

# COLLISION FREE PATH FINDING FOR MOBILE ROBOTS

## A DISSERTATION

*Submitted in partial fulfillment of the  
requirements for the award of the degree*

*of*

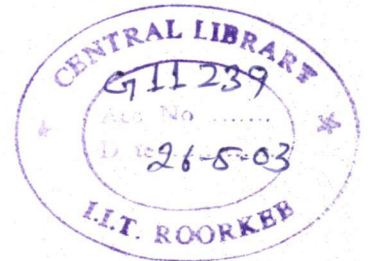
**MASTER OF TECHNOLOGY**

*in*

**INFORMATION TECHNOLOGY**

By

**RAMVATI PATHYA**



**ER & DCI  
NOIDA**

**DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING**

**ER & DCI - IIT (ROORKEE) CAMPUS**

**NOIDA - 201301 (INDIA)**

**February, 2003**

12

ROLL No. : 019035

621.380285  
PAT

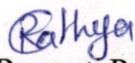
## CANDIDATE'S DECLARATION

I hereby declare that the work presented in this dissertation title **“COLLISION FREE PATH FINDING FOR MOBILE ROBOTS”**, in partial fulfillment of the requirements for the award of the degree of **Master of Technology in Information Technology**, submitted in **IIT, Roorkee – ER&DCI Campus, Noida**, is an authentic record of my own work carried out during the period from August 2002 to February, 2003 under the guidance of **Mr. P.N. Astya**, Software Consultant, New Delhi.

The matter embodied in this dissertation has not been submitted by me for award of any other degree of diploma

Date: 24-02-03

Place: Noida

  
(Ramvati Pathya)

## CERTIFICATE

This is to certify that the above statement made by the candidate is correct to the best of my knowledge and belief.

Date: 24-02-03

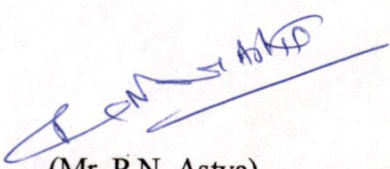
Place: Noida

Co-Guide : (Munish Kumar)

Project Engineer

ER&DCI, Noida.

Guide:

  
(Mr. P.N. Astya)

Software Engineer

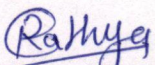
Sky InfoTech Pvt Ltd.

New Delhi -16

## ACKNOWLEDGEMENT

The work presented in this report would not have been completed without the guidance and support of many people. In first place, I would like to thank my guide, **Mr. P. N. Astya**, for his constant support, incredible enthusiasm and encouragement. I am grateful to **Mr. V. N. Shukla** Course coordinator, ER&DCI for his support and guidance. My sincere thanks are due to **Mr. Munish Kumar** for the encouragement and valuable suggestions he provided me with during the course of my work.

Most of all I would like to thank my family. My parents provided me a perfect environment for my studies and supported me throughout. Finally, I would like to extend my gratitude to all those persons who directly or indirectly helped me in the process and contributed towards this work.

  
(**Ramvati Pathya**)

# CONTENTS

---

<b>CANDIDATE'S DECLARATION</b>	<b>(i)</b>
<b>ACKNOWLEDGEMENT</b>	<b>(ii)</b>
<b>ABSTRACT</b>	<b>01</b>
<b>1. INTRODUCTION</b>	<b>02</b>
1.1 Objective	02
1.2 Scope	02
1.3 Outline of Dissertation	03
1.4 Organization of Dissertation	04
<b>2. LITERATURE SURVEY</b>	<b>05</b>
2.1 Robot Characteristics	05
2.1.1 Robot capability, components and intelligence	05
2.1.2 Robot Classification	05
2.1.3 Technical Approach	06
2.2 Mobility	06
2.2.1 Surfaces and Locomotion	06
2.2.2 Control	07
2.3 Robot Programming Languages	07
2.4 Aspect of Path Planning	07
<b>3. DESIGN AND ANALYSIS</b>	<b>09</b>
3.1 Problem Definition	09
3.2 Classifications of Path Planning	10
3.2.1 Complete and Incomplete Information	10
3.2.2 Local and Global Path Planning	10
3.3 Quadtree Data Structure	10
3.3.1 Advantages of quadtree	11
3.3.2 Disadvantage of quadtree	11

3.4	Space Efficiency of Quadtree	11
3.4.1	Best Case	12
3.4.2	Worst Case	13
<b>4.</b>	<b>IMPLEMENTATION OF QUADTREE APPROACH</b>	<b>19</b>
4.1	Algorithm for Converting Rasters to Quadtree	19
4.1.1	Block Decomposition	19
4.1.2	Processing of Image	21
4.1.3	Neighbor Adding	22
4.2	Quadtree Generation	25
<b>5.</b>	<b>IMPLEMENTATION OF NEIGHBOR FINDING TECHNIQUE</b>	<b>35</b>
5.1	Significance of Neighbor Finding Technique	35
5.2	Neighbor Finding Algorithms for Quadtree	35
5.2.1	World Map & Its Decomposition	36
5.2.2	Representation of Children	37
5.2.3	Adjacency Relation Table	39
5.2.4	Reflection Relation Figure	39
5.2.5	Neighbor Finding	40
<b>6.</b>	<b>RESULT AND CONCLUSION</b>	<b>43</b>
6.1	Result	43
6.2	Conclusion	45

## **REFERENCES**

## **APPENDIX-A**

## **APPENDIX-B**

## ABSTRACT

---

The problem of automatic collision-free path planning is central to mobile Robot applications. The basic problem of a mobile robot is that of navigation (moving from one place to another) by the coordination of planning, sensing and control. In any navigation the desire is to reach a destination without getting lost anywhere. The image of the region, which is full of obstacle, is scan converted and stored in the form of quadtree. And then this quadtree is used to find a collision free path for a mobile robot from any given source to any given destination. An approach to automatic path planning based on a quadtree representation is presented. Hierarchical path-searching methods are introduced, which make use of this multi-resolution representation, to speed up the path planning process considerably. The applicability of this approach to mobile robot path planning is discussed.

## INTRODUCTION

---

### 1.1 Objective

The objective of the project is first to scan convert the image of the region which is full of obstacle and store it in the form of quadtree. And then this quadtree is used to find a collision free optimal path for a mobile Robot from any given source to any given destination using Neighbor finding technique.

A flexible, intelligent robot is regarded as a general purpose machine system that may include effectors, sensors, computers and auxiliary equipment and like a human can perform a variety of tasks under unpredictable conditions. Development of intelligent robots is essential for increasing the growth rate of today's robot population in industry and elsewhere. Robotics research and development topics include manipulation, end effectors, mobility, sensing, adaptive control, robot programming languages and manufacture process planning.

### 1.2 Scope

Collision avoidance and robot path planning problems have emerged as a potential domain of robotics research because of its indispensable requirements in the field of manufacturing vis-à-vis material handling, such as picking-and-placing an object, loading/unloading a component to/from a machine or storage bins. It could be used in the hazardous places like coal mines, dense and deadly forest where humans may not reach safely and easily. It could be used for fire fighting too. Indeed mobile robots were and still are a very convenient and powerful support research in artificial intelligence oriented Robotics. They possess the capacity to provide a variety of problems at different levels of generality and difficulty in a large domain including perception, decision making, communication etc which all have to be considered within the scope of the specific constraints of Robotics as line computing, cost consideration, operating ability and reliability.



The main scope of this project includes:

- ❖ Quadtree representation of Image[1]
- ❖ Path planning using Neighbor finding technique[2]
- ❖ Obstacle avoidance

### 1.3 Outline of Dissertation

The problem of automatic collision free path planning is central to mobile robot applications. In this section we have presented the automatic path planning using the quadtree. Region representation is an important aspect of image processing. There has emerged a considerable amount of interest in the quadtrees. This is because of its hierarchical nature, which puts itself into compact representation. It is quite efficient for a number of traditional image processing operations such as computing parameters, finding genus of an image, computing centroids and set properties.

The conventional path planning algorithms can be divided broadly into two categories. The methods in first category make trivial (if any) changes to the representations of the image map before planning a path. The Regular grid search fall into this category. Though this method keep the representational cost to minimum, it's applicability to mobile robot navigation is limited. The Regular grid approach consists of subdividing an environment into discrete cells of a predefined shape (for example squares) and size and then searching an undirected graph based on the adjacency relationships between the cells and connecting the neighbors with four to eight arcs. Apart from the fact that discretization of space allows for control over the complexity of the path planning, it also proposes a flexible representation for obstacles and cost maps, and eases implementation. This approach has the advantage of being able to generate accurate paths, though are Efficient when environments contain large areas of obstacle free regions. Its path planning cost and memory requirements increases with its grid size, rather than with number of obstacles present. The multi-resolution (quadtree) addresses the later problem.

The methods in the second category make elaborate representation changes to convert to a representation, which is easier to analysis before planning the path. Free space methods, medial axis transform methods will come into this category. The

practical shortcoming of such methods is that path-planning cost is still very high, because of the representation conversion process involved.

The quadtree (multi-resolution) approach is a compromise between these two categories. The hierarchical nature of the quadtree data structure makes it a popular choice for other approaches because; It is adaptive to the clutter of an environment. As the image map being converted into a smaller number of nodes, the quadtree gains a lot of computational saving. In this approach we have concentrated on two aspects. A mobile robot will ordinarily negotiate any given path only once. This implies that **it is more important to develop a negotiable path quickly than it is to develop an “optimal” path, which is usually a costly operation.** A mobile robot should keep as far away from obstacles as possible.

#### 1.4 Organization of Dissertation

This project finds the collision free path for the mobile robot. The first chapter contains objective and scope of the dissertation under the heading of introduction. The literature survey is included in the second chapter, which further contains brief description, and the characteristics of mobile robot. Design and Analysis part is described in chapter in the chapter 3. It also describes the data structure and its advantages and disadvantages. The chapter 4 describes Implementation part of quadtree. It also explains the quadtree with example. The chapter 5 includes path finding with example and its implementation. The Result and Conclusion are contained in chapter 6. The appendix A contains the class and methods of the quadtree. The appendix B contains list of all figures with their page numbers.

## LITERATURE SURVEY

---

The objective is to survey the state-of-the-art of intelligent robots. The terms “robot” and “artificial intelligence” classify the intelligent robots according to their level of intelligence and a discussion is made about the various aspects of the intelligent robots.

### 2.1 Robot Characteristics

#### 2.1.1 Robot capability, components and intelligence

A robot is a general purpose machine system that, like a human, can perform a variety of different tasks under conditions that may not be known a priori. A robot system may include any of the following major functional components:

- a) Effectors- “arms,” “hands,” “legs,” “feet;”
- b) Sensors- contact, non-contact;
- c) Computers - top controller, lower level controller (including Communication channels);
- d) Auxiliary equipment - tools, jigs, fixtures, tables, pallets, conveyors, part feeders etc.

#### 2.1.3 Robot Classification

Like human intelligence, robot intelligence is variable. This observation is compatible with the Japanese classification of Industrial robots into five categories :

- a) A slave manipulator teleoperated by a human master;
- b) A limited sequence manipulator (further classified into “hard-to- adjust” and “easy-to-adjust” categories )
- c) A teach-replay robot;
- d) A computer controlled robot
- e) An intelligent robot

### **2.1.2 Technical Approach**

The technical approach to intelligent robot development should be based on application of artificial intelligence (AI) techniques to robotics Under four engineering constraints:

- a) High reliability—the robot must be robust; if it fails, it should be able to detect the error and recover from it or call for help;
- b) High speed—the robot should be able to perform its functions as fast as necessary.
- c) Programmability—the robot should be flexible (able to perform a class of different functions for a variety of tasks), easily trainable (for new tasks or modification of old one's) and intelligent (able to perceive problems and solve them);
- d) Low cost—the cost of the robot should be low enough to justify its application.

## **2.2 Mobility**

Robot mobility is needed for a wide variety of robot functions in unstructured environment, such as mining, military operations and aid to the handicapped. Some robot mobility are described below.

### **2.2.1 Surfaces and Locomotion**

The mechanism for robot locomotion depends strongly upon the type of surface the robot must be able to move on. Indoor surfaces include floors, ramps, stairs, and cluttered environments. Outdoor surfaces include roads, Smooth ground, terrain with holes and ditches and terrain with obstacles.

Robot locomotion is realized with wheels, tracks and legs. Wheels perform well if the terrain is not rough and the traction is sufficient. Tracks perform well if the terrain slope is not too high or no major obstacles are encountered.

Legged vehicles have been developed for robot mobility in rough terrain, where wheels and tracks are useless. The major issues are stability, strength, speed and control.

