IJRDO International Journal of Research & Development Organisation

ISSN-4567-7860

JOURNAL OF ELECTRICAL AND ELECTRONINS ENGINEERING

A Review on Remotely Controlled Vision Based Industrial Robot for optimum movement

NAVEEN KUMAR C., AMRUT ANILRAO PUROHIT

cnaveenkumar71@gmail.com,amrutpurohit@revainstitution.org

Department of Electronics and Communication Engineering,
REVA Institute of Technology and Management, Bangalore 560064

***Abstract*— This paper is review of finding shortest path to destination by avoiding the obstacles using Dijkstra's algorithm to move robot from source to destination. Today's technology is making its way into our daily lives. Goal of the proposed work is to design and implement a real-time robot. A robot has been designed, prototyped, such that it involves various applications in innovative technologies like accident avoidance and moving through the shortest path in a harsh industrial environment. The concept of "Anti-Collision and shortest path" is one of the solutions to it. Microcontroller is the heart of robot where robot is moving blindly as per the instructions given by microcontroller. Microcontroller instructions are given through MAT lab program which calculates the shortest path to the destination.**

Index Terms—obstacle avoidance, dijkstra algorithm, RF module

I. INTRODUCTION

Technology today is making its way into our daily lives. This paper reviews finding shortest path to reach destination by avoiding the obstacles, Dijkstra's algorithm logic is used to find the shortest path and to guide the robot. This paper will be a review about the vehicle or device control for avoiding accident and finding the shortest path by different algorithms. By knowing the Cartesian points of the three points(locations) we can find the angle between the two lines with respect to the reference line x-axis. Dijkstra's algorithm also explains the logic which is used to compute the shortest path using MATLAB code.

The central processing unit consists of a camera and a processor. The image is captured by the camera, and the robot position is located in the image using the MATLAB. The destination to which the robot has to reach is identified by the user. Depending on the destination provided, the MATLAB code identifies the shortest path. The shortest path to the destination is notified to the robot with the help of RF transmitter.

At the receiving end the RF receiver decodes the received data. Microcontroller is programmed to control dc motor upon receiving the data from the computer. The robot traverse the path specified by the computer.

The central processing unit captures an image every 500 ms and provides an alternative path for the robot to reach its destination if there are any dynamic obstacles in the path. It also provides the makes sure that the robot follows the path specified by the computer. Hence providing a feedback path.

Control method for moving robot in closed area is based on creation and sharing maps through shortest path finding and obstacle avoidance is proposed.[1]Through simulation study, a validity of this method is confirmed. Furthermore, the effect of map sharing among robotics is also confirmed together with obstacle avoidance with ultrasonic sensors.[1]

The real-time path-planning[2] problem for autonomous robots is considered with the presence of arbitrary moving and static obstacles in the workspace. It is solved as a dynamic shortest route problem by adapting Dijkstra's algorithm to the workspace motion-scene graphs generated at successive scan intervals. In this obstacle-avoidance scheme, the robot takes the relative importance of detected obstacles into consideration [2]. This is the path planning problem for autonomous mobile robots that executes obstacle-avoidance.

ROBOT has sufficient intelligence[3] [8] to cover the maximum area of provided space. It has an infrared sensor which is used to sense the obstacles coming in between the path of ROBOT. It will move in a particular direction and avoid the obstacle which is coming in its path. Autonomous Intelligent Robots are robots that can perform desired tasks in unstructured environments without continuous human guidance.

A low cost solution[4] to obstacle avoidance for a mobile robot and dynamic steering algorithm which ensures that the robot doesn't have to stop in front of an obstacle which allows robot to navigate smoothly in an unknown environment avoiding collisions. Obstacle avoidance strategy and working of robot is greatly dependent on the detection of obstacles by sensors and corresponding to the response of robot [6].

Path plan for autonomous robot is based on image processing techniques in the unknown environment. The system finds and analyzes an optimal path for robots, while avoiding obstacles along the way. The environment is first captured as an image using a camera. Obstacles detecting methods are then performed to identify the existence of obstacles within the

unknown environment [8] recognized as obstacles and then shortest path is obtain by A-Star algorithm. A* is commonly used for the common path finding problem in applications such as games, but was originally designed as a general graph traversal algorithm[5].

II. ALGORITHMS

The Shortest Path problem is defined on a directed, weighed graph, where the weights may be taken as distances. The objective is to find a path from a source node to destination node, that minimizes the sum of weights along the path.

