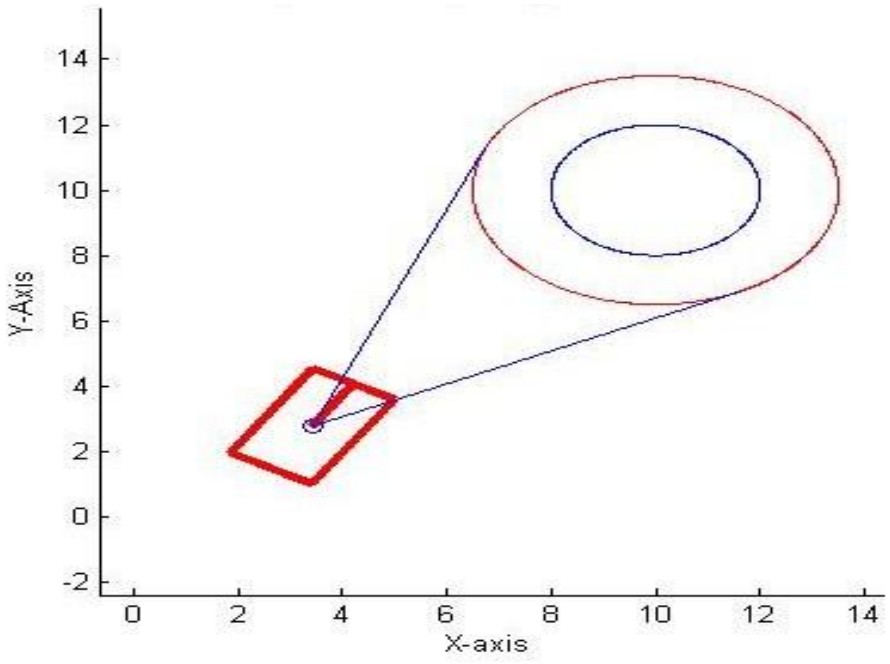

Fig 3.8: Velocity Obstacle Method implementation [22]

3.2 Brief description of Search algorithms

A search algorithm is an algorithm for finding an optimal solution from the best available collection of items for a particular problem. These items might be stored in a database as records or they might be search space elements. The most common search algorithms which are used extensively for the purpose of UAV/MAV path planning are A* search algorithm and dijkstra algorithm.

3.2.1 A* Search Algorithm

A* is a search algorithm which is commonly used in the problem of finding an optimal path and graph traversal. Due to its better performance and accuracy, it is widely used in a search problem. This algorithm is first introduced by Peter Hart, Nils Nilsson and Bertram Raphael in 1968. It is an extension of Edsger Dijkstra's algorithm and heuristic function used in A* search is for the purpose to increase the speed of Dijkstra's algorithm which was introduced in 1959. Due to the use of heuristics in its searching approach, A* achieves better performance than the dijkstra's algorithm. It along with other search algorithms examines a number of possible paths and gives an optimal solution. The total cost of each node traversed has a distance which is usually denoted by f(x) and is defined as "the sum of cost of current node from starting node and the heuristic cost from that node to the destination node".

$$f(x) = g(x) + h(x) \quad \ldots\ldots (3.2)$$

Where

g(x) = cost of the current node from the starting node.

h(x) = A "heuristic estimate" of the distance of that visited node to the goal node.

The h(x) value of the function f(x) should be an admissible heuristic, i.e. it should not overestimate the distance to the goal. Normally, it is taken as the straight line distance between the current node and the goal node, since it actually gives the shortest distance between two points.

3.2.2 Dijkstra Algorithm

Dijkstra's algorithm was first conceived in 1956 by a Dutch scientist, named Edsger Dijkstra. It was then published in 1959 as a graph search algorithm which can solve the problem of finding a shortest path among number of possible outcomes. Through dijkstra algorithm, for a particular node in the graph, it finds the path having lowest cost with every other vertex. It can also be used for finding costs of shortest paths from a single vertex to a single destination vertex by stopping the algorithm once the shortest path to the destination vertex has been determined.

A* Search algorithm is an extension of dijkstra algorithm which was formulated in 1968 by Peter Hart, Bertram Raphael and Nils Nilsson. Also one can say that dijkstra algorithm is a special case of A*, having h(x) value equal to zero. For dijkstra algorithm:

$$f(x) = g(x) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (3.3)$$

Where

$g(x)$ = cost of the current node from the starting node.

This is the basic difference between A* search algorithm and dijkstra algorithm that in dijkstra we do not take the heuristic function h (x) which is the straight line distance between the current node and the goal node. A* search algorithm is more a generalization of Dijkstra's algorithm which can be minimized on the size of the sub-graph which has to be explored.

A* search shows better results by using heuristic function. It has been successfully used for both land and aerial navigation. It gives the optimal path out of all possible solutions. For original A* to implement, the decomposition of the workspace is required especially from aerial navigation point of view as it gives different set of waypoints and by taking the combination of best waypoints, it leads to an optimal path. The workspace can be decomposed by using Delaunay triangulation method, voronoi diagrams etc. Through Delaunay triangulation, polygonal obstacles can be easily modeled and the workspace can be divided into flyable and obstacle triangles [11]. It shows very less computational expense but leads to a sub-optimal path. The Voronoi graph similar to Delaunay also provides a method to create waypoint path. These voronoi graphs and Delaunay are then searched through the proposed path planning algorithms normally A* search, dijkstra algorithm, eppstein’s k-best paths algorithm etc [16]. The final path

will may be the optimal considering the limitation of generated waypoints through the decomposition method and it may help in online path planning in which one might tackle with the moving obstacles and goal as in the case of guided missiles but from static environment point of view, it leads to a sub-optimal path and it may also have extra waypoints to be coped with during flight.

The proposed algorithm developed in this work does not use any voronoi graph or Delaunay triangulations to create waypoints for the path planner instead it emit rays in the direction of goal. It is a hybrid type of an algorithm in which two driving steps (algorithms) are blended together in order to get the best results. As in the case of [17] in which the researchers have used a hybrid algorithm by combining virtual force and A* search algorithm in order to cater the problem of local minima which comes in the case of virtual force algorithm and tried to get an optimal solution. The hybrid algorithm also results in the increase of computational cost. In the proposed algorithm, the combinations of rays which give the best result are picked. More the number of rays are emitted in the configuration space, more will be the optimality but it will be less computationally efficient. But computational inefficiency is not something to bother about in our case as we are dealing in an environment which is static. Also this proposed algorithm not only helps in achieving an optimal path but a power efficient path i.e. it generates minimum number of waypoints on its way to goal which is the sole requirement for an MAV path planner.

3.3 Proposed Algorithm (theory)

The proposed algorithm's search is different from [16] in which A* search algorithm is used, in a way that here the workspace is not decomposed into any polygons or quads and the original workspace remains as it is. In our proposed algorithm, we have tried to cut down the number of waypoints (turn points) for the MAV in order to make the system more power efficient and at the same time, we do not have to trade off on an optimal solution. In this thesis, we have used a hybrid approach that combines velocity obstacle method (VO) and depth first search (DFS) to navigate from initial position to goal position. VO method is normally used in dynamic environments [22] where we have pop up obstacles in addition to priori known obstacles on our way to the goal. DFS is a search technique that expands its first child node of the search tree and goes deeper and deeper until it finds the goal node. DFS is usually preferred to perform where

there is a limited depth and due to the fact that MAV has limited memory availability so one typically cannot use large data structures which other search algorithms do have for keeping track of the set of all previously visited nodes. So for the purpose of MAV path planning, we prefer DFS over other known search algorithms like A*, dijkstra etc.

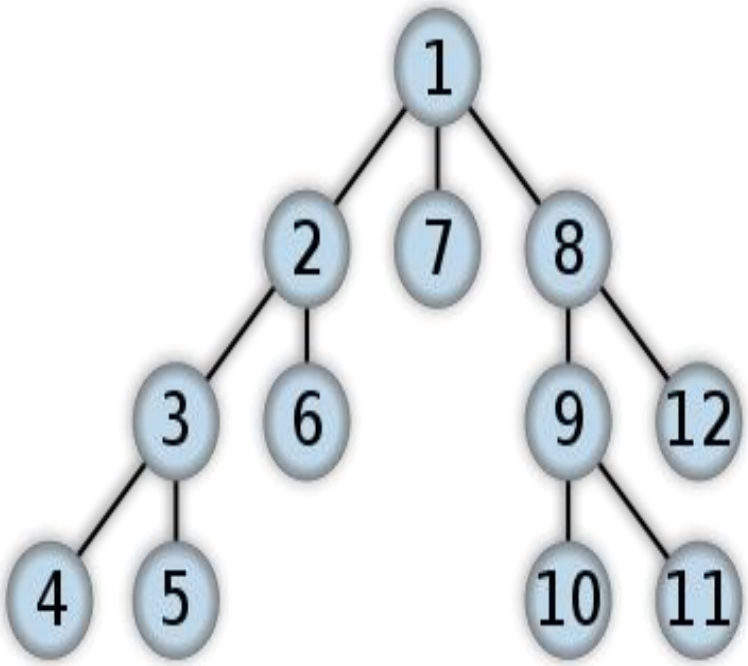

Fig 3.9: Depth First Search Tree [e]

We emit number of rays in the workspace from the starting point and check each ray for the closest point to the goal (the point which makes shortest distance to the goal), before striking an obstacle. Either we will reach the goal, or the closest point to goal will become our turning point. From each turn point on a single ray, we again emit rays, and repeat the process of finding the next turning point, or the goal point. We may end up with many rays reaching the goal. So a set of all the rays which we have traversed till reaching the goal will make our path. Finally we will choose the path with the least cost just like A*. We also refined that shortest path ray further in order to achieve optimality as it will be clear in the simulation results that with the proposed search technique, we may not get the optimal path but after refining, we get the optimal solution. After refining our path, we may further increase the computational cost of the algorithm

but that should not bother us as we are dealing in a known environment in which we have fixed obstacles and a fixed goal point. After further refining our proposed algorithm, the cost does not only improve but also the power efficient. That is why this proposed search algorithm has its worth when it comes to MAV path planning problem. Modification in the A* has been done previously [14],[15] and with good results but none has tried to refine it further which leaves a serious mark on its optimality. The proposed search algorithm takes input of vehicle initial position, goal position and obstacles position and returns a path which is optimal.

Normally, the researchers have taken the cost as distance with the effort to minimize the number of turn points. But in our case, we have taken the cost as distance plus the turn point cost with the turn point has been assigned with a multiplying factor. We have taken this factor as 10

Cost = total distance + (no. of turning points*10)

We can define the cost function as:

$$J = \sum_{i=1}^{n} di + a.T \quad \text{..... (3.4)}$$

Where

$d \in \mathbb{R}^2$

n = total no of paths

di = length of the ith path

a = no. of turns

T = turn point cost

The block diagram of our proposed algorithm is given below.