## **2.2.2 Control**

A major research issue in the robot mobility is autonomous control, which includes motor control, sensing, navigation, communication, obstacle avoidance, and task performance

## **2.3 Robot Programming Languages**

The main goal of using a robot language is to facilitate the programming of a robot system for a new task or modification of an old one. To achieve this goal, a robot language provides the user with high-level programming capabilities. These capabilities are implemented by means of a language processor and a robot controller- the processor accepts and checks the user statement and translates them into commands for the controller; the controller then generates lower-level commands for the corresponding device.

There are different commercially robot language is available like AML(IBM Corporation), HELP (General Electric company) etc. An alternative approach is to utilize the general programming capabilities of a high-level language(c, c++, pascal or ADA) which could reduce the cost and enhance the portability of the robot software as well as improve programming flexibility.

## **2.4 Aspect of Path Planning**

The mobile robots were found to be appealing for various artificial intelligence techniques. In these projects the emphasis was more on the decision-making or motion planning and not so much on motion control. A definition of a mobile robot can be derived from that of the classical robots: "A robot vehicle capable of self propulsion and (re) programmed locomotion under automatic control in order to perform a certain task. The keywords are that a robot is programmable to execute a variety of tasks and that does so under automatic control (as opposed to human control). There is large diversity of mobile robots. Fig-2.1. illustrates a division based on vehicle characteristics.

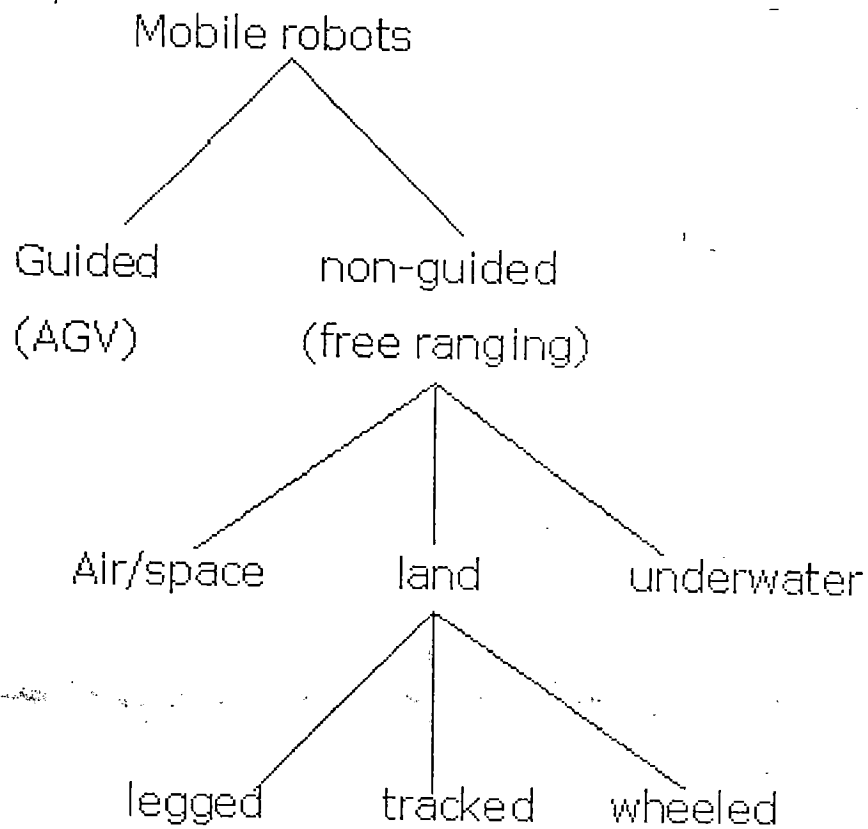


Fig. 2.1 The Division of Mobile Robot

The some of main subjects of research in the field of mobile robots today are:

- ❖ Path planning and trajectory planning.
- ❖ World modeling (environmental mapping)
- ❖ Position and course estimation(localization)
- ❖ Obstacle avoidance

## DESIGN AND ANALYSIS

---

### 3.1 Problem Definition

The basic problem of a mobile robot is that of navigation[5]: moving from one place to another by a coordination of planning, sensing and control. In any navigation scheme the desire is to reach a destination without getting lost or crashing into anything. Put simply the navigation problem is to find a path from start (S) to target (T) and traverse it without collision.

Navigation may be decomposed into three sub-tasks: mapping and modeling the environment; path planning and selection; and path following and collision avoidance. The relationship between these tasks is shown in fig 3.1

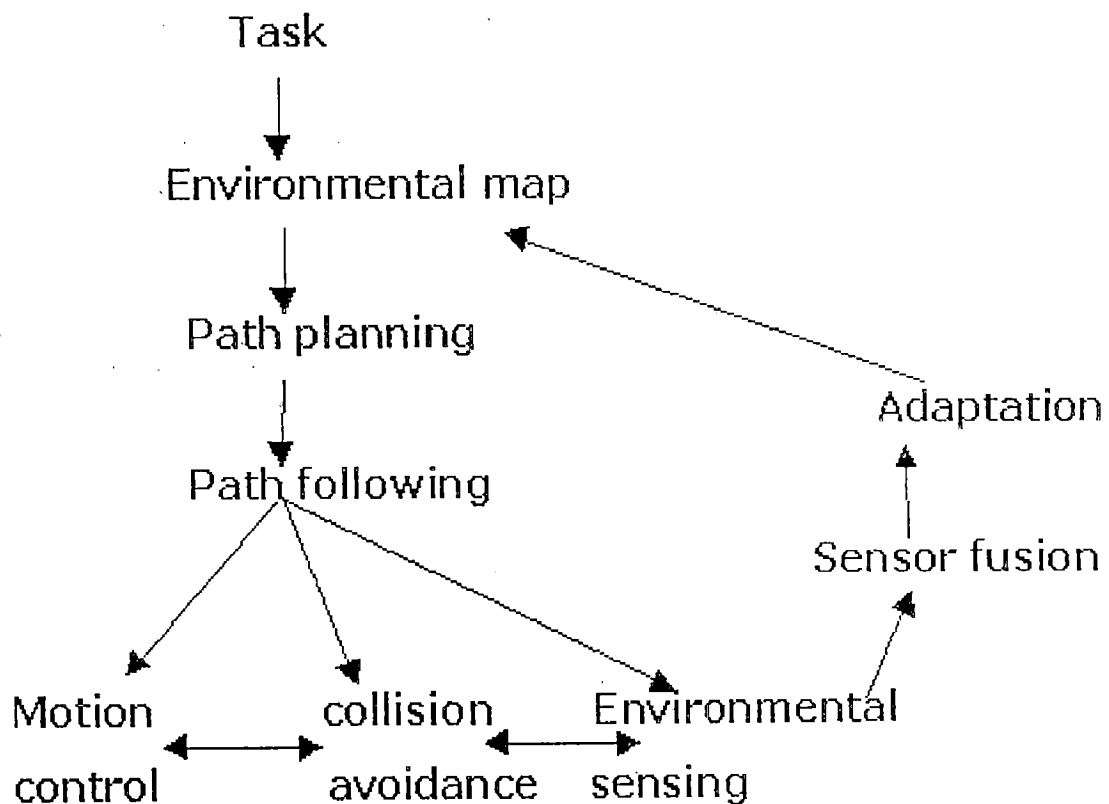


Figure- 3.1 The relationship between different tasks

## **3.2 Classifications of Path Planning**

### **3.2.1 Complete and Incomplete Information**

A path planning algorithm for a mobile robot has as input data two coordinates\_(the actual robot position start and the desired robot position target) and must yield output a possible path between these points. Related to the priori knowledge about the environment, in which the robot is supposed to travel, there are two different cases:

1. The environment is completely known.
2. The environment is completely unknown.

The path planning algorithm for the former case are called path planning with complete information, while for the latter, the problem of finding a path between a start location and a target location is called path planning with incomplete information or path planning with uncertainty.

### **3.2.2 Local and Global Path Planning**

The path planning problem can be divided into global path planning and local obstacle avoidance. Global path planning requires a pre-learned model of the domain, which may be some what simplified description of the real world and might not reflect recent changes in the environment. This global model must provide the planning algorithm with a network of landmark points which are connected by simple local movements.

A local navigation system carries out the steps in the global plan, maintaining an estimate of the robot's position with respect to the global model and planning local paths as needed to avoid unexpected obstacles.

## **3.3 Quadtree Data Structure**

There are different ways to represent the spatial information using various data structures. It has been proven that the efficiency of the different kinds of path planning algorithm developed till present is closely related to the type of data structures used to store the map.

In this work the quadtree data structure is chosen because of it's advantages over others.



### 3.3.1 Advantages of quadtree

In Quadtree approach the region representation based on recursively subdividing an image into quadrants until blocks are obtained that are either entirely contained in the region or entirely disjoint from it. The advantages of this quadtree data structure are:

1. It provides a variable resolution encoding of a region according to the sizes and number of 'maximal non-overlapping blocks' that are contained in it.
2. It provides topological relations involving block adjacency, connectedness and borders are easily computable.
3. The quadtrees provides a relatively compact representation [1] for many regions.

### 3.3.2 Disadvantage of quadtree

The quadtrees are shift variant i.e., two identical regions differing only by a translation in an image may have quite different quadtrees e.g., a  $2^m$  by  $2^m$  square region may be represented by a single node or as many as  $O(2^m)$  nodes depending on its position in the image.

### 3.4 Space Efficiency of Quadtree

The quadtree of a  $2^n$  by  $2^n$  binary image is defined recursively as follows let the root of the quadtree be associated with the entire image; the level of the root is  $n$ . If the  $2^k$  by  $2^k$  block of the image associated with an arbitrary node at level  $k$  doesn't consist of either all of 1's or 0's, then subdivide the block into four  $2^{k-1}$  by  $2^{k-1}$  quadrants and associate these sub blocks with four nodes designated as the four sons of the given node; each son is defined to be at level  $k-1$ .

Each node in a quadtree contains six fields. Five fields contains pointers to a node's father and four sons and the sixth field describes the block of the image associate with the node-its value is black if the block contains only pixels in the region (i.e., all 1's), WHITE if the block contains no pixels in the region (i.e., all 0's) or gray if it contains pixels of both types ( 0's and 1's) thus all non leaf nodes are gray and all leaf nodes are either BLACK or WHITE.

Let  $N$  be the total number of nodes in a quadtree,  $B$  the number of black nodes,  $W$  be the number of white nodes and  $G$  th number of gray nodes. Thus  $N=B+W+G$ .

The following relations are true for any quadtree:

$$N=4G + 1 = (4(B+W)-1)/3,$$

$$B=3G-W+1,$$

$$W=3G-B+1,$$

$$G=(B+W-1)/3$$

There are three different cases occurs i.e., best, worst, average[3] values for the variables N,B,W and G for which will be computed as a function of n and m for a quadtree representation of a  $2^n$  by  $2^n$  image in which a single rectilinearly ordered  $2^m$  by  $2^m$  region occurs. Each pixel will be referred by its matrix co-ordinates in the  $2^n$  by  $2^n$  input array. Thus the upper-left corner pixel has co-ordinates (0,0). The position of the region in the array will be specified by the position of its upper left corner.

### 3.4.1 Best Case

The Best Case [3] occurs when the region can be represented by a single black node at level m. As shown in the fig.3.2.1 this implies that there are no nodes at level 0 through m-1, four nodes at each of levels m through n-1 and one node at level n. Thus  $N = 4(n-m)+1$ . Of these nodes, one gray node occurs at each of levels m+1 through n and three white nodes occurs at each of levels m through n-1. Thus  $B=1$ ,  $W=3(n-m)$  and  $G=n-m$ .

This case occurs when ever the position (r,c) of the region is such that  $r \bmod 2^m = c \bmod 2^m = 0$ ; thus only  $O(n-m)$  nodes are required when the region is any of  $2^{n-m+1}$  positions in the image. At any other position G would increase since there would now exist black nodes at levels less than m and there must be at least one gray node at each level except the lowest in any quadtree, hence  $G \geq n-m$ . We know  $n=4G+1$ ; thus there can be no position of the region which results in a quadtree containing fewer than  $4(n-m)+1$  nodes.

