A Novel Algorithm for Path Planning of the Mobile Robot in Obstacle Environment

Chun-li Yang
College of Engineering, Dali University, Yunnan, Dali671003, China

Received: February 22, 2021. Revised: March 10, 2021. Accepted: March 17, 2021. Published: March 30, 2021.

***Abstract*—In this paper, a design method of smoothing the path generated by a novel algorithm is proposed, which makes the mobile robot can more rapidly and smoothly follow the path and reach the target point. No matter the attitude vector angle is an acute angle or obtuse angle, there is no doubt that we can find the right curve, including polar polynomial curves and piecewise polynomial functions, which makes the path length and the circular arc tend to be similar and guarantees the shorter path length. In the condition of meeting the dynamic characteristics of the mobile robot, the tracking speed and quality are improved. Therefore, the symmetric polynomial curve and the piecewise polynomial function curve are used to generate a smooth path. This novel algorithm improves the path tracking accuracy and the flexibility of the mobile robot. At the same time, it expands the application range of mobile robot in structured environment.**

***Keywords*—Mobile Robot, Obstacle Environment, Path Planning, Smoothing Path**

I. INTRODUCTION

WITH the acceleration of the global modernization process, the related industries has an unprecedented demand in the application of industrial robots, and the function of robot in various fields is significant. As the main body of the robot, the research of mobile robot [1-2] has been developed in a large extent, in the industrial production, people took the lead in the use of mobile robots. Before and after 1968, as the representative of the mobile robot, the production of automatic guided vehicles began to rise. Along with the gradually increasing application of the mobile robot, the development of the social civilization is increasingly close.

The mobile robot has played a pivotal role in the national production and life. Path planning usually has many obstacles, such as the environmental information quantity is large and it also has many obstacles as non-deterministic polynomial difficult. At present, many algorithms have been applied to path planning. Latombe research indicates that the traditional method mainly has the graphics method and the analysis method. As to Graphics methods [3], there are road map method, genetic method, ant method, taboo search method, taboo search intelligent algorithm, and the hybrid method. Intelligent algorithms such as GA have many problems, such as the encoding length change range is big, the efficiency is low and the solution range is small. Dijkstra algorithm directly search the global space without considering the target information, which leads to the time of the path length is long for the fast path planning. The planning algorithm [4] is a kind of path planning method, which is not only suitable for the global environment information, but also for the two time planning of the path. The planning algorithm introduced the heuristic function, therefore, the planning algorithm is also called the heuristic algorithm.

The differential planning algorithms are proposed to solve some problems of path planning under the circumstance of local environmental information. The algorithm process is complex, and it requires a lot of mathematical calculation and derivation. The planning algorithm has many problems in the planning path, such as it has lots of broken lines and turning point. The path planning algorithm is deeply studied to improve the quality of the line, and a novel algorithm model is established, and the method and comparison results are presented

II. RESEARCH METHODS AND PRESENT SITUATION

The path planning of multiple robots focuses on the optimal path of the whole system [5-8], such as the total time of the system is short, or, the shortest path of the system etc. From the current domestic and foreign research, in the planning of multi robot path, we tend to pay more attention on the coordination and cooperative path planning of multi robots. Among them, the traditional method is based on graph theory, such as visual graph, free space method, grid method, Voronoi [9] method, artificial potential field method etc. The Intelligent optimization methods includes: genetic algorithm, ant colony algorithm, immune algorithm, neural network, reinforcement learning, etc. And the other methods include dynamic programming, optimal control algorithm, and fuzzy control method etc. Most of them are extended from a single robot path

planning method.

A. Traditional Method

The characteristics of multi robot path planning are mainly based on graph theory. Most of the method is to build the environment into a graph firstly, and then find the optimal path from the graph. Its advantages are relatively simple to achieve, and the disadvantage is that the path may not the optimal path, but the sub optimal path instead [9-10]. As to the basic idea of the artificial potential method, it is regarded the motion as a kind of virtual force field. Obstacle will generate a repulsive force to the mobile robot, and the target points are generated by gravity, the gravity and repulsion force is generated by a certain algorithm. Meanwhile, the robot is subjected to abstract forces in the potential field, which makes the robot bypass obstacles. The advantages are suitable for the planning of the unknown environment, there will not be the dimension explosion problem. However, there may be some useful information which may be lost. Goo Guo [11] Chang et al proposed reference total potential reduction of dynamic scheduling techniques of multi robot path planning, which can better solve the problem.

