Shortest Path Finding and Tracking System Based on Dijkstra's Algorithm for Mobile Robot

WaiWai Maw, WaiPhyoEi

***Abstract*— Roads play a vital role to the people live in different places and, from day to day, they travel to schools, to work, to shops, and to transport their goods. Even in this modern world, roads remain one of the mediums used most frequently for travel and transportation. The computation of a shortest path between different locations appears to be a key problem in the road networks. Even now the problem still persists to find the shortest path on road networks. The wide range of applications was introduced to overcome the problem by developing various shortest path algorithms. Among then, Dijkstra's algorithm is used to find the shortest path from one node to another node in a graph. It is also known as a single source shortest path algorithm. It is applied only on positive weights. Simulation results are carried out to find and track for the shortest path by using Dijkstra's algorithm**

***Index Terms*—Shortest Path Finding and Tracking, Wheeled Mobile Robot, Point to Point Motion, Dijkstra's algorithm**

1) INTRODUCTION

Motor vehicles, as a kind of modern transport means, with the advantages of high speed and convenience, are important to people's daily travel. Activities like going out, travelling to work, shopping, and visiting friends and relatives are often done by using motor vehicles. With the development of economics and technology, the number of motor vehicles has increased rapidly in the past decades. Davis in 2012 showed that from 1990 to 2009 in selected countries, the average annual percentage change of the number of cars is 2.3%, for trucks and buses the average annual percentage change is 3.9%, and the growth is still proceeding. Availability of more vehicles makes it more convenient for people to travel and merchandise transport. The increase of the number of vehicles also brings stresses to public traffic and pollution to the environment.

A key problem in network and transportation analyses is the computation of shortest paths between different locations on a network. Sometimes this computation has to be done in real time. For the sake of illustration, let us have a look at the case of a 911 call requesting an ambulance to rush a patient to a hospital. Today it is possible to determine the fastest route and dispatch an ambulance with the assistance of GIS. Because a link on a real road network in a city tends to possess different levels of congestion during different time periods of a day, and because a patient's location can't be expected to be known in advance, it is practically impossible to determine the fastest route before a 911 call is received. Hence, the fastest route can only be determined in real time. In some cases the fastest route has to be determined in a few seconds in order to ensure the safety of a stuck people. Moreover, when large real road networks are involved in an application, the determination of shortest paths on a large network can be computationally very intensive. Because many applications involve real road networks and because the computation of a fastest route (shortest path) requires an answer in real time.

The shortest path computations are one of the problems in graph theory. In shortest path problems, a directed weighted graph is given & the goal is to determine the shortest path among vertices [1]. The shortest path problem can be categorized in to two different problems; single source shortest path problem and all pair shortest algorithm. In Single source shortest path problems, a graph is being given and the goal is to find a shortest path from a given fixed vertex to all other vertices of the graph. In all pair shortest path problems [2], the goal is to finding the shortest paths between all pairs of vertices of a graph.

Traffic information systems use Dijkstra`s algorithm in order to track the source and destinations from a given particular source and destination. Dijkstra`s algorithm is used in SPF, shortest path first. There are two types of routing, Link State routing and Distance Vector routing. Dijkstra`s is based on Link State routing. In the Link-state routing approach, using Dijkstra`s algorithm each router calculates the shortest path to each node into the route table. Dijkstra's Algorithm is a graph search algorithm that solves the single-source shortest path problem for a graph with nonnegative edge path costs, producing a shortest path tree. The shortest distance between two points is a straight line. But in the real world, if those two points are located at opposite ends of the country, or even in different neighbourhoods, it is unlikely to find a route that enables to travel from origin to destination via one straight road. Find out a map to determine the fastest way to drive somewhere, but these days, it is just as likely to use a Web-based service or a handheld device to help with driving directions. The popularity of mapping applications for mainstream consumer use once again has brought new challenges to the research problem known as the "shortest-path problem."[are increasingly present in industrial and service robotics, particularly when flexible motion capabilities are required on reasonably smooth grounds and surfaces. Several mobility configurations (wheel number and type, their location and actuation and single or multi-body vehicle structure) can be found in different

applications (De Luca *et al*., 2001). The most common for single-body robots are differential drive and synchro drive (both kinematic equivalent to a unicycle), tricycle or car-like drive and omnidirectional steering (De Luca *et al*., 2001).

