International Journal of Computer Science and Informatics

Volume 1 | Issue 3 Article 5

January 2012

Knowing the Robot's World

Radha Krishna Jana
Dept of CSE, Mallabhum Institute of Technology, Bishnupur, Bankura, radhakrishnajana@gmail.com

Debanjan Sengupta
Dept of CSE, Mallabhum Institute of Technology, Bishnupur, Bankura, dave4u90@gmail.com

3Sumanta Banerjee
Dept of CSE, Mallabhum Institute of Technology, Bishnupur, Bankura, bsumanta41@yahoo.com

Smita Chowdhury
Dept of CSE, Mallabhum Institute of Technology, Bishnupur, Bankura, smitachowdhury22@gmail.com

Available online at www.interscience.in

Knowing the Robot's World

[1]Radha Krishna Jana, [2] Debanjan Sengupta, [3]Sumanta Banerjee, [4]Smita Chowdhury
Dept of CSE, Mallabhum Institute of Technology, Bishnupur, Bankura
E-mail : radhakrishnajana@gmail.com, dave4u90@gmail.com,
bsumanta41@yahoo.com,smitachowdhury22@gmail.com

Abstract **-** The term robotics is used in broader aspects now-a-days. One of its main implementation is in the game development. Robotics & artificial intelligence are used in this field almost interchangeably. In this demonstration project of Patnaik's algorithm in the field of artificial intelligence we are trying to demonstrate how the intelligent robots traces path during a game. This is implemented using the most fundamental graphics tool known i.e. BGI graphics in C. Our demonstration project will also help in getting the idea of a robot's path tracing algo.

Keyword **-** *Artificial Intelligence, Patnaik's Algorithm, Procedures Traverse Boundary, Procedure Map_building, State Space Diagram, Context Diagram, Formal Model of The Problem, Problem Discussion, Bot Shape, Map Generation.*

I. INTRODUCTION

The term robot is used for machines that can think or which perceive, understand and react to their environment. Robots are designed to mimic human intelligence. This is mainly implemented using the branch of science that is artificial intelligence. In recent times this field has been credited with huge popularity. Now days robots are equipped with various types of sensors to recognize the world around them .A robot can also construct high level knowledge from relatively lower level data .The main function of robots is the sensing action. The project that we have discussed here concerns a bot present inside a known boundary and it senses and detects the obstacles that comes its way and instead of colliding with them traces a new obstacle free path.

II. PROBLEM ANALYSIS

Formal Model of our problem:

We are trying to build a formal model of one particular bot. A model for bots in which the bot is present inside a fixed room or boundary and there is also the presence of some obstacles .Each time for every keyboard hit the fixed and static number of obstacles (bricks) are generated at different positions inside the fixed boundary or room. All the obstacles are of different shape. The bot starts it movement from the centre of the room and traverses the boundary of the room and, senses the obstacles that comes its way and then finds its own way out. The room size is (max allowed X coordinate-60)*(max allowed Y coordinate*-60) sq units and its fixed .The entire room is divided into grids of 10*10 units. The bot is of circular shape and is of radius 10 units. For the obstacles we have used a brick() function. In order to generate the different shapes for the obstacles we use the pattern () function. In order to trace its path we use a recursive function circle move (x,y, dx, dy) for the bot. It takes four parameters x,y are used for the current position of the bot and dx, dy for the direction of the new position the bot is supposed to go. The bot checks it path for boundary or for obstacle in 8 directions going in clockwise direction viz - north, north-east, east, south-east, south, south-west, west, north-west.

III. FORMULATION /ALGORITHM

Patnaik,s Algorithm:

Procedure Traverse-boundary (current-coordinate)

Begin

Initial-coordinate=current-coordinate;

Boundary-coordinate=NULL;

Repeat

Move-to(current-coordinate)and mark the path of traversal;

Boundary-coordinate= Boundary-coordinateU{current-coordinate};

Find the next point such that the next point is obstacle free and the perpendicular distance from the next such that the next point is obstacle boundary is minimum and path of traversal from the current to next is not marked;

Current-coordinate:=next-coordinate;

Until current-coordinate=initial-coordinate

Return Boundary-coordinate;

End

The above algorithm is self-explanatory and thus needs no elaboration.

We now present an algorithm for map building,where we will use the above procedure.

ProcedureMap-building(current-coordinate)

Begin

Move to(current-coordinate);

Check-north-east-direction();

If(new obstacle found)**Then do**

Begin

Current-obstacle=Traverse-boundary(new-obstacle-coordinate);

Add-obstacle-list (current-obstacle);//adds current obstacle to list//

Current-position=find-best-point(current-obstacle)//finding the best take off point from the current obstacle//

Call Map-building(current-position);

End

Else do begin

Check-north-east-direction();

If (new obstacle found) **Then do**

Begin

Current-obstacle=Traverse-boundary(new-obstacle-coordinate);

Add-obstacle-list(current-obstacle);

Current-position=find-best-point(current-obstacle);

Call Map-building(current-position);

End;

End;

Else do Begin

Check east direction();

//Likewise in all remaining directions//

End

Else backtrack to the last takeoff point on the obstacle

(or the starting point);

End.

The above algorithm is self explanatory and easy to understand.