When we found multiple paths in reaching goal point, the best path would be that path which has minimum cost and that would be calculated as:

Best_Path = min ($J_1, J_2, J_3, \ldots\ldots\ldots\ldots J_n$) …… (3.5)

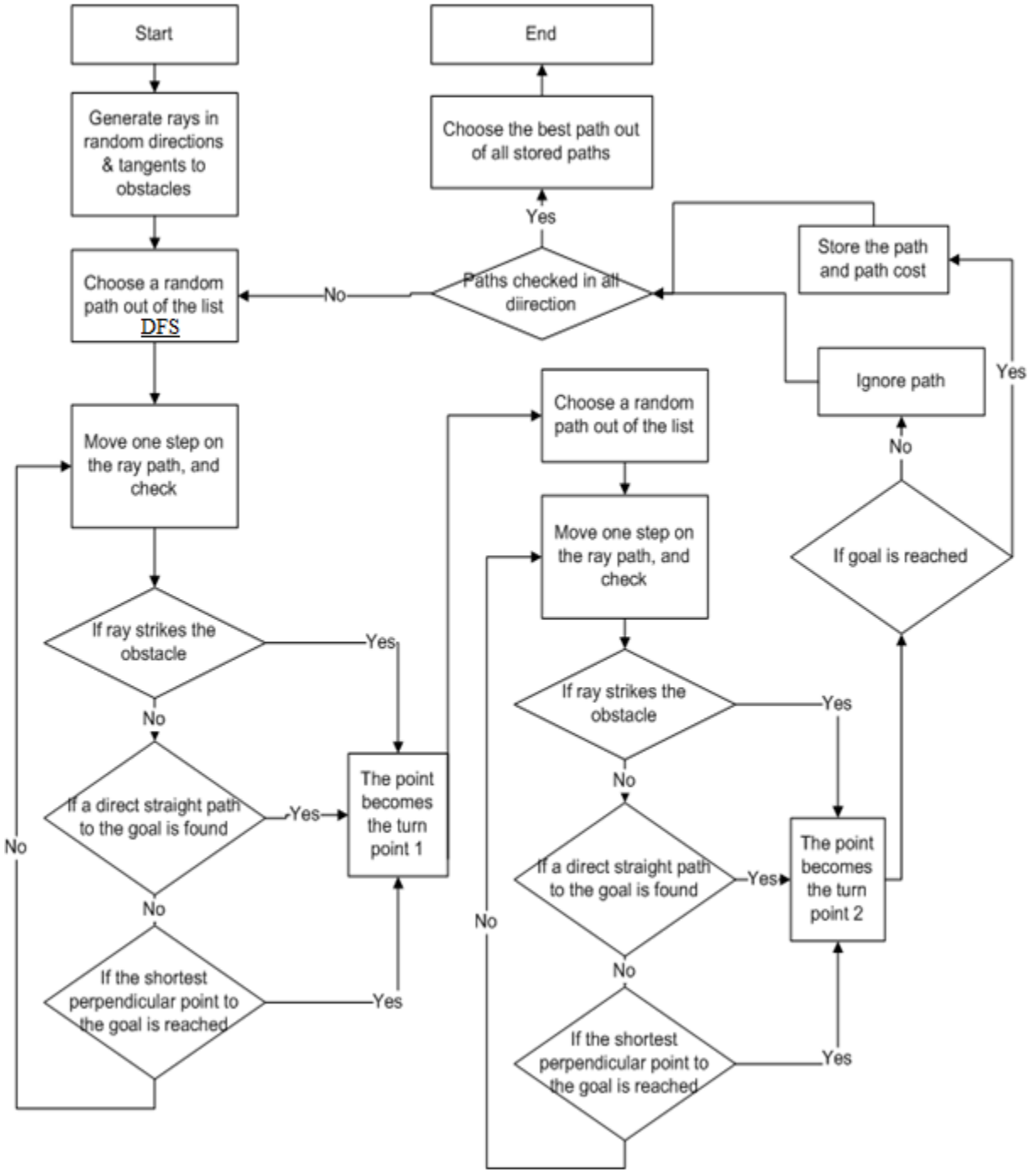

Fig 3.10: Block diagram of the proposed algorithm

3.4 Proposed Algorithm (Description)

The methods discussed above all are for the scenario similar like ours, but there is a basic difference in our case i.e. we do not just want to reach the goal, rather we want to reach the goal in minimum number of turns. This is not addressed by planning methods discussed above. And for vehicles like MAV, this becomes a rather different task to generate such a method by avoiding obstacles.

In our proposed algorithm, we emit number of rays in the workspace from the starting point in order to reach the goal. Also we emit rays in the direction of the obstacle tangent point in order to ensure the optimality. One of the rays is headed towards the goal point. If there is no obstacle in the path, then that ray will be our path.

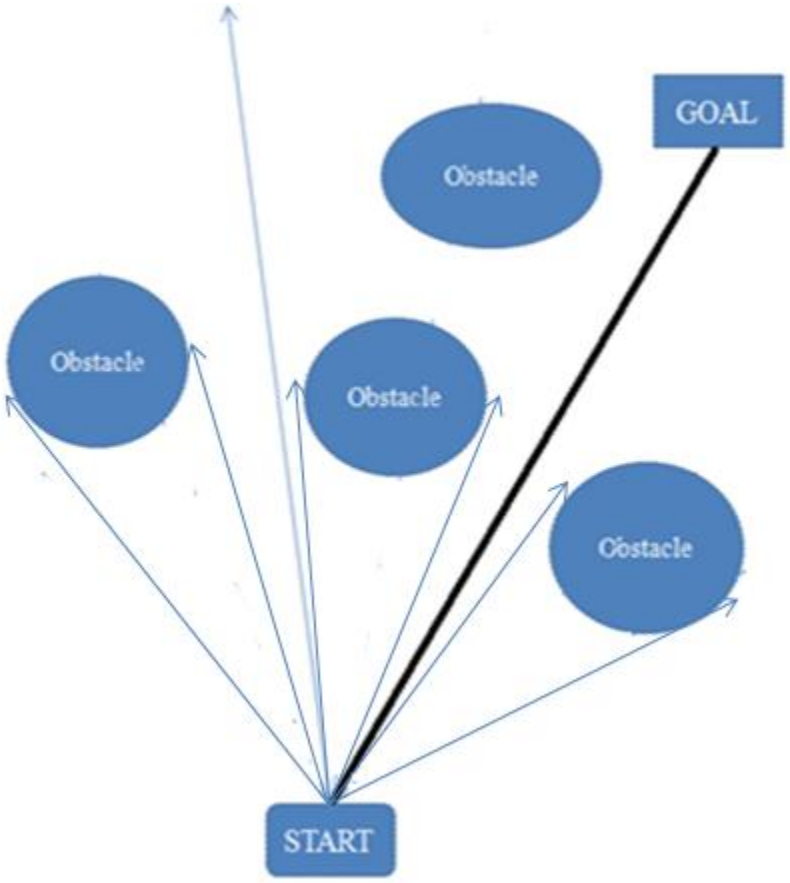

Fig 3.11: Proposed algorithm (greater angle between rays)

In fig 3.12, we emit rays in different directions and we can see that there is no obstacle in the direction of that ray which is heading towards the goal. So we will end up with straight line navigation despite of the fact that there were number of obstacles on its way but none of the obstacles was in the goal heading ray direction.

The angle between the rays plays a critical role in algorithm's optimality and computational efficiency. More the angle between the rays, more it will be computationally efficient but less will be its optimality. There will be a tradeoff between algorithm's optimality and computational efficiency but in either case power efficiency of the algorithm will not be sacrificed.

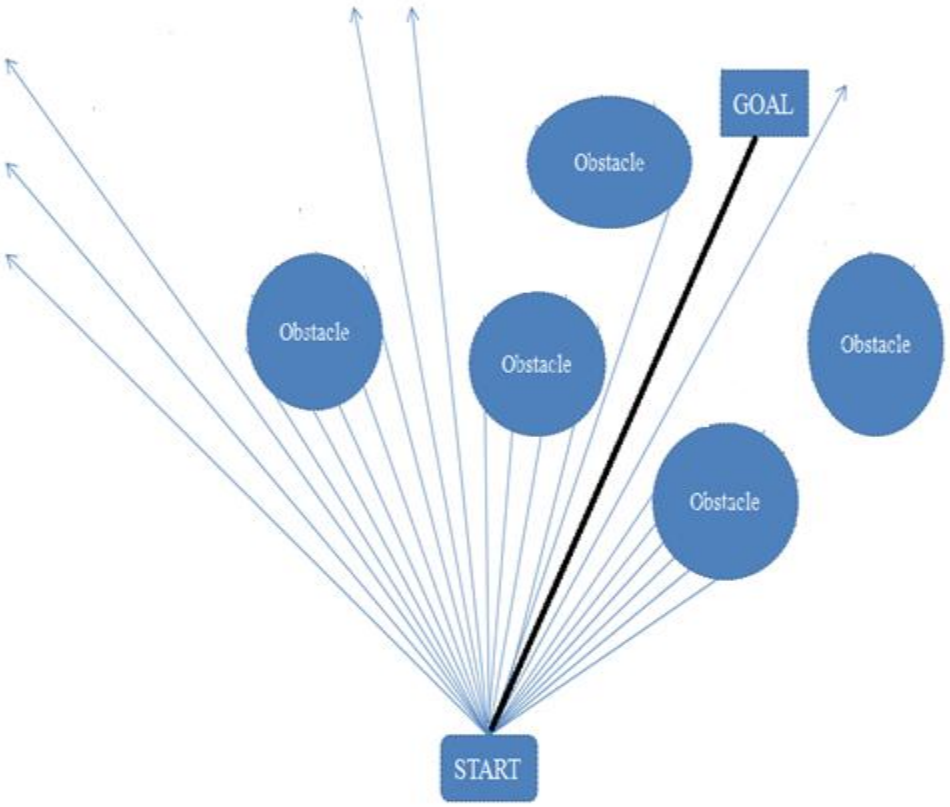

Fig 3.12: Proposed algorithm (lesser angle between rays)

Here we can see that, we have applied more rays in the workspace but it has given us the same result to that when applied to the earlier case which has rays with the larger angle or has minimum number of rays. This is simply because the goal heading ray has not been obstructed by any obstacle and we have the same path as we got previously with the lesser number of rays. This concludes that if the goal heading ray has not been obstructed, then we will have only one solution, no matters what angle between the rays has been taken. It only adds the computational cost if we keep on increasing the number of rays in the workspace.

The pseudo code for obstacle avoidance for MAVis given below:

```
for i = no of obstacles
        Distance of P2 from obstacle_centre = obstacle_centre - P2
        if (Distance of P2 from obstacle_centre < radius of obstacle)
                Path is not safe;
                P2 = turnpoint1
        else
                Path is safe
                P2 = P1 + incremental step;
        end if
end for
```

3.5 TANGENT POINTS GENERATION

It is important to ensure the optimality of the path. So we generate some rays which are tangent to the obstacles and these tangent rays tend to give us the optimal results. Here we use equation of a circle and the cosine law to find the tangent points on a circle.

Equation of Circle:

$$x^2 + y^2 = r^2 \quad \text{..... (3.6)}$$

We know the radius of the circle and we know the circle points in x-axis. Points for x-axis lies in the range given below.

circle_x_points = (center_x – rad) : 0.1 : (center_x + rad);

So from the above equation of circle, we will find y-values of the circle.

y_1 = center_y + $\sqrt{(\text{rad}^2 - (circle_x points - center_x)^2}$;

y_2 = center_y - $\sqrt{(\text{rad}^2 - (circle_x points - center_x)^2}$;

Now, from the Law of Cosine, we will find the tangent points on the respective circle.

$$a^2 = b^2 + c^2 - 2bc*CosA \quad \text{..... (3.7)}$$

where

a = distance or length from start point to centre of circle

b = length from start point to tangent point

c = radius of circle

A = tangent angle

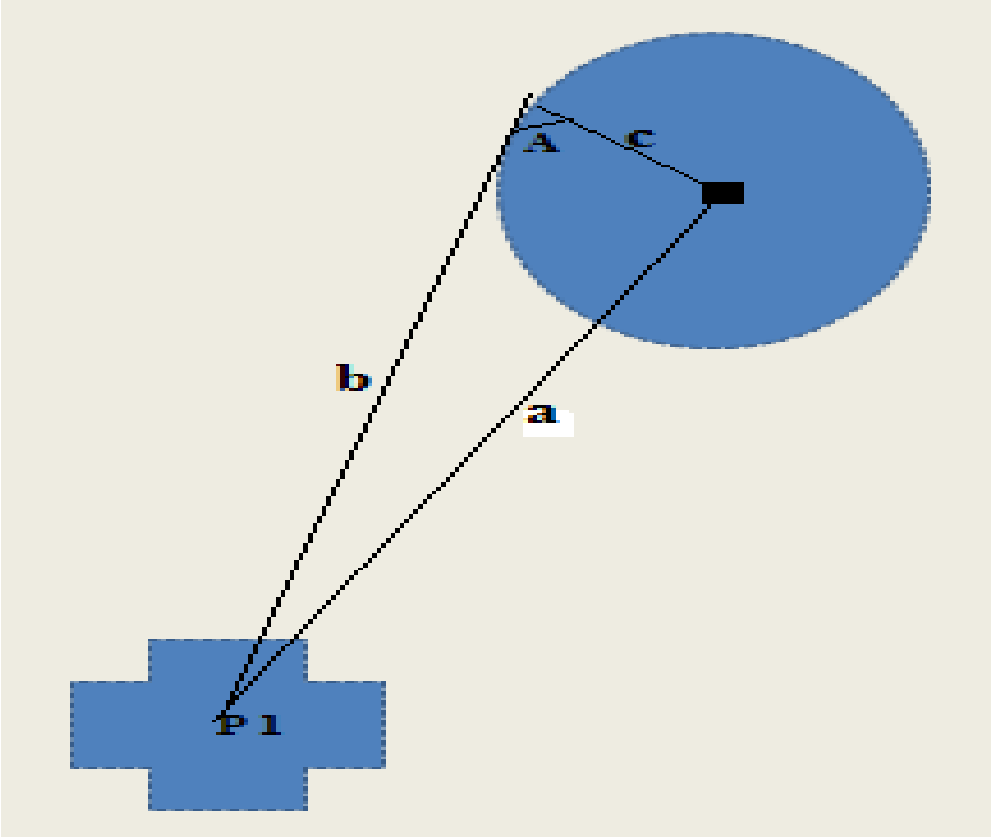

Fig 3.13: Tangent point generation

The pseudo code for tangent point generation on the obstacles for MAV is given below:

for i = 1: length(y_1)

