CHAPTER 4: ON-LINE PATH PLANNING

Unlike off-line path planning, on-line planning can consider only a division of the environment in the vicinity of robot to plan a path. Even within the neighbourhood environment, which is continuously changing, information with regards to obstacles needs to be refreshed on-line from sensory data input. Working with such incomplete information is a challenge and poses a limitation that the path planned cannot be a global optimum path. Further, there is a possibility of getting struck in traps due to lack of wider perspective of the environment (Tarokh, 2008; Ghosh, 2010). Therefore, effective functioning of on-line path planning algorithms very much depends on the capabilities of sensors, their signal processing, efficiency of path planning algorithm and the performance of control system. However, within the path planning domain if all the available information at any given instant of time is taken into consideration it can provide a near optimal solution. Hence, the mathematical model depicting the scene of the neighbourhood environment should be a complete one without neglecting any small instantaneous changes in obstacle information. Thus the objective of on-line path planning algorithm based on such a complete mathematical model is to primarily avoid collision and reach the target by following the shortest path. This is possible by suitably manipulating the instantaneous velocity of robot for the given neighbourhood information.

This chapter explains the establishment of a mathematical model, consideration of dynamic constraints, working of the algorithm, illustrations of collision-avoidance to determine the optimal velocity of robot at every instant and validation of the algorithm
through simulated environments. The effectiveness of the proposed algorithm has been demonstrated by considering a variety of environments. Comparative performance analysis in terms of computational efficiency is done against an existing algorithm. The ruggedness of the algorithm is demonstrated using Monte Carlo simulations trials.

4.1. MODELLING OF ENVIRONMENT

Robot is assumed to be a point which can move at a prescribed velocity. Obstacles of any shape, static or dynamic are modelled by enclosing circles (Min et al., 2005; Hui-zhong et al., 2006; Yu et al., 2008; Qingquan and Bi, 2009), amplified by a value so as to take care of physical dimensions of the robot and a safe maneuverable distance from the obstacles without collision. It is assumed that robot travels towards its target and gets information of nearing obstacles’ instantaneous position and velocity on-line, by refreshing sensory data.

4.2. MATHEMATICAL MODEL FOR COLLISION-AVOIDANCE

Fig. 4.1 shows the motion model for collision-avoidance, in which point R represents robot and point O represents the centre of the circle enclosing a moving obstacle. A coordinate frame XY is attached at R with its X axis along the line joining the current position of the robot R and the current position of the target T. Let \( \alpha \) be the angle made by the velocity of robot \( \mathbf{v}_r \) with respect to the X axis and \( \beta \) be the angle made by the velocity of obstacle \( \mathbf{v}_o \) with respect to the X axis of the same frame. Relative velocity between the robot and the obstacle \( \mathbf{v}_{or} \) makes an angle \( \gamma \) with the line joining R and O.
Two extreme tangents (RP and RQ) are drawn from the robot to the obstacle forming a collision cone with $\mu$ as semi collision cone angle. For the mobile robot to avoid collision with the obstacle, relative velocity $V_{OR}$ should lie outside the collision cone, i.e.,

$$\mu \leq \gamma \leq 2\pi - \mu$$

(4.1)

where

$$\gamma = \tan^{-1}\left(\frac{V_{ORA}}{V_{ORa}}\right) = \tan^{-1}\left(\frac{V_R \sin(\alpha - \theta) - V_O \sin(\beta - \theta)}{V_R \cos(\alpha - \theta) - V_O \cos(\beta - \theta)}\right)$$

(4.2)

$V_{ORA}$ and $V_{ORn}$ are the relative velocities along and normal to the line joining R and O respectively while $\theta$ being the angle between $X$ axis and the line joining R and O.

Fig. 4.1. Model for collision-avoidance
For tan function as in (4.2), the first derivative can be written as:

\[
\frac{d\gamma}{df} = \frac{1}{1+f^2} \text{ where } f = \left( \frac{V_R \sin(\alpha - \theta) - V_o \sin(\beta - \theta)}{V_R \cos(\alpha - \theta) - V_o \cos(\beta - \theta)} \right)
\]  
(4.3)

Simplifying \( \frac{1}{1+f^2} \) yields:

\[
\frac{1}{1+f^2} = \frac{k^2}{V_R^2 + V_o^2 - 2V_kV_o \cos[(\alpha - \theta) - (\beta - \theta)]} \text{ where } k^2 = [V_R \cos(\alpha - \theta) - V_o \cos(\beta - \theta)]^2
\]  
(4.4)