B. Intelligent Optimization Method

The intelligent optimization algorithm of multi robot path planning is a new developed method. Compared with the traditional method, this method is more intelligent, and has become the focus of domestic and foreign research. Genetic algorithm [12] is a hot research topic in recent years, as a kind of probability optimization method based on population evolution, it is suitable for solving the complex and nonlinear problems which are difficult to solve by traditional search algorithms, such as the multi machine path planning problem. The basic idea of path planning is to construct the environment map into a path node link network with the graph method, then the path is expressed as a series of intermediate nodes in the path, and is converted to binary string; next, the genetic operations, such as selection, crossover, replication, mutation, are carried out, after N evolution [13], it will output the current optimal individual, that is, the optimal path of the robot. The disadvantage of genetic algorithm is that the numerous planning will occupy a large storage space and computing time. The advantage is that the computation is small.

C. Ant Colony Algorithm

The ant colony algorithm is a bionic algorithm [14] for random search. Through the interaction between the individual in the ant colony, we can solute the combination optimization problem concurrently. The algorithm is also suitable for the path planning. Zhu Qingdao [15] proposed a multi robot motion ant navigation algorithm in the unknown global environment. In this method, the global target points are mapped to the robot's field of vision and the local navigation is used as the target. The local optimal path of the robot's visual field is searched by the two groups of ants, then, on the basis of the collision prediction and collision avoidance with other robots, we can make our choices. Therefore, the path of the robot's forward path is constantly modified, so as to lead the way in each local optimal path, which can make the robot reach the target point along a path of global optimization. But the problem is that the time overhead of path planning in dynamic uncertain environment is increased, at the same time, the robot lacks the necessary learning, so that the whole robot system path is difficult to be the optimal path.

III. Design of Improving Path Planning Algorithm

A. The Basic Principle of Algorithm

Because of the complex structure of the robot, so, it is difficult to design the ideal motion control law. As to the formation control of multiple robots, in addition to considering the motion of each robot, we must consider the coordination of the system as a whole, especially in 3D space [16], the movement is more complex. The improved path planning algorithm can be used to control the nonlinear object, so it is an effective method which can deal with the problem properly. The mathematical model of the robot with practical significance is often nonlinear; at the same time, the feedback linearization method is also difficult to be used in 3D space. Therefore, it has a very strong practical significance for the robot group control design of nonlinear object in 3D space.

Dijkstra algorithm [17] is the most classical method for path search, the main idea is to calculate the path weight of each node to the target node, and then select the path of the most consistent with the requirements of the output. Its characteristic is to start node as the center, and the search will continue until reaching the target node. In the practical application, the efficiency is very low due to the large number of nodes, which need to traverse every node in the state space.

Path planning starts from the starting node, and then updates the node weights of the current node in turn. It will update the node with the minimum weight [18] of the current node until he node is traversed or the current node is the destination node. The following definitions are given in the path planning algorithm.

l -- A node that has been planned in the path planning;

g_i -- Grid node in path planning.

$h_i x$, $h_i x$ -- Respectively, the horizontal and vertical coordinates of the nodes.

$h(l)$-- The actual moving distance of the initial node QI.

$I(l)$ -- Heuristic function, the distance from L to UF.

$g(l)$ -- $h(l)+i(l)$ Node L path evaluation function.

P --- Queue set of nodes waiting to be expanded.

D ---Storage the expanded cohort set of the nodes.

O_f -- Expand the node function, when the node is not an obstacle. or it has not been previously extended, implement the insertion node function, and then insert P to the list.

O_j -- Insert the node function, according to the magnitude of g , then descend the nodes into the P list.

Using the grid map and the eight neighbor node expansion method, the mobile robot is used as the Euclidean distance, and the current node l to the target point U_f will regard as a heuristic function

$$i(l) = \sqrt{(lx - U_f x)^2 + (ly - U_f y)^2} \tag{1}$$

The evaluation function $i(l)$ is usd as an extension node, and put the node in the D list until it is extended to the target node for r path planning. The definition of the function file E_o is given in the algorithm, which is used to calculate the heuristic function values. B_f Obtain the subsequent extension list of nodes, P_f will calculate and then send the value to the P list, and then J_o will back to P node to get the location index.

