Mathematical Modeling and Simulation of Target Oriented Obstacle Avoidance Robot

Subhasis Behera
National Institute of Technology Jalandhar,
Subhasis.behera@gmail.com

Abstract- The aim of the project is to design an autonomous robot platform capable of reaching a desired destination starting from a known position in an environment filled with unknown obstacles. We are using the **Virtual Force Field (VFF)** method by Dr. J. Borenstein for the navigation purpose. The VFF method is the integration of two concepts; Certainty grids for obstacle representation and potential fields for navigation. The system can be summarized as follows. In the first phase distance measuring ultrasonic sensors mounted strategically on the robot chassis scans the environment and represents the obstacles in a grid-type world model. This grid-type world model has been developed at the Carnegie-Mellon University (CMU). In the certainty grid, the robot's work area is represented by a two-dimensional array of square elements, denoted as cells. Each cell contains a certainty value (CV) that indicates the measure of confidence that an obstacle exists within the cell area. The ultrasonic sensor has a conical field of view and returns a radial measure of the distance to the nearest obstacle within the cone. In the second phase the idea of imaginary forces acting on the robot has been considered. Location of the obstacles and the target with respect to the robot are fed into the computer or microcontroller. The obstacles exert repulsive forces, while the target applies an attractive force to the robot. A resultant force **R,** comprising the sum of a target-directed attractive force and repulsive forces from obstacles, is calculated for a given robot position. This resultant force has a specific magnitude and direction and the robot follows the same. Traditionally such systems use 12 to 24 distance measuring ultrasonic sensors placed circumferentially around the robot chassis. However due to cost considerations, we are using only one sensor mounted on a stepper motor (SRF-08 by devantech). The sensor will take the range readings for multiple positions by revolving the stepper motor to specific locations about the chassis. This interferes with the robot's overall motion and our system may not match the speed of the traditional systems. However to overcome the above problem, we are modifying the ranging technique so that the time spent on the ranging operation can be minimized to some extent. The system takes the range readings for the entire circumference at 36-degree interval once every five readings. This information is updated for the successive readings. In the 4 successive readings the system takes 3 ranging readings across 54 degree about the line joining the robot position with the destination point. The readings for the remaining positions are extrapolated from the first reading. The problem of low speed can be partially solved by the above approach. However it will introduce some inaccuracies in the location of the obstacles. This can be taken care by giving higher weightage to the current readings than the extrapolated readings. A rough weightage will be estimated from the MATLAB model of the above system.

The hardware of the system consists of a differentially driven wheeled platform, driven by geared stepper motors and an ultrasonic rangefinder SRF-08 mounted on a stepper motor. SRF-08 gives the distance of the measured obstacle in IIC protocol. The sensor and the stepper motors communicate with the computer through the data pins of parallel port.

I. Introduction

Obstacle Avoidance Robots (OARs) have been an active area of research and development over the past three decades. This long-term interest has been mainly fueled by the myriad of practical applications that can be uniquely addressed by mobile robots due to their ability to work in large (potentially unstructured and hazardous) domains. The types obstacles can be classified into two types.

1.1 Stationary Obstacles:

Walls, heavy furniture like beds, tables, chairs etc fall into this category.

1.2 Non-Stationary Obstacles: These obstacles result from "disorder" in the environment. The positional coordinates of these obstacles keep on changing. In our application, detected obstacles are added to the temporary map, & consequently processed like stationary obstacles until the temporary readings are finally extrapolated and overwritten by the new readings. Thus, the real time obstacles can also be taken care by temporarily considering them as stationary obstacles.

Trajectory following: A reference point on the robot must follow a trajectory in the Cartesian space (i.e., a geometric path with an associated timing law) starting from a given initial configuration.
One approach to obstacle avoidance is wall-following method. Here robot navigation is based on moving alongside walls at a predefined distance. If an obstacle is encountered, the robot regards the obstacle just as another wall following its contour until it may resume its original course. A more commonly employed method for obstacle avoidance is based on edge detection. The line connecting the two edges is considered to represent one of the obstacle's boundries. A disadvantage with the edge detection method is the need of the robot to stop in front

of an obstacle in order to allow for a more acurate measurement.

