CHAPTER 3: OFF-LINE PATH PLANNING

This chapter presents the proposed off-line path planning algorithm. Details such as modelling of environment, modelling of path, design of evaluation function and determination of optimal path using PSO are discussed in sequence. Further, the effectiveness of the proposed algorithm has been studied by considering a variety of environments and also by comparing with the results obtained using GA in lieu of PSO. In addition, the performance of the algorithm is compared with a published literature for computational efficiency. Also, the ruggedness of the algorithm is demonstrated using Monte Carlo simulation trials.

3.1. MODELLING OF ENVIRONMENT WITH OBSTACLES

An off-line path planning algorithm necessarily uses a given global map of the environment which may have stationary as well as moving obstacles. Environments containing only stationary obstacles are referred to as static environments while environments containing stationary as well as moving obstacles are termed as dynamic environments.

While modelling an environment, obstacles are enclosed by a rectangular or square boundary and then enlarged as the robot is modelled to be a point. The enlargement of obstacles is done taking into consideration of physical dimensions of the robot and a safe maneuverable distance from the obstacles. The goal is also modelled as a point. A typical environment and its configuration space model (Lozano-Perez and Wesley, 1979;
Latombe, 1991) are shown in Fig. 3.1(a) and 3.1(b) respectively. Static obstacles are represented by black shading and moving obstacles by red shading.

(a) A typical environment

(b) Configuration space model

Fig. 3.1. A typical environment and its configuration space
3.2. MODELLING OF PATH

Path of a robot is represented by

\[ x_s, y_s, [v_1] \rightarrow x_1, y_1, [v_2] \rightarrow \ldots \ldots \rightarrow x_n, y_n, [v_{n-1}] \rightarrow x_g, y_g, [0] \] (3.1)

where \( x_s, y_s \) and \( x_g, y_g \) are the absolute coordinates of start and goal points of the robot as shown in Fig. 3.2. In order to limit the complexity, path length is restricted by the number of discrete points which can be 2 at the minimum and \( n + 2 \) at the maximum where \( n \) is the total number of vertices of all the obstacles in the environment.

Intermediate points in the path are represented by \( x_1, y_1, \ldots, x_n, y_n \). Velocity of the robot is
represented by \( v_i \) where \( i \) varies from \( 1 \) to \( m \) and \( m \) is the number of line segments of robot’s path. Similarly, the trajectory of a moving obstacle is given by
\[
x_{m1}, y_{m1} \rightarrow \ldots \rightarrow x_{mt}, y_{mt}.
\]

3.3. DETERMINATION OF SHORTEST PATH

The first step in path planning is generation of population containing alternative paths. The population of such paths is then subjected to evaluation function. For each generation, the population is evaluated for its fitness to choose the shortest path. The fitness function is defined in terms of path length and given by:

\[
F = \frac{1}{m} \sum_{i=1}^{m} d(p_i, p_{i+1})
\]

(3.2)

where \( d(p_i, p_{i+1}) \) denotes Euclidean distance between successive points and determined by:

\[
d(p_i, p_{i+1}) = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}
\]

(3.3)

As the fitness would increase with decrease in path length, the objective function of path planning turns out to be a minimization problem.
3.3.1. Generation of Valid Paths for Static Environments

The path is considered to be invalid if the intermediate point between start to target is contained within the obstacle and should be discarded during the generation of population. Further, in order to generate the required size of population with valid alternative paths, an invalid path interfering with a static obstacle is glided along the edges of the enlarged obstacle and made a valid path. This is done by using direction concept. One such invalid path is represented in Fig. 3.3(a). The path $x_s, y_s \rightarrow x_1, y_1 \rightarrow x_2, y_2 \rightarrow x_g, y_g$ is invalid, as intermediate path $x_1, y_1 \rightarrow x_2, y_2$ passes through the obstacle.