B. A Conventional Algorithm

Mobile robot is considered as the point moving object [19] in the two-dimensional plane environment, and the certain obstacles in the environment are mapped into a dangerous area in the plane. Mark the mobile robot r move on the planar finite area a , a random number of obstacles were distributed on the A, the obstacle $p_i = (i=1,2,\cdots,n)$, its shape distribution is not determined, meanwhile, a is set to any shape of a convex polygon. Firstly, complete a to the specifications of rectangular, at the same time, the filled area (the obstacle region) is not allowed pass by. After the processing of the grid, the obstacle on a is converted into an obstacle region. In a , its lower left corner is O , transverse is X axis, longitudinal axis is Y , then the environmental information of the grid is mapped to the plane coordinate system XOY .

Hypothesis: the step size on the u is a , the biggest value of a on X axis and Y axis is $x_{\max}$ and $y_{\max}$ respectively, so, the a ranks of the grid number is $N_c = x_{\max} / u$, $N_1 = y_{\max} / u$ respectively, as shown in figure 1, the obstacle is randomly distributed in the a , and the position of the starting and ending points is not fixed. Mark g as the arbitrary grid, suppose that the grid in a is a collection of B , mark $P_{obs} = \{p_1, p_2, \cdots, p_n\} \in B$ is the obstacle grid $\forall p_i \in P_{obs} \ (i=1,2\cdots,n)$, the corresponding coordinates of $\forall h \in B$ is (x, y) IN xOy , and it was marked as $h(x, y)$, then defines the first grid coordinates of the lower left corner is $(1,1)$, $N_{num} = \{1,2,3\cdots,M\}$ is the grid serial number, $h(1,1)$ is $1;...;$ $h(2,1)$, and its serial number is $N_c + 1$. The coordinate (x_i, y_i) of $h_i \in B$ and the series $i \in N_{num}$ have a mapping relationship between each other, and the coordinates of serial number i is determined by the following formula

$$\left.\begin{aligned} x_i &= \left((i-1) \bmod N_c\right) + 1 \\ y_i &= \operatorname{int}\left((i-1)/N_c\right) + 1 \end{aligned}\right\} \tag{2}$$

In this formula, int. mod are respectively represent: get integer and give up remainder and the operation of getting the remainder.

Path planning under Grid Environment, that is, on the a , r is made from an arbitrary starting point of T to reach the target point along a path that is safe and free from collision U_f . Among them, $u \in P_{obs}$ and t , $f \in N_{num}$, $t \neq f$.

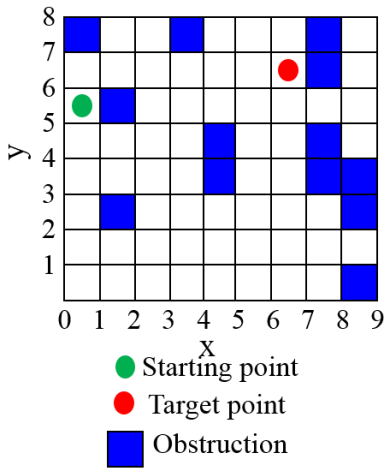


Fig. 1 the grid mobile robot environment

C. Improving Algorithm

The $h(x)$ of path planning algorithm, which use the classical Manhattan heuristic [20] function, that is, getting the evaluation of the current node, the target node, the horizontal axis, the vertical coordinate difference, and then sum the absolute value of the two. The value of the current node and the actual path of the target node is used as the evaluation value $h(x)$. According to the admissibility condition, when search the all nodes in the graph x and $h'(x) \le h(x)$, $h'(x)$ is the optimal path for the starting node to the target node. Path planning algorithm can guarantee a minimum cost path to reach the goal. $h'(x)$ is Manhattan function, which cannot guarantee $h'(x) \le h(x)$, that cannot guarantee to meet the

acceptance conditions, so we need to find a suitable heuristic function.

If path planning include two versions: A1 and A2, the difference lies in all non-target nodes $h_1'(x) \le h_2(x)$, that is, A2 is more informed than A1. If A1 is more informed than A2, in that way, a path from the X0 to the destination node, when the search terminates, each node that has been extended by A2 is also extended by A1. It can be concluded that the A1 extension of the nodes is at least as much as A2, this is a more effective algorithm, so A2 is more effective. As a result, we need to look at the function of the $h(x)$ as close as possible to the real function (in order to search efficiency), at the same time, we cannot exceed them as well (in order to be acceptable). When selecting the function, we must consider the amount of computation, the less strict the model is, the better the heuristic function is, and the more difficult it is. The value should be taken between the exact function and the computational cost. There are two heuristic function which can be improved, the two heuristic functions satisfy the admissibility theorem.