The system can be summarized as follows. In the first phase distance measuring ultrasonic sensors mounted strategically on the robot chassis scans the environment and represents the obstacles in a grid-type world model. In the certainty grid, the robot's work area is represented by a two-dimensional array of square elements, denoted as cells. Each cell contains a certainty value (CV) that indicates the measure of confidence that an obstacle exists within the cell area. In the second phase the idea of imaginary forces acting on the robot has been considered. The obstacles exert repulsive forces, while the target applies an attractive force to the robot. A resultant force **R,** comprising the sum of a target-directed attractive force and repulsive forces from obstacles, is calculated for a given robot position. This resultant force has a specific magnitude and direction and the robot follows the same. The modified system takes the range readings for the entire circumference at 36-degree interval once every five readings. This information is updated for the successive readings. In the 4 successive readings the system takes 3 ranging readings across 54 degree about the line joining the robot position with the destination point. The readings for the remaining positions are extrapolated from the first reading. The problem of low speed can be partially solved by the above approach. However it will introduce some inaccuracies in the location of the obstacles. This can be taken care by giving higher weightage to the current readings than the extrapolated readings. A rough weightage will be estimated from the MATLAB model of the above system.

II. TheVirtual Force Field (VFF) Method[4] :The idea of objects conceptually exerting forces onto a mobile robot has been suggested by Khatib. Our implementation treats each ultrasonic range reading as a repulsive force vector. The reading from the target acts as an attractive force vector. If the magnitude of the sum of the repulsive forces exceeds a certain threshold , the robot stops, turns into the direction of the resultant force vector, and moves on.

The Basic VFF Method[4] : As the robots move around, range readings are taken and projected into the Certainity Grid. Each occupied cell inside the Certainity grid window applies a repulsive force to the robot, "pushing" the robot away from the cell. The magnitude of this force is proportional to the cell contents, C(i,j), and inversely proportional to the square of the distance between the cell and the robot.

$$F(i,j) = \frac{F_{cr} C(i,j)}{d^2(i,j)} \left(\frac{x_t - x_0}{d(i,j)} \hat{x} + \frac{y_t - y_0}{d(i,j)} \hat{y} \right)$$

where

F_{cr} =Force Constant (repelling)

$d(i,j)$ = Distance between cell (i,j) and the robot

$C(i,j)$ = Certainty level of cell (i,j)

(x_0, y_0) =Robot's present coordinates

(x_i, y_j) =Coordinates of cell (i,j)

The resultant repulsive force, F_r, is the vectorial sum of the individual forces from all the cells.

$$F_{cr} = \sum_{i,j} F(i,j) = F_r$$

At any time during the motion, a constant-magnitude attracting force, F_t, pulls the robot toward the target. F_t is generated by the target point T, whose coordinates are known to the robot . The target-attracting Force, F_t is given by

$$F(t) = F_{ct} \left(\frac{x_t - x_0}{d(t)} \hat{x} + \frac{y_t - y_0}{d(t)} \hat{y} \right) = F_t$$

where,

F_{ct} = Force constant (attraction to the target)

d_t = Distance between the target and the robot

x_t, y_t = Target coordinates

Note: $F(t)$ is independent of the absolute distance to the target.

The vectorial sum of all forces, repulsive from occupied cells and attractive from the target position, produces a **resultant force vector R:**

$$R = F_t + F_r$$

The direction of R, $\delta = R/|R|$ (in degrees), is used as a reference for the robot's steering-rate command.

III. The kinematical model:

The kinematical model for the obstacle avoidance robot (OAR) under the nonholonomic constraint of pure rolling and non-slipping is given as follows.

$$\dot{q}_1 = v_1 * \cos q_3 \quad \ldots \ldots \ldots \ldots (3.1)$$

$$\dot{q}_2 = v_1 * \sin q_3 \quad \ldots \ldots \ldots \ldots (3.2)$$

$$\dot{q}_3 = v_2 \quad \ldots \ldots \ldots \ldots (3.3)$$

v_1 = The longitudinal velocity applied to the vehicle

v_2 = The instantaneous angular deflection provided to the wheels of the vehicle.

v_1 and v_2 depend on k1 and k2 respectively.

