International Journal For Technological Research In Engineering
Volume 4, Issue 8, April-2017 *ISSN (Online): 2347 - 4718*

A* ALGORITHM FOR ENHANCING THE SHORTEST PATH PROBLEM DURING REAL TIME IMFORMATION: A REVIEW

Parveen Kumar[1], Dr. Rishu Bhatia[2]
[1]M.Tech(ECE), [2]Asst. Prof. , Dept. Of ECE, GITAM, Kablana

ABSTRACT: To find the shortest distance always suffers struggle and more complexity rather than simple and easy path. Here we are finding the position & shortest path to an object from image taken from camera or any visual sensing device by making use of fuzzy logic & Artificial Intelligence. First we take image of the desired area then take out the desired place where we want to reach as quick and by proper way. By Artificial Intelligence we can find the best way out of all possible good ways to reach. Shortest path not always depends upon the shortest path it may lead the rush, large number of traffic signals, worst condition of road which may cause for delay. So we take all into account to make our modify model for shortest path. Here our model provides the user with more than one way/path to go to the destination. Position tracking system is to find the shortest path by means to save time. Earlier model give the shortest path on the basis of shortest distance, it does not include the time factor where our model include the factors that affect the time to reach the destination earlier than the previous shortest path. Here, we are making use of AI & Fuzzy-Logic to track the position to the object & then finding the shortest path from source to destination. Shortest path is the best path to reach from source to destination & it will make use of AI that helps in decision making power. Proposed algorithm including every factors that we face in our daily life. It give user more ways to choose and provide shortest path in term of very less time as compare to previous model.

Keywords: Shortest Path, A Star Algorithm, AI & Fuzzy-Logic

I. 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. Traffic congestion almost happens all over the world in every day, especially in big cities. The reasons which cause traffic congestion are various, like traffic accident and atrocious weather (for example, heavy rains and snows). In many countries, the government and the institute of environmental protection advocate people to take public transport means, like bus and metro, instead of personal cars to reduce the number of motor vehicles on the road. Besides this, there are other solutions for traffic problems in different countries, like widening the streets, rebuilding the existing roads, building more new roads, and enlarging the area of city. Properly priced on-street parking can greatly reduce traffic congestion.

Figure 1: Worst traffic jams from around the world

The researches about how to solve traffic problems can somehow make traffic flow more efficient, but one of the presumable results of following these measures is to affect the utilization of motor vehicles or increase the cost of using motor vehicles. In this condition, we try to provide other solutions to the traffic problems, by providing a shortest-time route plan to make the traffic flow more time-efficient. By taking the shortest-time path, it will increase traffic control and management, make the motor vehicles running more rapidly and efficiently.

1.1 The Shortest-Time Path

In general, the shortest-time path, as the name implies, is the path which costs the shortest time of all possible paths from one place to another. In order to figure out such path, the time cost of each path should be able to be measured or calculated. However, in reality, the time cost of a journey not only depends on the fixed length of the roads which the driver chooses, but also depends on the complex traffic condition and the variable driving speed, so in reality, the shortest path is probably not the shortest-time path. The hinder factors of traffic are various, like the amount of vehicles, the number of pedestrians, weather condition and the road condition (width, pavement condition, gradient and visibility). The driving speed is also impacted by many factors, such as vehicle type, driver's choice, traffic condition, weather and so on. Although there are methods to calculate the driving speed, like it is studied by Zhao (2010), but in reality, it is impossible to measure the speed of all

vehicles on the roads, therefore, the time cost of motor vehicles' travel is impossible to estimate without measuring it individually. By taking advantages of mobile GNSS and web GIS as background knowledge and technology, and using open source data and tools, the purpose of our study is to combine web GIS and mobile phone with GPS module (which also must be able to connect to internet), to design and develop a web-based application which can provide intelligent vehicle navigation service as in Figure 2.