Larger value heuristic function $h'(x)$ is max value between difference of horizontal coordinates and vertical coordinates. Euclidean heuristic function $h'(x)$ is the line distance between current point and target point.

D. Design of Improving Path Planning Algorithm

(1) Polar polynomial curve

In the convergence curve, the arc length is shorter. So, on the one hand, the smooth curve should not too far from the arc, on the other hand, it must be applied to the position of the initial point and the end point, meanwhile, it should meet the requirements of the curvature of the constraint. Therefore, we should choose the polar coordinate system (r,φ), and the variable r is the polynomial function of the φ, its general form is

$$r(\varphi) = \alpha_0 + \alpha_1\varphi + \alpha_2\varphi^2 + \alpha_3\varphi^3 + \alpha_4\varphi^4 + \cdots \quad (3)$$

Figure 2 is the polar polynomial curve [21] of the joint posture of A and B, the constraints imposed on the curve can be expressed as

$$\left.\begin{array}{ll} r=R' r'=0 \kappa=0 & \varphi=0 \\ r=R' r'=0 \kappa=0 & \varphi=\Phi \end{array}\right\} \quad (4)$$

Where R is the radius of the arc of the place, $r' = \dfrac{dr}{d\varphi} r'' = \dfrac{d^2 r}{d\varphi^2} \kappa$, κ represents the curvature of r, r indicates the steering angle V, assuming that the mobile robot tracks the curve at a constant speed V, at this moment, the only acceleration is the radial acceleration, that is $\alpha_c = \dfrac{V^2}{r} = V^2 \kappa$. Since the curvature of the curve κ is continuous, the acceleration of the robot is also continuous. That is, if the robot is followed by a polar polynomial curve, the position, velocity and acceleration are all continuous. It is the direction of the car center of Figure 3. In polar coordinate system (r,φ), the tangent angle of the path can be expressed as

$$\theta = \frac{\pi}{2} + \varphi - \tan^{-1}\left(\frac{r'}{r}\right) \quad (5)$$

The differential can be obtained for φ

$$\frac{d\theta}{d\varphi} = 1 - \frac{rr'' - r'^2}{r^2 + r'^2} \quad (6)$$

The infinitesimal variation of the path length is ds, which can be expressed in polar coordinates.

$$ds = \left(r^2 + r'^2\right)^{\frac{1}{2}} d\varphi \quad (7)$$

From (4), (6)and (7), we can get the expressions for curvature:

$$\varphi\kappa = \frac{r^2 + 2r'^2 - rr''}{\left(r^2 + r'^2\right)^{\frac{3}{2}}} \quad (8)$$

By calculating the constraint conditions [22] in the formula (4), the coefficients of the formula (3) can be obtained, and the polar polynomial curve of the connecting line is finally obtained

$$r(\varphi) = R\left[1 + \frac{\varphi^2}{2} - \frac{\varphi^3}{\Phi} + \frac{\varphi^4}{2\Phi^2}\right] \quad (9)$$

The curve shows in Figure 2 is the symmetrical curve, that is, it symmetric with the symmetric axis line $PQ: \varphi = \dfrac{\Phi}{2}$, $r(\varphi) = r(\Phi - \varphi)$. Please see Figure 3 for the curvature variation of the curve $\kappa(\theta) = \theta' \kappa(\Phi) = \theta'$ And the curvature variation law is θ —Maximum value— θ, It satisfies the condition of continuous curvature continuity when connected with a straight line.

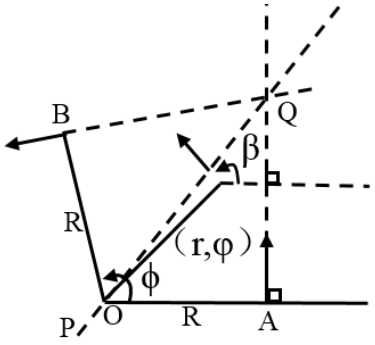


Fig.2 Polar polynomial curves under polar coordinates.