Where

k_1 = Forward velocity constant

k_2 = Angular deflection constant of the wheels

$$q = \begin{bmatrix} x_c \\ y_c \\ \theta_c \end{bmatrix} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (3.4)$$

$$\dot{q} = \begin{bmatrix} \dot{x}_c \\ \dot{y}_c \\ \dot{\theta}_c \end{bmatrix} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (3.5)$$

q is the instantaneous position of the origin of the reference frame attached to the vehicle.

x_c (t) and y_c (t) denote the position of the center of mass of the Obstacle Avoidance Robot (OAR) along the X and Y Cartesian coordinate frames.

And velocity vector v(t) is defined as

$$v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} v_1 \\ \theta_d \end{bmatrix}$$

The vehicle is to start at a position (x, y, θ) and has to reach a given point (x_d, y_d, θd) with respect to the global reference plane.

The model is a tricycle type model having two rear wheels driven independently and a front wheel on a castor. The ultrasonic sensor (sonar) is strategically mounted on a rotating platform. The platform is rotated at a constant angle by the geared stepper motor at specified constant intervals. Thus, a single sensor maps its complete environment (360-degree). This modified scheme cuts down the need of using 12 or 24 ultrasonic sensors for the mapping purpose, thereby reducing the overall cost of the system. The disadvantage comes in the form of overhead time taken to do so. Since here a single sensor is replicating 12 sensors and that too with the same efficiency. But this can be taken care by including certain modification in the VFF algorithm.
The modified system takes the range readings for the entire circumference at 36-degree interval once every five readings. This information is updated for the successive readings. In the 4 successive readings the system takes 3 ranging readings across 54 degree about the line joining the robot position with the destination point. The readings for the remaining positions are extrapolated from the first reading. The problem of low speed can be partially solved by the above approach.

IV. The MATLAB model

The behavior of the model and the strategy can be tested if we can obtain the trajectory of the path, when subjected to a given set of conditions. For that we need to get all the values of the state variables (q_1, q_2, q_3) at small intervals of time, which can later be plotted to obtain the trajectory of the path followed. Hence the above set of first order differential equations have to be integrated in a time interval; given the values of the initial conditions and parameters using ode23; A powerful tool of matlab. Ode23 is a function for the numerical solution of ordinary differential equations. It can solve simple differential equations or simulate complex dynamical systems. It integrates a system of ordinary differential equations using 2nd & 3rd order Runge-Kutta formulas. This particular 3rd-order method reduces to Simpson's 1/3 rule and uses the 3rd order estimate for xout. The process of ode23 is as follows: A string variable with the name of the M-file that defines the differential equations to be integrated. The function needs to compute the state derivative vector, given the current time and state vector. It must take two input arguments; scalar t (time) and column vector q (state), and return output argument qdot, a column vector of state derivatives. The above set of first order differential equations was converted into the following M-file, to execute ode23.

4.1 Testing the strategy to reach destination points:

The above model is modeled in MATLAB and the strategy is tested for the vehicle to reach different destination points in all the four different quadrants. The matlab program has been included which explains the constants k1 and k2.
Given a global reference plane in which the instantaneous position and orientation of the model is given by (q(1), q(2), q(3)) with respect to the global reference system. The vehicle is to start at a position (x, y, θ) and has to reach a given point (x_d, y_d, θd) with respect to the global reference plane.
The longitudinal axis of the reference frame attached to the vehicle and the lateral axis perpendicular to the longitudinal axis. Since this reference frame's position changes continuously with respect to the global reference system, the instantaneous position of the origin of the reference frame attached to the vehicle is given by (q_1, q_2, q_3).