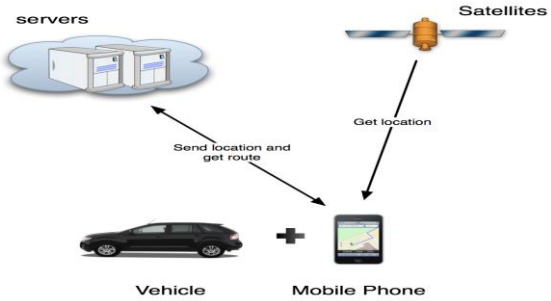

Figure 2: Mobile-based navigation web application

1.2 Dijkstra's Algorithm

A common example of a graph-based path finding algorithm is Dijkstra's algorithm. This algorithm begins with a start node and an "open set" of candidate nodes. At each step, the node in the open set with the lowest distance from the start is examined. The node is marked "closed", and all nodes adjacent to it are added to the open set if they have not already been examined. This process repeats until a path to the destination has been found. Since the lowest distance nodes are examined first, the first time the destination is found, the path to it will be the shortest path. Dijkstra's algorithm fails if there is a negative edge weight. In the hypothetical situation where Nodes A, B, and C form a connected undirected graph with edges AB = 3, AC = 4, and BC = −2, the optimal path from A to C costs 1, and the optimal path from A to B costs 2. Dijkstra's Algorithm starting from A will first examine B, as that is the closest. It will assign a cost of 3 to it, and mark it closed, meaning that its cost will never be reevaluated. Therefore, Dijkstra's cannot evaluate negative edge weights. However, since for many practical purposes there will never be a negative edge weight, Dijkstra's algorithm is largely suitable for the purpose of path finding.

1.3 A Algorithm*

A* is a variant of Dijkstra's algorithm commonly used in games. A* assigns a weight to each open node equal to the weight of the edge to that node plus the approximate distance between that node and the finish. This approximate distance is found by the heuristic, and represents a minimum possible distance between that node and the end. This allows it to eliminate longer paths once an initial path is found. If there is a path of length x between the start and finish, and the minimum distance between a node and the finish is greater than x, that node need not be examined. A* uses this heuristic to improve on the behavior relative to Dijkstra's algorithm. When the heuristic evaluates to zero, A* is equivalent to Dijkstra's algorithm. As the heuristic estimate increases and gets closer to the true distance, A* continues to find optimal paths, but runs faster (by virtue of examining fewer nodes). When the value of the heuristic is exactly the true distance, A* examines the fewest nodes. (However, it is generally impractical to write a heuristic function that always computes the true distance.) As the value of the heuristic increases, A* examines fewer nodes but no longer guarantees an optimal path. In many applications (such as video games) this is acceptable and even desirable, in order to keep the algorithm running quickly.

1.4 SHORTEST PATH BY A ALGORITHM*

A* is a computer algorithm that is widely used in pathfinding and graph traversal, the process of plotting an efficiently traversable path between multiple points, called nodes. Noted for its performance and accuracy, it enjoys widespread use. However, in practical travel-routing systems, it is generally outperformed by algorithms which can pre-process the graph to attain better performance,[1] although other work has found A* to be superior to other approaches.

Peter Hart, Nils Nilsson and Bertram Raphael of Stanford Research Institute (now SRI International) first described the algorithm in 1968. It is an extension of Edger Dijkstra's 1959 algorithm. A* achieves better time performance by using heuristics.

A* uses a best-first search and finds a least-cost

II. LITERATURE SURVEY

Mobile robots are more efficient than legged or treaded robots on hard as well as smooth surfaces, and have potential enough to find widespread application in industry, because of the hard, smooth plant floors in existing industrial environments [1]. Several configurations for mobility can be found in the applications as mentioned by Jones et al. [2]. The most common form single-body robots are the differential drive and synchronic drive tricycle or car-like drive, and omnidirectional steering robots [3]. Besides the relevance in applications, the problem of autonomous motion planning and control of mobile robot has attracted the interest of many researchers to view its theoretical challenges [4]. The motion control of wheeled mobile robots has been able to draw considerable attention over the past few years. The nonholonomic behaviour in robotic systems is particularly interesting; since it points out that the mechanism can be completely controlled by using a reduced number of actuators.