To negotiate the interfering obstacle, the algorithm commences with a straight line path between the intermediate starting point $S_i(x_1, y_1)$ and intermediate terminus $T_i(x_2, y_2)$ as shown in Fig. 3.3(b). Then the algorithm segregates the ‘pass through’ and ‘no pass through’ gliding edges based on direction concept. The edge which can be maneuvered by the robot is designated as the ‘pass through’ gliding edge and a non-maneuverable edge as ‘no pass through’ edge. The direction concept assigns $(-)$ sign to vertices on the left and $(+)$ sign to the vertices on the right of the shortest path. These vertices are considered as maneuverable gliding points. An edge is found to be maneuverable if the product of their vertices yields a positive sign. Joining all the maneuverable gliding edges on the left or on the right gives an obstacle-free path.
For example, for the polygonal obstacle as shown in Fig. 3.3(b) the vertices, $A$ and $B$ are to the left of the shortest path $S_T$ and hence $A$ & $B$ are assigned ($-$) sign. Therefore, the product of $S_A$ and $S_B$ yields a ($+$) sign, which means that, the edge $AB$ is maneuverable and hence is ‘pass through’ gliding edge. Similarly, other ‘pass through’ and ‘no pass through’ edges are identified as given in table 3.1. From the table 3.1, two alternative paths through $AB$ and $CD$ are identified.

For the environment as shown in Fig. 3.3(b) the obtained paths after the application of direction concept are $x_1,y_1 \rightarrow x_2,y_2 \rightarrow x_3,y_3 \rightarrow x_4,y_4 \rightarrow x_2,y_2$ and $x_1,y_1 \rightarrow x_3,y_3 \rightarrow x_4,y_4 \rightarrow x_2,y_2$. 
Table 3.1. Identification of ‘pass through’ edges using direction concept

<table>
<thead>
<tr>
<th>Edge</th>
<th>Vertex</th>
<th>Assigned Sign</th>
<th>Product of vertex signs</th>
<th>Inference</th>
</tr>
</thead>
<tbody>
<tr>
<td>AB</td>
<td>A</td>
<td>Left (-)</td>
<td>(-)(-) = +</td>
<td>AB is ‘pass through’ gliding edge</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>Left (-)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BC</td>
<td>B</td>
<td>Left (-)</td>
<td>(-)(+) = -</td>
<td>BC is ‘no pass through’ edge</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>Right (+)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>C</td>
<td>Right (+)</td>
<td>(+)(+) = +</td>
<td>CD is ‘pass through’ gliding edge</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>Right (+)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DA</td>
<td>D</td>
<td>Right (+)</td>
<td>(+)(-) = -</td>
<td>DA is ‘no pass through’ edge</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>Left (-)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Therefore, the interfering path sequence \( x_5, y_5 \rightarrow x_1, y_1 \rightarrow x_2, y_2 \rightarrow x_2, y_2 \rightarrow x_2, y_2 \rightarrow x_8, y_8 \) is modified to be either

\( x_5, y_5 \rightarrow x_1, y_1 \rightarrow x_2, y_2 \rightarrow x_2, y_2 \rightarrow x_2, y_2 \rightarrow x_2, y_2 \rightarrow x_8, y_8 \) or \( x_5, y_5 \rightarrow x_1, y_1 \rightarrow x_3, y_3 \rightarrow x_3, y_3 \rightarrow x_3, y_3 \rightarrow x_2, y_2 \rightarrow x_2, y_2 \rightarrow x_8, y_8 \)

and these will join the population of valid paths.

### 3.3.2. Generation of Valid Paths for Dynamic Environments

In an environment with moving obstacles, an invalid path predicted to be interfering with a moving obstacle is discarded. Any attempt to convert an invalid path to a valid path by negotiating the moving obstacle may delay the process of population generation. Fig. 3.4 represents a dynamic environment. To judge the validity of the path, the first crossing
Fig. 3.4. Valid and invalid path for a dynamic environment

point P is obtained by joining the path of robot and moving obstacle. The time taken by the robot to reach the point P is computed. At this time, the corresponding location of the obstacle is also calculated. The path is considered to be valid if it does not interfere with the present location of the obstacle. The path is considered to be invalid if it interferes with the present location of the obstacle. In Fig. 3.4, valid and invalid path are represented by continuous and dashed lines respectively.