Startpoint _center_distance = sqrt ((circle_center_y – start_point_y)^2 +

(circle_center_x – start_point_x)^2);

tangent_line_1 = sqrt (y_1 - start_point_y)^2 + (circle_x_point - start_point_x)^2);

tangent_angle_1 = (rad^2 + tangent_line_1^2 - Startpoint _center_distance ^{2})

/(2*rad*tangent_line_1);

if (tangent_angle_1 = 90)

tangent1_xpoint = circle_x_point;

tangent1_ypoint = circle_y_point;

```
        end if
end for
for  j = 1: length( y2 )
        Startpoint _center_distance =  sqrt ( (circle_center_y – start_point_y )^2  +
        (circle_center_x – start_point_x )^2 );
        tangent_line_2 = sqrt ( y2  - start_point_y)^2  + (circle_x_point -  start_point_x)^2 );
        tangent_angle_2 = (rad^2  + tangent_line_2^2  - Startpoint _center_distance ^2 )
        /(2*rad*tangent_line_2);
        if (tangent_angle_2 = 90)
        tangent2_xpoint = circle_x_point;
        tangent2_ypoint = circle_y_point;
        end if
end for
```

Here for each value of y-points of a circle, we will keep on checking that at what point it makes the angle 90. That particular y-point and its corresponding x-point will be our tangent point. Since equation of a circle is a quadratic equation, so it will give us two tangent points on the circle from a particular start point

Now coming back to the ray generation scheme, if the goal heading ray has been obstructed by an obstacle, then the optimality of the algorithm will depend on the number of rays that are spread in the workspace. Obviously, more the number of rays, more will be its optimality and lesser it will be a computationally efficient algorithm.

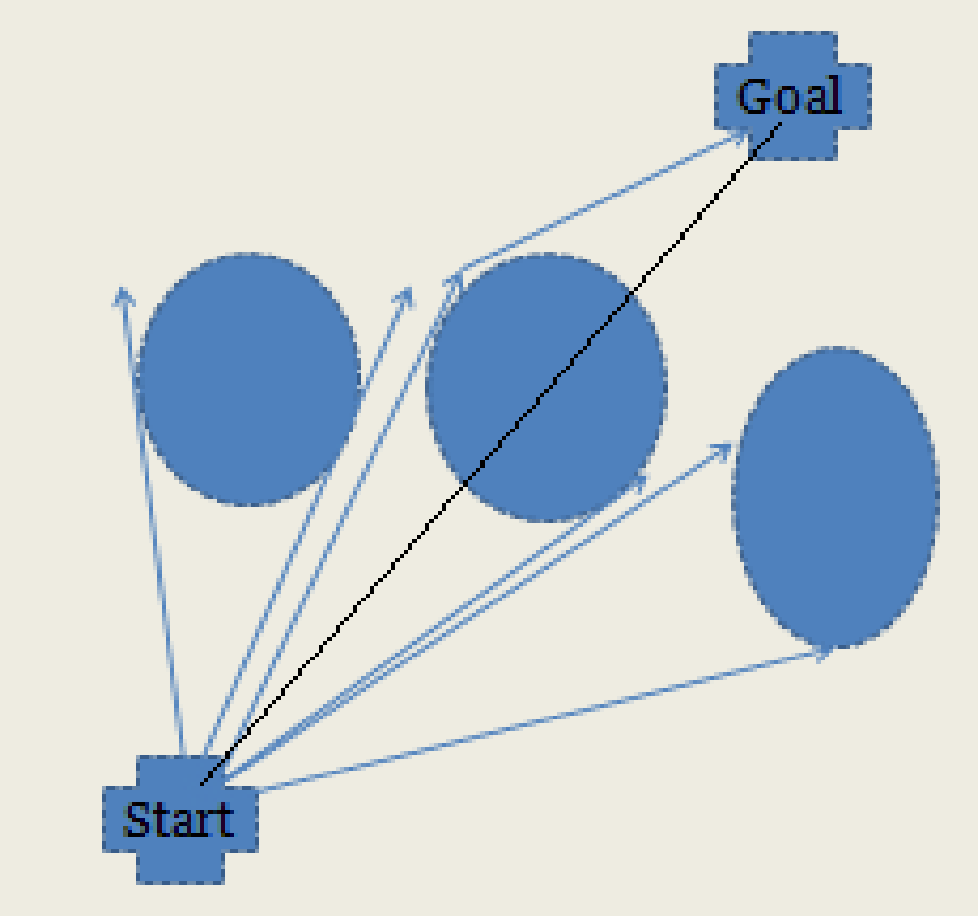

Fig 3.14: Ray heading goal is obstructed

In fig 3.15, we can see that the goal heading ray is obstructed by an obstacle in its path, so we have to choose a path other than that ray which gives us the best optimal solution. Here clearly, we can see that it gives us the optimal solution and it gives us the best solution out of these inputs just like A* search and dijkstra algorithm. So for achieving further optimality, we refined our path so that we could get the better result out of it. One important thing to note that in achieving the best result, the robot just has to deviate from its path one time only. So by this proposed algorithm, we not just only getting the optimal results but also the power efficient solution as with each waypoint actuator is switched on & off which consume the vehicle battery power and we are doing just that.

Now we will see how the refined path will help in further reducing our cost.

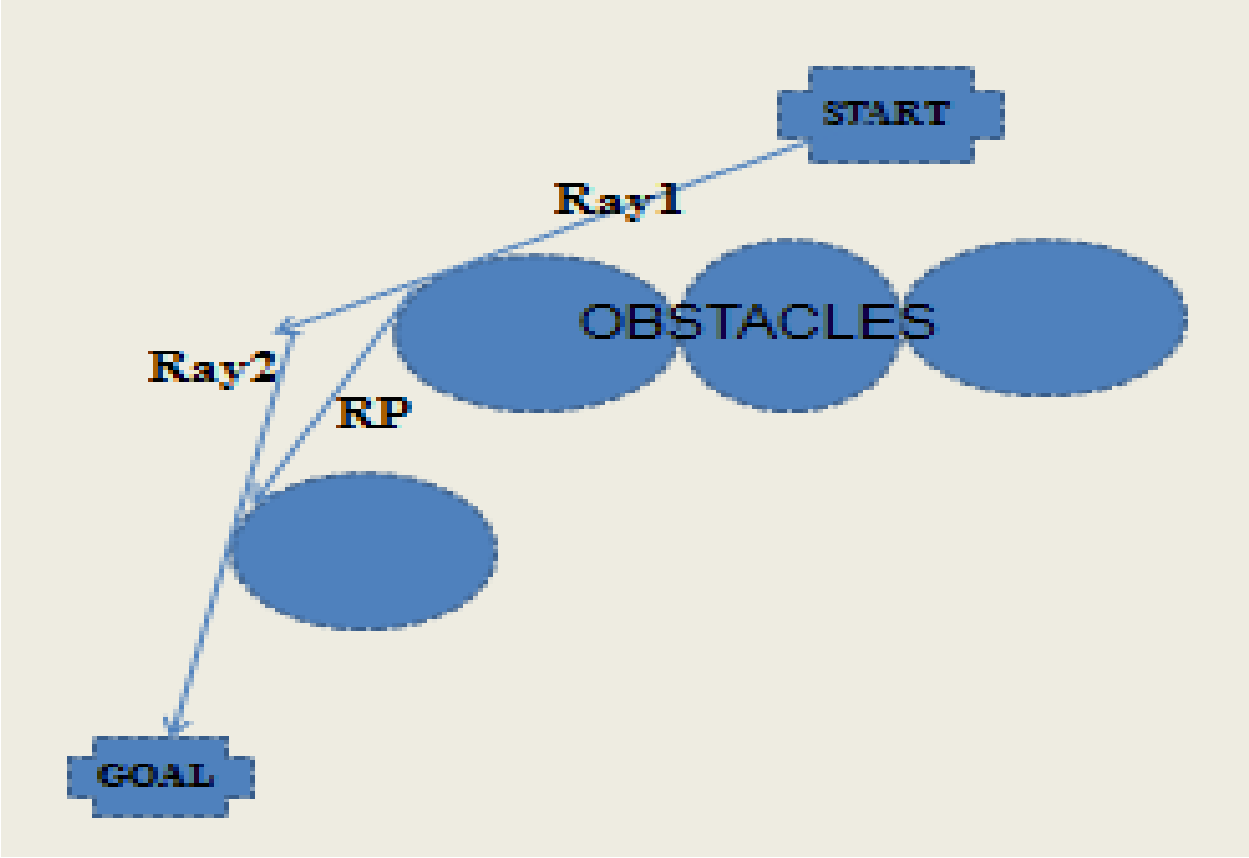

Fig 3.15: Proposed algorithm with the Refined Path

We can see that after refining our path, we have made our algorithm stronger in claiming that it has achieved an optimal path keeping in view that number of turn points to that of the original proposed algorithm has not been increased.

In refining, we check for the ray1 and ray2 that if we can further cut down our path. We check from ray1 on ray2 that which point from ray1 makes more difference from the original unrefined path keeping in view the position of all the obstacles. That combination of points on ray1 and ray2 are now our new path which we have termed as the refined path.

The pseudo code for refining path is given below:

for i = 1:P1_array

```
        P1_array  = Ray1_points;
for j = 1:P2_array
        P2_array  = Ray2_points;
If (path_from_P1_array_to_ P2_array = obstacle free path)
        New path = start_point to P1_array, P1_array to P2_array, P2_array to goal point;
end if
end for
end for
If (refined path < unrefined path)
        New_path = refined path
Else
        New_path = unrefined path
end if
```

4. IMPLEMENTATION

4.1. Two Dimensional path planning

The proposed algorithm explained in Chapter 3 has been applied for path planning of MAV in static environment both in 2D and 3D. The implementation of the proposed algorithm for path planning in 2D and 3D is explained in this chapter.

In our proposed algorithm, we emit number of rays in different directions in the workspace from the starting point in order to reach the goal. Then we applied depth first search algorithm in order to find the optimal solution. We will see in this chapter how the angle between the two rays or the number of rays emitted in the workspace affect the overall optimality of the algorithm. Also we will check the difference in cost between the refined and unrefined paths. Here green and red marks are showing the start point and red point respectively. We took circles as our obstacles as shown in the diagram below.