Particularly, these systems are typical examples of nonholonomic mechanisms due to the perfect application of the rolling constraints on the wheel motion [5]. Several controllers have been proposed for the motion control of mobile robots with nonholonomic constraints, where the two main approaches to controlling mobile robots are posture stabilization and trajectory tracking.

The procedure of modelling can be inspired by definition of a wheeled mobile robot according to Muri and Neuman [6] as this, "A robot capable of locomotion on a surface solely through the actuation of wheel assemblies mounted on the robot and in contact with the surface. A wheel assembly is a

device that provides a relative motion between its mount and a surface on which it is intended to have a single point of contact." However it is required that the vehicle kinematic design has the appropriate degrees of freedom (mobility) so that it can adapt to the variations in the surface and the wheels roll without slip. Mobility is enhanced by the use of omnidirectional wheels instead of conventional wheels [7]. The necessity of ideal rolling without sideways slipping for wheels enforces non-holonomic (nonintegrable) constraints on the motion of the wheels of mobile robot [8]. The relation between the rigid body motion of the robot and the steering and drive rates of wheels has been developed by Alexander and Maddocks [9] based on constraint as "rolling without sliding". Slippage due to misalignment of the wheels is investigated here by minimization of a nonsmooth convex dissipation functional that is derived from Coulomb's Law of friction. This minimization principle is equivalent to the construction of quasi-static motions.

Three related but different kinematical aspects have to be considered when designing a robot. They can be listed as mobility, control and positioning [10, 11]. The first one, mobility, deals with the possible motions that the robot can follow in order to reach its final destination in any orientation. The second aspect, control, relates to the choice of the kinematical variables: generalized velocities or coordinates. Finally, the third aspect, positioning, considers the localization system that is used to estimate the actual position and orientation of the robot by reducing the robot's region of uncertainty based on sensor measurements necessary to achieve an autonomous operation [12].

The motion along the configuration space is limited using the kinematic constraints. Kinematic limitations can be applied at any speed, while dynamic constraints are important to apply as an agent operates at higher speeds. Robot design has to tackle agent dynamics issues, as even a holonomic robot without any kinematic constraints will have to face some form of dynamics limitations, and in particular bounds on acceleration and velocity.

Dynamics constraints limit the acceptable values for derivatives of an agent's position over Time

Moon et al. [13] have proved that a wheeled mobile robot is not able to move along a straight line exactly, even if the kinematic problems are corrected perfectly, and this phenomenon is related to acceleration constraints on motor controllers. Kinematic model of a parallel wheeled mobile robot fails to meet Brockett's necessary condition for feedback stabilization thereby implying that no smooth or continuous time invariant. Stabilization and control of nonholonomic systems with dynamic equations have been considered in [14] whereas back stepping based methods are presented in several papers [15, 16, 17].

Internal error occurs from unsuitable setting up of the parameters and the time constant. External error inescapably appears when a WMR is being driven and it occurs by virtue of the two driving wheel's different friction and radius. In order to minimize such errors, Chung et al. [18] has proposed a feedback controller having two separated feedback loops; one of which is a position feedback, and the other an orientation feedback.

Based on back stepping algorithm, a robust adaptive controller has been proposed in [19, 20] to design an auxiliary wheel velocity controller in order to make the tracking error as small as possible as compared to the uncertainties in the kinematics of the robot and fuzzy logic techniques employed to learn the behaviours of the unknown dynamics of the robot and the wheel actuators. The major advantage of this method is that previous knowledge of the robot kinematics and the dynamics of the robot and wheel actuators is unnecessary. The parameters characterizing the robot dynamics are to be updated online, thereby providing smaller errors and better performance in applications in which these parameters can vary, such as load transportation. The stability of the whole system is analyzed using Lyapunov theory, and the control errors are ultimately bounded [21].

Deng et al. [22] designed a combined feedback control scheme based on Lyapunov function candidate [23] has been discussed for four obstacle cases in dynamic environments considering local minima problem. The controller includes virtual attractive force, repulsive force and detouring force, whereas the potential field function used for the design of the controller considers the Euclidean distance information and the magnitude information of the relative velocity between the robot and the target [24].