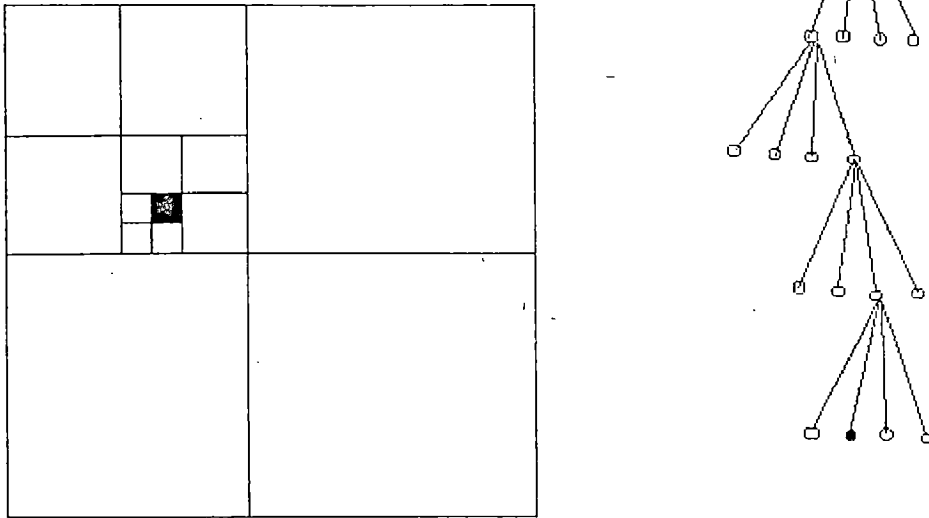


Fig. 3.4.1. Best case position of a  $2^m$  by  $2^m$  region in a  $2^n$  by  $2^n$  binary image(a) Block decomposition (b) Quadtree Representation

### 3.4.2 Worst Case

The worst case [3] occurs when the obstacle is presented as per the figure. In this case the region is at position  $(r,c)$  such that  $r \bmod 2^m = c \bmod 2^m = 1$  i.e., the region is shifted to the right and down 1 pixel from the best case position. In this case there is a border of level 0 black blocks along the left and top sides of the region. Inside these two borders there is a row and column of level 1 black blocks; this filling in along the left and top side continues until a single black block at level  $m-1$  occurs. Finally a band of black blocks at level 0 fills in the right and bottom borders of the region. The map and quadtree structure is represented in the figure 3.4.2.

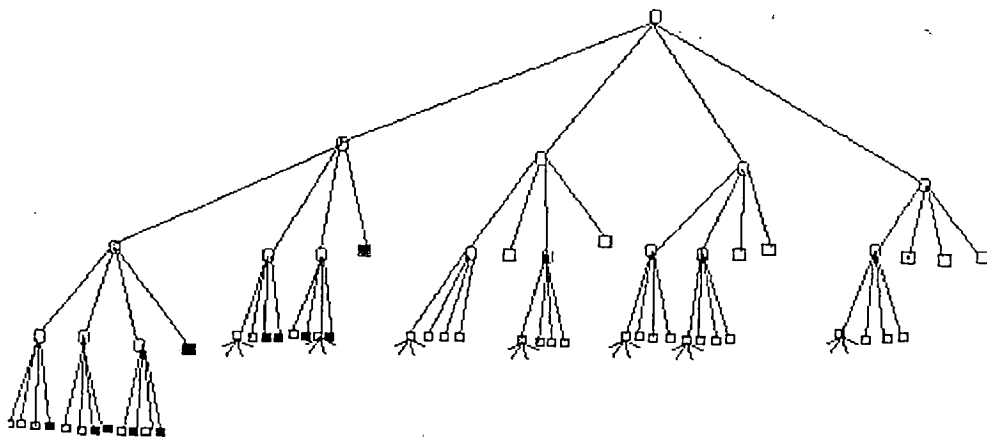
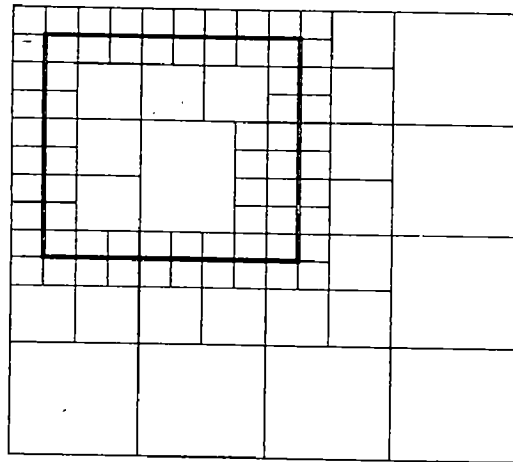


Fig 3.4.2. Worst case position of a  $2^m$  by  $2^m$  region. (a) Block decomposition of a 16 by 16 region at position (1,1). (b) Portion of the quadtree representation of (a).