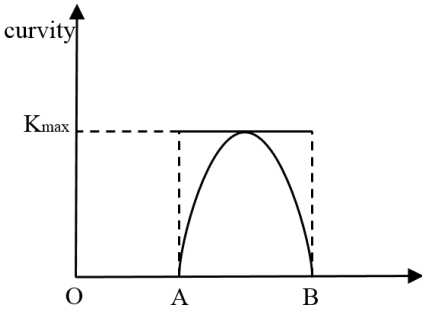
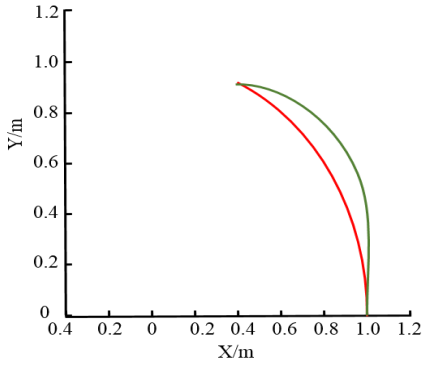
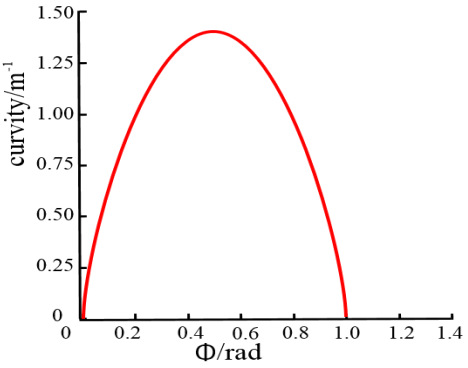


Fig.3 Polar polynomial curve curvature variation

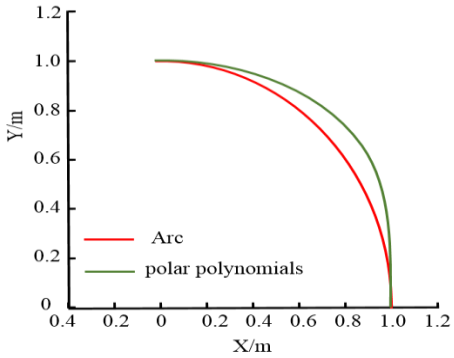
From this, we can further obtain the explicit expression of the various parameters of the projection point [23]. At the same time, it is simple to calculate, and there is no complicated integral operation, what's more, it does not need the accumulation of historical data, so, it can meet the real-time requirements of the mobile robot in the path tracking. Figure 4 is a very polynomial curve of the different angles of the MATLAB, and it is also the contrast with the same radius. Among them, the solid is polynomial curve, the dotted line is the arc. At the same time, the variation of the curvature with the polar angle is plotted.



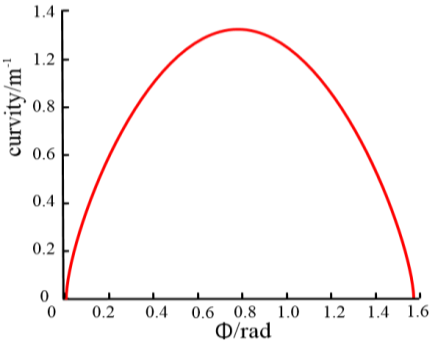
(a) $\Phi = \pi/3$



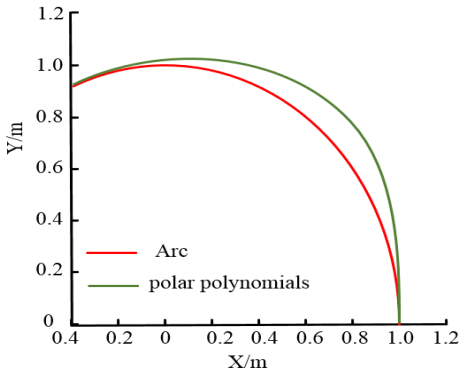
(b) $\Phi = \pi/3$



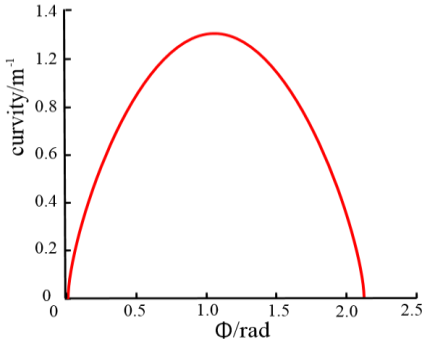
(c) $\Phi = x/2$



(d) $\Phi = \pi/2$



(e) $\Phi = 3x/2$



(f) $\Phi = 2\pi/3$

Fig. 4 Three pole polynomial curves and the curvature function of the unit circle