The shortest distance between two points is a straight line. But in the real world, if those two points are located at opposite ends of the country, or even in different neighbourhoods, it is unlikely to find a route that enables to travel from origin to destination via one straight road. Find out a map to determine the fastest way to drive somewhere, but these days, it is just as likely to use a Web-based service or a handheld device to help with driving directions. The popularity of mapping applications for mainstream consumer use once again has brought new challenges to the research problem known as the "shortest-path problem."[1]

The shortest-path problem, one of the fundamental quandaries in computing and graph theory, is intuitive to understand and simple to describe. In mapping terms, it is the problem of finding the quickest way to get from one location to another. Expressed more formally, in a graph in which vertices are joined by edges and in which each edge has a value, or cost, it is the problem of finding the lowest-cost path between two vertices.[1] considered as a graph with positive weights. The nodes represent road junctions and each edge of considered as a graph with positive weights. The nodes represent road junctions and each edge of the graph is associated with a road segment between two junctions. The weight of an edge may correspond to the length of the associated road segment, the time needed to traverse the segment or the cost of traversing the segment.[4]

The paper is organized as follow. Mathematical modeling of the mobile kinematicsis described in Section II. In Section III ,Related work. Methodology is described in Section IV. The simulation results for the proposed control scheme are given in Section V to demonstrate the performance of the whole system. Finally, conclusion is given in Section VI.

2) Mathematical Modeling Of The Mobile Kinematics

In this section kinematic model of four-wheel skid-steering mobile robot is presented. In order to simplify the mathematical model of SSMR we consider the following model assumptions:

The mass center of the robot is located at the geometric center of the body frame;

The two wheels of each side rotate at the same speed;

The robot is running on a firm ground surface, and four wheels are always in contact with the ground surface.

A. Autonomous positioning

The experimental platform used in this work is a car-like mobile robot. Such robot is a mobile vehicle where the front wheels can turn on the left or right, but kept always parallel. The rear wheels are fixed parallel to the car body. Such wheeled robots are systems with hard nonholonomic constraints where differential expressions involving velocity and state vectors are non-integrable. Each w heel presents kinematic constraints due to nonsleeping and non-tilting conditions (Cuesta, Gomes-Bravo &Ollero, 2004). Figure 1 shows a car-like robot with corresponding parameters: [x y] are the rear point *R* coordinates, and θ is the robot's heading. Linear velocity of the point *R* is represented by *v*. The orientation of the front wheels is noted φ, and *l* is the distance between the rear and the front wheels axes.

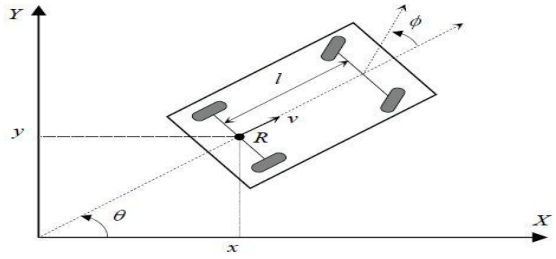

Figure.1. Car-like mobile robot.

The kinematic equation of the robot is described by:

$\dot{x} = v\cos\theta\cos\emptyset$(1)

$\dot{x} = v\sin\theta\cos\emptyset$(2)

$$\dot{\theta} = \dot{v}\frac{\sin\emptyset}{l} \quad (3)$$

The vector $(X, Y, \theta)^T$ is called configuration vector, and the vector [*v*(t) φ(t)]T is the control vector. This work is divided into two steps. The first step consists on the implementation of a simple positioning. This makes the robot moving from any initial position to a final destination, respecting nonholonomic constraints. The second task is the oriented positioning.

The robot is placed on a plane an inertial frame (X,Y) (global frame) and a local (robot body) frame (x,y) as shown in figure (1). Suppose that the robot moves on a plane with a linear velocity expressed in the local frame as v = v_l, v_r and rotates with an angular velocity ω =ω_z. If q = $(X, Y, \theta)^T$is the state vector describing generalized coordinate of the robot, then $\dot{q} = (\dot{x}, \dot{y}, \dot{\theta})^T$denotes the vector of generalized velocities. It is straightforward to calculate the relationship of the robot velocities in both frames as follows:

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}\cos\theta & -\sin\theta & 0\\ \sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}v_x\\ v_y\\ \omega_z\end{bmatrix} \quad (4)$$

And then the velocity of the robot can be taken from each wheel velocity by the following equation:

$$v_l = \frac{2v - \omega B}{2R},\ v_r = \frac{2v + \omega B}{2R} \quad (5)$$

So, for the first step is to consider each wheels velocities from the parameters of the mobile robot such as the axial distance between two wheels, B (wheel span) and the wheel radius, R.