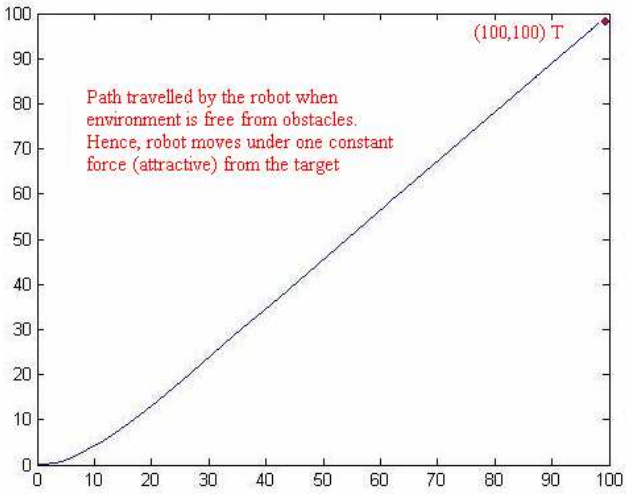

Fig.1 The trajectory of the robot vehicle in an environment free of obstacles.

In Fig.1 the robot is initially at origin. There are no obstacles in the environment, thus the robot moves under a constant attractive force from the target. Hence the path of the robot is a straight line. It moves on the displacement line joining the robot with the target T.

Fig. 2 simulates the path of the robot vehicle reaching the target, which has been placed in the 2nd quadrant.

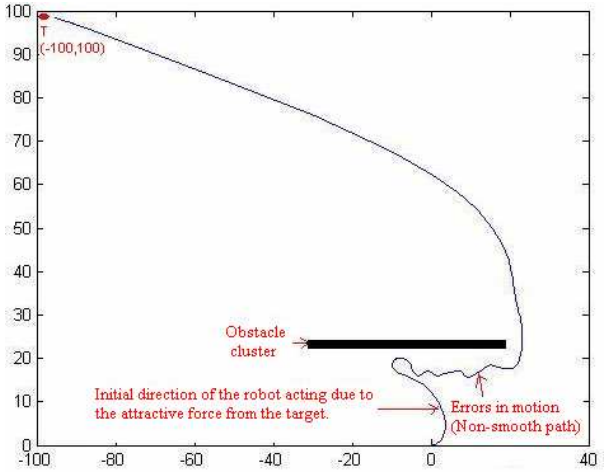

Fig2(a). The trajectory of the robot vehicle in an environment with obstacle clusters.

The target is in the 2nd quadrant at (-100,100). The robot initially moves along the displacement line joining the robot with the target. But as soon as the ultrasonic sensor maps the obstacle cluster, the repulsive forces begin acting. Thus, the robot then calculates the resultant of the attractive and the repulsive forces and traverses in the direction of the resultant force. The speed of motion depends on the magnitude of the **resultant force F.**
We can see the zigzag motion in fig 2.1
This is an error, which can be rectified by adjusting the values of the parameters k1 and k2.

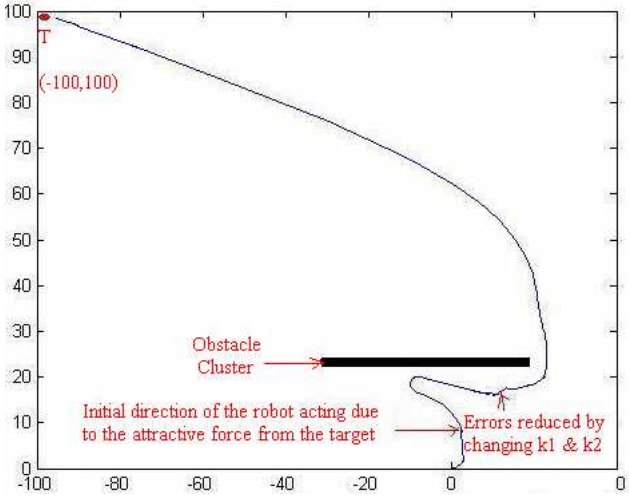

Fig 2(b) The trajectory of the robot with k1=7 and k2=5

4.2 The optimal values of the parameters k1 and k2

k_1 = Forward velocity constant

k_2 = Angular deflection constant of the wheels

On comparing figures, 2.1 and 2.2, we see that the error (zigzag motion) near the obstacle cluster decreases in fig 2.2 This has been possible by varying the values of the parameters k1 and k2.
In fig. 2.1 k1=10 and k2=7.
In fig. 2.2 k1=7 and k2=5.

This mathematical model considers just one obstacle cluster. Nevertheless, a number of obstacle clusters can be considered by modifying the MATLAB program slightly. That is, by considering two obstacle cluster equations. The obstacle cluster may not be in a straight line. Obstacles of any shape and size can be depicted with the help of suitable equations.