IV. OBSTACLE GENERATION ALGORITHM

The steps of the algorithm are:

1. Default pattern is set with four rectangular boxes set to form a larger rectangle.
2. Each side of the smaller rectangle is of can is of length 10 units.
3. The diameter of the bot is of 10 units.
4. Pattern() function is used to generate the four rectangle boxes which are used to generate the different patterns of obstacles.
5. Pattern1(),Pattern2(),Pattern3(), Pattern4() functions are used to generate the four different patterns having colors red, yellow, cyan and green(colors going clockwise direction in the default pattern which is a rectangle formed by joining the four patterns).
6. A brick function is used to generate all the patterns.
7. brick1(),brick2(),brick3(),brick4(),brick5() are used to generate the five different shapes of obstacles.
8. Four co-ordinates are taken likewise- left, top, right and bottom.
9. The different patterns are generated by moving these four co-ordinates e.g. left+10, top+10, right+10, bottom+10.
10. The differences between left and right co-ordinates should always be 10 units.
11. The differences between top and bottom co-ordinates should always be 10 units.
12. Thus the required patterns are generated.

V. PROBLEM DISCUSSION

Strategy:

The main reservations regarding our project is that for operation on a closed environment, a robot is required to know its environment for its safety and security and also for reliability of its performance.

i) In this project we provide a bounded room of size (max X coordinate value-60)*(max Y coordinate value-60)sq units.

ii) There are five obstacles present in the room.

iii) The positions of the obstacles change with each key board hit and they are not in motion.

iv) We provide with a pattern generation algorithm for generation of obstacles of different shapes and size.

v) The bot senses in 8 directions viz-N(north), NE(north-east), E(east),SE(south-east),

S(south), SW(south-west), W(west), NW(north-west) .

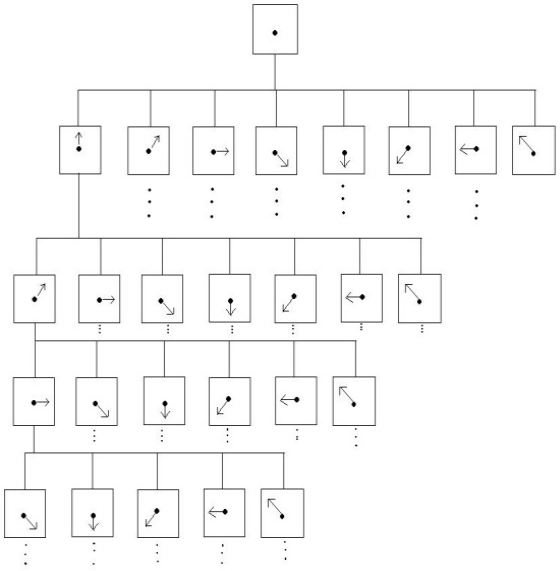
vi) When the bot senses the boundary of the room it traverses the entire boundary in clockwise direction and when it reaches the initial position from where it had first sensed the boundary it goes into a new path.

vii) When an obstacle is sensed by the bot it checks for the next possible obstacle free path in a clockwise direction for its movement.

VI. IMPLEMENTATION DETAILS

In this problem of bot movement at first we have taken a header file named **"design.h"** which when included in the program helps in calling functions like pattern(), brick() both of which returns void. These functions help creating obstacles. The boundary wall is being designed using rectangle() functions and floodfill() functions. Two rectangle() functions are called and then the region between is filled using floodfill() function with desired color (here Green) .The bot is designed very simply using circle() and floodfill() functions. We have also defined a function named **circlemove()** passing four parameters **circlemove(int x,int y,int dx,int dy)**. Here x,y denotes the current co-ordinates and dx, dy denotes the directional increment by desired units. The objective of the function is to detect and move along a free obstacle or boundary path. This movement is being done in a recursive way. This function is placed in a loop that continues to get called until a keyboard hit from **main()** function .

State Space Model:



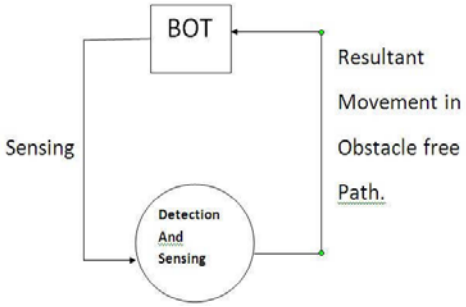
Explanation of State Space Diagram:

In the root node that is at the beginning of the bot movement it was at the center of the room.

At the next level there are eight available directions for the bot. For the time being, movement of the bot is initiated towards **north** direction. Encountering obstacle at the north direction it checks for the other seven directions except north for opting an obstacle free path. Same is true for all other direction.

Again, if any obstacle is detected in the **north-east** direction then it checks all other direction except **north** and **north-east**. Further if any obstacle is found in the **east** it checks all the remaining direction except **north**, **north-east** and **east** direction. This detection and sensing continues in a **clockwise** way. Thus finding its own free path.

Context Diagram:



VII. APPLICATIONS

The field of AI application is huge and the concept of our project has a wider range of application. Some of its applications are

Roomba-a vacuum cleaner, Fields of path detection algorithms, In the field of game development, A more developed version can be applied to automobiles, Can be applied to robot arm for various purposes.

VIII FUTURE SCOPE OF WORK

This project now is on a 2D platform having only X and Y axis. The future scope of the work lies in the fact that we will try to implement the same project in 3-D having X, Y and Z axis. The Bot architecture needs to be more efficient, smart and qualified. The map/field would be more realistic and dynamic.

IX. REFERENCES

[1] Stuart Russel, Peter Norvig"Artificial Intelligence- A Modern Approach".

[2] Amit Konar, "Artificial Intelligence and Soft Computing".