The parameters of the mobile robot based on the kinematic model can be written as follows-

V_r, V_l −Right wheel and left wheel velocity respectively

x, y− Relative position of the robot local frame

θ− Relative heading of the robot in inertia frame

V − Linear velocity of the mobile robot

ω− Angular velocity of mobile robot

B -Distance between left and right wheel (wheel span)

B= 0.2m

R-Radius of wheel

R = 0.0325m

Above parameters of differential drive mobile robot are useful as input of basic kinematics model of robot.

Linear velocity of the mobile robot is expressed in equation (6).

$$v = \frac{v_r + v_l}{2} \quad (6)$$

Angular velocity of the mobile robot is expressed in equation (7).

$$\omega = \frac{v_r - v_l}{B} \quad (7)$$

Wheel can's not contribute to move on sideway, so $v_y = 0$. By the above assumptions, the two wheels of each side rotate at the same speeds, so ;

$v_x = v_r, v_l$, $v_r = v_3 = v_4$, $v_l = v_1 = v_2$.

B. Simple Positioning

To accomplish a simple positioning of a car-like mobile robot, the control vector [*v*(t) φ(t)]T must be computed in real-time, to make the robot moving from its initial position [xi yi] to any desired goal [xGyG], respecting its nonholonomic constraints. For that, "Robot Positioning Controller" (RPC) has been implemented in order to insure the convergence of the robot to the desired goal. The variables used by the RPC are represented on figure 2 :

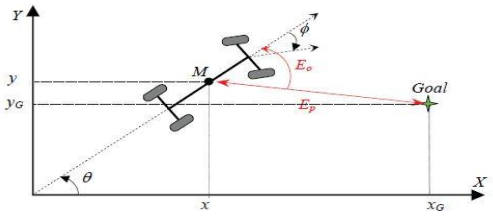

Figure.2. Simple positioning task.

From this figure:

$$E_p = \sqrt{\Delta x^2 + \Delta y^2} \quad (8)$$

$$E_o = \theta - \Delta\theta \quad (9)$$

$$\Delta x = x_G - x \quad (10)$$

$$\Delta y = y_G - y \quad (11)$$

$$\Delta\theta = \operatorname{artan}(\Delta y, \Delta x) \quad (12)$$

Where, $[x\ y\ \theta]^T$ is the configuration vector of the robot with respect to the middle of its axis.$[x_G\ y_G]^T$, are the coordinates of the goal (desired position).

C. Oriented Positioning

Oriented positioning task consists on the accomplishment of a robot positioning, with a desired final orientation. In theory, to accomplish this task, the control vector [*v*(t) φ(t)]T can be computed in real-time by a controller, in order to make the robot moving from any initial configuration [xi yiθi]T to any final desired configuration $[x_G y_G \vartheta_G]$T, respecting the nonholonomic constraints, as it has been done for simple positioning. Figure 3 shows the used variables in this case:

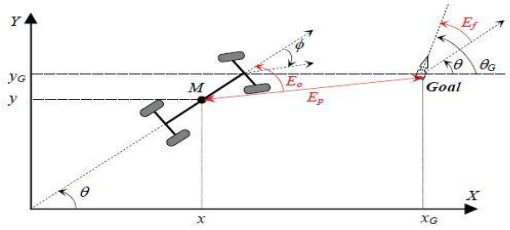

Figure.3. Oriented positioning task.

The variables E_p and E_o are those used for the case of simple positioning. $E_f$is computed as follow:

$$E_f = \theta_G - \theta \quad (13)$$

E_f expresses the "Final Orientation Error". To achieve an oriented positioning task, three input variables are necessary, which are E_p ,E_o and E_f. Unfortunately, only two outputs variables (*v* and φ) are available. The two input variables E_o and E_f are hardly coupled, and can not be controlled in the same time by only the two variables of the robot's control vector.