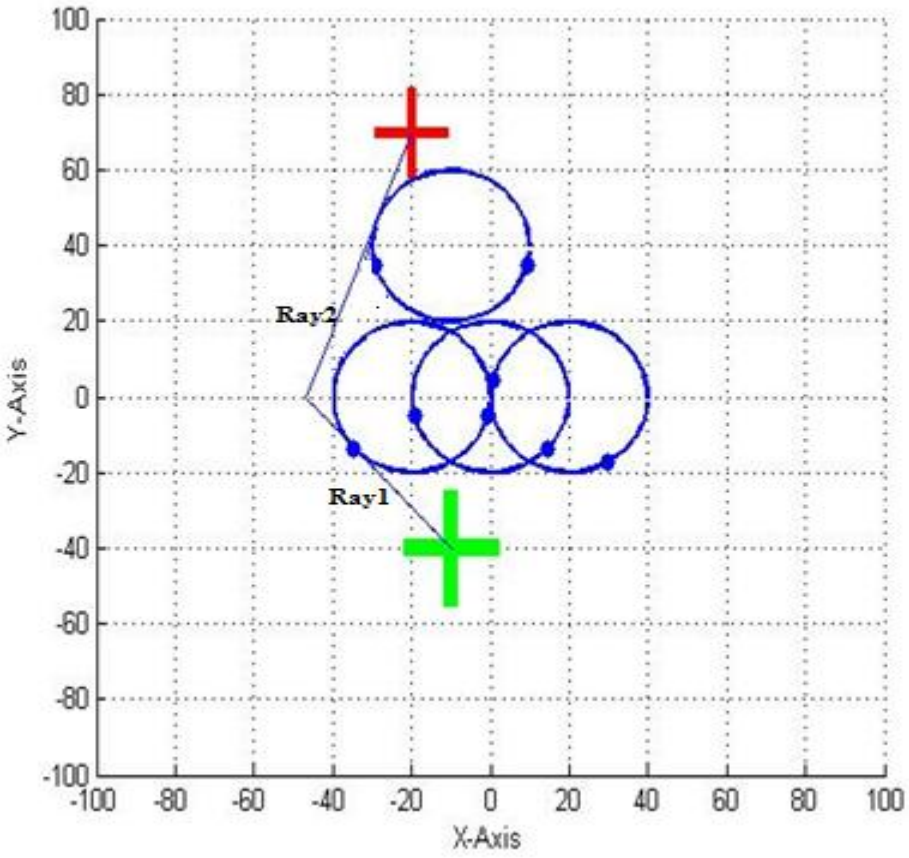

Fig. 4.1 UnRefined Path 1

Here in the fig 4.1, we have not refined our path and this unrefined path is giving us the total cost 129.5 +10=139.5. From the naked eye, we can see it is not the shortest path available but with only one turning point, it has the added advantage over the shortest path which we will see in the fig 4.2

We take another look of difference in cost of refined and unrefined path.

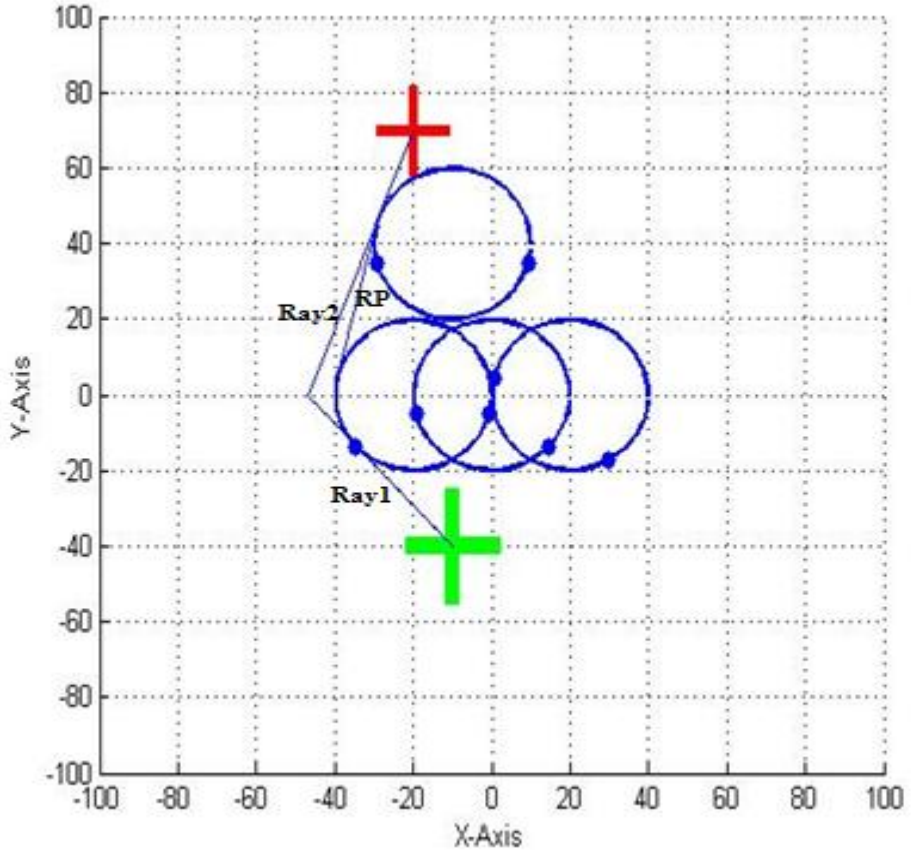

Fig. 4.2 Refined Path 1

Here in the fig 4.2, RP stands for refined path which appears to be the shortest path but it has one extra turning point when we compare it to the unrefined path. Now according to our new cost definition, the total cost of the refined path is 124.3016 + 20 = 144.3016. This cost is clearly more than the unrefined cost which was 139.5. So our algorithm will give us the unrefined path

as our best path which has more distance but due to less number of turning points, it has lower cost than the refined path. Now we will see another example of the difference between the refined cost and the unrefined cost in fig 4.3.

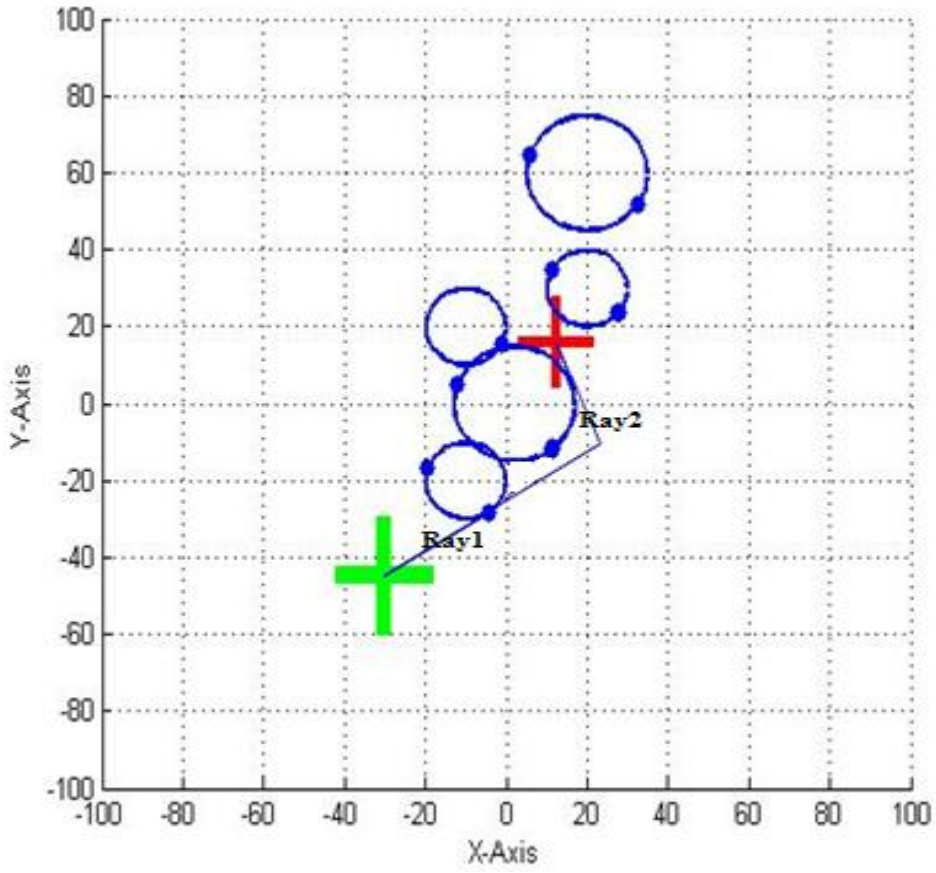

Fig. 4.3 UnRefined Path 2

Here in the fig 4.3, we have not refined our path and this unrefined path is giving us the total cost 92 + 10 = 102. Here 92 is the distance cost and 10 is the turning point cost. From the naked eye, we can see it is not the shortest path available but with only one turning point, it has the added advantage over the shortest path which we will see in the fig 4.4.

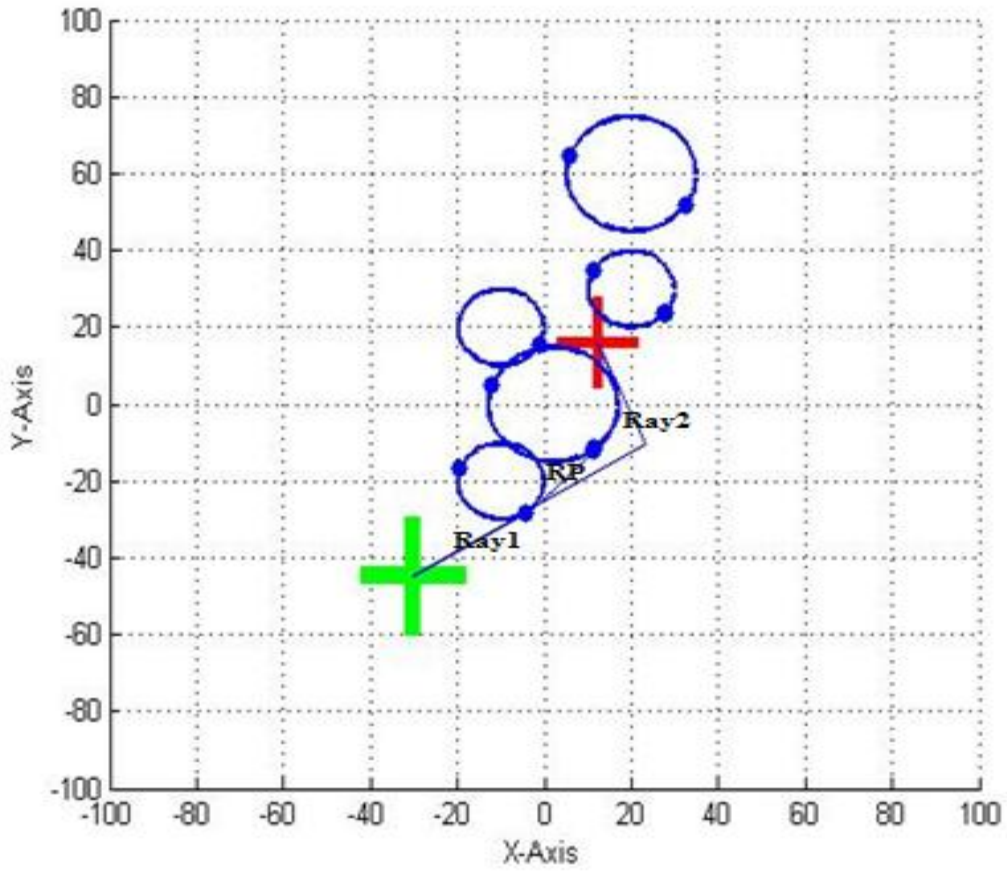

Fig. 4.4 Refined Path2

Here in the fig 4.4, RP stands for refined path which appears to be the shortest path but it has one extra turning point when we compare it to the unrefined path. Now according to our new cost definition, the total cost of the refined path is 84.1322 + 20 = **104.1322**. This cost is clearly more than the unrefined cost which was **102**. So our algorithm will give us the unrefined path as our best path which has more distance but due to less number of turning points, it has lower cost than the refined path. Now we will see another example of the difference between the refined cost and the unrefined cost in fig 4.5 and 4.6.