3.3 DATA FLOW DIAGRAM

3.3.1 DFD 0 LEVEL OF PATH PLANNING SYSTEM

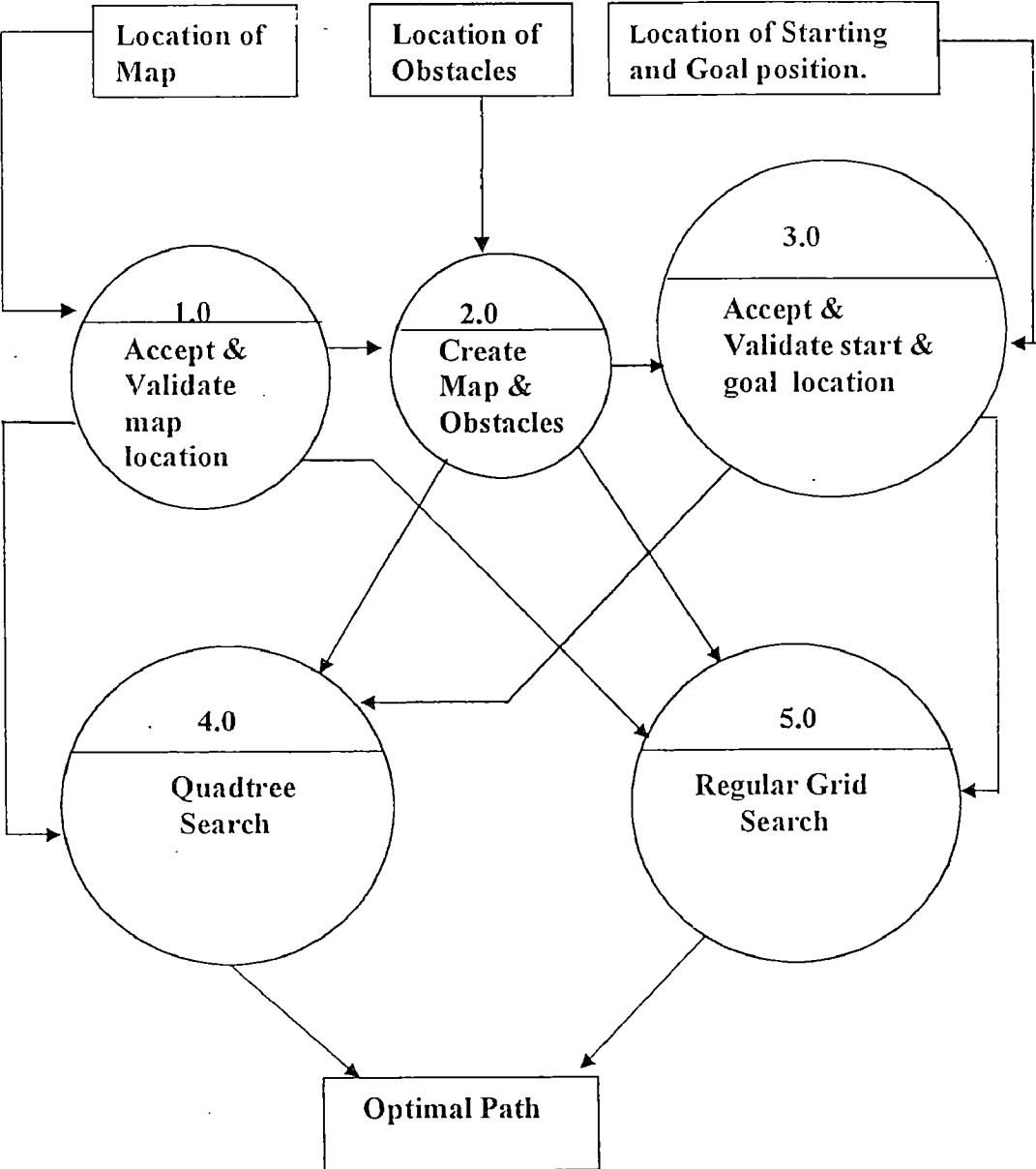


Fig. 3.2

### 3.3.2 DFD 1 LEVEL OF PATH PLANNING SYSTEM

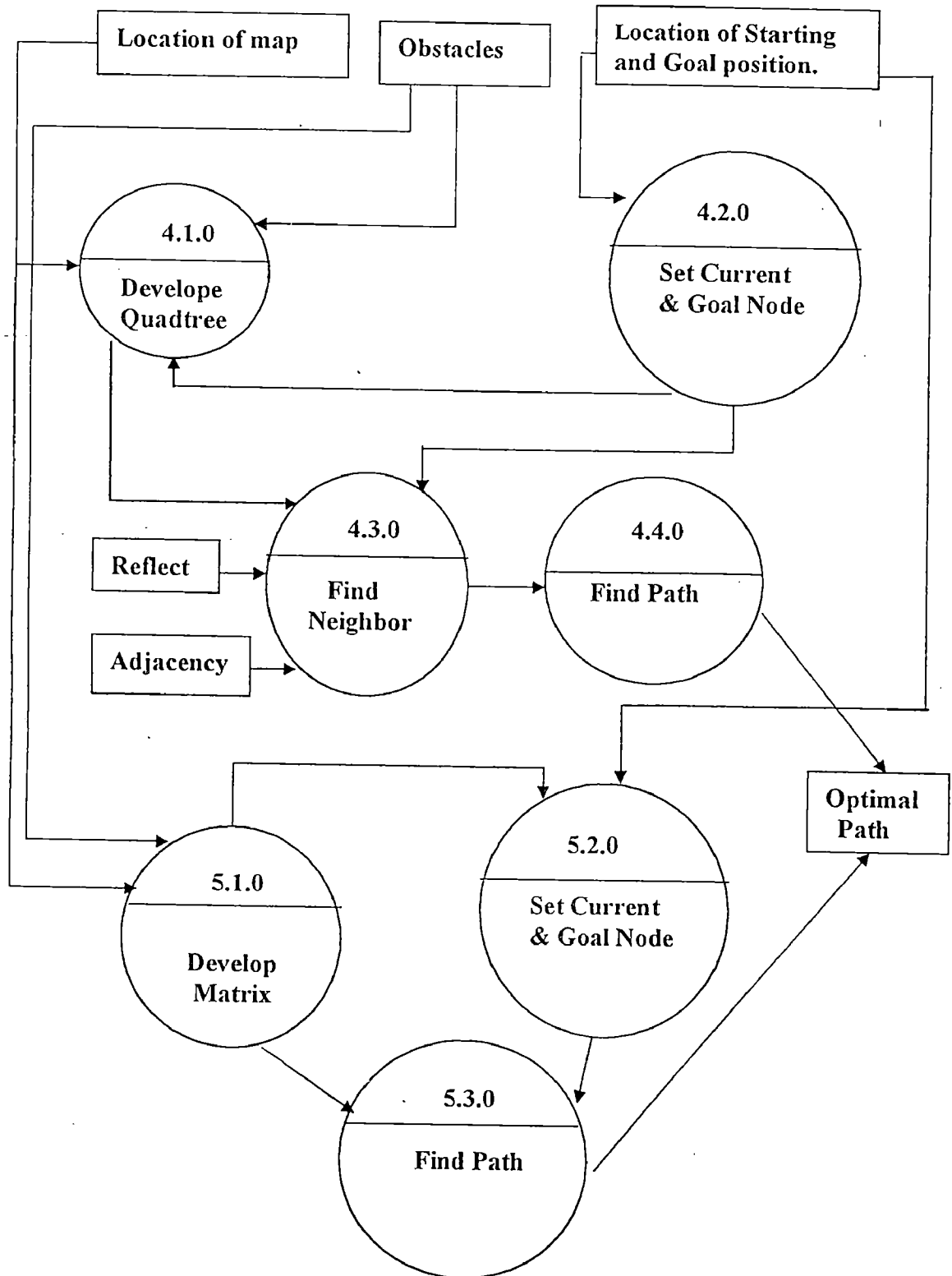


Fig. 3.3

### 3.3.3 DFD 2 LEVEL OF PATH PLANNING SYSTEM

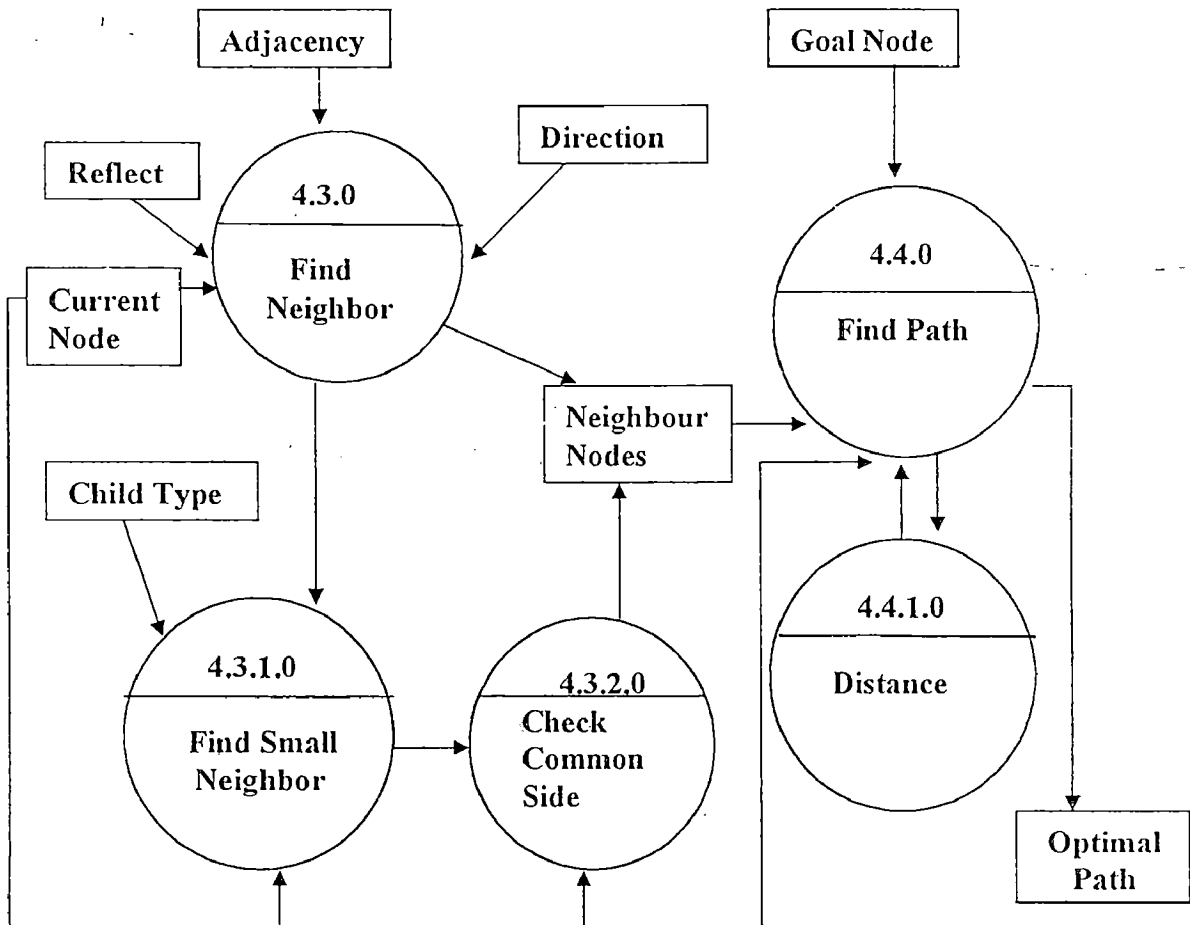


Fig. 3.4

## IMPLEMENTATION OF QUADTREE APPROACH

---

---

### 4.1. Algorithm For Converting Rasters to Quadrees

This algorithm[1] is used for obtaining in-core quadtree representation given the row-by-row description of a binary image. The quadtree[1],[6] is a compact hierarchical representation, there by facilitating search.

Assume that the image is a  $2^n \times 2^n$  array. Each row of the image is thus a bit string of length  $2^n$ . The quadtree[1] is an approach to image representation best on successive subdivision of the image into quadrants. In essence we repeatedly subdivide the array into quadrants, sub quadrant, until we obtain blocks (possibly single pixels) which consist of entirely of either 1's or 0's. This process is represented by a tree of out-degree 4 in which the root node represent the entire array, the four sons of the root node represents in order of the NW, NE, SW and SE quadrant and the terminal node corresponds to those blocks of the array for which no further subdivision is necessary.

#### 4.1.1 Block Decomposition

The block decomposition is shown in the fig 4.1.1.



1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

Fig. (a)

1	2	3	4		6	7	
A	10	11	12	13	14	15	
B		C		21	22	23	D
				29	30	31	32
				37	F	38	G
	E			45	46		
				H		I	

Fig. (b)

Fig 4.1.1. An image, its maximal blocks, and the corresponding quadtree. Blocks in the image are shaded (a) sample image (b) block decomposition of the image in (a). (c) quadtree representation of the blocks in (b).

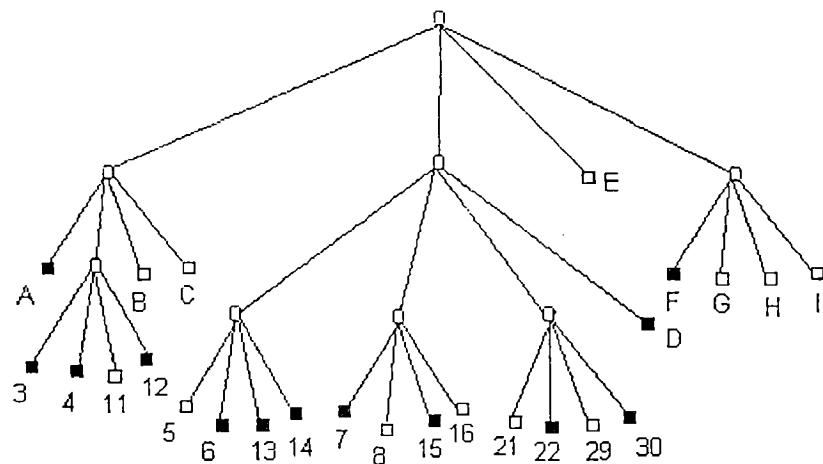


Fig. (c)

In general BLACK and WHITE square nodes represents blocks consisting entirely of 1's and 0's respectively. Circular nodes, also term GRAY nodes, denote non-terminal nodes.

### 4.1.2 Processing of Image

We assume that the image contains an even number of rows, if the image contains an odd number of rows, then it is presumed that one extra row of WHITE has been added.

For an odd-numbered row, the tree is constructed by processing the row from left to right adding a node to the tree for each pixel. The example is given below shows the construction of a quadtree corresponding to the first four pixels of the binary image of figure-4.1.1.(a) (i.e. pixels 1,2,3 and 4). This is done by invoking a procedure described below, called ADD-NEIGHBOR[5]. As the quadtree is constructed, non-terminal nodes must be added. As the quadtree is constructed, non-terminal nodes must be added. Since we wish to have a valid quadtree after processing each pixel, whenever we add a non-terminal node we also add, as is appropriate, 3 or 4 WHITE nodes as its sons.

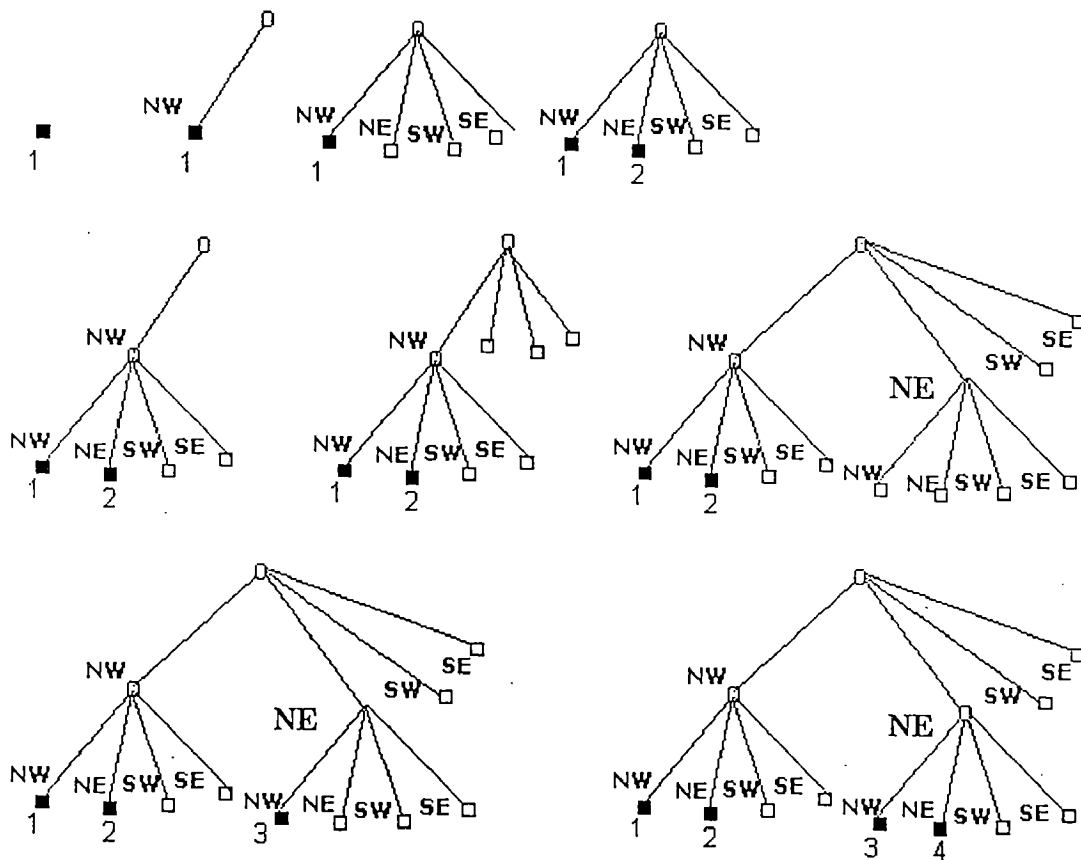


Fig. 4.1.2.(a, b, c, d, e, f, g, h, i) Intermediate trees in the process of obtaining a quadtree for the first part of the first row in fig. 4.1.1 (a).

### 4.1.3 Neighbor Adding

We now describe ADD-NEIGHBOR[5] more formally. Adding a neighbor of a node, say P, in a specified direction consist of traversing ancestor links until a common ancestor of the two nodes is found. Once the common ancestor is found, we descend along a path that retraces the previous path with the modification that each step is a reflection of the corresponding step about the axis formed by the common boundary between two nodes. For example when attempting to add the eastern neighbor of node 3 (i.e., node 4) in figure 4.1.2(h), node X is the common ancestor and the eastern edge of the block corresponding to node 3 is the common boundary. Thus having ascended a north-west link to reach node X, reflection about the eastern edge of node 3's block causes us to descend to the NE son of X. If a common ancestor doesn't exist then a non-terminal node is added with its 3 remaining sons being WHITE [example for Fig 4.1.2.(c) and (f) ] once the common ancestor and 3 sons have been added, we once again descend along retrace path modified by reflecting each step about the access formed by the boundary of the node whose neighbor we seek. During this descend, a WHITE node is converted to a GRAY node and four WHITE sons are added [e.g., Fig 4.1.2(g) ]. As a final step, the terminal node is colored appropriately (e.g., Fig. 4.1.2(d) and (h) ). In example Fig.4.1.2. (a),(b)-(d),(e)-(h) and (i) are snapshots of the quadtree construction process for the nodes corresponding pixels 1,2,3 and 4 respectively of Fig.4.1.1(a).

Even-numbered rows required more work since merging may also take place. In particular, a check for a possible merge must be performed at every even numbered vertical position (i.e., every even-numbered pixel in a row) once a merge occurs, we may have to check if another merge is possible. We wish to maintain the position in the tree where the next pixel is to be added as well as the next row. Therefore, prior to attempting a merge, a node corresponding to the next pixel in the image is added to the quadtree [e.g., node 1] is added to the quadtree in Fig. 4.1.3 (which is given below) prior to attempting to merge nodes 1,2,9 and 10 of Fig.4.1.1(a)]. Similarly we precede the processing of each even-numbered row by adding to the quadtree a node corresponding to the first pixel in the next row [e.g., the addition of the node 17 to the tree of the Fig.4.1.3 prior to processing row 2 of Fig.4.1.1(a)].