(2) Piecewise polynomial function curve

From figure 4 we can see, when $\Phi > 90^\circ$, the polar polynomial curve has a large deviation from the arc, and the path length is obviously increased, and the optimal condition for the smooth path of the mobile robot is not consistent. At this point, we can use the "piecewise polynomial function curve" [24] to replace the polar polynomial curve. The piecewise polynomial function curve is composed of a polar polynomial curve with two ends and a middle circular arc segment. This piecewise polynomial function curve cannot only guarantee the continuity of curvature, but also more close to the circular arc, the path length is shorter than the original polynomial curve, the price is higher curvature and curvature change rate. As shown in Figure 5, the piecewise polynomial function curve is composed of 3 parts: the ordinary polynomial form from $\varphi = 0$ to $\varphi = \beta$, and it satisfy the constraints (10). From $\varphi = \beta$ to $\varphi = \Phi - \beta$, they are symmetric polynomials, that is

$$\left.\begin{array}{ll} r=R' r'=0' \kappa=0 & \varphi=0 \\ r=R_b' r'=0' \kappa=\dfrac{1}{R_b} & \varphi=\beta \end{array}\right\} \tag{10}$$

Similarly, the mathematical form of the first segment of the polar polynomial curve can be obtained

$$r(\varphi) = R\left[1+\frac{\varphi^2}{2}-\frac{\varphi^3}{2\beta}+\frac{\varphi^5}{10\beta^3}\right] \tag{11}$$

Among them, R_b and β meet the following constraints

$$\beta^2 = 10\left(\frac{R_b}{R}-1\right) \tag{12}$$

In the third part, the expression of the polar polynomial curve is only required to be replaced by φ of (11) into $\Phi - \varphi$.

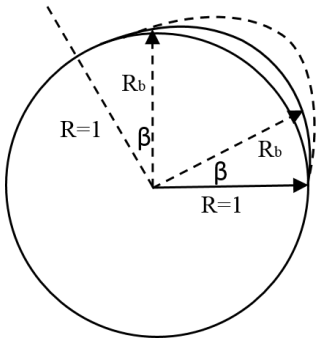


Fig. 5 Piecewise polynomial function curve

Please see Figure 6, we can see that the piecewise polynomial function curve firstly guarantee the continuity of curvature, but its peak curvature is larger than the other 2 kinds of curves. Therefore, formula (12) is required to combine the kinematic and dynamic constraints of the robot, so as to ensure the reasonable value of R_b and β, in this way, we can avoid the higher peak curvature as well.

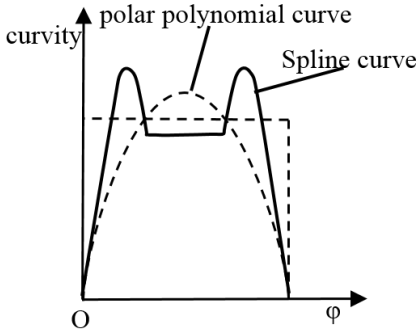


Fig.6 The comparison of piecewise polynomial function curve and normal curve curvature.

IV. The Main Steps and Programs to Improve Path Planning Algorithm

At present, most path searching algorithm research tend to focus on the improvement of the algorithm itself. It also provides a way for the optimization and improvement of the algorithm[25]. Since the dual core CPU can simultaneously carry on the dual thread operation, in this case, we can base on MPI parallel programming technology, then convert the single direction of the path planning search process to an improved path planning algorithm. The specific steps are as follows.

Step 1. According to the actual situation, the information of each node is set as a factor a. Then calculate the Euclidean distance of the starting node to the end node, and note it as d, which was shown below:

$A = (a, i = 0, n)$ Among them, n is the sum point.

$$d = \sqrt{(x_n - x_0)^2 + (y_n - y_0)^2} \tag{13}$$

Step 2. Two tables are set up for the forward path search and the reverse path search respectively, they are FOPEN table, FCLOSE table, BOPEN table and BCLOSE table. The first two tables are stored in the forward path search node that has been visited; the following two tables are stored in the reverse path search node and the node that has been visited.

Step 3. According to the standard path planning algorithm, the forward path search and the reverse path search are respectively processed. But in the calculation of the value of each node, we need to use the valuation function.

Step 4. Search the current node with the forward and reverse paths, then calculate the Euclidean distance between the respective starting points to the current node, d1 and d2 respectively, then return to operation 3, if $d_1 + d_2 < d$, then turn to step 5.

Step 5. Compared table FCLOSE and table BCLOSE, then check if there is the same node, if it exists, shift to step 6; if it doesn't exist, then keep a record of the two tables, the starting node is the current node in the FCLOSE table, and the current node in the BCLOSE table is terminated, then shift to step 2.