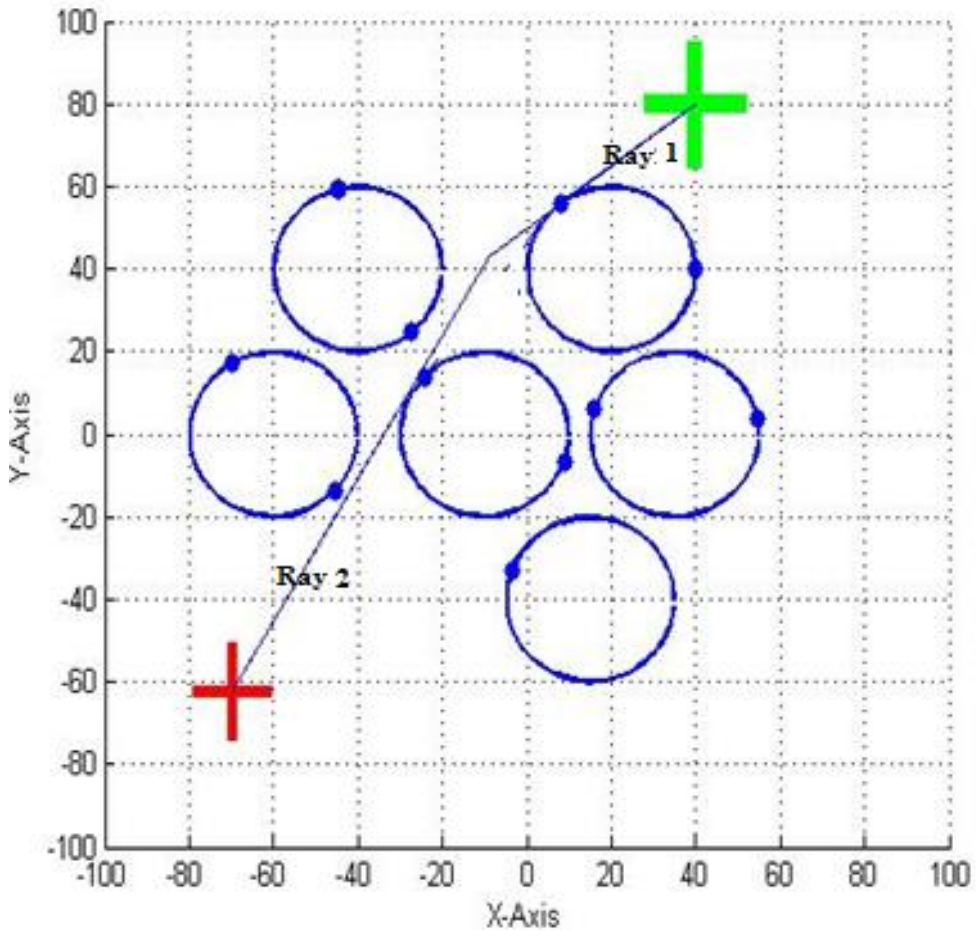

Fig. 4.5 UnRefined Path 3

Here in the fig 4.5, we have not refined our path and this unrefined path is giving us the total cost 183+10=193. Here 183 is the distance cost and 10 is the turning point cost. From the naked eye, we can see it is not the shortest path available but with only one turning point, it has the added advantage over the path that looks to be the shortest path which we will see in the fig 4.6.

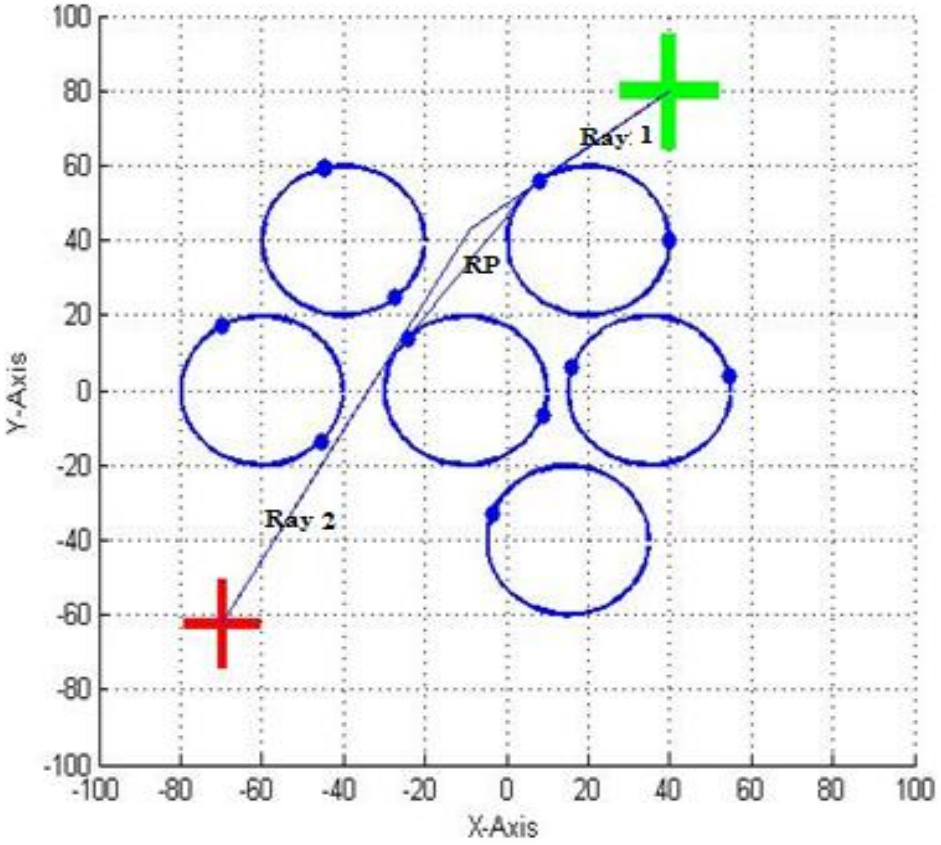

Fig. 4.6 Refined Path 3

Here in the fig 4.6, RP stands for refined path which appears to be the shortest path but it has one extra turning point when we compare it to the unrefined path. Now according to our new cost definition, the total cost of the refined path is 182 + 20 = **202**. This cost is clearly more than the unrefined cost which was **193**. So our algorithm will give us the unrefined path as our best path which has more distance but due to less number of turning points, it has lower cost than the refined path. Now we will see another example of the difference between the refined cost and the unrefined cost in fig 4.7 and 4.8.

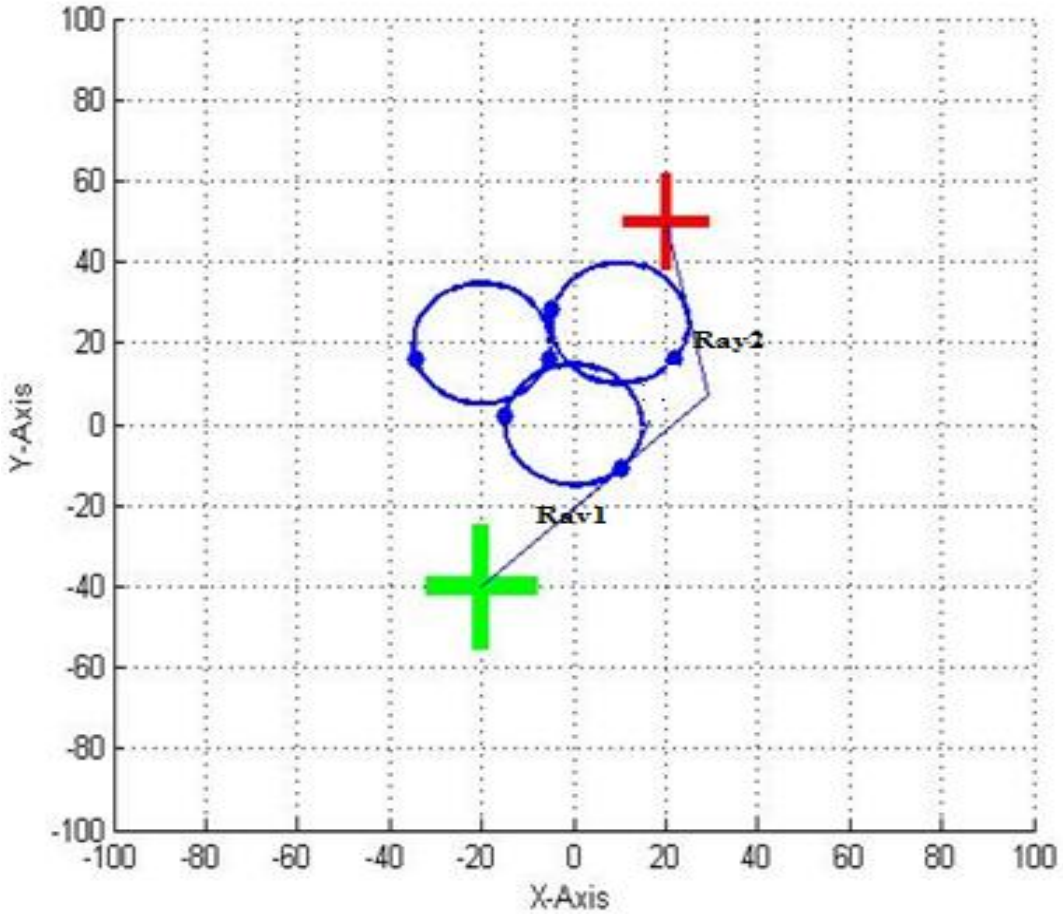

Fig. 4.7 UnRefined Path 4

Here in the fig 4.7, we have not refined our path and this unrefined path is giving us the total cost 112 + 10 = 122. Here 112 is the distance cost and 10 is the turning point cost. From the naked eye, we can see it is not the shortest path available but with only one turning point, it has the added advantage over the path that looks to be the shortest path which we will see in the fig 4.8.

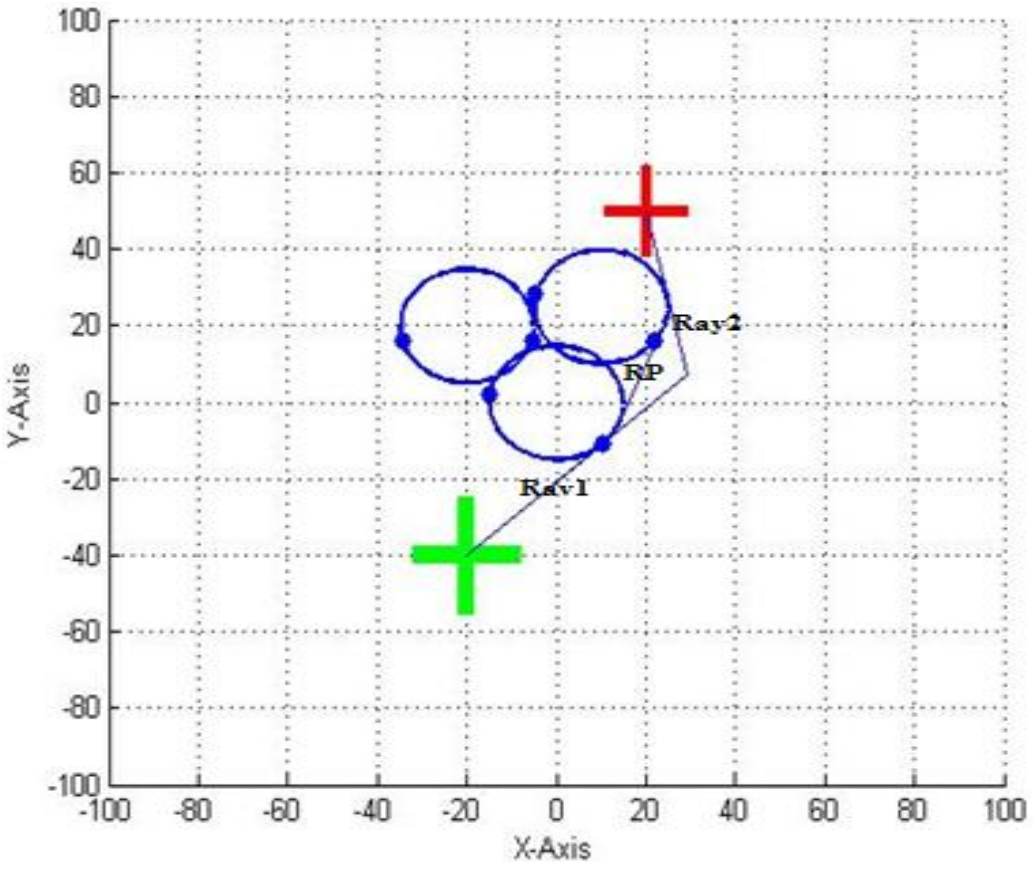

Fig. 4.8 Refined Path 4

Here in the fig 4.8, RP stands for refined path which appears to be the shortest path but it has one extra turning point when we compare it to the unrefined path. Now according to our new cost definition, the total cost of the refined path is 106.81 + 20 = **126.8086**. This cost is clearly more than the unrefined cost which was **122**. So our algorithm will give us the unrefined path as our best path which has more distance but due to less number of turning points, it has lower cost than the refined path. Now we will see another example of the difference between the refined cost and the unrefined cost in fig 4.9 and 4.10.