Equation (4.3) can also be written as:

\[
d\gamma = \left[ \frac{1}{1+f^2} \right] df
\]  
(4.5)

We know that, \( \gamma \) is a function of \( V_R \) and \( V_O \), i.e., \( \gamma = \tan^{-1}f(V_R, \alpha, V_O, \beta) \). By total differentiation, \( df \) can be written as:

\[
df = \frac{\partial f}{\partial V_R} dV_R + \frac{\partial f}{\partial \alpha} d\alpha + \frac{\partial f}{\partial V_O} dV_O + \frac{\partial f}{\partial \beta} d\beta
\]  
(4.6)

where

\[
\frac{\partial f}{\partial V_R} dV_R = \frac{V_O \sin(\beta - \alpha)}{k^2} dV_R
\]  
(4.7)

\[
\frac{\partial f}{\partial \alpha} d\alpha = \frac{V_R \left[ V_R - V_o \cos(\alpha - \beta) \right]}{k^2} d\alpha
\]  
(4.8)

\[
\frac{\partial f}{\partial V_O} dV_O = \frac{V_R \sin(\alpha - \beta)}{k^2} dV_O
\]  
(4.9)
\[
\frac{\partial f}{\partial \beta} \, d\beta = \frac{V_o^2 - V_r V_o \cos(\alpha - \beta)}{k^2} \, d\beta \tag{4.10}
\]

Substituting (4.7-4.10) in (4.6), we get:

\[
df = \frac{V_o \sin(\beta - \alpha)}{k^2} \frac{dV_r + V_r \left[V_r - V_o \cos(\alpha - \beta)\right] d\alpha + V_o \sin(\alpha - \beta) dV_r + V_r \left[V_o - V_r \cos(\alpha - \beta)\right] d\beta}{V_r^2 + V_o^2 - 2V_r V_o \cos(\alpha - \beta)} \tag{4.11}
\]

Substituting (4.4) and (4.11), in (4.5) and further simplifying:

\[
\Delta y = \frac{V_o \sin(\beta - \alpha) \Delta V_r + V_r \left[V_r - V_o \cos(\alpha - \beta)\right] \Delta \alpha + V_o \sin(\alpha - \beta) \Delta V_r + V_r \left[V_o - V_r \cos(\alpha - \beta)\right] \Delta \beta}{V_r^2 + V_o^2 - 2V_r V_o \cos(\alpha - \beta)} \tag{4.12}
\]

In order to reduce the complexity of the above expression, the relationship between $V_r$ and $V_o$ is modelled in Fig. 4.2.

---

**Fig. 4.2. Relationship between $V_r$ and $V_o$**
From Fig. 4.2, using sine theorem,

\[ V_O \sin (\beta - \alpha) = V_{OR} \sin \phi \] (4.13)

\[ V_R \sin(\alpha - \beta) = -V_{OR} \sin \left[ 180 - (\beta - \alpha + \phi) \right] \] (4.14)

where \( \phi \) is the angle between \( V_R \) and \( V_{OR} \).

Similarly, through geometrical relations:

\[ V_R - V_O \cos (\alpha - \beta) = -V_{OR} \cos \phi \] (4.15)

\[ V_O - V_R \cos (\alpha - \beta) = V_{OR} \sin \left[ \phi - \frac{\pi}{2} - (\beta - \alpha) \right] \] (4.16)

\[ V_R^2 + V_O^2 - 2V_RV_O \cos (\alpha - \beta) = V_{OR}^2 \] (4.17)

Substituting (4.13-4.17) in (4.12),

\[ \Delta \gamma = \frac{1}{V_{OR}} \left[ \sin \phi \Delta V_R - V_R \cos \phi \Delta \alpha - \sin \left[ 180 - (\beta - \alpha + \phi) \right] \Delta V_O + V_O \left( \sin \left[ \phi - \frac{\pi}{2} - (\beta - \alpha) \right] \right) \Delta \beta \right] \] (4.18)

The above expression establishes \( \Delta \gamma \) as a function of velocity parameters of robot as well as obstacle, namely \((V_R, \alpha)\) and \((V_O, \beta)\). Among these, only the incremental changes in parameters of robot \((\Delta V_R, \Delta \alpha)\) can be adjusted. Changes in obstacle velocity parameters \((\Delta V_O, \Delta \beta)\) cannot be adjusted but can be measured through sensors. Therefore, for the robot to move from its current location to next location, \( \gamma_{new} = \gamma_{current} + \Delta \gamma \) subject to satisfying (4.1). However, there can be numerous combinations of \((V_R, \alpha)\) which result in \( V_{OR} \) lying outside the collision cone. The
selection of \( V_R \) cannot be arbitrary and it should be governed by the consideration of dynamic constraints of robot and actuator specifications.