Step 6. Combine all the FCLOSE table and BCLOSE table.

Under Windows operating system, we can use C + + language to write the simulation program, the hardware platform is IBM Think Centre series, CPU is intel Core 2 Duo E6320, the frequency is 2.93 GHz. The simulation data is generated by the program, we give the starting point and the end point, then set up the number node.

Path planning (shortest path) algorithm is to find the path with the least cumulative weight from node A to node B. The algorithm program with C + + language is as follows.

Firstly, we can abstract "directed edges" as Edge Classes.

```
public class Edge
{
    public string StartNodeID ;
    public string EndNodeID  ;
    public double Weight     ;
}
```

A node is abstracted into a node class, and a "out side" table with this node as the starting point is hung on a node.

```
public class Node
 {
    private string iD ;
    private ArrayList edgeList ;

    public Node(string id )
  {
        this.iD = id ;
        this.edgeList = new ArrayList() ;
    }
}
```

In the process of calculation, we need to record the path to each node with the smallest weight.

```
// <summary>
/// </summary>
public class PassedPath
{
    private string     curNodeID ;
    private bool     beProcessed ;
    private double     weight ;
    private ArrayList passedIDList ;

    public PassedPath( string ID)
    {
        this .curNodeID  =  ID ;
        this .weight     =  double .MaxValue ;
        this .passedIDList  =  new  ArrayList() ;
        this .beProcessed  =  false  ;
    }
    #region property
    public bool BeProcessed
    {
        get
        {
            return this .beProcessed ;
        }
        set
        {
            this .beProcessed  =  value ;
        }
    }
    public string CurNodeID
    {
        get
        {
            return this .curNodeID ;
        }
    }
    public double Weight
    {
        get
        {
            return this .weight ;
        }
        set
        {
            this .weight  =  value ;
        }
    }
    public ArrayList PassedIDList
    {
        get
        {
            return this .passedIDList ;
```

```
        }
      }
      #endregion
   }
```

In addition, a table plancourse is needed to record the intermediate results of the plan, that is, it manages the passedpath of each node.

```
   /// <summary>
   /// </summary>
   public class PlanCourse
   {
      private Hashtable htPassedPath ;

      #region ctor
      public PlanCourse(ArrayList nodeList , string originID
)
      {
         this .htPassedPath = new Hashtable() ;

         Node originNode = null ;
         foreach (Node node in nodeList)
         {
            if (node.ID == originID)
            {
               originNode = node ;
            }
            else
            {
               PassedPath pPath = new PassedPath(node.ID)
;
               this .htPassedPath.Add(node.ID ,pPath) ;
            }
         }

         if (originNode == null )
         {
            throw new Exception( " The origin node is not ex
ist ! " ) ;
         }

         this .InitializeWeight(originNode) ;
      }

      private void InitializeWeight(Node originNode)
      {
         if ((originNode.EdgeList == null ) || (originNode.E
dgeList.Count == 0 ))
         {
            return ;
         }

         foreach (Edge edge in originNode.EdgeList)
         {
            PassedPath pPath = this [edge.EndNodeID] ;
            if (pPath == null )
            {
               continue ;
            }

            pPath.PassedIDList.Add(originNode.ID) ;
            pPath.Weight = edge.Weight ;
         }
      }
      #endregion

      public PassedPath this [ string nodeID]
      {
         get
         {
            return (PassedPath) this .htPassedPath[nodeID] ;
         }
      }
   }
```

The main steps of the algorithm are as follows.

Step1. A table (plancourse) is used to record the minimum weight of the source point to any other node. When initializing this table, if the source point can pass through a node, the weight is set to the weight of the corresponding edge, otherwise it is set to double.MaxValue .

Step2. Select the node targetnode that has not been processed and the current cumulative weight is the smallest, use its edge accessibility to update the path and weight to other nodes (if the weight of other nodes becomes smaller after passing through this node, it will be updated, otherwise it will not be updated), and then mark targetnode as processed.

Step3. Repeat (2) until all reachable nodes are processed.

Step4. Get the passedpath of the destination point from the plancourse table, which is the result.