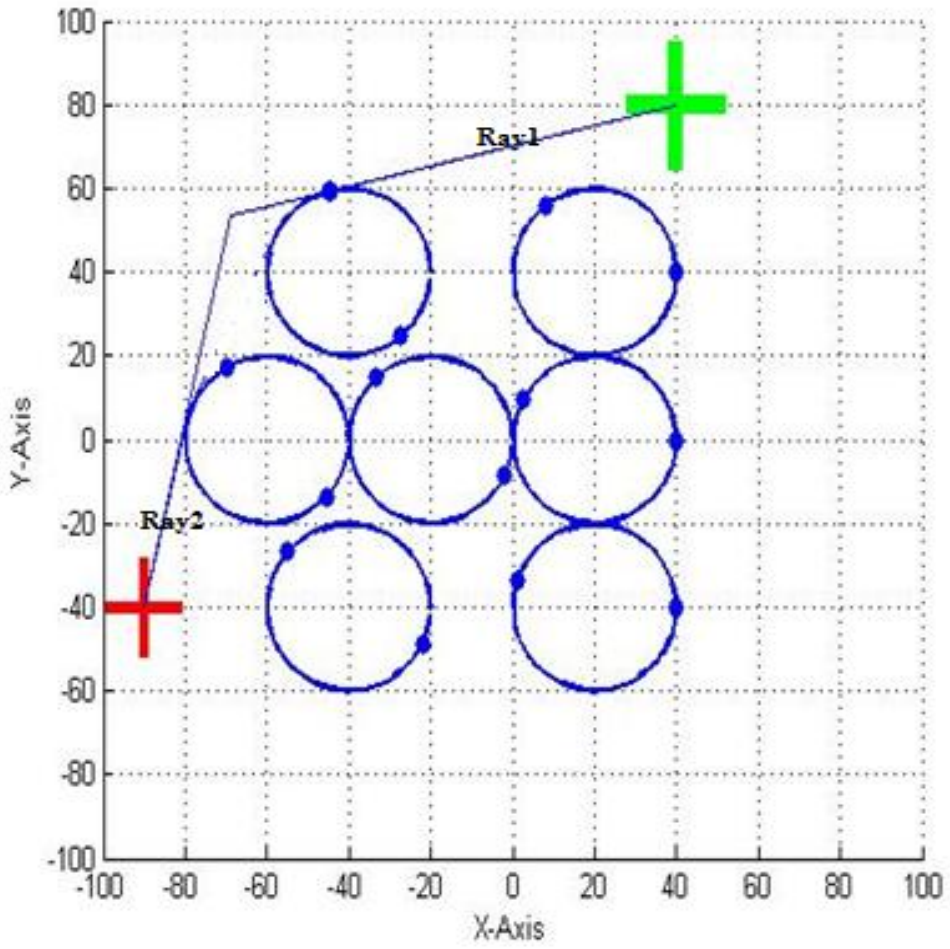

Fig. 4.9 UnRefined Path 4

Here in the fig 4.9, we have not refined our path and this unrefined path is giving us the total cost 208 + 10 = 218. Here 208 is the distance cost and 10 is the turning point cost. From the naked eye, we can see it is not the shortest path available but with only one turning point, it has the added advantage over the path that looks to be the shortest path which we will see in the fig 4.10

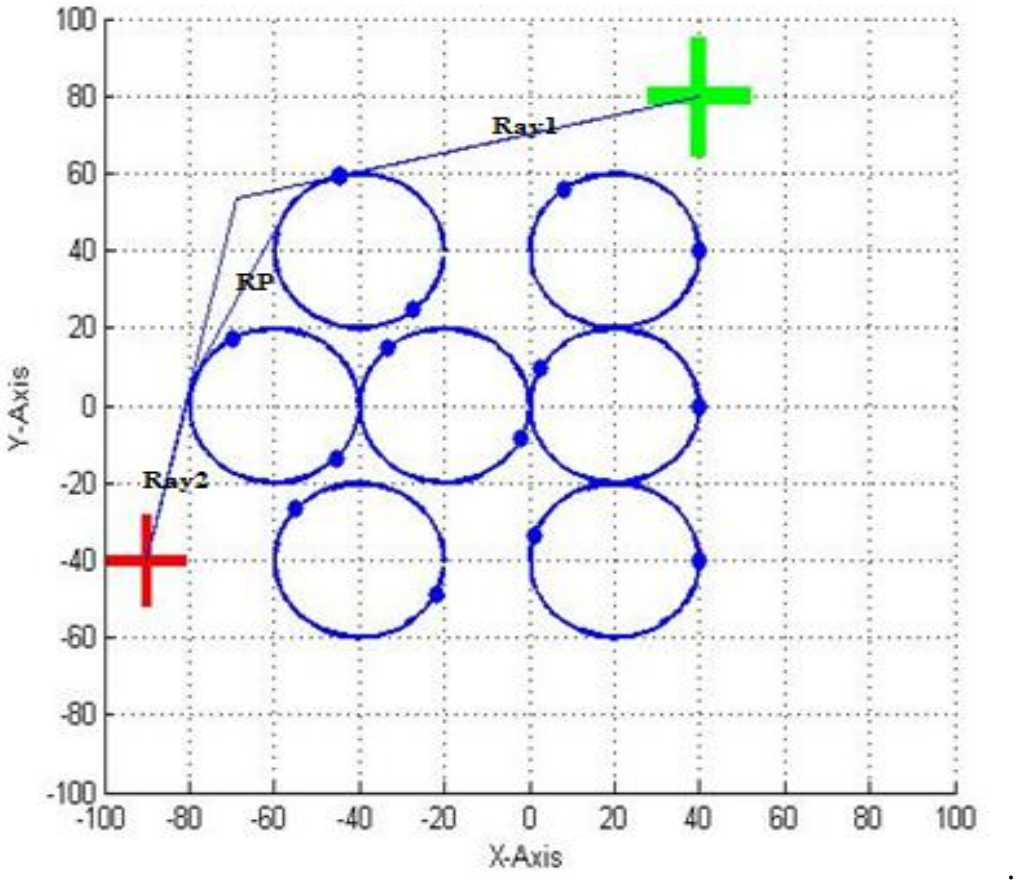

Fig. 4.10 Refined Path 4

Here in the fig 4.10, RP stands for refined path which appears to be the shortest path but it has one extra turning point when we compare it to the unrefined path. Now according to our new cost definition, the total cost of the refined path is 200.0812 + 20 = **220.0812**. This cost is clearly more than the unrefined cost which was **218**. So our algorithm will give us the unrefined path as our best path which has more distance but due to less number of turning points, it has lower cost than the refined path. Now we will see another example of the difference between the refined cost and the unrefined cost in fig 4.11 and 4.12.

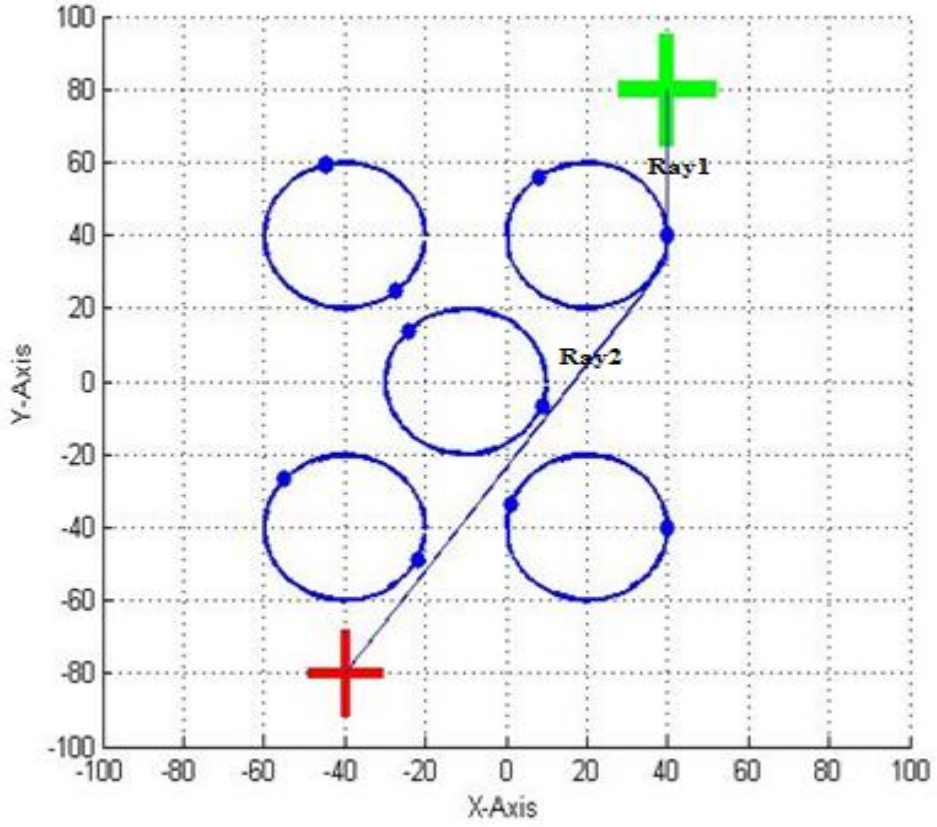

Fig. 4.11 UnRefined Path 5

Here in the fig 4.9, we have not refined our path and this unrefined path is giving us the total cost 185.5 + 10 = **195.5**. Here 185.8 is the distance cost and 10 is the turning point cost. From the naked eye, we can see it is also the shortest path available and with only one turning point and it cannot be further refined as it is giving us the shortest solution. So here refined path will be same as the unrefined path, i.e. 185.5 +10= **195.5**. Now we will see another example of the difference between the refined cost and the unrefined cost in fig 4.13 and 4.14.

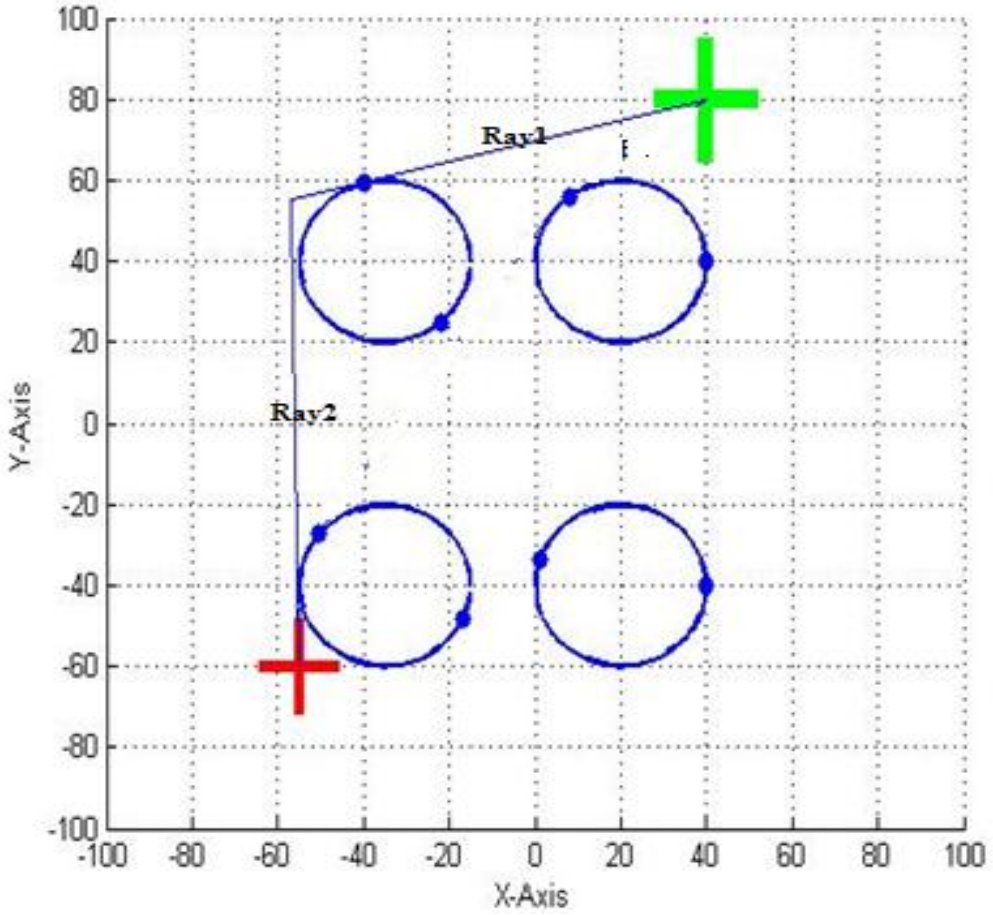

Fig. 4.12 UnRefined Path 6

Here in the fig 4.12, we have not refined our path and this unrefined path is giving us the total cost 215 + 10 = 225. Here 215 is the distance cost and 10 is the turning point cost. From the naked eye, we can see it is not the shortest path available but with only one turning point, it has the added advantage over the path but it has way too much distance cost as compared to the refined path which will clearly indicate that it will not be our best path but the best path will be the refined path which we will see in the fig 4.13.