The population of valid paths is then subjected to PSO and evaluated for its shortest Euclidean distance, as explained in the next section.
3.4. IMPLEMENTATION OF PSO FOR OPTIMAL PATH PLANNING

PSO is one of the widely used optimization techniques in mobile robot path planning as it is easier to implement because of its real coded nature and there are fewer parameters to be adjusted (L. Lu and D. Gong, 2008). For each generation, at the end of fitness evaluation of every particle in the population, a particle which has the highest fitness value is identified as the best particle of that generation. Here, a particle refers to an alternative path and a population refers to a set of alternative paths. A new population of valid paths is then generated by using the governing equations of PSO (Kennedy and Eberhart, 1995) as given below:

\[
\dot{v}[i] = c_1 (rand( ))(pbest[i] - present[i]) + c_2 (rand( ))(gbest[i] - present[i])
\]
\[
present[i] = present[i] + v[i]
\]

where \( v[i] \) is particle velocity, \( present[i] \) is current particle (solution), \( rand( ) \) is random number between 0 and 1, \( 'gbest[i]' \) is the best particle identified by comparing \( 'pbest[i]' \) 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. For each generation, the best particle is identified and at the end of thousand iterations the overall best particle is identified by comparing the best particles of all the generations. The overall best particle provides the optimal collision-free path of the robot. The flow chart for implementing the PSO technique is given in Fig. 3.5.
Fig. 3.5. Flowchart for optimal path determination by PSO technique
3.5. SIMULATION RESULTS

The effectiveness of the proposed algorithm is verified by simulations using MATLAB 7.0 on an Intel(R) Core(TM) 2 to 2.4 GHz processor. Experiments are conducted by considering different terrain dimensions \( (m \times n) \). A map is created for every environment where points \( S \) and \( T \) are assigned as the start and target points respectively. The mobile robot is required to move from \( S \) to \( T \) while avoiding any obstacles. The maximum velocity of the robot is taken to be 0.7 m/s.

3.6. VALIDATION OF ALGORITHM

To study and validate the performance of the algorithm, simulated experiments are performed in a variety of environments, containing convex & concave polygonal and curved obstacles. The results obtained using the proposed algorithm using PSO technique are compared with that of binary coded GA and real coded GA. This comparative analysis is presented in section 3.6.1.

In order to study the computational efficiency of the algorithm, it has been compared with Wang et al. (2007) algorithm for environments containing static as well as moving obstacles. The results are analyzed in terms of path length and computation time and presented in section 3.6.2.
To ascertain the ruggedness of the proposed algorithm Monte Carlo simulation trials (Connors and Elkaim, 2007; Perez et al., 2011) are performed. Trial simulations are performed for different start and target points and presented in section 3.6.3.

3.6.1. Variety of Environments

The first set of experiments was performed for an environment with only convex polygonal obstacles. Fig. 3.6(a) shows an environment comprising of eight convex obstacles. After initializing population, 50 valid paths are obtained by eliminating particles containing obstacle cells. Out of the 50 valid paths, only ten paths are shown for clarity in the figure. Obstacle-free optimal path is arrived by using the proposed algorithm employing the PSO technique and shown in Fig. 3.6(b). For comparison, valid path sequences for the same environments are optimized using real coded GA and binary coded GA. Fig. 3.6(c) and 3.6(d) show the best path obtained by using real coded GA and binary coded GA.

The second set of experiments was performed for concave polygonal obstacles. Fig. 3.7(a) shows an environment comprising of nine concave obstacles. Valid initial population is subjected to PSO, to develop obstacle-free optimal path, till solution convergence is reached. A few of the valid paths are shown in the Fig. 3.7(a). Fig. 3.7(b) shows the best path obtained by using PSO technique. Fig. 3.7(c) and 3.7(d) show the best path obtained by subjecting the same population to real coded GA and binary coded GA.
Fig. 3.6. Path planning for environment with regular convex polygonal obstacles
Fig. 3.7. Path planning for environment with concave polygonal obstacles

(a) Valid paths
(b) Optimal path obtained by PSO
(c) Optimal path obtained by real coded GA
(d) Optimal path obtained by binary coded GA
The third set of experiments was performed in an environment having convex & concave polygonal obstacles, curved obstacles and obstacles having curved as well as straight edges. Fig. 3.8(a) shows such an environment with ten obstacles. A few of the valid paths generated are shown for clarity in the figure. Obstacle-free optimal path is arrived by using PSO, after reaching the solution convergence as shown in Fig. 3.8(b). Fig. 3.8(c) and 3.8(d) show the best path obtained by subjecting the same valid population to real coded GA and binary coded GA.