A dynamic model of a two-wheeled mobile robot has been derived in [25, 26] which shows that the translational motion and the rotational motion with 3 degrees of freedom of the body and here, the dynamic model is reduced to the kinematic model under certain assumptions. Arvin et al. [27] have presented mobile robots motion control technique based on pulse-width modulation (PWM).

The wheels of mobile robot have been modelled as a torus by Chakraborty and Ghosal [28] and used as a passive joint thereby enforcing a lateral degree of freedom so as to get a slip free motion in an uneven terrain without using variable length axle (VLA) as it has several limitations in application. Zhang et al. [29] have developed a feedback control law [30, 31], allowing a 2-wheel differentially driven mobile robot to track a prescribed trajectory by using the integral backstepping method and Lyapunov function for ensuring a trajectory tracking controller with global asymptotic stability.

Zohar et al. [32] recently proposes control schemes for trajectory tracking of mobile robot model which includes kinematic and dynamic effects on motion by using the notion of virtual vehicle [33] and the concept of flatness [34], and applying the backstepping [35] methodology. Gandhi and Ghorbel [36] have proposed the harmonic drive system for non-linear controller to compensate for kinematic error in the presence of flexibility in high-speed regulation and trajectory tracking application. Pathak et al. [37] have discussed the behaviour of space robots with torque and attitude controller. A single curvature trajectory, having a constant and large rotation radius, has been proposed by Han et al. [38] as an optimal trajectory, in order to minimize the tracking error of the differential drive mobile robot while capturing a moving object along with the pre-determined initial and final states. A receding horizon controller may be used for tracking

control of wheeled mobile robots subject to non holonomic constraint in the environments without obstacles. The control policy is derived from the optimization of a quadratic cost function, which penalizes the tracking error and control variables in each sampling time [39, 40].

III. EARLIER WORKS

3.1 NAVIGATION USING FUZZY LOGIC CONTROLLER FOR MOBILE ROBOT

Fuzzy Logic technique plays an important role in the designing of an intelligent controller for mobile robot. This technique is used for navigation of mobile robots. Fuzzy set theory provides a mathematical framework for representing and treating uncertainty in the sense of vagueness, imprecision, lack of information and partial truth. Fuzzy control systems employ a mode of approximate reasoning that resembles the decision-making process of humans.

3.2 FUZZY DECISIONS AND CONTROL ALGORITHM DESIGN

The robot operates in a 2 ×2.4 m2 arena (viewed entirely by the web cam) with moving obstacles and a target to be reached. The positions of the target and of the obstacles are not known in advance; therefore the navigation algorithm has to implement a reactive paradigm relying only on sensory information. At first, two simple behaviors, namely reach the target and avoid obstacles, are carried out with two different fuzzy controllers, hereinafter called FLC1 and FLC2 respectively. The reach the target behavior as well as avoid obstacles behavior depends on artificial vision information and is the primary task for the mobile robot It has the highest priority and takes place only if an obstacle appears on the robot path. Subsequently, a fuzzy supervisor takes charge to combine the reference wheel speeds calculated by each FLC following a priority code. The final commanded speeds are sent to the built in speed control loop of the robot. These controllers are sufficient to guarantee satisfactory navigation performances for the mobile robot in most of the navigation tasks. The explore the environment behavior makes the mobile robot mark regions already visited and look for unexplored areas. The mobile robot is endowed with a type of spatial local memory, which is used by a further fuzzy controller, henceforth called FLC3, to localize and avoid the box canyons. The structure of the whole control scheme is shown in Fig.4. The modular architecture of our controller has the following main advantages with respect to a monolithic solution: 1. Debugging and tuning operations are faster and easier since each behavior is described by few rules and inputs; 2. The final structure is more flexible as new simple behaviors can easily be added in order to expand mobile robot skill. Reach the target This behavior reacts to the stimuli of the vision system, providing information about the relative position between mobile robot and target. The behavior ignores the presence and position of obstacles. The robot is equipped with two different diameter circles of same color on top for position and orientation detection, and the target is marked with a red ball. The image processing is based on conventional colour coding on RGB formats using the Matlab image Processing Toolbox. The information captured by the vision system is updated every 200 ms and passed through mlib/mtrace to the FLC1 which is running on the dSpace board. In this behavior, the robot firstly turns until it is aligned to the target, and then moves in a straight line. The information on the distance (DIST) between robot and target, and the alignment error (DIR) of the robot is provided by the vision system passed as inputs to FLC1.