The Shortest Path Problem (SPA) is one of the fundamental and most important in combinatorial problem.SPA is an important problem in graph theory and has applications in communications, transportation and electronic problem. Here different algorithm is described for solving SPA.

a) The Bellman-Ford algorithm

The Bellman-Ford algorithm solves the single-source shortest-path problem in the general case in which edge weights may be negative. Given a weighed, directed graph G = (V, E) with source s and weight function w : E → R, the Bellman-Ford algorithm returns a Boolean value indicating whether there is negative-weight cycle that is reachable from the source. If there is such a cycle, the algorithm indicates that no solution exists. If there is no such cycle, the algorithm produces the shortest paths and their weights.

The algorithm uses relaxation, progressively decreasing an estimate d[v] on the weight of a shortest path from the source s to each vertex v * V until it achieves the actual shortest-path weight δ(s, v).

The algorithm returns TRUE if and only if the graph contains no negative-weight cycles that are reachable from the source.

b) Directed acyclic graphs

By relaxing the edges of a weighted dag (directed acyclic graph) G = (V, E) according to a topological sort of its vertices, we can compute shortest paths from a single source in (V + E) time. Shortest paths are always well defined in a dag, since even if there are negative-weight edges, no negative-weight cycles can exist.

The algorithm starts by topologically sorting the dag to impose a linear ordering on the vertices. If there is a path from vertex u to vertex v, then u precedes v in the topological sort. We just make one pass over the vertices in the topologically sorted order. As we process each vertex, each edge leaving the vertex is relaxed.

c) Johnson's algorithm

Algorithm solves all pair shortest paths in a sparse weighed directed graph. It allows some of the edge weights to be negative numbers, but no negative-weight cycles exist.. Johnson's algorithm uses as subroutines of both Dijkstra's algorithm and the Bellman-Ford algorithm.

d) Dijkstra's algorithm

Dijkstra's algorithm solves the single-source shortest-paths problem on a weighed, directed graph G = (V, E) for the case in which all edge weights are non-negative. In this section, therefore, we assume that w(u, v) ≥ 0 for each edge (u, v) E. As we see a good implementation, the running time of Dijkstra's algorithm is lower than that of the Bellman-Ford algorithm.

Application of Dijkstra's Algorithm

- Robot path planning
- Logistics Distribution Lines
- Link-state routing protocols
- Open Shortest Path First
- Intermediate System to Intermediate System

The Bellman-Ford algorithm and Dijkstra's algorithm proved to be much more efficient than brute-force, with Dijkstra proving to run in the least amount of time for very large networks. It appears that for complete and fully meshed networks the Bellman-Ford algorithm actually run faster than Dijkstra's algorithm for networks of size 425 or less.

Let the node at which we are starting be called the initial node. Let the distance of node Y be the distance from the initial node to Y. Dijkstra's algorithm will assign some initial distance values and will try to improve them step by step.

1. Assign a tentative distance value to every node: set initial node as zero and infinity for all other nodes.
2. Mark all nodes unvisited. Set the initial node as current. Create a set of the unvisited nodes called the unvisited set consisting of all the nodes.
3. For the current node, consider all of its unvisited neighbors and calculate their tentative distances. Compare the newly calculated tentative distance to the current assigned value and assign the smaller one. Otherwise, keep the current value.
4. When we are done considering all of the neighbors of the current node, mark the current node as visited and remove it from the unvisited set. A visited node will never be checked again.
5. If the destination node has been marked visited (when planning a route between two specific nodes) or if the smallest tentative distance among the nodes in the unvisited set is infinity (when planning a complete traversal; occurs when there is no

connection between the initial node and remaining unvisited nodes), then stop. The algorithm has finished.

6. Select the unvisited node that is marked with the smallest tentative distance, and set it as the new "current node" then go back to step 3.

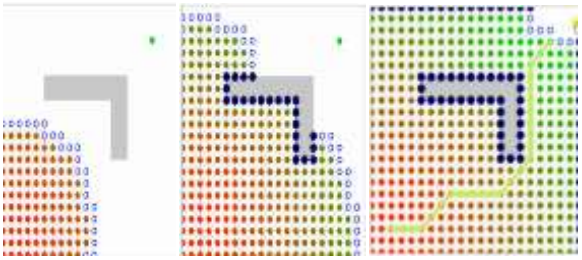

Figure2: shows the working of algorithm

e) A* Algorithm

A* algorithm is a graph search algorithm that finds a path from a given initial node to a given goal node. It employs a "heuristic estimate" h(x) that gives an estimate of the best route that goes through that node. It visits the nodes in order of this heuristic estimate. It follows the approach of best first search. The secret to its success is that it combines the pieces of information that Dijkstra's algorithm uses and information of Best- First-Search. In the standard terminology used when talking about A*, g(n)represents the exact cost of the path from the starting point to any vertex n, and h(n) represents the heuristic estimated cost from vertex n to the goal.