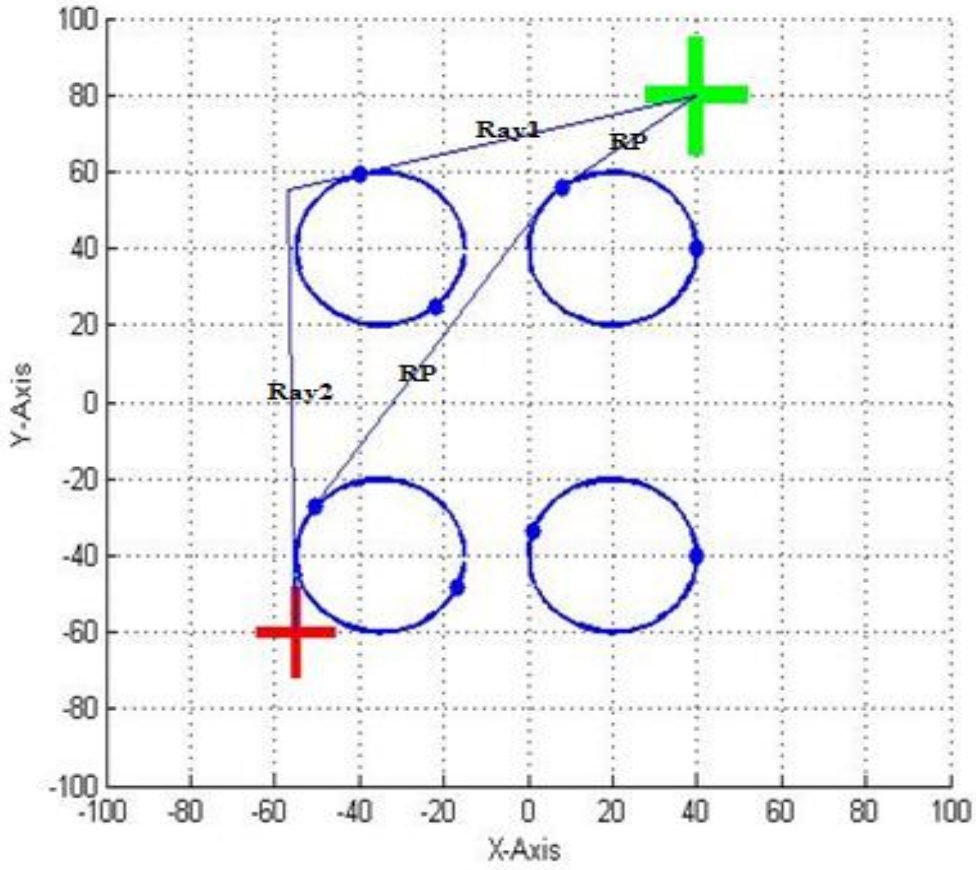

Fig. 4.13 Refined Path 6

Here in the fig 4.13, RP stands for refined path which appears to be the shortest path but it has one extra turning point when we compare it to the unrefined path. Now according to our new cost definition, the total cost of the refined path is 175.8982+20= **195.8982**. This cost is clearly less than the unrefined cost which was **225**. So our algorithm will give us the refined path as our best path which has less distance than the unrefined path, so finally, it has lower cost than the unrefined path. Now we will see another example of the difference between the refined cost and the unrefined cost in fig 4.14.

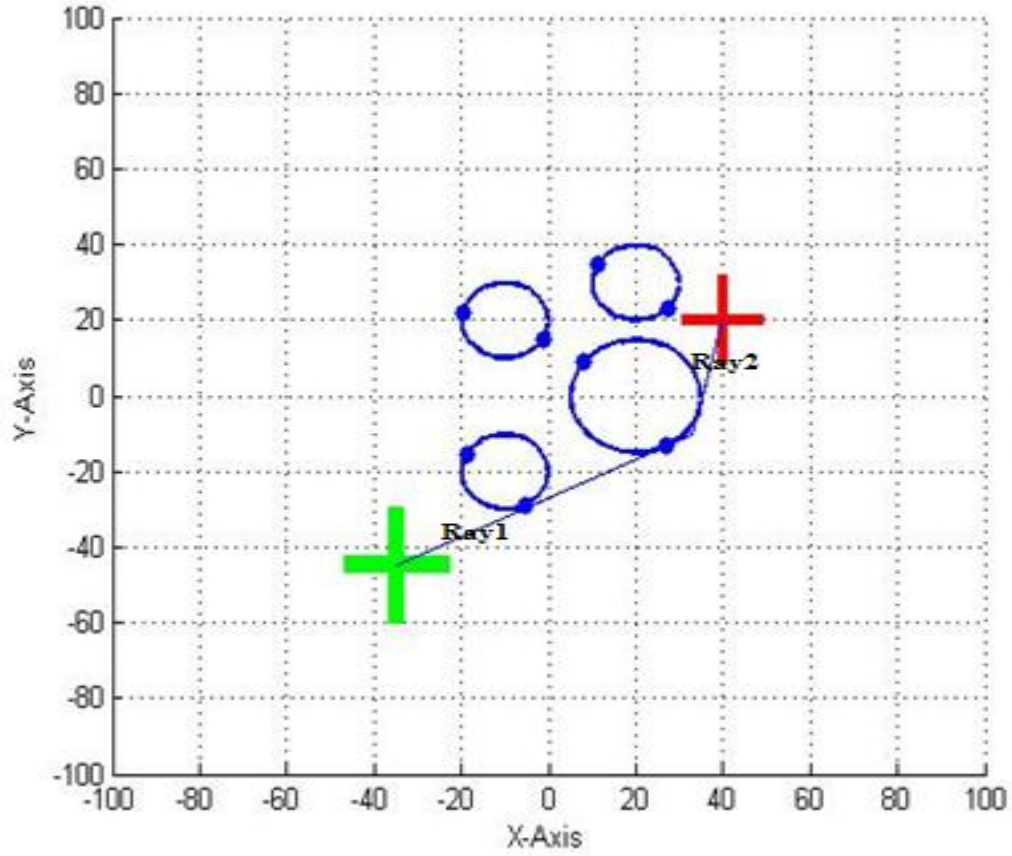

Fig. 4.11 UnRefined Path 7

Here in the fig 4.9, we have not refined our path and this unrefined path is giving us the total cost 107.5 + 10 = **117.5**. Here 107.5 is the distance cost and 10 is the turning point cost. From the naked eye, we can see it is also the shortest path available and with only one turning point and it cannot be further refined as it is giving us the shortest solution. So here refined path will be same as the unrefined path, i.e. 107.5 + 10 = **117.5**.

5. COMPARISON AND CONCLUSION

5.1 Comparison of Results

In this thesis, we have proposed a new method for the problem of finding a collision free path for a MAV moving in a static environment. Static environment means that both the obstacles and the target are fixed and that MAV has the global knowledge of its environment i.e. terrain map is pre-loaded in it. We took circles as our obstacles in 2D. Circles are the bounded circles. Any shape of the obstacle can be replaced by the bounded circles in 2D. The main advantage of the proposed algorithm is its simplicity of architecture, optimality, power efficiency and most importantly, its decision making whether to pick the refined path as best solution or the unrefined path which usually have lesser number of turn points as the best solution. This is a true advantage in a way that previously when researchers try to make their algorithm power efficient, they lose optimality with a considerable amount. Further refining of the proposed path helps in achieving the best results which might not be delivered from the unrefined path.

However, the robustness of this algorithm also ensures it performs well in all scenarios. In this section, a comparison of results of this algorithm having both refined path and the unrefined path will be done with each other. We will not compare our algorithm results to the any other algorithm available in literature as no algorithm can give us the best solution in each and every scenario and also due to the fact we have changed our cost definition according to the need of the MAV flight as our cost definition is

Cost = total distance + (no. of turning points*10)

So comparison with the other existing algorithm will not justify our algorithm's intelligent approach rather we will show how our algorithm behaves in a particular scenario when it has to pick a path from refined and the unrefined path. Although we have done it already in the previous chapter but just to conclude our debate, we are presenting the difference in this section also.

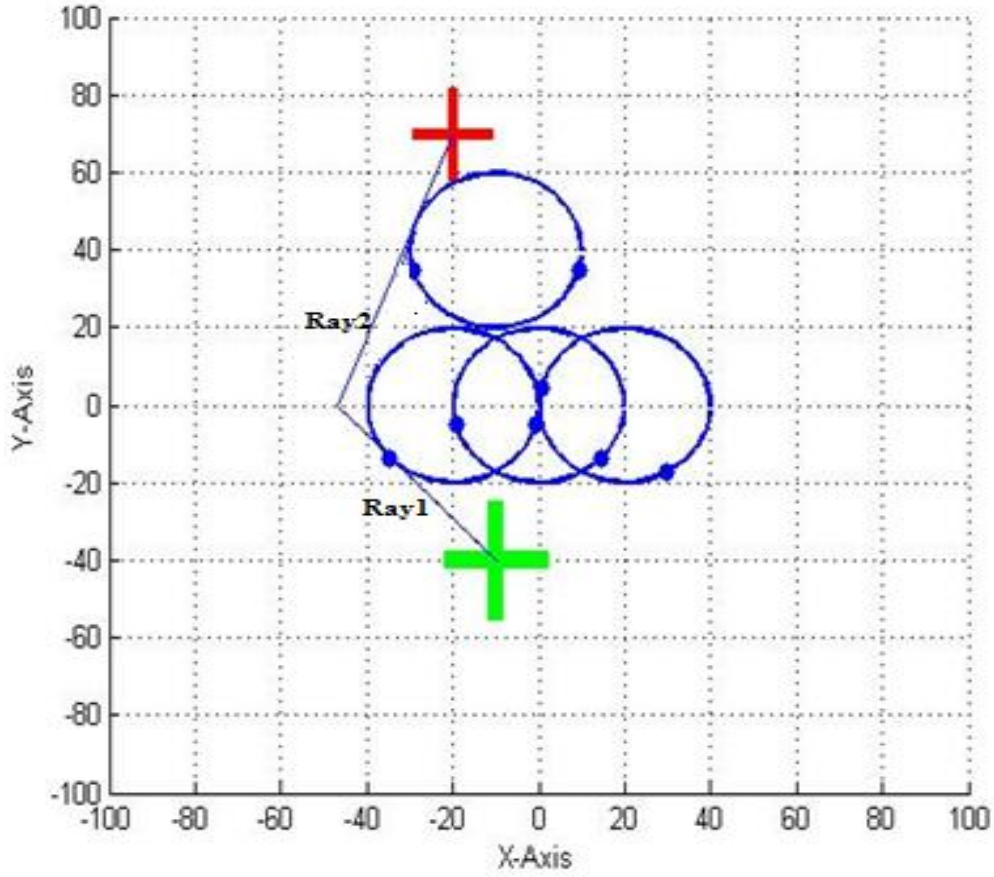

Fig. 5.1 UnRefined Path

Here in the fig 5.1, we have not refined our path and this unrefined path is giving us the total cost 129.5 +10 = **139.5**. From the naked eye, we can see it is not the shortest path available but with only one turning point, it has the added advantage over the shortest path which we will see in the fig 5.2.