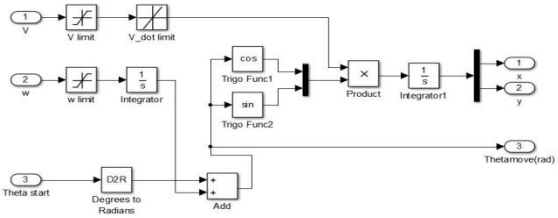

Figure.4.Simulink block for the unicycle kinematic model

3) RELATED WORK

A. Graph Theory

A graph is mathematical concept formally defined by a set of vertices (or nodes) *V* and a set of edges *E* connecting these vertices. Edges are called directed if for a pair of vertices, an edge can be used to travel from one node to the other but not the other way around. Edges may also have a numerical value, often called the weight or cost, which are an abstract representation of the "work" needed to move along that edge. Only these basic concepts are needed to understand the shortest path problem; for the more advanced concepts, please refer to . Graphs are often used as abstract representations of networks, e.g. a road-map, map of linguistic distance between words etc. and can discretize an environment into "checkpoints" to enable navigation by artificial intelligence agents.

B. Overview of Shortest Path Algorithm

Over the past decades there are various shortest path algorithm were developed. The shortest path algorithm are classified into three categories they are Single source shortest path algorithms, Single destination shortest path algorithms, All-Pairs shortest path algorithms and it is shown in below figure 6.

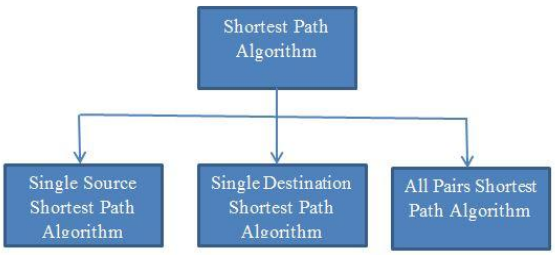

Figure.5. Classification of Shortest Path Algorithm

The shortest path problem can be defined for graphs whether undirected, directed, or mixed. It is defined here for undirected graphs; for directed graphs the definition of path requires that consecutive vertices be connected by an appropriate directed edge.

The problem is also sometimes called the **single-pair shortest path problem**, to distinguish it from the following variations:

The **single-source shortest path problem**, to find shortest paths from a source vertex *v* to all other vertices in the graph.

The **single-destination shortest path problem**, to find shortest paths from all vertices in the directed graph to a single destination vertex v. This can be reduced to the single-source shortest path problem by reversing the arcs in the directed graph.

The **all-pairs shortest path problem**, to find shortest paths between every pair of vertices *v*, *v'* in the graph.

These generalizations have significantly more efficient algorithms than the simplistic approach of running a single-pair shortest path algorithm on all relevant pairs of vertices.

4) METHODOLOGY

A. *Dijkstra's Algorithm*

Dijkstra's algorithm is an algorithm for finding the shortest paths between nodes in a graph, which may represent, for example, road networks. This algorithm is a solution to the single-source shortest path problem in graph theory. The algorithm exists in many variants; Dijkstra's original variant found the shortest path between two nodes, but a more common variant fixes a single node as the "source" node and finds shortest paths from the source to all other nodes in the graph, producing a shortest path tree. For a given source node in the graph, the algorithm finds the shortest path between that node and every other. It can also be used for finding the shortest paths from a single node to a single destination node by stopping the algorithm once the shortest path to the destination node has been determined . For example, if the nodes of the graph represent cities and edge path costs represent driving distances between pairs of cities connected by a direct road, Dijkstra's algorithm can be used to find the shortest route between one city and all other cities.

The single source shortest path problem can be described as follows:

newd_j =min $\{d_j , d_i + c_{ij}\}$

where ,

d_i=distance of current node

d_j =distance of label node

c_{ij} =weight set for the edges(i,j)

The algorithm maintains and step by step updates the states of the nodes.

Dijkstra's algorithm to calculate the shortest distance between any two nodes. The Dijkstra's algorithm calculates the shortest path between two points.

Step 1 Label the start vertex as 0 and all other vertex as infinity.

Step 2 Box this number (permanent label).

Step 3 Label each vertex that is connected to the start vertex with its distance (temporary label).

Step 4 Box the smallest number.

Step 5 From this vertex, consider the distance to each connected vertex.

Step 6 If a distance is less than a distance already at this vertex, cross out this distance and write in the new distance. If there was no distance at the vertex, write down the new distance.

Step 7 Repeat from step 4 until the destination vertex is boxed.