Dijkstra's is essentially the same as A*, except there is no heuristic (H is always 0), because it has no heuristic. It searches by expanding out equally in every direction. But A* scans the area only in the direction of destination. As we might imagine, because of this Dijkstra's it usually ends up exploring a much larger area before the target is found. This generally makes it slower than A*. But both have their own importance. For example A* is mostly used when we know both the source and destination. Dijkstra's is used when we don't know where our target destination is. Say you have a resource-gathering unit that needs to go get some resources of some kind. It may know where several resource areas are, but it should approach closer one. In this case, Dijkstra's is better than A* because we don't know which one is closest. Our only alternative is to repeatedly use A* to find the distance to each one, and then choose that path. There are probably countless similar situations for the kind of location we might be searching for. In order to find the closest one, but we don't find where it is or which one is closest. So, A* is better when we know both starting point and destination point. A* is both complete and optimal if you use an Admissible heuristic function. If the function is not admissible - all bets are off.

III. IMPLIMENTATION METHODOLOGY

IR sensor or ultrasonic sensor are used to detect obstacles and the alternative routes can be taken to avoid the obstacle. To calculate the shortest path from source to destination continuously and also to track the dynamic obstacles. The camera is used to capture the images and constantly monitor the movement of the robot and the obstacles. The images are processed to find the obstacles in between source and destination. Upon processing the image, the shortest path is identified and the robotic motions are controlled. The robot upon finding an obstacle stops and intimates the computer about the obstacle in its path. The computer reroutes the path and guide the robot through the new path. RF transmission is used to communicate between the robot and the computer, as it is cost effective. MATLAB is used to process the images and provide the shortest path.

By knowing the three Cartesian points the angles of the two lines making with respect to the reference line x-axis can be found, if angle is taken either clockwise or anti-clockwise there are chances of getting negative angle depending on the position of two lines, to make it positive subtract it from 360°.There fore, the angle between the two lines with respect to the reference line is calculated as shown in the figure below.

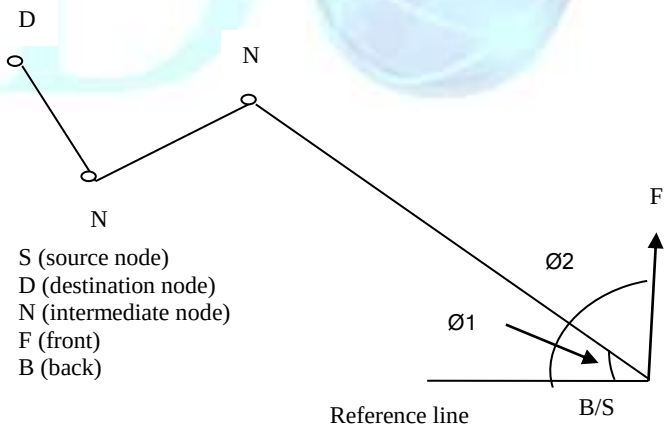

Figure1: showing robot front, back, next node and the destination

Therefore the angle formed between the two lines (theta) with respect to the reference line that joins at the common point is calculated. To know that, we need to know the distance of the two lines by using Equ(1) &Equ(2).

$$R1 = \sqrt{(Bx - Fx)^2 + (By - Fy)^2} \dots\dots\dots\dots\dots\dots Equ\ (1)$$
$$R2 = \sqrt{(Nx - Bx)^2 + (Ny - By)^2} \dots\dots\dots\dots\dots\dots Equ\ (2)$$

To calculate the angle forming between the line and the reference line individually, therefore two angles are formed separately for two lines with respect to the reference line x-axis.by using Equ(3), Equ(4), the formulae are given below.

$$\Phi 1 = \cos((Bx - Fx) / R1) \quad \ldots\ldots(3)$$

$$\Phi 2 = \cos((Nx - Bx) / R2) \quad \ldots\ldots(4)$$

Initially check which of the two angles is greater, to make the decision to move left or right and then if the angle is more than the threshold value then it should turn in the opposite direction, if the angle is less than threshold then move forward. Here time taken to move forward i.e., forward time and turn time (left/right) can be adjusted in the robot program. The turn time should made as optimal as possible for quicker response time, Therefore it should be set by trial and error basis and also the value depends on motor specification and weight of the vehicle.

The robot moves forward when the angle falls within the threshold limit, if not it stops and checks for the threshold, and it continues to turn till it falls in the threshold region. The image is captured continuously in a real time so that the dynamic obstacles are avoided to calculate the shortest path from the recent captured image till it reaches the destination.