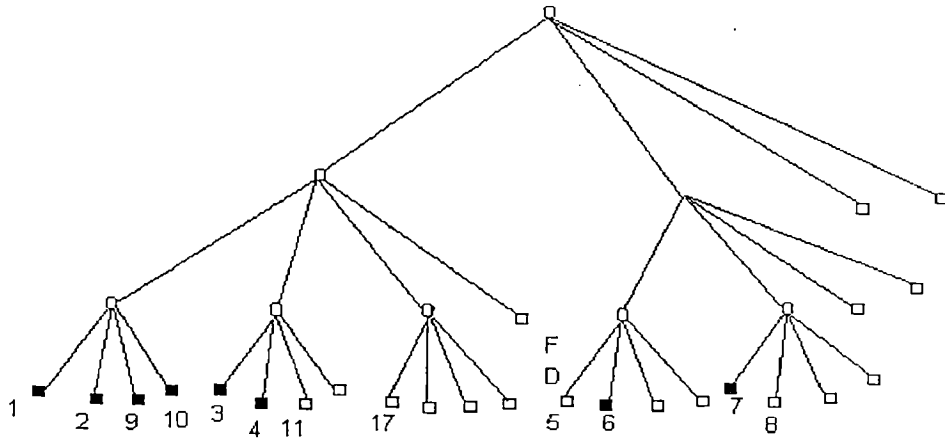


Fig. 4.1.3. Quadtree prior to merging nodes 1,2,9 and 10

In this algorithm the execution time has a time complexity proportional to the number of pixels in the image. This is obtained by examining the number of nodes that are visited as the tree is constructed. In particular, the number of nodes visited by the merging process is bounded by the number of pixels in the image. While the remaining part of the tree construction process visit four times as many nodes as there are pixels in the image. The algorithm is also space wise efficient in that merging is attempted whenever possible. Thus, after processing each pixel in a given row the resulting quadtree contains a minimum number of nodes. Then the algorithm is one dimensional in the sense that it processes the image a row at time.

The example of the algorithm represented in Fig.4.1.1(a) and Fig.4.1.2(b) is the corresponding block decomposition and Fig.4.1.1(c) is its quadtree representation. Fig.4.1.2(a)-(i) shows the steps in the construction of the quadtree corresponding to the first part of first row and Figs.4.1.4 and 4.1.5 shows the resulting trees after the first and second rows have been processed.

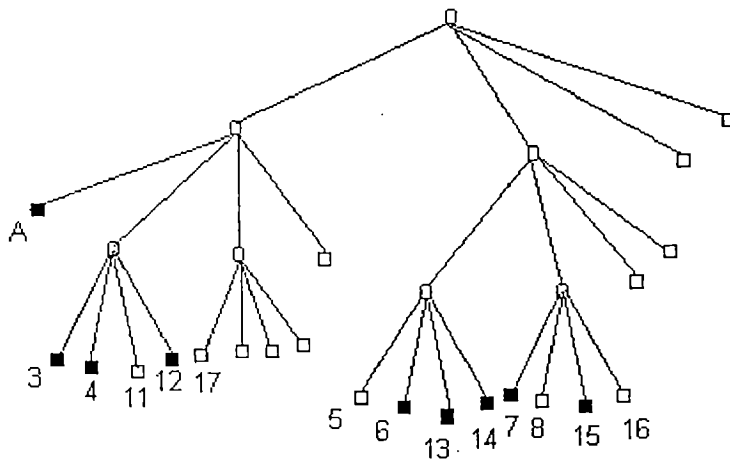


Fig. 4.1.4. Quadtree after processing the first row in Fig. 4.1.1.(a).

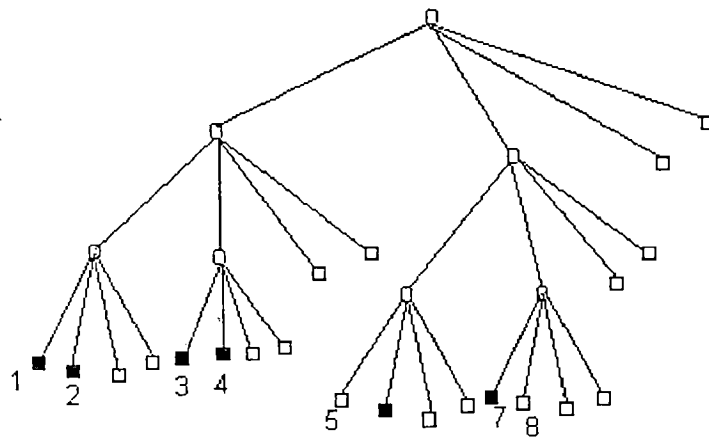


Fig. 4.1.5. Quadtree after processing the second row in Fig. 4.1.1.(a).

## 4.2 Quadtree Generation

The quadtree is a tree, in which each node in the tree will have four children nodes. We can represent the given 2-dimensional image map into the form of quadtree by recursive decomposition. Each node will represent a square block of the given image Map. The size of the square block may be differed from node to node. The nodes in The quadtree can be classified into three groups.

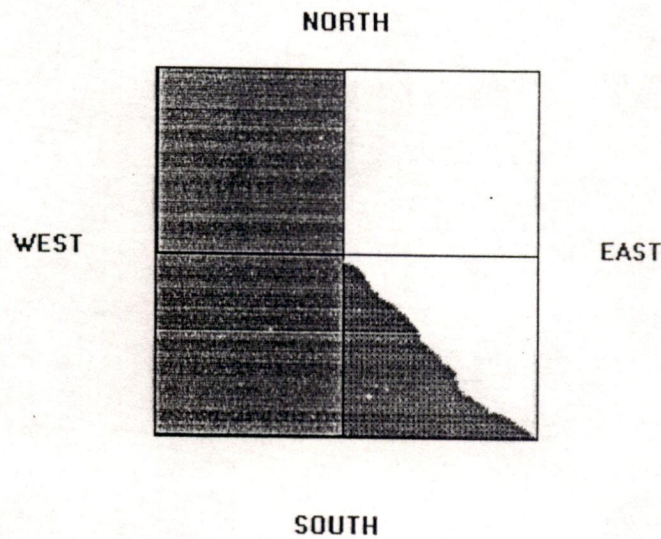
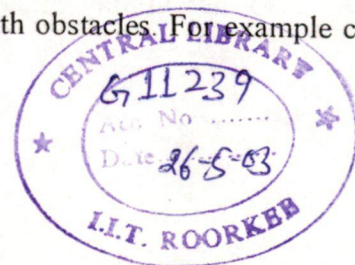


Fig-4.2.1 Representation of a simple 2D world map, in which gray area representing obstacle region, and white core presenting obstacle free region

1. Free nodes.
2. obstacle nodes.
3. mixed nodes.

A free node is a node , which has no obstacles in the square region represented by it. An obstacle node is a node ,whose square region is totally filled with obstacles. A mixed node's square regions partially filled with obstacles. For example consider the image map given in fig. 4.2.1.



In the decomposition of above 2-dimensional image map above ,it is divided into four sub square regions(four children),namely NW, NE, SW, and SE according to directions. The square region NW,SW are fully occupied with obstacle(gray region)

So they will come into “Obstacle mode” category. Square region NE is not having any obstacle region in it. So it will be under” free node” category. The square region SE is partially filled with obstacle and remaining spaces is an obstacle-free region. So it will come under “mixed mode” category. The quad tree representation of the above decomposition is shown in figure 4.2.2

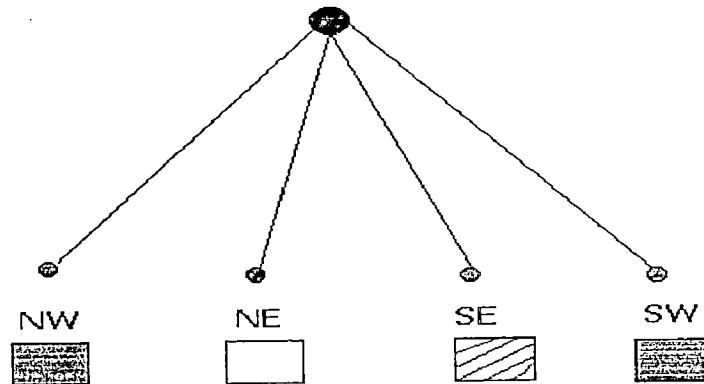


Fig 4.2.2 The decomposition of 2D World map in Fig 4.2.1 into quadtree nodes.

The type of each node is represented by small box , with different fill pattern . Grey fill for obstacle node , Empty fill for free node ,and hatched line fill for gray node. The data structure needed for a node is represented as given below in Programming language C syntax.

### **Struct node**

```
{  
    node*pointer_to_child1;  
    node*pointer_to_child2;  
    node*pointer_to_child3;  
    node*pointer_to_child4;  
    node* pointer_to_parent_node;  
    int node_status;  
};
```

The first four fields are pointers for four children. The fifth field is a pointer to the parent node. The pointer to the parent node of root will be NULL. The pointers to children for leaf nodes are also equal to NULL.

From the given 2-dimensional image map, we will generate the quadtree by dividing the map into four equal sized square quadrants. For both free nodes, and obstacles nodes there is no need to decompose to further. They will remain as leaves of the quadtree. Each mixed node must be recursively, subdivided into four sub quadrants, which form children of that node. This subdivision procedure is repeated continuously until either of the conditions below is satisfied.



1. The node is either a free node or an obstacle node.
2. The size of the square region represented by the Square region represented by the node's children is less than the size of the mobile Robot.

The complete example for generating the quadtree for a given 2-Dimensional image map given below. The image Map is given in figure-4.2.3

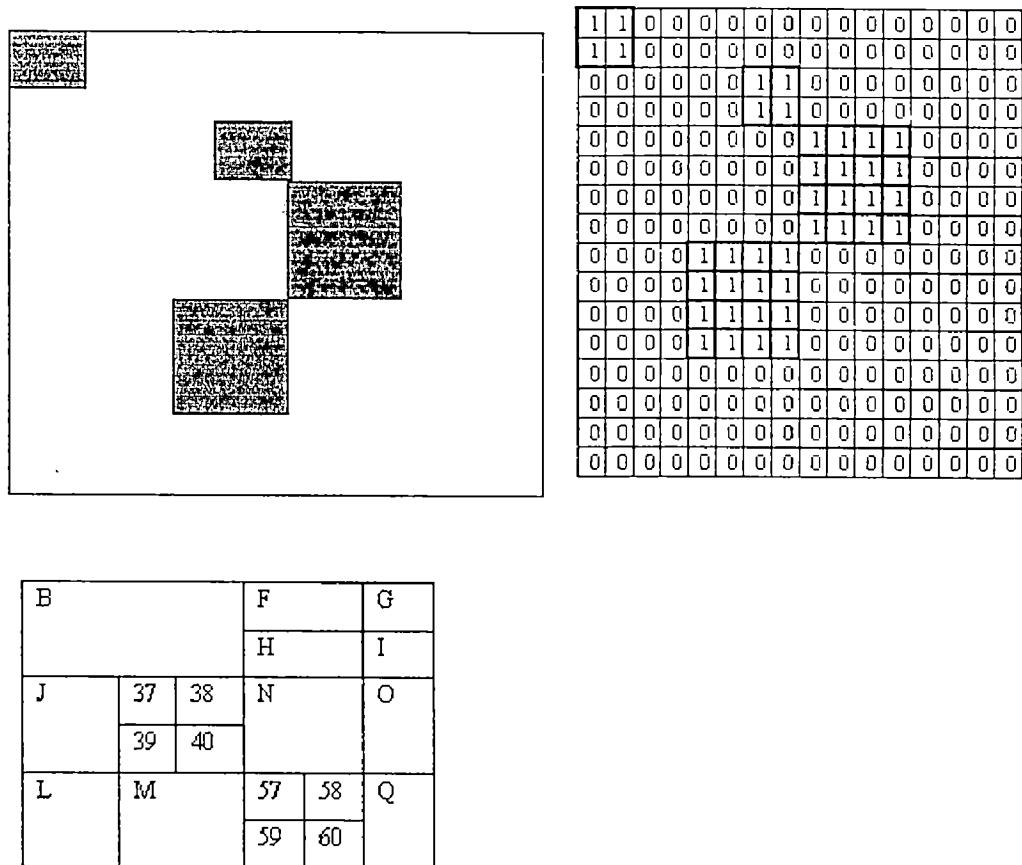


Fig 4.2.3. Representation of a 2D world map ,in which gray area representing obstacle region , and white color area representing obstacle free region 1. A region, 2. binary array 3. maximal blocks.

The gray area in the figure are obstacle regions. For simplicity we have considered the square obstacle regions only. In the first stage of its decomposition, the image map is divided into four square regions of equal size as shown below. The image map itself is the root of the quadtree. The name given to it is A.