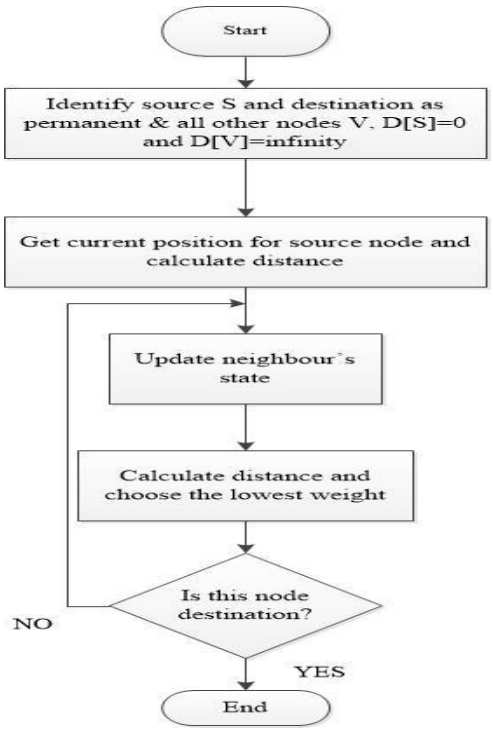

Figure.6. Flow chart of the shortest path calculation

B. *Point-to-Point Shortest Path Algorithms with Preprocessing*

The classical point-to-point shortest path problem (P2P): given a directed graph G = (V, A) with non-negative arc lengths and two vertices, the source s and the destination t, find a shortest path from s to t. More interesting is to find exact shortest paths. Preprocessing is allowed, but limit the size of the precomputed data to a constant times the input graph size. Preprocessing time is limited by practical considerations. For example, for computing driving directions on large road networks, quadratic-time preprocessing is impractical.

Note that preprocessing-based algorithms have two parts: preprocessing algorithm and query algorithm.

The former may take longer and run on a bigger machine or a cluster of machines. The latter need to be fast and may run on a small device. For the P2P case, note that when the algorithm is about to scan the destination t, d(t) is exact and the s-t path defined by the parent pointers is a shortest path. One can terminate the algorithm at this point. This termination rule gives Dijkstra's algorithm for the P2P problem. This algorithm searches a ball with s in the center and t on the boundary.[5]

The shortest path problem is a fundamental problem with numerous applications. This is one of the most common variants of the problem, where the goal is to find a point-to-point shortest path in a weighted, directed graph. This problem is referred to as the P2P problem. It is assume that for the same underlying network, the problem will be solved repeatedly. Thus, preprocessing is allowed with the only restriction that the additional space used to store precomputed data is limited: linear in the graph size with a small constant factor. One can spend time preprocessing maps for these applications, but the underlying graphs are very large, so memory usage much exceeding the graph size is prohibitive.

5) Simulation And Results

A. . *Simulation of Shortest Path Finding*

Proposed algorithm can also be used for finding shortest path from origin node (i.e. node 1) to destination node (i.e. node 7) in a graph. The map will be built similar with the real city. There are several special places in the artificial map same with the real ones, such as main building, library, departments, etc. These special places will be shown in the artificial map as an icon. Each special places has a standard length and width regardless the size of the icon in the map. The map has seven nodes and eleven edges. The map can be seen in the figure below:

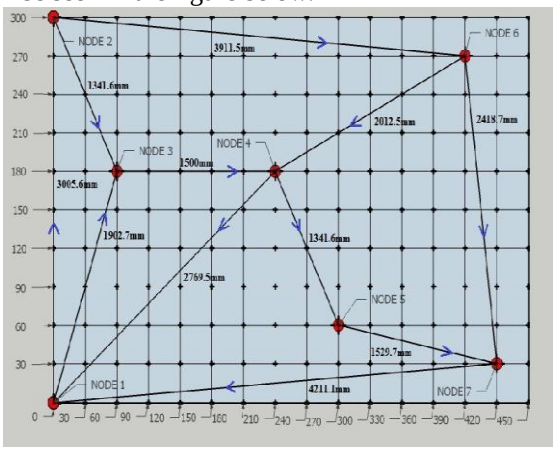

Figure.7. The map of 7 nodes and 11 edges

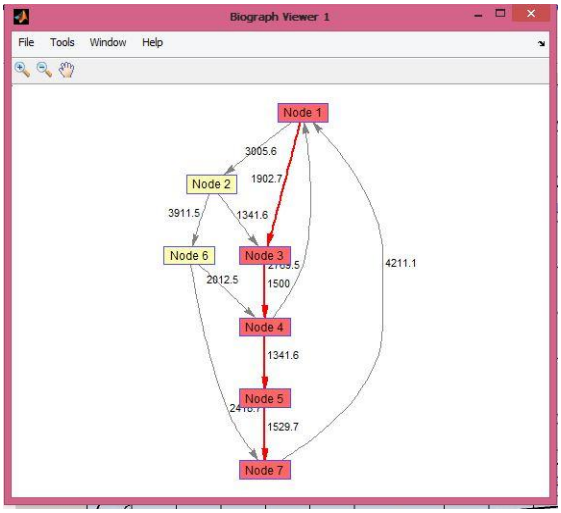

Figure.8. Shortest path result of the map for directed graph

B. *Simulation of Shortest Path Tracking*

The input of the algorithm consists of distances between nodes and a source data from origin and destination paths in the graph. If the distance between two nodes can be defined as path cost, the total cost of a network graph is the sum of all route distances between nodes in a graph.

The positioning of the mobile robot is accomplished the control vector [v(t) ω(t)] must be computed in real time, to make the robot moving from its initial position to any desired goal , respecting its non-holonomic constraints. Controller has been implemented in order to insure the convergence of the robot to the desired goal. Figure 9 is shown the overall block diagram of the system, where E_p is the position error, E_o is the orientation error, v is the robot's velocity and ω is the steering angle of the robot.

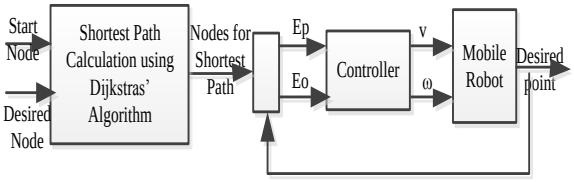

Figure.9. Overall block diagram of the whole mobile robot control system

The simulation of the present work includes motor driving system and control system. Motor driving system was done using kinematics model and its differential drive system. Figure 10 shows the overall flow chat for the whole system.

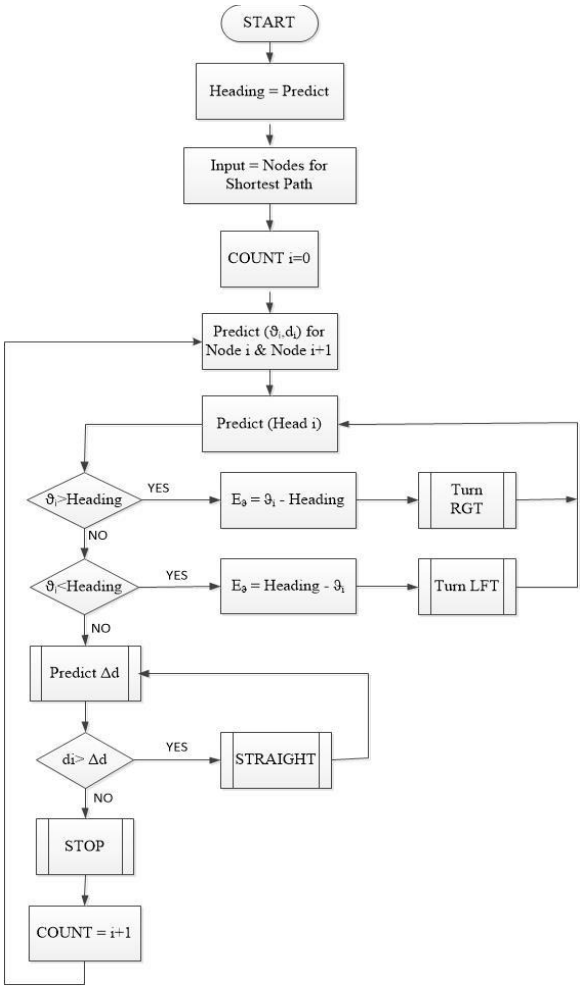

Figure.10. Overall flow chart of the whole mobile robot control system

The effectiveness of the proposed strategy is a computer simulation has been performed. In all tests, robot's localization is computed by Incremental Odometer. The simulation block diagram is using by MATLAB. Figure 11 shows the simulation block diagram for shortest path tracking control for mobile robot.