Table 3.2 presents the results of path length and computation time obtained by the proposed algorithm employing PSO for the three environments with real coded and binary coded GA. The data presented are average values obtained from 1000 trials.

<table>
<thead>
<tr>
<th>Environment</th>
<th>Path length (m)</th>
<th>Computation time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSO technique</td>
<td>Real coded GA</td>
</tr>
<tr>
<td>1</td>
<td>13.57</td>
<td>13.76</td>
</tr>
<tr>
<td>2</td>
<td>13.83</td>
<td>14.13</td>
</tr>
<tr>
<td>3</td>
<td>9.74</td>
<td>9.97</td>
</tr>
</tbody>
</table>
Fig. 3.8. Path planning for environment with convex & concave polygonal and curved obstacles
It is evident from the results that in terms of shortest path as well as computation time, the algorithm using real coded GA shows superior performance over using binary coded GA, for all the three environments. In terms of computation time, the performance of real coded GA is superior when compared to binary coded GA due to the following reasons:

1. Binary and real coding do not perform in the identical manner at all times. For instance, the two real numbers which are successive in real representation may entirely change in the binary version. As a consequence, it consumes more computing time, often delaying the convergence.

2. Also, in binary coded GA, string length decides solution accuracy and higher accuracy can be achieved at the expense of computation efficiency.

3. Further, elimination of coding and decoding procedures in real coded GA results in shorter computation time when compared to binary coded GA (Ripon et al., 2007).

Comparison of results shows that the algorithm using PSO obtains the shortest path length. The algorithm achieves a minimum of 1.38% improvement in path length while a maximum of 2.3% improvement in path length over real coded GA. As it is an off-line optimization problem only marginal improvement in path length reduction is possible. More importantly, in terms of computation time over 29% improvement was achieved. The above major improvements are due to simple optimization steps and easier
implementation of PSO. Also, it requires only less number of parameters to be adjusted. Hence, it is evident that the performance of the proposed algorithm using PSO is found to be more efficient than real coded GA and binary coded GA.

3.6.2. Comparison of Results with Wang et al. Algorithm

In order to study the relative performance of the proposed algorithm in terms of computational efficiency, published results of Wang et al. algorithm are taken as a reference. Their algorithm generates the initial population containing the vertices of the enlarged obstacles as via points. In the process invalid paths are also considered initially and then subjected to penalty function evaluation subsequently. The generated population is then subjected to binary coded GA. During the evolutionary process, special operators like repair and speed mutation are introduced in addition to crossover and mutation to convert invalid paths to valid paths.

Experiments are designed for sparse as well as cluttered dynamic environments. The maximum velocity of robot is taken to be the same 0.7 m/s as adopted by Wang et al. Fig. 3.9 shows four simulated dynamic environments. In first two environments obstacles are sparsely located and the other two are cluttered environments. Static obstacles are represented by black colour and moving obstacles by red colour. Lines with arrowheads show the path of moving obstacles. The corresponding obstacle velocities are written alongside. S and T are the start and target points of the mobile robot. Simple paths for the moving obstacles are designed for the first two sparse environments as shown in Fig. 3.9(a) and 3.9(b). More cluttered environments are presented in Fig. 3.9(c) and 3.9(d).
Fig. 3.9. Comparison of path with Wang et al. algorithm
with obstacles having different velocities when they move through a series of path segments. Optimal paths are determined based on minimum path length. The optimal path generated by the proposed algorithm as well as that of Wang et al. are shown in the figures.