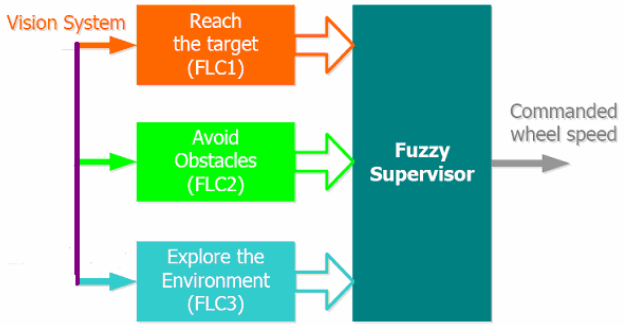

Fig.4- Behavior based control scheme

Obstacle Avoidance The distance between the robot and the target causes the robot seeking towards the target when the robot is very close to the target. Similarly when the robot is very close to an obstacle, because of it the robot must change its speed and heading angle to avoid the obstacle.All the rules in those tables have been obtained heuristically using common sense. Some rules for extreme conditions when the obstacles have to be avoided as quickly as possible. This is for three-membership function.If the left obstacle distance is “near”, right obstacle distance is “far”, front obstacle distance is “medium” and no unobstructed target is around the robot, then the robot should turn to right side as soon as possible to avoid collision with the left obstacle. For the above condition the left wheel velocity should increase fast and right wheel velocity should decrease slowly. Similarly some rules are used for extreme conditions when the obstacles have to be avoided as soon as possible. These rules are for five-membership function. For example in rule 12, the left obstacle distance is “ very far”, right obstacle distance is “near”, front obstacle distance is “ very near” and no target is located around the robot, then the robot should turn to left side to avoid collision with the obstacle in front and towards right of it. For the above condition the right wheel velocity should increase very fast and left wheel velocity should decrease very slowly.

IV. PROPOSED WORK

To find the shortest distance always suffers struggle and more complexity rather than simple and easy path. Here we are finding the position & shortest path to an object from image taken from camera or any visual sensing device by making use of fuzzy logic & Artificial Intelligence[8]. First we take image of the desired area then take out the desired place where we want to reach as quick and by proper way [6]. By Artificial Intelligence we can find the best way out of all possible good ways to reach. Shortest path not always depends upon the shortest path it may lead the rush, large number of traffic signals, worst condition of road which may cause for delay [7]. So we take all into account to make our modify model for shortest path. Here our model provides the

user with more than one way/path to go to the destination.

4.1 REAL TIME INFORMATION MODEL FOR SHORTEST PATH

- Real Time information model with grid algorithm is the proposed Algorithm for shortest path problem.
- In this algorithm, we provide real time information to shortest path algorithm.
- In this we include various factors like jamming, damage of roads, no. of traffic signals, rush which affects the shortest path in time domain

4.2 A ALGORITHM FOR REAL TIME SYSTEM*

Construct grid, where a * = obstacle and you can move up, down, left and right, and you start from S and must go to D, and 0 = free position:

```
S 0 0 0
* * 0 *
* 0 0 *
0 0 * *
* 0 0 D
```

You put S in your queue, then "expand" it:

```
S 1 0 0
* * 0 *
* 0 0 *
0 0 * *
* 0 0 D
```

Then expand all of its neighbours:

```
S 1 2 0
* * 0 *
* 0 0 *
0 0 * *
* 0 0 D
```

And all of those neighbours' neighbours:

```
S 1 2 3
* * 3 *
* 0 0 *
0 0 * *
* 0 0 D
```

And so on, in the end you'll get:

```
S 1 2 3
* * 3 *
* 5 4 *
7 6 * *
* 7 8 9
```

So the distance from S to D is 9. The running time is O(NM), where N = number of lines and M = number of columns. I think this is the easiest algorithm to implement on grids, and it's also very efficient in practice. It should be faster than a classical dijkstra..