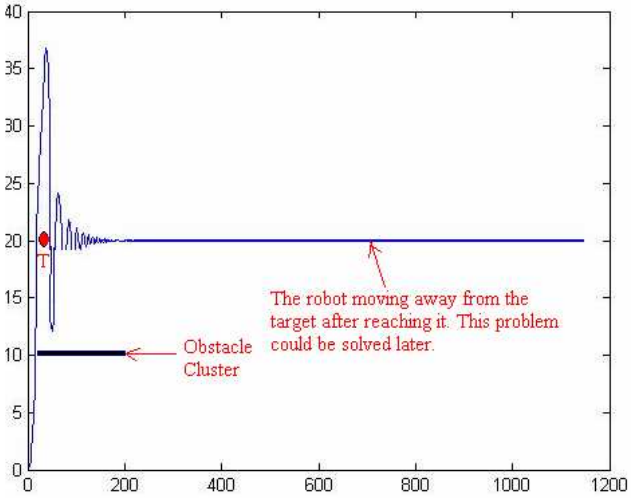

Fig3. This fig. shows the robot moving away from the target after reaching it. This problem was encountered by suitably choosing the values of k1 (the velocity constant) and k2 (the steering constant).

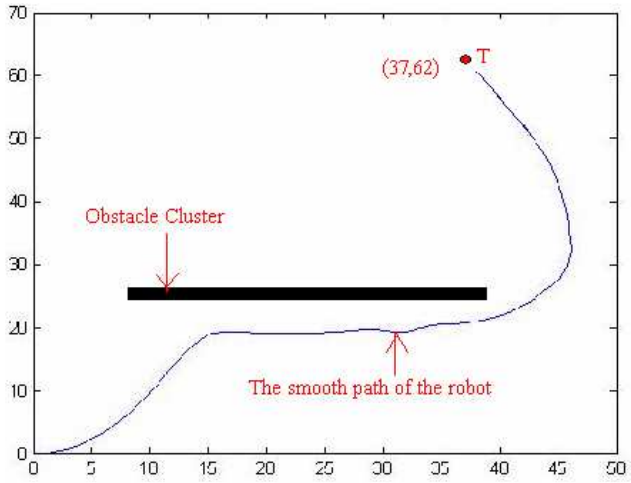

Fig. 4) The smooth trajectory of the robot after adjusting the values of the parameters.

V. The MATLAB Program for simulation of Obstacle Avoidance Robot (OAR)

```
% ---------Attractive force------------
theta = atan2((y-q(2)),(x-q(1))) - q(3);
r = sqrt((x - q(1))^2 + (y - q(2))^2);
% fctx = (fct*(cos(theta));
% fcty = (fct*(sin(theta));
fctx = fct*((x - q(1))/r);
fcty = fct*((y - q(2))/r);

%-----------------------------------------------------------
angle = [ pi/6 pi/ pi/2 (2/3)*pi (5/6)*pi pi (7/6)*pi (4/3)*pi
(3/2)*pi  (5/3)*pi, (11/6)*pi 2*pi ];
for n=1:11,
m0(1,n) = tan(ang(1,n) + q(3));
m = 1;
c = 100;
c0(1,n) = q(2) - m0(1,n)*q(1);

% ----------End points of the Obstacle-----------
xm(1,n) = (c - c0(1,n))/(m0(1,n) - m);
ym(1,n) = (m0(1,n)*c - m*c0(1,n))/(m0(1,n) - m);

if ((xm(1,n)<x2) && (xm(1,n)>x1))
dsqr(1,n) = ((xm(1,n) - q(1))^2 + (ym(1,n) - q(2))^2);
p = fcr/dsqr(1,n)/sqrt(dsqr(1,n));
fcrx(1,n) = p*((xm(1,n)) - q(1));
fcry(1,n) = p*((ym(1,n)) - q(2));
else
dsqr(1,n) = 0;
fcrx(1,n) = 0;
fcry(1,n) = 0;
end
end
xm
ym
Fx = fctx -
(fcrx(1,1)+fcrx(1,2)+fcrx(1,3)+fcrx(1,4)+fcrx(1,5)+fcrx(1
,6)+fcrx(1,7)+fcrx(1,8)+fcrx(1,9)+fcrx(1,10)+fcrx(1,11));
%Fxa = fctx - sum(fcrx');
Fy = fcty -
(fcry(1,1)+fcry(1,2)+fcry(1,3)+fcry(1,4)+fcry(1,5)+fcry(1
,6)+fcry(1,7)+fcry(1,8)+fcry(1,9)+fcry(1,10)+fcry(1,11));
beta = (atan2(Fy,Fx)) ;
v = k1*sqrt(Fx^2 + Fy^2);

qd(1,1) = v*cos(q(3));
qd(2,1) = v*sin(q(3));
qd(3,1) = k2*(beta - q(3));

%endfunction
```