Table 3.3 compares the path lengths obtained by the proposed algorithm for the above dynamic environments with that obtained by Wang et al. algorithm. The data presented are average values of results from 1000 trials. Results show that the proposed algorithm yields shorter path lengths in all the four different environments. The algorithm achieves a minimum of 1.6% improvement in path length for a sparse dynamic environment and a maximum of 6.7% improvement in path length for a cluttered dynamic environment which is significant by considering the fact that only marginal improvement in path length reduction is possible in off-line optimization. Also, the path obtained by the proposed algorithm has less number of path segments which is another quality indicator. These improvements are due to the use of open initial search space including vertices of the enlarged obstacles as against only these vertices as via points by Wang et al.

Further, more importantly the computation time required by the proposed algorithm is only around 1% of the time required by Wang et al. algorithm which is a very significant improvement. Generation of valid paths using direction concept instead of using special genetic operators like repair & speed mutation and use of real coded PSO technique are the reasons for the improved computational efficiency.
Table 3.3. Comparison of results with Wang et al. algorithm

<table>
<thead>
<tr>
<th>Environment</th>
<th>Path length (m)</th>
<th>Computation time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>algorithm</td>
<td>algorithm</td>
</tr>
<tr>
<td>Sparse environment 1</td>
<td>401</td>
<td>408</td>
</tr>
<tr>
<td>Sparse environment 2</td>
<td>306</td>
<td>311</td>
</tr>
<tr>
<td>Cluttered environment 1</td>
<td>280</td>
<td>300</td>
</tr>
<tr>
<td>Cluttered environment 2</td>
<td>334</td>
<td>346</td>
</tr>
</tbody>
</table>

3.6.3. Ruggedness of the Algorithm

Algorithms which make use of random numbers as input are validated by using Monte Carlo method. The method finds wide applications like computational biology, network routings, localization of mobile robots, etc. Depending upon the application, thousands of trials are performed, to assess the performance of the algorithm. In this work of path planning for mobile robots, to assess the algorithm’s ruggedness, Monte Carlo simulations have been performed for different sparse as well as cluttered dynamic environments.
Four different simulated dynamic environments are considered as shown in Fig. 3.10. The same environments which are presented in Fig. 3.9 are considered with the obstacles at different locations in respective environments. For each environment, five different tasks with different start and target locations are considered to check the collision-avoidance and successful reaching of the target. Among the five tasks as shown in Fig. 3.10, one is dedicated for barrier-free negotiation represented by brown colour, one is for static obstacles negotiation represented by green colour and the other three paths are for negotiation of static as well as moving obstacles represented by cyan, blue & magenta colours respectively.

For each set of start and target locations in each environment, 1000 trials are conducted. A trial of the algorithm is considered successful if a collision-free path is found from robot’s start point S to target point T while avoiding interfering obstacles. Out of 1000 successful trials, one of the obtained optimal paths is shown (with robot velocities written alongside). Table 3.4 presents the results of obtained path lengths along with their standard deviation for each of the four dynamic environments. The data presented are of the result of an average of 1000 path planning trials.

In terms of path length, for barrier-free cases (brown colour path in Fig. 3.10) the algorithm always finds a straight line path as the best route, as there is no interference of obstacles with the shortest path. Therefore, there is no variation in path length from all
Fig. 3.10. Results of Monte Carlo simulations
Table 3.4. Results of Monte Carlo simulation for different environments

<table>
<thead>
<tr>
<th>Environment</th>
<th>Path length (m)</th>
<th>Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Brown</td>
</tr>
<tr>
<td>Sparse environment 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Path length (m)</td>
<td>Mean</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Standard deviation</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>130.6</td>
</tr>
<tr>
<td></td>
<td>Standard deviation</td>
<td>0</td>
</tr>
<tr>
<td>Cluttered environment 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Path length (m)</td>
<td>Mean</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Standard deviation</td>
<td>0</td>
</tr>
<tr>
<td>Cluttered environment 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Path length (m)</td>
<td>Mean</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Standard deviation</td>
<td>0</td>
</tr>
</tbody>
</table>

the 1000 trials and hence the standard deviation is zero. In finding the optimum path, the algorithm achieves a minimum standard deviation of 0.12 m for sparse dynamic environments and a maximum standard deviation of 0.29 m for cluttered dynamic 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. Thus the ruggedness of the proposed algorithm is evident.

Having developed an efficient and effective off-line path planning algorithm, on-line path planning, is addressed in the next chapter.