The following program calculates the minimum point of a multi-variable function using the grid search method. This method performs a multi-dimensional grid search. The grid is defined by a multiple dimensions. Each dimension has a range of values. Each range is divided into a set of equal-value intervals. The multi-dimensional grid has a centroid which locates the optimum point. The search involves multiple passes. In each pass, the method local a node (point of intersection) with the least function value. This node becomes the new centroid and builds a smaller grid around it. Successive passes end up shrinking the multidimensional grid around the optimum.

The function Grid_Search has the following input parameters:
N - number of variables
XLo - array of lower values
XHi - array of higher values
NumDiv - array of number of divisions for each range
MinDeltaX - array of minimum ranges
Eps_Fx - tolerance for difference in successive function values
MaxIter - maximum number of iterations
myFx - name of the optimized function
The function generates the following output:
X - array of optimized variables
BestF - Function value at optimum
Iters - number of iterations
Here is a sample session to find the optimum for the following function:
y = 10 + (X(1) - 2)^2 + (X(2) + 5)^2
The above function resides in file fx1.m. The search for the optimum 2 variables has the search range of [-10 -10] and [10 10] with a divisions vector of [4 5] and a minimum range vector of [1e-5 1e-5].
The search employs a maximum of 10000 iterations and a function tolerance of 1e->> [XBest,BestF,Iters]= Grid_Search(2, [-10 -10], [10 10], [4 4], [1e-5 1e-5], 1e-7, 10000, 'fx1')
XBest =2.0001 -5.0000
BestF =10.0000
Iters = 200
Notice how close the located optimum is to the actual one [-2 5].
Here is the MATLAB listing:

```
Function y=fx1(X, N)
y = 10 + (X(1) - 2)^2 + (X(2) + 5)^2;
end
function [XBest,BestF,Iters]=Grid_Search(N, XLo, XHi, NumDiv, MinDeltaX, Eps_Fx, MaxIter, myFx)
% Function performs multivariate optimization using the
% grid search.
% Input
% N - number of variables
% XLo - array of lower values
% XHi - array of higher values
% NumDiv - array of number of divisions along each dimension
% MinDeltaX - array of minimum search values for each variable
% Eps_Fx - tolerance for difference in successive function values
% MaxIter - maximum number of iterations
% myFx - name of the optimized function
% Output
% XBest - array of optimized variables
% BestF - function value at optimum
% Iters - number of iterations
```



```
Xcenter = (XHi + XLo) / 2;
XBest = Xcenter;
DeltaX = (XHi - XLo) ./ NumDiv;
BestF = feval(myFx, XBest, N);;
if BestF >= 0
  LastBestF = BestF + 100;
else
  LastBestF = 100 - BestF;
end
X = XLo; % initial search value

Iters = 0;
bGoOn = 1;

while (bGoOn > 0) && (abs(BestF - LastBestF) > Eps_Fx)
&& (Iters <= MaxIter)

bGoOn2 = 1;
while bGoOn2 > 0
 Iters = Iters + 1;

 F = feval(myFx, X, N);

 if F < BestF
     LastBestF = BestF;
     BestF = F;
     XBest = X;
   End
```

The next For loop implements a programming trick that simulated nested loops using just one For loop search next grid node

```
   for i = 1:N
     if X(i) >= XHi(i)
       if i < N
         X(i) = XLo(i);
       else
         bGoOn2 = 0;
         break
       end
     else
       X(i) = X(i) + DeltaX(i);
       break
     end
   end

 end
 while bGoOn2 > 0
  XCenter = XBest;
  DeltaX = DeltaX ./ NumDiv;
  XLo = XCenter - DeltaX .* NumDiv / 2;
  XHi = XCenter + DeltaX .* NumDiv / 2;
  X = XLo; % set initial X
  bGoOn = 0;
  for i=1:N
    if DeltaX(i) > MinDeltaX(i)
      bGoOn = 1;
    end
  end
end
while bGoOn > 0 && () && ()
```