- Case (1): When both Ø1 & Ø2 are positive and Ø1 >Ø2, so move left.
- Case (2) When both Ø1 & Ø2 are positive and Ø1 <Ø2, so move right.
- Case (3) When Ø1 is negative & Ø2 is positive Ø1=2π-Ø1 and then if Ø1 <Ø2, so move left.
- Case (4) When Ø1 is positive & Ø2 is negative Ø2=2π-Ø2 and then if Ø1 <Ø2, so move right.
- Case (5) When both Ø1 & Ø2 are negative Ø1=2π-Ø1 and Ø2=2π-Ø2 then if Ø1<Ø2, so move left.
- Case (6) When both Ø1 is positive & Ø2 are negative Ø1=2π-Ø1 and Ø2=2π-Ø2 then if Ø1 >Ø2, so move right.

IV. CONCLUSION

Some of the algorithms explained in this paper would be used to control the movement of the robot by avoiding the obstacle and finding the shortest path to reach destination. Here we can conclude that Dijkstra's algorithm and by using visual sensors finding an obstacle is easier compared to Dynamic Obstacle-Avoidance proposed [2].

REFERENCE

[1] Moving Domestic Robotics Control Method Based on Creating and Sharing Maps with Shortest Path Findings and Obstacle Avoidance (*IJARAI) International Journal of Advanced Research in Artificial Intelligence, Vol. 2, No.*

[2] A Dynamic Obstacle-Avoidance Scheme for Autonomous Robots[2014] O.S. Asaolu*, V.O.S. Olunloyo, *Department of Systems Engineering, University of Lagos, Nigeria*

[3] OBSTACLE AVOIDING ROBOT-International Journal of Advanced Research in Electrical, Electronics and Instrumentation EngineeringVol. 2, Issue 4, April 2013

[4] Low Cost Obstacle Avoidance RobotVivekHanumante, Sahadev Roy, SantanuMaityInternational Journal of Soft Computing and Engineering (IJSCE)ISSN: 2231-2307, Volume-3, Issue-4, September 2013

[5] An Image Based Path Planning Using A – StarAlgorithm HerambNandkishor Joshi, Prof J. P. Shinde,International Journal of Emerging Research in Management &Technology ISSN: 2278-9359 (Volume-3, Issue-5)-2014

[6] Basem M. ElHalawany, HalaM.Abdel-Kader, AdlyTagEldeen, AlaaEldeenElsayed "Modified A* Algorithm for Safer Mobile Robot Navigation" U2013 Proceedings of International Conference on Modelling, Identification & Control (ICMIC)Cairo, Egypt, 31stAug.- 2nd Sept. 2013

[7] Wang Shaokun, Xiao Xiao, and Zhao Hongwei, "The Wireless Remote Control Car System Based on ARM9," in Instrumentation, Measurement, Computer, Communication and Control, International Conference on , October 2011 , pp. 887-890.

[8] Ding Chengjun, Yan Bingsen, and Duan Ping, "The Remote Control of Mobile Robot Based on Embedded Technology," in Measuring Technology and Mechatronics Automation, International Conference on , January 2011, pp. 907-910.

[9] NiuZhigang and Wu Yanbo, "Research on Wireless Remote Control for Coal Mine Detection Robot," in Digital Manufacturing and Automation, International Conference on , December 2010 , pp. 315-318.

[10] Ofir H Goldstain, Irad Ben-Gal, and Yossi Bukchin, "Evaluation of Telerobotic Interface Components for Teaching Robot Operation," in IEEE TRANSACTIONS ON LEARNING TECHNOLOGIES, 2011, pp. 365-376.

[11] NiuZhigang and Wu Yanbo, "Research on Wireless Remote Control for Coal Mine Detection Robot," in

Digital Manufacturing and Automation, International Conference on, December 2010, pp. 315-318.

[12] EvsyakovArtem Dmitry Bagayev, "System remote control of the robotized complex - Pegas," in East-West Design & Test Symposium , September 2010 , pp. 358-361.

[13] Dijkstra, E.W. (1959). A note on two problems in connexion with graphs. In NumerischeMathematik, 1 (1959), S. 269 ～271.

[14] Huijuan Wang et. al , "Application of Dijkstra algorithm in robot path-planning", Second International Conference on Mechanic Automation and Control Engineering (MACE), pp. 1067 - 1069 ,2011

[15] N. L. Cassimatis, J. G. Traftonb, M. D. Bugajskab and A. C. Schultz, ―Integrating cognition, perception and action through mental simulation in robots.‖ Robotics and Autonomous Systems, 49: 13–23, 2004.