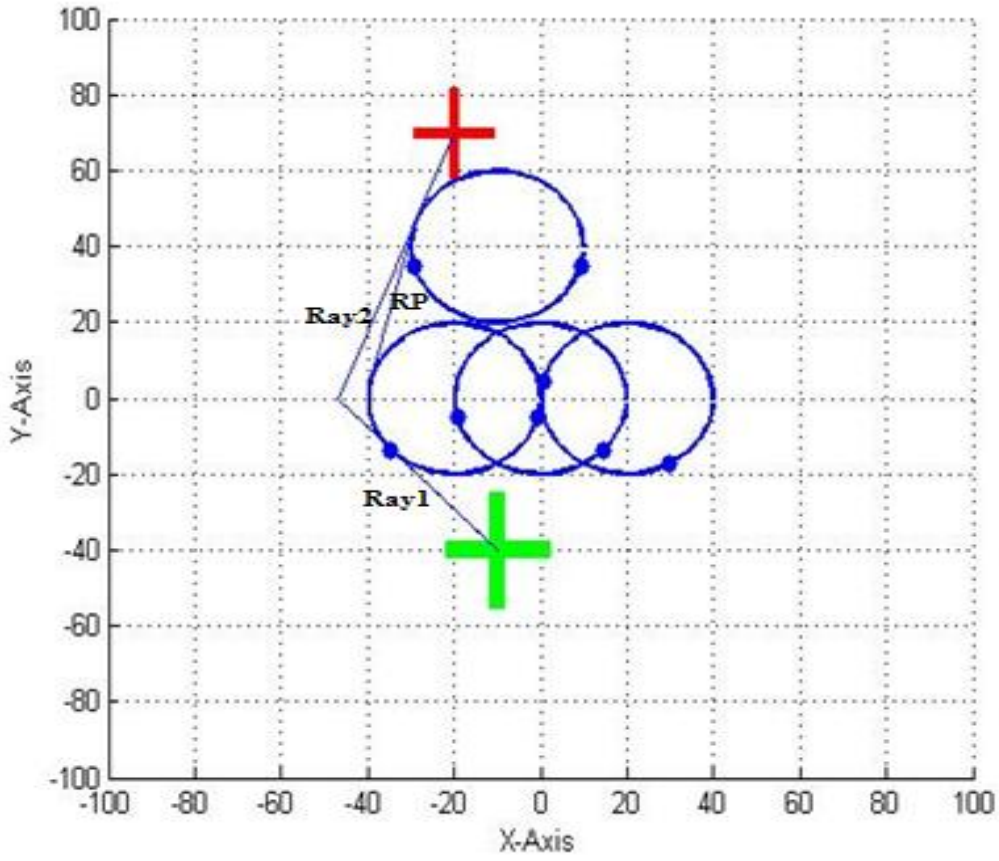

Fig. 5.2 Refined Path

Here in the fig 5.2, RP stands for refined path which appears to be the shortest path but it has one extra turning point when we compare it to the unrefined path. Now according to our new cost definition, the total cost of the refined path is 124.3016 + 20 = **144.3016**. This cost is clearly more than the unrefined cost which was **139.5**. So our algorithm will give us the unrefined path as our best path which has more distance but due to less number of turning points, it has lower cost than the refined path.

So this comparison will justify our debate that how critical are the turning points when it comes to MAV path planning.

5.2 Conclusions

In this thesis, we proposed a hybrid algorithm that combines VO method and DFS algorithm for the path planning of micro air vehicle (MAV) in static environment. MAV path planner needs to provide an optimal and power efficient path. Power efficient means it consume minimum power on its way towards goal. So to minimize this power as much as possible, researchers have done a lot of work. MAV path planner is being established based on the minimum number of turn points on its way to the goal as with each turn point actuator is switched on & off which consume the vehicle battery power and also the planner provides an optimal path.

In this thesis, we proposed a path planner which provides the best results in terms of optimality and power efficiency and also it is an intelligent search algorithm which tells us the best solution out of the refined and unrefined path based on the cost function which has been presented for the NAV navigation scheme. It may take a bit longer time for the algorithm to execute but it does not bother us as the planner is made for the path planning in static environment in which the environment is known and path planning can be done offline. There is always a tradeoff between algorithm's optimality, power efficiency and computational efficiency. For the case of offline path planning, we can compromise on computational efficiency.

Detailed simulations results show the effectiveness of this algorithm. Different scenarios are presented in results of the algorithm to show the robustness. This also shows that the proposed method can be easily applied for any environment. The results have shown that it works best and has given better results in terms of optimality and power efficiency.

5.3 Recommendations and Future Work

The proposed algorithm is a hybrid algorithm which has been implemented on software. In future, it would be implemented on the hardware as well. In this thesis, we have designed our algorithm for 2-D environment where we have taken circles as obstacles. In future, we will also look to add this hybrid approach in 3-D environment where we will replace circles as obstacles with spheres as obstacles.

As this work is done in static environment where the obstacles are fixed, the work or algorithm can be implemented in a dynamic environment where obstacles are moving which will give us more realistic results as normally we deal with flying objects in 3-D space.

This work can also be applied on unmanned ground vehicles (UGVs) and unmanned underwater vehicles (UUVs) as the algorithm is not only restricted to micro air vehicles.

References:

[1] Girish Chowdhary, John Ottander, Erwan Salaün, Eric Johnson, "1) Low Cost Guidance, Navigation & Control solutions for MAV in GPS denied areas," First Symposium on Indoor Flight Issues, International Aerial Robotics competition, 2009.

[2] Ryan Donovan Hurley, "Three-dimensional trajectory generation for flight within an Obstacle rich environment," a thesis presented to the graduate school of the university of florida in partial fulfillment of the requirements for the degree of master of science University of florida, 2009.

[3] Yohannes Ketema, Yiyuan Zhao, "Travel trajectory planning for micro air vehicles in winds," AIAA Guidance, Navigation, and Control Conference, Chicago, Illinois, 2009.

[4] Jeffery Saunders, Brandon Call, Andrew Curtis, Randal Beard, Timothy McLain, "Static and dynamic obstacle avoidance in miniature air vehicles," American Institute of Aeronautics and Astronautics.

[5] Javier Minguez and Luis Montano, "Nearness diagram (nd) navigation: Collision avoidance in troublesome scenarios," Transactions on Robotics and Automation, 2004.

[6] Stephen Griffiths, Jeff Saunders, Andrew Curtis, Tim McLain, Randy Beard Senior, "Obstacle and Terrain Avoidance for Miniature Aerial Vehicles," IEEE robotics and automation magazine.

[7] Steven LaValle, "Rapidly-exploring random trees: A new tool for path planning", Computer Science Department, Iowa State University, 1998.

[8] Sayan Ghosh and Abhishek Halder, Manoranjan Sinha, "Path planning for a fixed wing MAV in Fuzzy Quadtree Framework," 3rd US-European Competition and Workshop on Micro Air Vehicle Systems (MAV07) & European Micro Air Vehicle Conference and Flight Competition, Toulouse, France , 2007.

[9] Jeffery Saunders, Randal Beard, "Vision-based Reactive Multiple Obstacle Avoidance for Micro Air Vehicles,"American Control Conference Hyatt Regency Riverfront, St. Louis, MO, USA, 2009.

[10] Peng Cheng, Zuojun Shen, Steven LaValle, "Using randomization to find and optimize trajectories for nonlinear systems," In Proceedings of Annual Allerton Conference on Communications, Control, Computing, 2000.

[11] Rong Zhu, Xiaoying Guan, Zhaoying Zhou, "Collision-free Path Planning and Trajectory Generation for MAVs Flying in Urban Terrain", International Conference on Intelligent Robots and Systems, Beijing, China, 2006.

[12] Gargantini "An Effective Way to Represent Quadtrees", Communications of the ACM, Vol. 25, pp. 905-910, 1982.

[13] Paolo Fiorinit, Zvi Shillerz, "Motion planning in dynamic environments using the relative velocity paradigm," International conference on Robotics and automation 1(ICRA) , page 560-565, IEEE Computer Society Press, 1993.

[14] Robert Szczerba & Noah Ternullo, "Robust algorithm for real-time route planning", IEEE transactions on aerospace and electronic systems, vol. 36, 2000.

[15] Meng Bo-bo, Gao Xiaoguang, "UAV Path Planning based on Bidirectional Sparse A* Search Algorithm", Intelligent Computation Technology and Automation (ICICTA), Changsha, Hunan, China, 2010.

[16] Yao-hong Qu, Quan Pan, Jian-guo Yan, "Flight Path Planning of UAV Based on Heuristically Search and Genetic Algorithms," Industrial Electronics Society, 31st Annual Conference of IEEE, 2005.

[17] Zhuoning Dong, Zongji Chen, Rui Zhou, Rulin Zhang, "A Hybrid Approach of Virtual Force and A* Search Algorithm for UAV Path Re-Planning", International conference on Industrial Electronics and Applications (ICIEA), Beijing, China, 2011.

[18] Howie Choset, 'Principles of robot motion', theory, algorithm and implementations, Eastern economy edition, 2005.

[19] Meng Bo-bo, Gao Xiaoguang, "UAV Path Planning based on Bidirectional Sparse A* Search Algorithm" International Conference on Intelligent Computation Technology and Automation (ICICTA), Changsha, Hunan, China, 2010.

[20] Yao-hong Qu, Quan Pan, Jian-guo Yan, "Flight Path Planning of UAV Based on Heuristically Search and Genetic Algorithms," Industrial Electronics Society, 31st Annual Conference of IEEE, 2005.

[21] Usman Rafique and Kanwer Faraz, "Guidance Based Autonomous Parking Assistant", IEEE International Conference on Signal Processing, Robotics and Automation (ICSRA), China, 2010.

[22] Usman Rafique and Kanwer Faraz, "Modified Trajectory Shaping Guidance for Autonomous Parallel Parking", IEEE International Conference on Robotics, Automation and Mechatronics (RAM), Singapore, 2010.

[23] Zhang, Zhen, Wang, "UAV path planning method based on ant colony optimization", Proceedings of the IEEE International, Conference on Control and Decision (ICCD), China, 2010.

[24] Dong, Zhang, Chen, "The algorithms on three dimension route plan based on virtual forces" Journal of System Simulation, vol. 20, pp. 387-392, 2009.

[25] Jaryani"An effective manipulator trajectory planning with obstacles using virtual potential field method" Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (ICSMC), 2007.

[26] Alur, Courcoubetis, Henzinger, "an algorithmic approach to the specification and verification of hybrid systems" Lecture Notes in Computer Science, pp. 209-229, 1993.

[27] Randal Beard, Morgan Quigley, Sarita Thakoor, and Steve Zornetzer. "A new approach to observation of descent and landing of future mars mission using bioinspired technology innovations", AIAA Journal of Aerospace Computing, Information, and Communication, 2005.

[28] J. J. Kuffner and Steven LaValle. RRT-connect: "An efficient approach to single-query path planning", In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp 995–1001, San Francisco, 2000.

[29] Steven LaValle. "Rapidly-exploring random trees: A new tool for path planning", Computer Science Dept., Iowa State University, 1998.

[30] Javier Minguez and Luis Montano. Nearness diagram (nd) navigation: "Collision avoidance in troublesome scenarios", IEEE Transactions on Robotics and Automation, Vol 20, No. 1, 2004.

[31] Amato and Wu, "A randomized roadmap method for path and manipulation planning," in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 113–120, Minneapolis, 1996.

[32] Myung Hwangbo, James Kuffner, Takeo Kanade, "Efficient Two-Phase 3D Motion Planning for Small Fixed-Wing UAVs," IEEE International Conference on Robotics and Automation (ICRA), Roma, Italy, 10-14 April, 2007.

[33] James McMichael, McMichael.Francis, "MAV towards a new dimension in flight", Defense Advanced Research Projects Agency, USA.

Online Source:

a) J VanDomelen, http://blogs.mentor.com/jvandomelen/blog/tag/unmanned/, November, 2012.

b) D Bates, http://www.dailymail.co.uk/sciencetech/article-2138418/Mimicking-Mother-Nature-The-flying-robot-slow-land-like-bird.html, May 2012.

c) D Quick, http://www.gizmag.com/datron-scout-mav/18991/, June 2011.

d) G Warwick, http://www.aviationweek.com/Blogs.aspx?plckBlogId=%20Blog:27ec4a53-dcc8-42d0-bd3a-01329aef79a7&plckPostId=Blog:27ec4a53-dcc8-42d0-bd3a-01329aef79a7Post:1a54c3de-e542-42d7-b30a-829b984dea91, May 2008.

e) A Dirchel, http://en.wikipedia.org/wiki/File:Depth-first-tree.svg, March 2008.