VI. Results of simulations

The modified strategy when tested with the MATLAB model demonstrated very encouraging results. Fig 3. demonstrates a smooth trajectory. This model has better efficiency than its previous counterparts. Using one sensor instead of 24, the cost of the system has been reduced drastically. The mapping procedures and the avoidance of obstacles has been successfully achieved which reaching very close to the predefined target.

The following problems have been successfully rectified and a full proof strategy has been demonstrated.

1. **Solution of the Low speed Problem**
   Since in the modified model, only one sensor is mapping the complete environment (360-degree), thus the speed of mapping decreases. This problem has been encountered by modifying the mapping strategy slightly. The modified system takes the range readings for the entire circumference at 36-degree interval once every five readings. This information is updated for the successive readings. In the 4 successive readings the system takes 3 ranging readings across 54 degree about the line joining the robot position with the destination point. The readings for the remaining positions are extrapolated from the first reading. The problem of low speed can be partially solved by the above approach.

2. **Solution for the Errors in the motion**
   The zigzag motion, which can be observed in fig. 2.1, has been rectified in fig 2.2.
   This problem has been rectified by experimenting and adjusting the values of the parameters k1 and k2. The relatively lesser error can be clearly witnessed from figure 2.2. in which the values of constants k1 and k2 have been adjusted to
   k1=7 and k2 =5.

3. **Repulsion from the target once the robot reached the target.**
   It was observed that once the robot reached the target it moved away from it. (**See fig.4**) It seemed as if a repulsive force was acting from the target towards the robot. This cause for problem was wrong values of parameters. This problem was rectified by selecting suitable values of k1 and k2.

VII. Conclusions

The above model of the obstacle avoidance robot can reach a predefined target avoiding all the global and real time obstacles in its path. The modified model in this paper is different from the traditional model in the sense that it uses just one sensor instead of the 24 sensors used in the traditional model for mapping the environment. Thereby, it decreases the overall cost of the system significantly. The MATLAB model demonstrates the enhanced efficiency of this model. The success of this model has also been practically demonstrated by our robot. This model can be used further in all the obstacle avoidance robots for efficient path planning.

References

[1] Johann Borenstein and Yoram Koren. Optimum path algorithms for autonomous vehicles.

[2] J. Borenstein and Y. Koren. The Vector Field Histogram-Fast Obstacle Avoidance For Mobile Robots. In IEEE Journal of Robotics and Automation Vol 7, No 3, June 1991

[3] J. Borenstein and Y. Koren. Obstacle Avoidance with Ultrasonic Sensors. In IEEE Journal of Robotics and Automation.

[4] J. Borenstein and Y. Koren. Real-time Obstacle Avoidance for Fast Mobile Robots. In IEEE Transactions on Systems, Man and Cybernetics, Vol. 19, No. 5, Sept/Oct. 1989.

[5] J. Borenstein and Y. Koren. Tele-autonomous Guidance for Mobile Robots. In IEEE Transactions on Systems, Man and Cybernetics, Special Issue on Unmanned Systems and Vehicles Vol. 20, No. 6, Nov/Dec 1990.

[6] J. Borenstein and Y. Koren. Histogram In-motion mapping for Mobile Robot Obstacle Avoidance. In IEEE Journal of Robotics and Automation, Vol. 7, No. 4, 1991, pp. 535-539.