4.3 MATHEMATICAL ANALYSIS OF PROPOSED ALGORITHM

Integrate the real time model to the Grid algorithm
Inputs: Size of the Grid (N); No. of Paths: Z; Source and Destination Coordinates of each of the individual Z paths
Given: Each path comprises of the cells which are ADJACENT to each other and NOT diagonal. Time taken to cross each cell is EQUAL and is an unit time.
Output: Z no. of paths which are SHORTEST possible without clash.
Z(t) is time domain for AI
$\acute{Z}(t) = Mean(z)$;
$\acute{Z}(t+\delta t)= \acute{Z}(t)+R\text{-}1(Z)*\blacktriangledown z* z(t+ \delta t)$

4.4 HOW SHORTEST PATH CALCULATION PROCESSED BY REAL TIME INFORMATION MODEL

- Shortest path is being calculated by Grid Algorithm.
- It also give the position of origin and destination.
- After that we include the factors that affects the shortest path.
- We get the shortest path in real time information model that is more suitable for user.

V. CONCLUSION

- The use of real time information give us more dynamic model to find the shortest path.
- This real information makes it a best model for future.
- This enhancement brings us more closer to real time situations & save time for travelers.

FUTURE WORK

In our study, all the hinder factors are fixed constructions, we provide the path with less hinders to the traffic and driving speed. The method may serve as a reference while doing urban planning. Careful planning of building roads close to those constructions (schools, hospitals, traffic lights and residential areas) will diminish the risk of traffic congestion. To the researchers and students of Geo-science and computer science, the method to implement the function of our application can be a reference when they do similar work and study. Besides this, as utilizing various techniques and tools in the application, like GIS, traffic analysis, web development and so on, our application will be useful to people who are interested in these techniques or tools. The application works according to the description, but there are problems that should be solved in the further study. The algorithm needs to be improved in order to speed up the long response time. In the future, we hope It can maintain the application of shortest path project regularly and improve the method, to make our study be usable to the public.

REFERENCES

[1] Dudgeon, D.E. and R.M. Mersereau, Multidimensional Digital Signal Processing. 1984, Englewood Cliffs, New Jersey: Prentice-Hall.

[2] Castleman, K.R., Digital Image Processing. Second ed. 1996, Englewood Cliffs, New Jersey: Prentice-Hall.
[3] Oppenheim, A.V., A.S. Willsky, and I.T. Young, Systems and Signals. 1983, Englewood Cliffs, New Jersey: Prentice-Hall.
[4] Papoulis, A., Systems and Transforms with Applications in Optics. 1968, New York: McGraw-Hill.
[5] Russ, J.C., The Image Processing Handbook. Second ed. 1995, Boca Raton, Florida: CRC Press.
[6] Giardina, C.R. and E.R. Dougherty, Morphological Methods in Image and Signal Processing. 1988, Englewood Cliffs, New Jersey: Prentice–Hall. 321.
[7] Gonzalez, R.C. and R.E. Woods, Digital Image Processing. 1992, Reading, Massachusetts: Addison-Wesley. 716.
[8] Goodman, J.W., Introduction to Fourier Optics. McGraw-Hill Physical and Quantum Electronics Series. 1968, New York: McGraw-Hill. 287.
[9] Heijmans, H.J.A.M., Morphological Image Operators. Advances in Electronics and Electron Physics. 1994, Boston: Academic Press.
[10] Hunt, R.W.G., The Reproduction of Colour in Photography, Printing & Television,. Fourth ed. 1987, Tolworth, England: Fountain Press.
[11] Freeman, H., Boundary encoding and processing, in Picture Processing and Psychopictorics, B.S. Lipkin and A. Rosenfeld, Editors. 1970, Academic Press: New York. p. 241-266.
[12] Stockham, T.G., Image Processing in the Context of a Visual Model. Proc. IEEE, 1972. 60: p. 828 - 842.
[13] Murch, G.M., Visual and Auditory Perception. 1973, New York: Bobbs-Merrill Company, Inc. 403.
[14] Frisby, J.P., Seeing: Illusion, Brain and Mind. 1980, Oxford, England: Oxford University Press. 160.