4.3. DYNAMIC CONSTRAINTS OF ROBOT

In order to arrive at the governing equations of velocity and acceleration bounds, a differential drive nonholonomic mobile robot as shown in Fig. 4.3(a) is considered (Samuel and Keerthi, 1993). The two rear wheels are driven by two independent actuators and the front one is a self-aligning wheel. Let \( V_R \) and \( \omega_R \) be the linear and angular velocities of the center point \( R \) of the rear axle. Let \( \psi_R \) be the heading angle of the robot with respect to the global frame \( XY \). The posture of the robot is defined by the triplet \( (X_R, Y_R, \psi_R)^T \) (Soueres and Boissonnat, 1998). The velocity components of robot are given by

\[
X_R' = V_R \cos \psi_R, \quad Y_R' = V_R \sin \psi_R, \quad \psi_R' = \omega_R
\]

where prime (superscript) refers to derivative with respect to time \( t \) (Benayad et al., 2000).

Fig. 4.3(b) shows motion of the robot along a curved path. For a plane curve expressed parametrically in terms of \( X(t) \) and \( Y(t) \), the signed curvature \( (\kappa) \) is given by

\[
\kappa = \frac{X_R'Y_R'' - X_R''Y_R'}{(X_R'^2 + Y_R'^2)^{3/2}} \quad (4.20)
\]
The translation and rotation velocities \((V_R & \omega_R)\) are related to \(\kappa\) as \(\kappa = \frac{\omega_R}{V_R}\). A local coordinate frame is attached to the robot with \(X_B\) axis coinciding with the vector tangent to the curved path at \(R\) as shown in Fig. 4.3(b). Let \(F_{XB}, F_{YB}\) and \(F_{ZB}\) be the forces acting along \(X_B, Y_B\) and \(Z_B\) directions at \(R\). The motion of the robot along the path must obey Newtonian dynamics, i.e.

\[
F_{XB} = ma_R, \quad F_{YB} = m\frac{V_R^2}{r} = mkV_R^2, \quad F_{ZB} = mg
\]

(4.21)

where \(m\) is the mass of the robot, \(a_R\) is the acceleration of the robot, \(r\) is the radius of path followed by the robot and \(g\) is the acceleration due to gravity. For the robot to avoid sliding,
\[
\sqrt{F_{XB}^2 + F_{YB}^2} \leq \mu_f F_{ZB} \tag{4.22}
\]

where \( \mu_f \) is the coefficient of friction between the wheels and the floor.

The bound on the admissible acceleration \( a_R \), is obtained from (4.21) and (4.22). Also considering the maximum torque or force \( F_{\text{max}} \) applied on the wheels, the admissible acceleration (Ge et al., 2007) is written as,

\[
a_R \leq \min \left( \sqrt{\mu_f^2 g^2 - \kappa^2 V_R^4}, \frac{F_{\text{max}}}{m} \right) \tag{4.23}
\]

The bound on the admissible velocity \( V_R \), is obtained from the requirement that \( \sqrt{\mu_f^2 g^2 - \kappa^2 V_R^4} \) in (4.23) should be non-negative. Hence

\[
V_R \leq \min \left( \sqrt{\frac{\mu_f g^2}{\kappa}}, V_{R_{\text{max}}} \right) \tag{4.24}
\]

The bound on the angular acceleration and angular velocity are constrained by actuator specifications as given below:

\[
a_R \leq \alpha_{R_{\text{max}}} \quad \text{and} \quad \omega_R \leq \omega_{R_{\text{max}}}, \tag{4.25}
\]

### 4.4. COLLISION-AVOIDANCE FOR MULTIPLE OBSTACLES

Mathematical model for collision-avoidance of a single obstacle was discussed in section 4.2. In case of multiple obstacles being present in the neighbourhood, they can be
negotiated according to priority such that the most imminent collision is avoided first. At the same time, it is also necessary to consider other obstacles which may necessitate the robot to deviate further away.