The user can see the generalized coordinates vector in the inertial frame from the output results of kinematic model for mobile robot. The simulation result is the shortest path tracking of movement. The movement along X axis, Y axis, diagonal. The robot moves from start node to desired node along the shortest path. For this movement, we can see the error position and the robot can reach or not to its desired node exactly. This movement can be seen in figure 12.

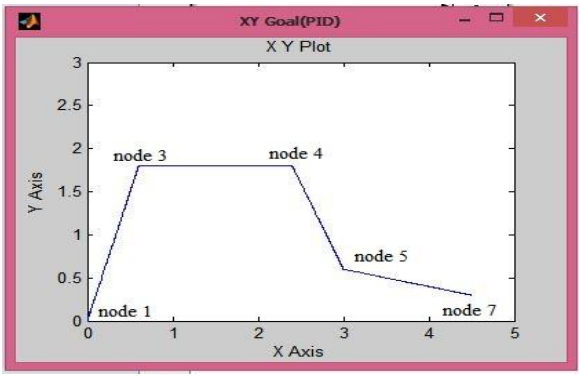

Figure.12. Simulation result of shortest path finding and tracking system for mobile robot

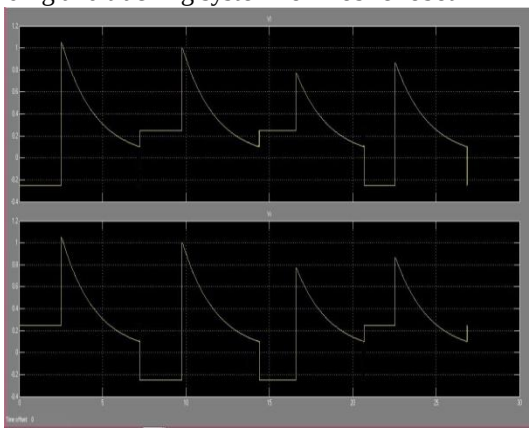

Figure.13. Response curve of the left and right motor

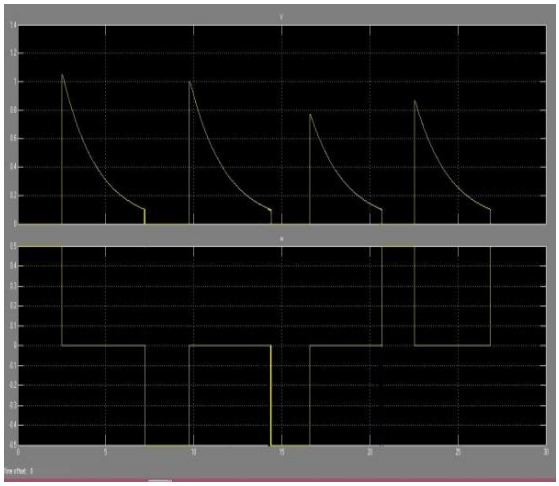

Figure.14. Response curve of the velocity and orientation

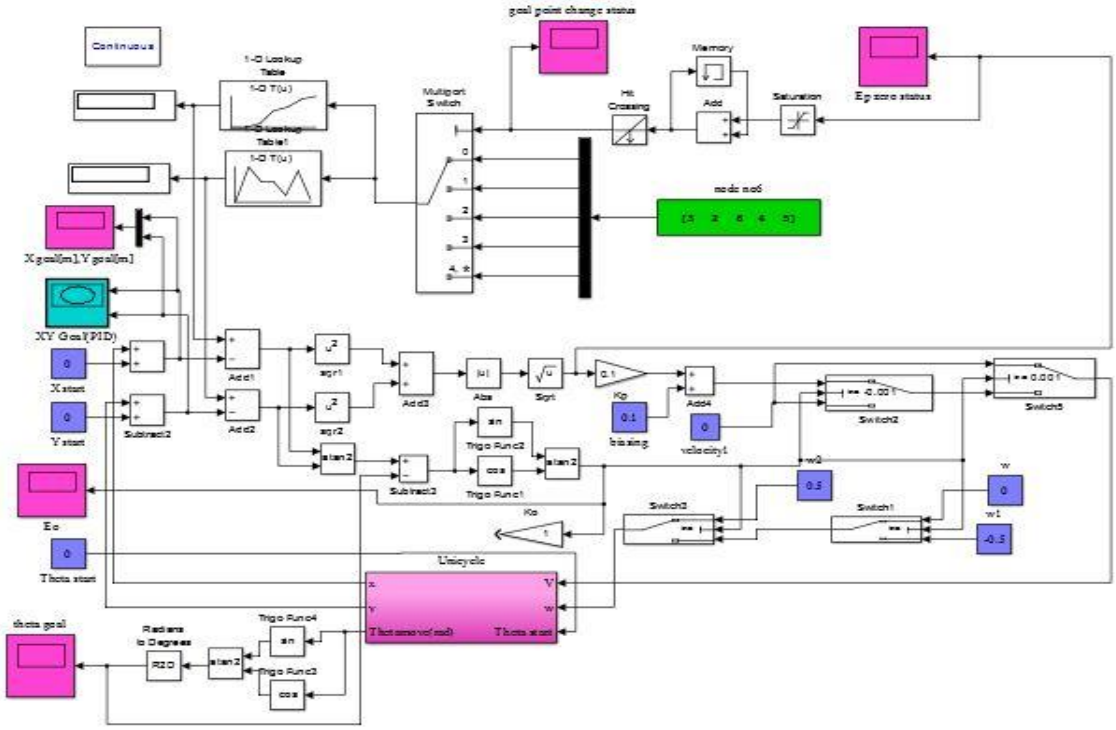

Figure.11. Simulation block for the shortest path finding and tracking system for mobile robot

6)CONCLUSIONS

Shortest Path Problem is an important problem in graph theory and has applications in communications, transportation, and electronics problem. In this paper, Dijkstra's algorithm is used to find the shortest path for directed and undirected graph. Simulation results using Dijkstra's algorithm for any kind of graphs are successfully implemented in MATLAB. In conclusion, the proposed method has been implemented in a shortest path tracking system so that the implement controller has the characteristics of high modularity and probability. By satisfying these facts, the robot can reach to its final point without any error in distance and direction. In addition these mobile robots can be used for communications, transportation, and electronics problem by moving from source point to target point in desired distance and direction in order to find the shortest path. And then, mobile robot can be also candidates for farming applications, as well as for transportation in nuclear plants and factories.

ACKNOWLEDGEMENT