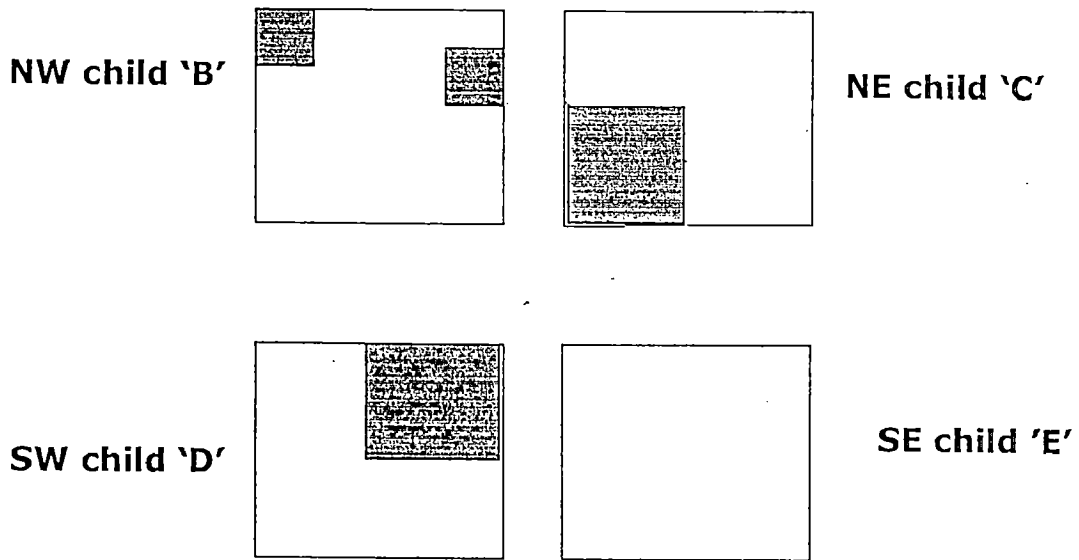


Fig-4.2.4: The decomposition of the 2D world map represented in Fig-4.2.3.

In the above decomposition the child 'E' has no obstacle in its square region. So it is a free node and will not be decomposed further. It will remain as a leaf node of the quad tree. The remaining children 'B', 'C', and 'D' are having some free space and some obstacle region, they will come under mixed nodes. The quadtree developed up to this stage is as follows, the small square box under the each node is representing the status of the node. The white box is free node. The gray box represent the obstacle node. The box with hashed lines represents the mixed node.

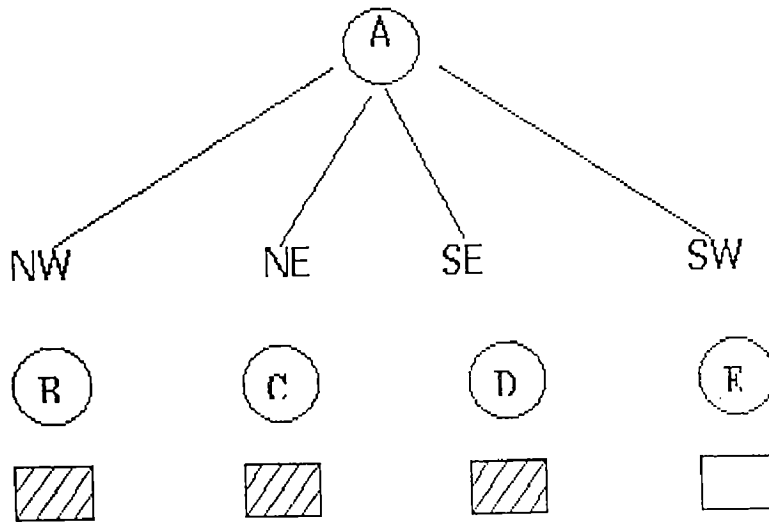
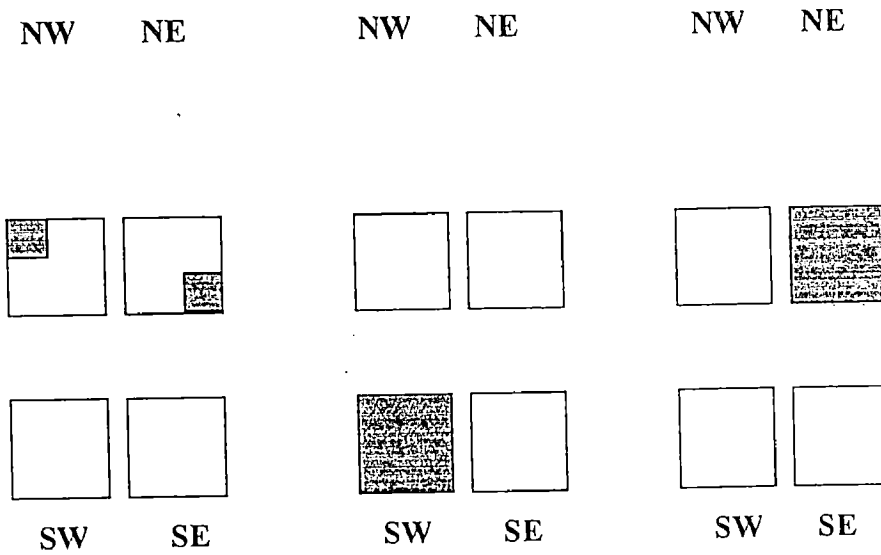


Fig-4.2.5. Quadtree representation of the decomposition .

They have to be decomposed further and the decomposition is shown in the Fig-4.2.6.



NODE 'B'

NODE 'C'

NODE 'D'

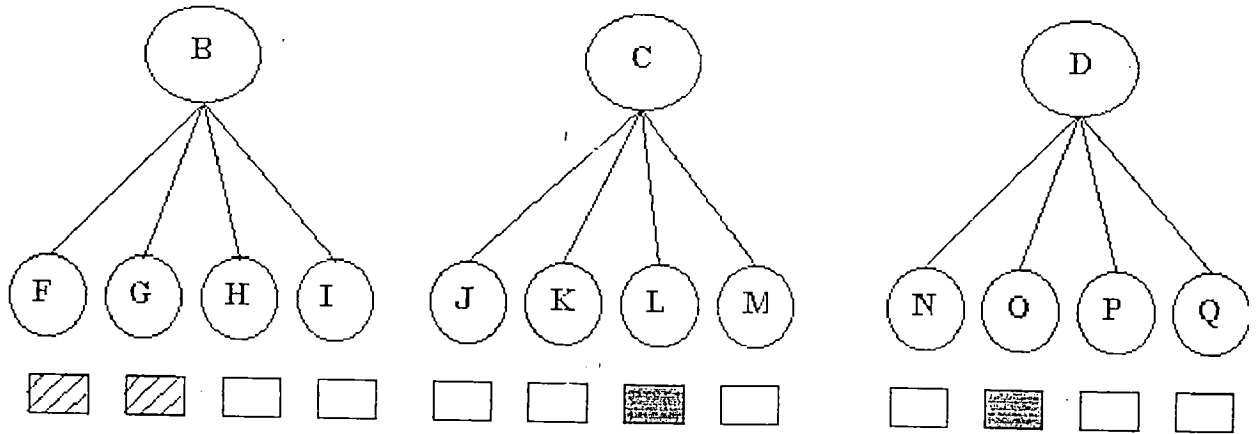


Fig-4.2.6. The second stage decomposition of the 2D world map given in Fig-4.2.3

As per the node status the nodes 'F' and 'G' are mixed nodes and will be expanded further.

NW NE



SW SE

NW NE



SW SE

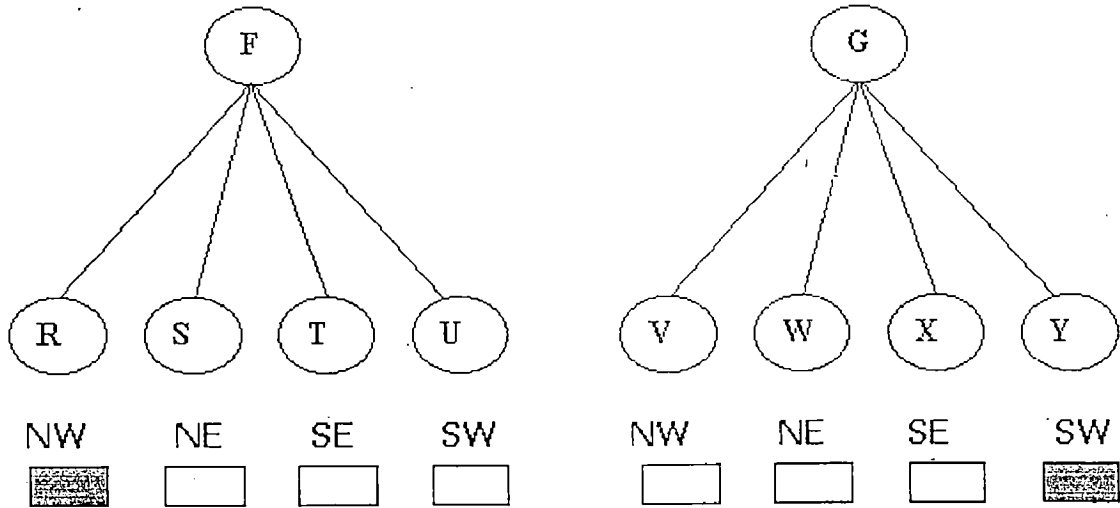


Fig-4.2.7 : Further decomposition of remaining mixed nodes.

At this stage all nodes are either obstacle nodes or free nodes. There are mixed nodes left unexpanded. This is satisfying the first condition of the two, mentioned earlier, we will stop the quadtree generation process here. The total quadtree generated in this process is shown in Fig-4.2.8.

We can observe from the fig 4.2.8, that all leaf nodes are either free nodes or obstacle nodes. A small box under each node represents node status. At this stage the process of generating the quadtree is completed.

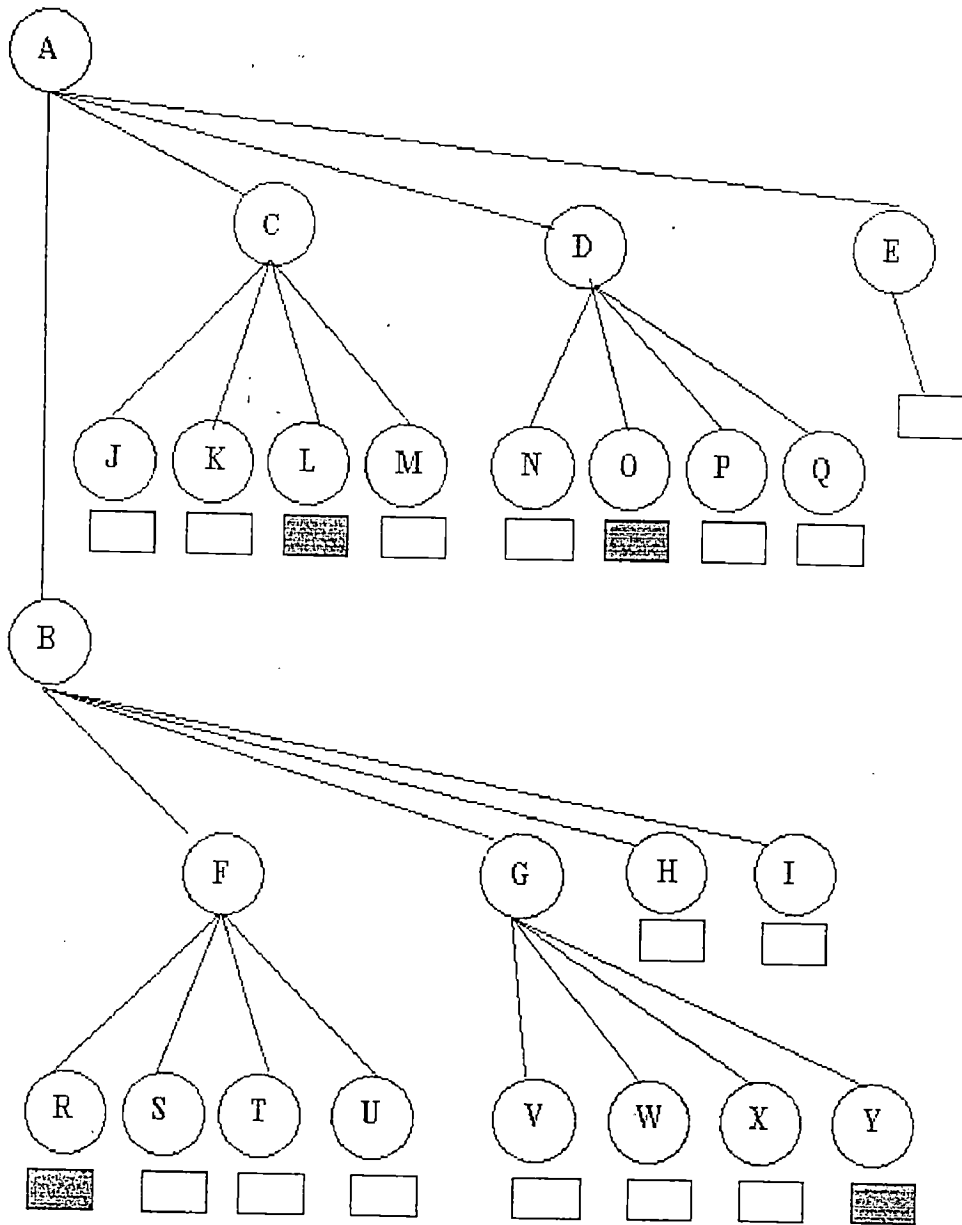


Fig-4.2.8. : Quadtree representation of the total world map given in Fig-4.2.3

# IMPLEMENTATION OF NEIGHBOR FINDING TECHNIQUE

---

## 5.1 Significance of Neighbor Finding Technique

There are different methods for moving between adjacent blocks in the quadtree. Different transitions can be made between blocks of equal size and blocks of different size, where the destination block is either of larger or smaller size than the source block. Such blocks are termed neighbors. There can also be possible to traverse along the diagonal as well as horizontal and vertical direction. The importance of these methods lies in their being corner stone of many of the quadtree algorithms, since they are basically tree traversal with a “visit” at each node. The significance of our method, Neighbor finding [2], lies in the fact that they don't use co-ordinate information, knowledge of the size of the image, or storage in excess of that imposed by the nature of quadtree data structure.