### 4.4.1. Identification of the Most Imminent Obstacle

The most imminent obstacle is the one which has the smallest collision distance index (Luh and Liu, 2007) as given below:

\[
\delta = \frac{d_{R,O_j}}{V_j T_s}, \quad J = 1, 2, \ldots, N_o \tag{4.26}
\]

where \(d_{R,O_j}\) is the Euclidean distance between the robot \(R\) and the \(J^{th}\) obstacle, \(V_j\) is the velocity of \(J^{th}\) obstacle and \(T_s\) is sampling rate or refreshment time.

### 4.4.2. Effect of Constraining Obstacles

In an environment with multiple obstacles, it is not enough to negotiate the most imminent obstacle alone, as it may not be the most constraining obstacle. If a small moving obstacle is close to the robot and a much larger one is farther away, it may be that the robot has to deviate more for the farther obstacle. Therefore, it is necessary to identify constraining obstacles and negotiate the most imminent one taking into account of further deviation required, if any.

An obstacle may be considered to be constraining obstacle if its collision cone falls outside the collision cone of the most imminent obstacle. One such case is presented in Fig. 4.4. The center of the most imminent obstacle is represented by \(O_1\) and the
constraining obstacle by \( O_2 \). There may be more than one constraining obstacle also, in a given situation. For the mobile robot to avoid collision with the most imminent obstacle as well as to take required deviation considering constraining obstacles, the relative velocity \( V_{OR} \) should lie outside the extreme tangents of all collision cones which overlap with that of the most imminent one thus forming a collision zone. Hence, (4.1) can be modified as:

\[
\max(\mu, \Omega_1) \leq \gamma \leq \min(2\pi - \mu, 2\pi - \Omega_2)
\]  

(4.27)

where \( \Omega_1 \) and \( \Omega_2 \) are the angles made by the extreme tangents of the overlapping collision cones with respect to the line joining robot \( R \) and the center of the most imminent obstacle \( O_1 \). While (4.1) gives the condition for avoiding a single obstacle, (4.27) is the condition to ensure that the relative velocity of robot \( V_{OR} \) lies outside the collision zone considering multiple obstacles in the vicinity.

---

Fig. 4.4. Negotiation of the most imminent obstacle considering the effect of constraining obstacle
4.5. WORKING OF THE PROPOSED ON-LINE ALGORITHM

For the mobile robot to move towards its target, at every instant the algorithm receives information about its environment by refreshing sensory data. Then the algorithm checks for the existence of any nearing obstacle. If there is no obstacle, the robot travels along the straight line towards the target satisfying kinematic and dynamic constraints. In the event of any obstacle being present, the algorithm checks whether the relative velocity between the robot and the obstacle $V_{\text{or}}$ lies within the collision cone. In case of multiple obstacles, collision distance index $\delta$ is computed to identify the most imminent threat. Also, the algorithm checks for the influence of constraining obstacles as discussed in section 4.4.2. The overlapping collision cones of obstacles result in a collision zone. In order to steer the robot out of the collision zone, the algorithm generates numerous collision-free combinations of $V_R$ and $\alpha$ randomly, however satisfying the dynamic constraints. The algorithm employs PSO technique to quickly select the best combination yielding the shortest collision-free trajectory of the robot. This is demonstrated by means of a flowchart, as shown in Fig. 4.5.

4.5.1. Implementation of PSO

The velocity of the robot at any instant can be within the range between 0 m/s and the maximum (specified). Incremental velocity $\Delta V_R$ at the next instant can be such that the velocity and acceleration constraints $V_R, a_r, \omega_r$ and $\alpha_r$ as in (4.23-4.25) are not violated.
Fig. 4.5. Flowchart of the PSO implementation
Similarly, the angle $\alpha$, made by $\mathbf{V}_R$ with the X axis as shown in Fig. 4.1, can lie between $0^\circ$ and $360^\circ$. Different combinations of $\mathbf{V}_R$ and $\alpha$ are obtained by random pairing. For the first iteration, the combinations are randomly chosen, and for every combination $\Delta \gamma$ is determined as per (4.18). Then, $\gamma_{new} = \gamma_{current} + \Delta \gamma$ is verified for satisfying (4.27). Such combinations of $\mathbf{V}_R$ and $\alpha$ that result in $\gamma_{new}$ satisfying (4.23-4.25) & (4.27) alone are considered as valid particles. Fifty such valid particles constitute the first generation. Out of these fifty particles, one that has the least Euclidean distance between the new position of the robot and the target and having the maximum possible velocity of the robot is identified as ‘pbest’. The same is repeated for 100 iterations.