The author is deeply gratitude to Dr. SintSoe, Rector of Mandalay Technological University (MTU) for his kind permission to submit this research. The author would like to thank to her supervisor, DawWaiPhyoEi, Lecturer of Mechatronic Engineering Department, Mandalay Technological University, for her valuable supervision guidance and comments for preparation of the research. The author would like to express her sincere and deepest gratitude to her co-supervisor Dr. KyawThiha, Professor, Mechatronic Engineering Department, Mandalay Technological University, for his suggestions. After that, the author especially appreciates and thanks all her teachers for paper support, and guidance during theoretical study and paper preparation durations.

REFERENCES

1] E. Hart, N.J. Nilsson, B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," IEEE Trans. Systems Science and Cybernetics, vol. SSC-4, no. 2, page no. 100-107, July 1968.

[2] D. Johnson L. McGeoch, "The Traveling Salesman Problem: ACase Study in Local Optimization," Local Search in Combinatorial Optimization,
Princeton Univ. Press, page no. 215-310, 1997

[3] A. Goldberg , C. Harrelson “Computing the Shortest Path: A¤ Search Meets Graph Theory” Technical Report MSR-TR-2004-24, Microsoft Research, 2004.
Algorithms Eng. and Experimentation, pp. 129-143, 2006
[4] K. M. Passino ,P. J. Antsaklis, “Metric space approach to the specification of the Heuristic function for the A* algorithm,” IEEE Transactions on Systems, Man and Cybernetics, vol. 24, no. 1, page no. 159–166, 1994.
[5] R. Agrawal and H.V. Jagadish, “Efficient Search in Very Large Databases,” *Proc. 14th VLDB Conf.* , pp. 407–418, 1988.
[6]Manikonda, V: Hendler, J. A. &Krishnaprasal, P.S.," Formalization behavior-based planning for non-holonomic robots", in Proc International Conference on Artifical Intelligence, pp. 142-149.
[7]*International Conference on Intelligent Robots andSystems*, 1304-1311, Osaka, Japan. Santos-Victor, J.; Minguez, J. & Montano, L. (2002)
[8] Cuesta, F.; Gomes-Bravo, F. &Ollero, A. (2004). Parking Maneuvers of Industrial-Like Electrical Vehicles With and Without Trailer. *IEEE Transaction on Industrial Electronics*, 51(2), 257-269.