## 5.2 Neighbor Finding Algorithms for Quadtree

In the quadtree approach the image representation base on the successive subdivision of the image into the quadrants. It is represented by a tree of outdegree 4 in which the root represents a block and the four sons represent in order the NW, NE, SW and SE quadrants. We assume that each node is stored as a record containing six fields. The first five fields contains pointers to the node's father and its four sons, which corresponds to the four quadrants. The example is given below, is shown in the Fig.5.2.1.

## 5.2.1 World Map And its Decomposition

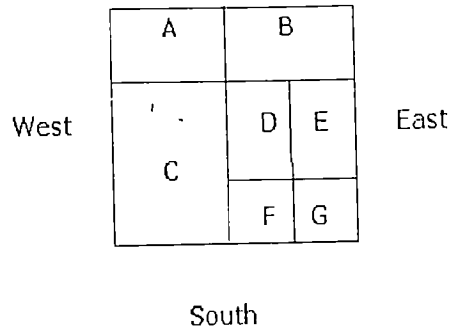


Fig 5.2.1. 2D World map

The neighbors of node D are B, E, F, and C regions in North, East, South and West. In our approach we are not taking the corner neighborhood (D and G are corner neighbors) because of possibility of absence of path between corner neighbors. For Example

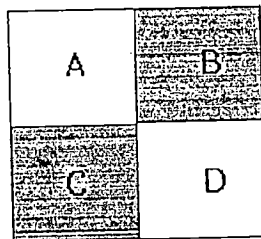


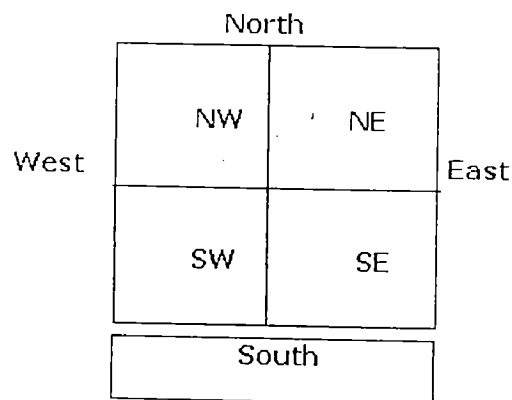
Fig-5.2.2 : Simple decomposed 2D world map.

In the fig 5.2.2 shown above, assume the robot is in the square region 'A', and the goal to be reached is in the square region 'D'. If we take corner neighbor into account, the regions A and D will be neighbors. But there is no path to move into the region D from the region A, since regions B and C are occupied with the obstacles. This is the reason we are neglecting corner neighbors.



### 5.2.2 Representation of Children

As mentioned in the previous section, for a mixed node, we will get four immediate children in four directions. These are called as 'NW','NE','SE' and 'SW', Which are represented below.



**Fig-5.2.3. Representation of children used in Algorithm.**

If 'P' is a node ,and 'I' is a Quadrant, then these fields are referenced as FATHER(P) and SON (P,I) respectively. We can determine the specific quadrant in which a node P lies relative to its father by the use of function SONTYPE(P),which has the value of I, if

$$\text{SON}(\text{FATHER}(P),I)=P$$

For example assume the figure shown above is a node named P, and the child node and the child node in the NW direction is named as Q. Then

$$\text{FATHER}(Q)=P$$

$$\text{SON}(P,\text{NW})=Q$$

$$\text{SONTYPE}(P)=\text{NW}$$

While generating the Quadtree we have stored the "node status" in every node. The integer values stored for the node status are

Node status =0=WHITE if node is a free node

Node status =1=BLACK if node is an obstacle node

Node status =2=GRAY if node is a mixed node

The four boundaries of a node's square region is called with names N,E,S and W for north, east, south and west directions respectively. We define the following predicates and functions, which will be used in the subsequent algorithm.

i)  $ADJ(B,I)$  is true if and only if the quadrant I is adjacent to the boundary B of the node's block.

For Example,  $ADJ(W, NW)=TRUE$ .

$ADJ(W, NE)=FALSE$

ii)  $REFLECT(B,I)$  yields the SONTYPE value of the block of equal size that is adjacent to the side B of a block having SONTYPE value I.

For Example,

$REFLECT(E, NW)=NE$  and

$REFLECT(N, SW)=NW$

$REFLECT$  gives the mirror image of the node I in the direction B.

For the world map is given as below:

- i) The mirror image of child SW in N (north) direction is NW.
- ii) The mirror image of child SW in E (east) direction SE.

These relations are represented into tables in fig.5.2.4(a) and Fig.5.2.4(b)

### 5.2.3 Adjacency Relation Table

**ADJ(S,Q)**

Quadrant 'Q'

		NW	NE	SW	SE
Side 'S'	N	T	T	F	F
	E	F	T	F	T
	S	F	F	T	T
	W	T	F	T	F

Fig 5.2.4(a) Adjacency Relation

### 5.2.4 Reflection Relation Table

**REFLECT(S,Q)**

Quadrant 'Q'

		NW	NE	SW	SE
Side 'S'	N	SW	SE	NW	NE
	E	NE	NW	SE	SW
	S	SW	SE	NW	NE
	W	NE	NW	SE	SW

Fig 5.2.4(b) Reflection Relation.

These diagrams shows predicate relations used in the neighbor finding algorithms.

#### 5.2.4 Neighbor Finding

For Quadtree corresponding to a  $2^n \times 2^n$  array, the root is at level 'n', and that a node at level 'l', is at distance 'n-l' from the root of the tree. In other words, for a node at level 'l' we must ascend 'n-l' FATHER links to reach the root of the tree.

```
Node procedure  GTEQUAL_ADJ_NEIGHBOR(P,D);
/* Locate a neighbor of a node P in horizontal or vertical direction D. If such a node
does not exists, then return NULL.*/
begin
    Value node P;
    Value direction D;
    Node Q;
    If not NULL(FATHER(P)) and ADJ(D,SONTYPE(P))
    then
        /*Find common ancestor */
        Q←GTEQUAL_ADJ_NEIGHBOR(FATHER(P),D)
    else  Q←FATHER (P);
    /*Follow the reflected path to locate neighbor */
    return (if not NULL (Q) and node status (q)=GRAY
    then
        SON (Q, REFLECT (D, SONTYPE(P)))
    else Q)
end
```

This algorithm will return a neighbor of greater or equal size. This is done by first finding the common ancestor. Next we retrace the path while making mirror image moves about an axis formed by the common boundary between the blocks associated with the nodes. The common ancestor is simple to determine. For example, to find an eastern neighbor the common ancestor is the first ancestor node, which is reached via its NW or SW son.

The procedure is shown in the fig 5.2.5.

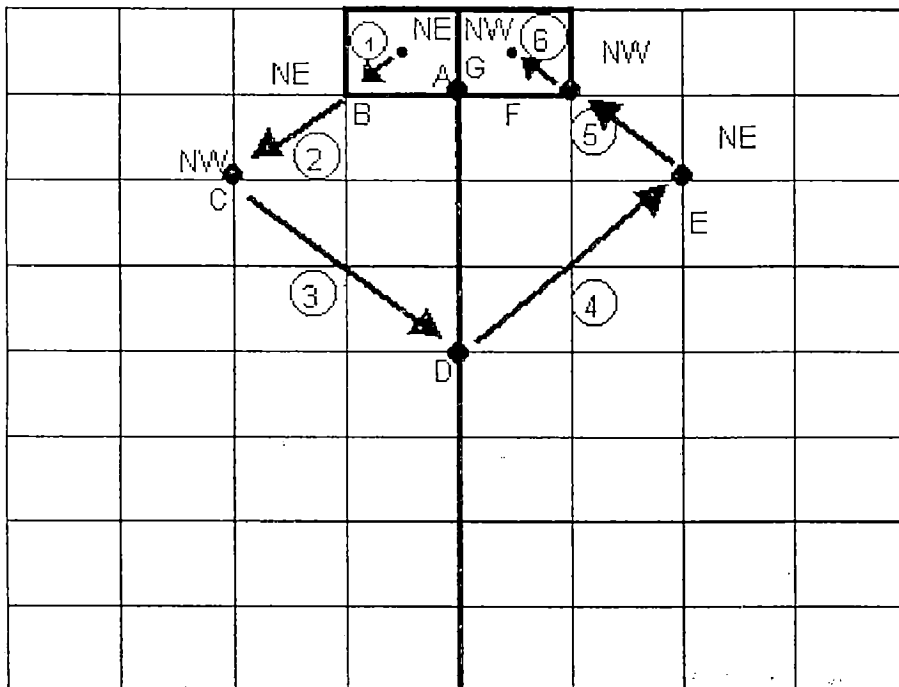


Fig 5.2.5 : Finding the neighbor of node A using a mirror image path from common ancestor (block-decomposition).

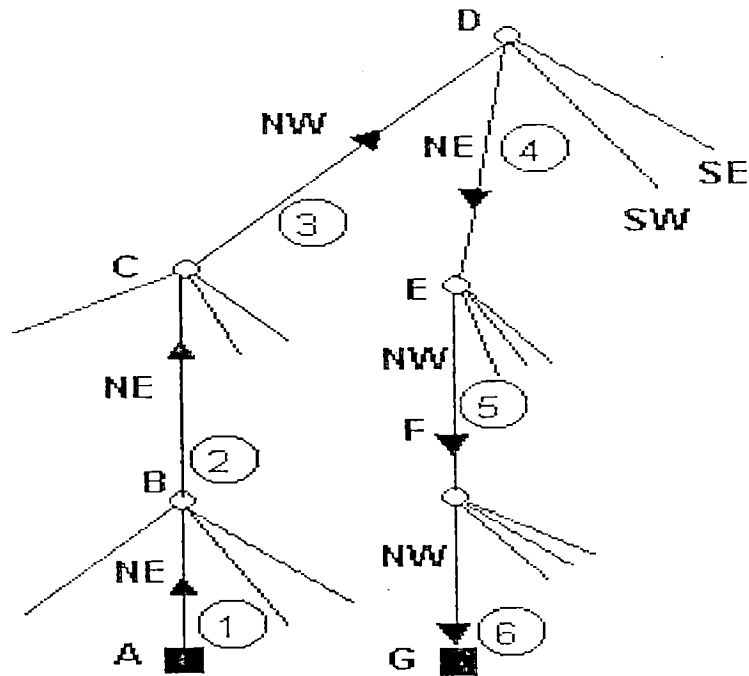


Fig 5.2.6 : Finding the neighbor of node A using a mirror image path from common ancestor (Tree decomposition).

In the Fig-5.2.5 the eastern neighbor of the node A is G. It is located by ascending the tree until the common ancestor D is found, from the Fig. 5.2.6. This requires going through a NE link to B, a NE link to C, and a NW link to reach D. The node G is now reached by backtracking along the previous path with appropriate mirror image moves. This requires descending a NE link to reach E and NW link to reach F and a NW link to reach G.

## RESULT AND CONCLUSION

---

### 6.1 Results

A simple example of a path obtained by Multiresolution (Quadtree Approach) path planning as we discussed earlier in previous chapters Algorithm has been shown in the Fig. 6.1. (Block Traversal), Fig 6.2 ( Optimal Path formed by Multiresolution approach).