For the subsequent iterations, particles are updated by using the following governing equations:

$$v[ ] = c_1(rand( ))(pbest[ ] - present[ ]) + c_2(rand( ))(gbest[ ] - present[ ])$$  \hfill (4.28)

$$present[ ] = present[ ] + v[ ]$$  \hfill (4.29)

where $v[ ]$ is particle velocity, present[ ] is current particle (solution), rand() is random number between 0 and 1, ‘gbest[ ]’ is the best particle identified by comparing ‘pbest[ ]’ of every iteration and $c_1$ & $c_2$ are learning factors. For quick convergence $c_1$ and $c_2$ are tuned to be 2 based on sample trials. Thus, at the end of 100 iterations the
algorithm offers an optimal set of $V_r$ and $\alpha$ which has the shortest Euclidean distance to target and the maximum possible $V_r$. Until the robot reaches its target, sensory data are refreshed at the prescribed sampling rate. The last maneuvered point is taken as the new start point and using the algorithm described earlier, the incremental motion of robot is planned. The procedure is repeated till the robot reaches its target.

4.6. SIMULATION RESULTS

In order to study the performance of the proposed algorithm, simulation is performed in MATLAB 7.0 on an Intel(R) Core(TM) 2 to 2.4 GHz processor. Velocity and acceleration constraints of mobile robot are $v_{\text{max}} = 0.7 \text{ m/s}$, $a_{\text{max}} = 0.5 \text{ m/s}^2$, $\omega_{\text{max}} = 2 \text{ rad/s}$ and $\alpha_{\text{max}} = 2 \text{ rad/s}^2$. Other parameters used in the simulation are the mass of the robot including its payload, $m = 10 \text{ kg}$ and the coefficient of friction between the wheels of the robot and the floor, $\mu_f = 0.3$. The obstacle detection range of the sensor is assumed to be 150 m. The sampling time, $T_s$ is 50 ms.

4.6.1. Illustration of Collision-Avoidance

In order to illustrate the functioning of the algorithm, an environment consisting of two static obstacles and a moving obstacle as shown in Fig. 4.6 is considered. Static obstacles are represented by dark shading and moving obstacles by red hatching. Even though the mobile robot is modelled as a point and the obstacles are enlarged, for showing the trajectory of the robot it is represented by small circles. R and T are the start and target locations of the robot and O is the start location of the moving obstacle. The velocity of
the obstacle is considered to be 0.5 m/s, with a maximum acceleration/deceleration of 0.3 m/s\(^2\).

![Illustrative environment](image)

*Fig. 4.6. Illustrative environment*

Fig. 4.7 explains different situations at different instances along the trajectory of the robot from its start to the target. Fig. 4.7(a) demonstrates the negotiation of the first static obstacle and the robot moves tangential to the obstacle towards its target. The negotiation of moving obstacle is demonstrated in Fig. 4.7(b) and 4.7(c). At the instant ‘A’, the algorithm predicts possible collision as the relative velocity \( V_{or} \) lies within collision cone as shown in Fig. 4.7(b). The algorithm steers the robot keeping \( V_{or} \) outside the collision cone as shown in 4.7(c). The robot is driven with a slower velocity till the instant ‘B’ by which time the obstacle has crossed the path of the robot. Subsequently the
second static obstacle is tangentially negotiated to reach the target as shown in Fig. 4.7(d).