#### INPUT FOR START AND GOAL LOCATION

```
set goal x-location : 90  
set goal y-location : 380  
set starting x-location : 240  
set starting y-location : 90
```

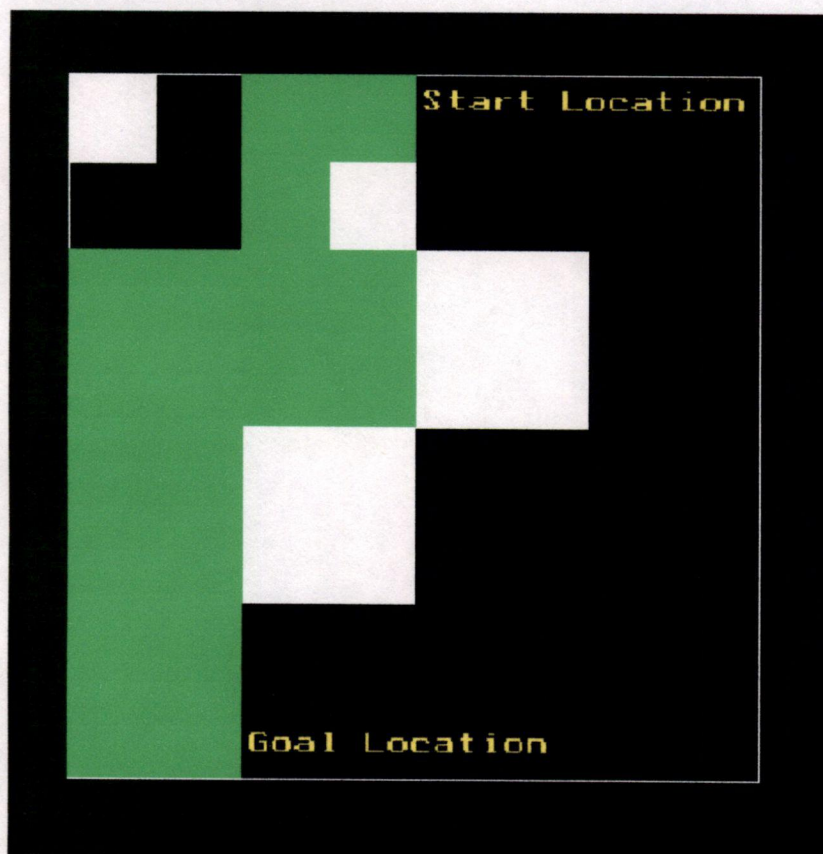
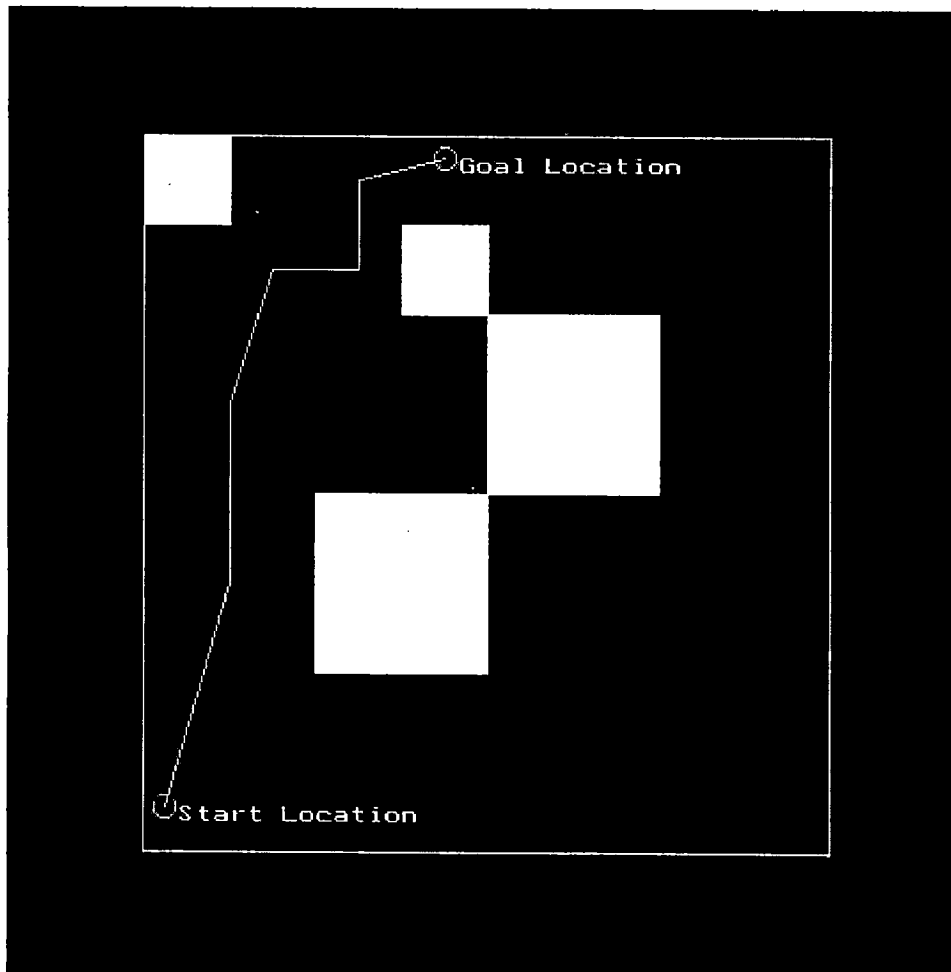


Fig.6.1 Block Representation of Path Formed by Quadtree Approach



**Fig 6.2 Path formed by Multi-Resolution (quadtree) Approach.**



## 6.2 Conclusion

Compared to the other path-planning algorithm, path planning cost for the quadtree-based search will be substantially lower because the number of nodes to be searched in Quadtree-based approach is considerably smaller. A hierarchy of different levels of description of the space that is available with quadtrees enables us to search for the path close to obstacles only when necessary. Corner clipping, inflexible paths are eliminated by considering only neighbors in horizontal and vertical directions.

The path produced by the quadtree algorithm, although not “optimal”, is a “negotiable” path which can be computed quickly. Apart from this, the hierarchical nature of the representation gives many advantages in path planning. For example, we can easily constrain the path to satisfy certain conditions, such as specification of minimal clearance of the path.

Though the generated path is collision free, but it is inferior to the exact optimal path that a robot can travel from the starting point to goal point in the given environment.

## 6.3 Further Scope

Mobile robots operating in vast outdoor unstructured environment often only have incomplete maps and must deal with new objects found during traversal. Path planning in such sparsely occupied regions must be incremental to accommodate new information and must use efficient representation. A path plan can be Implemented when the environment is not known ahead of time, but rather is discovered as the robot moves around. The planning can also be extended to 3-D map by using oct-trees.

## REFERENCES

---

1. H. Samet, "An Algorithm for converting rasters to quadtrees," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 3, pp. 93-95. 1981.
2. H. Samet, "Neighbor finding techniques for images represented by quadtrees," *Computer Graphics and Image Processing*, vol. 18, p.p. 37-57, 1982.
3. Charles R. Dyer, "The Space Efficiency of Quadtrees", *Computer Graphics and Image Processing*, vol. 19, p.p. 335-348, 1982.
4. David Nitzan, "Development of Intelligent Robots" *IEEE journal of Robotics and Automation*, vol. RA-1, No. 1, p.p. 3-12, 1985.
5. James L. Crowley, "Navigation for an intelligent Mobile Robot", *IEEE journal of Robotics and Automation*, vol. RA-1, No. 1, p.p. 31-40, 1985.
6. H. Samet, "An overview of quadtrees, oct-trees, and related Hierarchical Data structures", *NATO ASI Series*, vol. F40, 1988.
7. S.Ghoshray, K.K.Yen, "A comprehensive robot collision avoidance scheme by two dimensional geometric modeling" in *proc. Of the 1996. IEEE Int. conf. On Robotics and Automation*, minne polis, USA, 1996.
8. Danny Z Chen, Robert J. Szczerba and John J. Uhran. Jr, "A Framed-Quadtree approach for determining Euclidean shortest paths in a 2-D environment". *IEEE Transactions on Robotics and Automation*, vol. 13. No. 5. P.p. 668-680, 1997.
9. Alex Yahja, Anthony Stentz, Sanjiv Singh, and Barry L. Brummit, "Framed-Quadtree Path Planning for Mobile Robots Operating in Sparse Environments." In *proceedings, IEEE Conference on Robotics and Automation(ICRA)*, Leuven, Belgium, May 1998.
10. [www.dao.nrc.ca/abstracts/pasztor2.tex.html](http://www.dao.nrc.ca/abstracts/pasztor2.tex.html)
11. *Computer Graphics by Foley.*

## Appendix A

---

### A.1 Data Structure of Quadtree

The data structure of quadtree is described as follows

```
struct node
{
    int x1;
    int y1;
    int x2;
    int y2;
    int finished;
    node *child1;
    node*child2;
    node*child3;
    node*child4;
    node*parent;
    short status;
    int visited; }
```

Where  $(x1, y1)$  and  $(x2, y2)$  are the co-ordinates of top-left and bottom-right corner of a quadrant. The child1, child2, child3 and child4 fields of the node represent four quadrants of the region. Parent field of a node stores pointer to its parent. The status field can have three values BLACK (For obstacle), WHITE ( for free region), PARTIAL(for mixed region). The finished field of a node store 0 or 1 depending on whether that needs further expansion or not respectively.

### A.2 Quadtree Class

The class for quadtree is decided and is as follows.

```
class quadtree
{
    node *root;
    public:
    quadtree(void);
    void develop_tree(node* n);
    node *return_root(void);
    void set_current_node(node*n,int x,int y);
    void set_goal_node(node*n,int x,int y);
    node *find_neighbour(node*p,int direction);
    void cleanup(node* n);
}
```

- Names of the methods are reflecting their usage.
- The routines for initialization of quadtree is written. It is given below

```
quadtree::quadtree()
{
    root=new node;
    root->x1=80;
    root->y1=80;
    root->x2=400;
    root->y2=400;
    root->child1=NULL;
    root->child2=NULL;
    root->child3=NULL;
    root->child4=NULL;
    root->parent=NULL;
    root->finished=0;
}
```

- The region of obstacle is converted in to raster and then it is stored in the quadtree .

#### A.4 Neighbor Finding Routine

- The routine for finding the neighbor, using the adjacency matrix[2], is based on following algorithm.

```
Node procedure GTEQUAL_ADJ_NEIGHBOR(P,D);
/* Locate a neighbor of a node P in horizontal or vertical direction D. If such a
node does not exists, then return NULL.*/
begin
    Value node P; Value direction D;
    Node Q;
    If not NULL(FATHER(P)) and ADJ(D,SONTYPE(P))
    then
        /*Find common ancestor */
        Q ← GTEQUAL_ADJ_NEIGHBOR(FATHER(P),D)

    else Q ← FATHER (P);
        /*Follow the reflected path to locate neighbor */
        return (if not NULL (Q) and node status (q)=GRAY
        then
            SON (Q, REFLECT (D, SONTYPE(P)))
        else Q;
end
```

## LIST OF FIGURES

FIG. NO.	DESCRIPTION	PAGE.NO
2.1	The Division of Mobile Robot	8
3.1	The relationship between different tasks	9
3.2	DFD 0 level of path planning system	11
3.3	DFD 1 level of path planning system	12
3.4	DFD 2 level of path planning system	13
3.4.1	Best case position of a $2^m$ by $2^m$ region in a $2^n$ by $2^n$ binary image (a) Block decomposition (b) Quadtree Representation	13
3.4.2	Worst case position of a $2^m$ by $2^m$ region.(a) Block decomposition of a 16 by 16 region at position (1,1). (b) Portion of the quadtree	14
4.1.1	An image, its maximal blocks, and the Corresponding quadtree.	20
4.1.2	Intermediate trees in the process of obtaining a quadtree for the first part of the first row in fig. 4.1.1 (a).	21
4.1.3	Quadtree prior to merging nodes 1,2,9 and 10	23
4.1.4	Quadtree after processing the first row in Fig. 4.1.1.(a)	24
4.1.5	Quadtree after processing the second row in Fig. 4.1.1.(a)	24
4.2.1	Representation of a simple 2D world map, in which gray area representing obstacle region, and white core presenting obstacle free region	25
4.2.2	The decomposition of 2D World map in Fig 4.2.1 into quadtree nodes.	26
4.2.3	Representation of a 2D world map ,in which gray area representing obstacle region , and white color area representing obstacle free region .	28
4.2.4	The decomposition of map represented in Fig 4.2.3.	29
4.2.5	Quad tree representation of the decomposition .They have to decompose further and the decomposition is shown in the Fig-4.2.6.	30

FIG.NO.	DESCRIPTION	PAGE.NO
4.2.6	The second stage decomposition of the 2D world map given in Fig-4.2.3	31
4.2.7	Further decomposition of remaining mixed nodes	32
4.2.8	Quadtree representation of the total world map given in fig-4.2.3	33
5.2.1	2D World map	36
5.2.2	Simple decomposed 2D world map	36
5.2.3	Representation of children used in Algorithm	37
5.2.4	Predicate relations used in the neighbor finding algorithms	39
5.2.5	Finding the neighbor of node A using a mirror image path from common ancestor	39
5.2.6	Finding the neighbor of node A using a mirror image path from common ancestor(tree representation)	42
6.1	Block Representation of Path Formed by Quadtree Approach	43
6.2	Path formed by Multi-resolution (quadtree) Approach	44