(a) Negotiation of static obstacle

(b) Prediction of collision with a moving obstacle

Fig. 4.7. Illustration of collision-avoidance
Fig. 4.7. Illustration of collision-avoidance
In order to illustrate the effectiveness of the algorithm in negotiating multiple moving obstacles, an environment having three moving obstacles $O_1$, $O_2$ and $O_3$ with velocities of $0.3 \text{ m/s}$, $1.2 \text{ m/s}$ and $0.1 \text{ m/s}$ respectively is considered as shown in Fig. 4.8. Obstacles $O_1$ and $O_2$ pose the threat of collision as relative velocities $V_{O_1R}$ and $V_{O_2R}$ lie within their collision cones respectively (Fig. 4.8(a)). The collision distance indices of the obstacles $O_1$ and $O_2$ are computed as $\delta_1 = 6$ and $\delta_2 = 1.66$. Therefore, the algorithm identifies $O_2$ as the most imminent threat based on smaller collision index. Further, the algorithm also identifies $O_3$ as a constraining obstacle as its collision cone overlaps with that of $O_2$. The algorithm negotiates obstacle $O_2$ first steering $V_{O_2R}$ not just outside its collision cone but outside the extreme tangents of overlapping collision cones of $O_2$ and $O_3$, ahead of $O_1$ as shown in Fig. 4.8(b). Fig. 4.8(c) shows the negotiation of obstacle $O_1$ by steering $V_{O_3R}$ outside its collision cone.

### 4.6.2. Illustration of Trap Avoidance

Fig. 4.9 presents a trap situation, where multiple moving obstacles form a trap by hindering the robot’s path to reach its target. Fig. 4.9(a) shows the situation where the obstacles are moving and robot is directed towards its target. Robot is trapped at the instant ‘A’ as the moving obstacles become stagnant and form a U-shaped trap situation as shown in Fig. 4.9(b). At this instant, the algorithm generates a path tangential to the outermost collision cone. After following the contour till the instant ‘B’, the algorithm drives the robot towards its target, as shown in Fig. 4.9(c). Thus, by the inherent nature of the proposed algorithm, it does not require any special trap recovery procedure like the
ones employed in Borenstein and Koren (1989); Ge and Cui (2002); Luh and Liu (2008), provided there exists a path.

Fig. 4.8. Negotiation of multiple moving obstacles
Fig. 4.9. Trajectory of robot escaping from U-trap situation
4.7. VALIDATION OF ALGORITHM

To study and validate the performance of the algorithm, experiments are performed in a variety of environments containing polygonal and curved obstacles enclosed by different circle radius. The discussions are presented in section 4.7.1. In order to study the computational efficiency of the algorithm, it has been compared with the Min et al. (2005) algorithm for environments containing static as well as moving obstacles. The results are analyzed in terms of overall path length from start to target and computation time for each instant. This has been discussed in section 4.7.2. To ascertain the ruggedness of the proposed algorithm Monte Carlo simulation trials are performed. Trial simulations are performed for different start and target points and presented in section 4.7.3.

4.7.1. Variety of Environments

Fig. 4.10 shows the four simulated dynamic environments in which the first two environments are sparse and the other two are cluttered environments. Lines with arrowheads show the paths of moving obstacles. The corresponding obstacle velocities are written alongside. Simple curvilinear paths for the moving obstacles are designed for the first two sparse environments as shown in Fig. 4.10(a) and 4.10(b). More cluttered environments are presented in Fig. 4.10(c) and 4.10(d), with obstacles having different velocities when they move through a series of linear and curvilinear path segments. The optimal paths generated by the proposed algorithm are shown in the figures where S and T are the start and target points of the mobile robot. The generated optimal paths show
Fig. 4.10. Negotiation of obstacles by the proposed algorithm
that the algorithm is successful in negotiating polygonal as well as curved obstacles of different size which may be stationary or moving.

4.7.2. Comparison of Results with Min et al. Algorithm

The performance of the proposed algorithm is studied using the same four simulated environments by comparing with that of Min et al. Min et al. algorithm is selected for comparison as their algorithm also uses a collision cone based obstacle avoidance model. Their model neglects instantaneous changes in velocities of obstacles. Their algorithm finds optimum velocity of robot by using PSO and binary coded genetic operators of crossover and mutation. As explained in sections 4.2 to 4.5, the proposed algorithm in this work considers instantaneous changes in velocities of obstacles, dynamic constraints and influence of other obstacles while negotiating the most imminent threat. For comparison purpose, the optimal paths generated by the proposed algorithm as well as that by Min et al. are shown in the Fig. 4.11. Table 4.1 compares the results of the proposed algorithm for different dynamic environments with that of Min et al. algorithm. The data presented are average values obtained from 1000 trials. The algorithm achieves a minimum of 4.18% improvement in path length for sparse dynamic environments and a maximum of 32.67% improvement in path length for the cluttered environments over Min et al. algorithm.

This improvement can be attributed to the use of a mathematical model, which considers the complete environment information such as velocity parameters of both robot and
Fig. 4.11. Comparison of path with Min et al. algorithm
nearing obstacles. Further, due to real encoding and elimination of invalid particles from initial population generation, an improvement of over 70% in computation time per instant is achieved.

<table>
<thead>
<tr>
<th>Environment</th>
<th>Path length (m)</th>
<th>Computation time per instant (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Proposed on-line</td>
<td>Min et al. (2005) on-line</td>
</tr>
<tr>
<td>1</td>
<td>413</td>
<td>431</td>
</tr>
<tr>
<td>2</td>
<td>358</td>
<td>388</td>
</tr>
<tr>
<td>3</td>
<td>229</td>
<td>316</td>
</tr>
<tr>
<td>4</td>
<td>340</td>
<td>505</td>
</tr>
</tbody>
</table>

### 4.7.3. Ruggedness of the Algorithm

In order to study the ruggedness of the proposed algorithm, Monte Carlo simulation trials are carried out for four simulated dynamic environments as shown in Fig. 4.12. The same environments which are presented in Fig.4.11 are considered with the obstacles at different locations in each environment. For each environment, five different tasks with different start and target locations of robot are considered to check for collision-avoidance and optimal path planning. Among the five tasks as shown in Fig. 4.12, one is dedicated for barrier-free negotiation represented by brown colour, one is for the
Fig. 4.12. Results of Monte Carlo simulations
negotiation of static obstacles represented by green colour and other three tasks are for negotiation of static as well as moving obstacles represented by cyan, blue & magenta colours respectively.

For each set of start and target points in every environment, 1000 trials are conducted. A trial is deemed successful if a collision-free path is generated from the start to target while avoiding any interfering obstacles. Out of 1000 successful trials, one of the obtained optimal paths is shown as a representative sample. For each trial, the computation time required per instant is determined as the ratio of total computation time to the number of instances encountered by the algorithm till the robot reaches its target. For sparse as well as cluttered environments, average path lengths and computation time are computed for each task. Table 4.2 presents the results of path length and computation time along with their standard deviation for the four dynamic environments. The data presented are of the result of an average of 1000 path planning trials.

For all the barrier-free cases, the shortest path happens to be straight line connecting S and T with a standard deviation of zero. Therefore, the algorithm always finds the best route. In terms of computation time, the algorithm achieves a collision-free path with a minimum standard deviation of 0.01 ms for sparse environments and a maximum standard deviation of 0.11 ms for cluttered environments. In terms of path length, it can be seen from the table that the algorithm yields results with a minimum standard deviation of 1m for sparse environments and a maximum standard deviation of 3 m.
### Table 4.2. Results of Monte Carlo simulation for different environments

<table>
<thead>
<tr>
<th>Environment</th>
<th>Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Path length (m)</td>
</tr>
<tr>
<td>Sparse environment</td>
<td>237.7</td>
</tr>
<tr>
<td>Standard deviation</td>
<td>0</td>
</tr>
<tr>
<td>Computation time (ms)</td>
<td>6</td>
</tr>
<tr>
<td>Standard deviation</td>
<td>0.01</td>
</tr>
<tr>
<td>Sparse environment 2</td>
<td>294.5</td>
</tr>
<tr>
<td>Standard deviation</td>
<td>0.01</td>
</tr>
<tr>
<td>Computation time (ms)</td>
<td>8.2</td>
</tr>
<tr>
<td>Standard deviation</td>
<td>0.02</td>
</tr>
<tr>
<td>Cluttered environment 1</td>
<td>170.7</td>
</tr>
<tr>
<td>Standard deviation</td>
<td>0</td>
</tr>
<tr>
<td>Computation time (ms)</td>
<td>12.1</td>
</tr>
<tr>
<td>Standard deviation</td>
<td>0.04</td>
</tr>
<tr>
<td>Cluttered environment 2</td>
<td>179.2</td>
</tr>
<tr>
<td>Standard deviation</td>
<td>0</td>
</tr>
<tr>
<td>Computation time (ms)</td>
<td>15</td>
</tr>
<tr>
<td>Standard deviation</td>
<td>0.05</td>
</tr>
</tbody>
</table>
for cluttered environments. This demonstrates that the results obtained for a variety of cases, to be precise 20 cases resulting from 5 different tasks in 4 different environments, are well within around 1% deviation in terms of magnitude of path length as well as computation time per instant. Thus the ruggedness of the proposed algorithm is evident.