Let's take a look at the implementation of the above steps, which is encapsulated in the routeplanner class.

```
   /// <summary>
   public class RoutePlanner
   {
      public RoutePlanner()
      {
      }

      #region Paln
      public RoutePlanResult Paln(ArrayList nodeList , strin
g originID , string destID)
      {
         PlanCourse planCourse = new PlanCourse(nodeList
 ,originID) ;
         Node curNode = this .GetMinWeightRudeNode(pla
nCourse ,nodeList ,originID) ;
```

```
        #region
        while (curNode != null)
        {
            PassedPath curPath = planCourse[curNode.ID] ;
            foreach (Edge edge in curNode.EdgeList)
            {
                PassedPath targetPath = planCourse[edge.EndNodeID] ;
                double tempWeight = curPath.Weight + edge.Weight ;

                if (tempWeight < targetPath.Weight)
                {
                    targetPath.Weight = tempWeight ;
                    targetPath.PassedIDList.Clear() ;

                    for ( int i = 0 ;i < curPath.PassedIDList.Count ;i ++ )
                    {
                        targetPath.PassedIDList.Add(curPath.PassedIDList[i].ToString()) ;
                    }

                    targetPath.PassedIDList.Add(curNode.ID) ;
                }
            }

            planCourse[curNode.ID].BeProcessed = true ;

            curNode = this.GetMinWeightRudeNode(planCourse ,nodeList ,originID) ;
        }
        #endregion

        return this.GetResult(planCourse ,destID) ;

    }
    #endregion

    #region private method
    #region GetResult

    private RoutePlanResult GetResult(PlanCourse planCourse , string destID)
    {
        PassedPath pPath = planCourse[destID] ;

        if (pPath.Weight == int.MaxValue)
        {
            RoutePlanResult result1 = new RoutePlanResult( null , int.MaxValue) ;
            return result1 ;
        }

        string [] passedNodeIDs = new string [pPath.PassedIDList.Count] ;
        for ( int i = 0 ;i < passedNodeIDs.Length ;i ++ )
        {
            passedNodeIDs[i] = pPath.PassedIDList[i].ToString() ;
        }
        RoutePlanResult result = new RoutePlanResult(passedNodeIDs ,pPath.Weight) ;

        return result ;
    }
    #endregion

    #region GetMinWeightRudeNode

    private Node GetMinWeightRudeNode(PlanCourse planCourse ,ArrayList nodeList , string originID)
    {
        double weight = double.MaxValue ;
        Node destNode = null ;

        foreach (Node node in nodeList)
        {
            if (node.ID == originID)
            {
                continue ;
            }

            PassedPath pPath = planCourse[node.ID] ;
            if (pPath.BeProcessed)
            {
                continue ;
            }

            if (pPath.Weight < weight)
            {
                weight = pPath.Weight ;
                destNode = node ;
            }
        }

        return destNode ;
    }
    #endregion
    #endregion

}
```

The simulation program automatically generates the topological structure of the path graph, and sets the range of the specific gravity factor, meanwhile, the factor value of each node is generated randomly by the program. We can compare the performance and efficiency of different search algorithms in different number of nodes by adjusting the scale of nodes. After adding the orientation factor, the mobile path can be ensured to move in the direction of the target node. At the same time, due to the increase of heuristic factor, the operation

process is more close to the reality.

The improved path planning algorithm is based on MPI parallel programming technology of path search and optimization algorithm. In theory, it can reduce the time complexity of 50% compared with the traditional path planning algorithm. The experimental environment is: CPU Intel Core2 Duo, internal memory is 2 GB, the compiler tool is Mat lab 7.0. We simulate the path planning of the mobile robot with the random distribution of obstacles in different grid environments, and then give the simulation results.In this paper, the path between nodes and nodes is (s, x, u, v), the length is 10. In the simulation, two kinds of grid scale obstacles random were distributed: 50 ×50, 300 ×300. We choose a standard path planning algorithm in this paper, and then search the different size of the path graph. Inspire the specific gravity factor range, and we can get the results which was shown in Table 1.

Table 1. Comparison of algorithm and founder results

	The traditional path planning algorithm			Improved path planning algorithm		
Scale	Number of point	Sum of weights	Tine	Number of point	Sum of weights	Tine
30	14	72	0.83	16	38	0.75
150	-	-	-	70	490	3.59
300	396	435	45.37	407	1134	32.85
1000	933	702	56.59	892	1983	48.53

From the comparison of the simulation results, in the search of the small-scale path graph, the standard path planning algorithm and the improved path planning recursive algorithm are quite similar.

A number of straight lines and a number of symmetrical polar polynomial curve are used to connect it with transitions between adjacent gestures, and when the attitude angle is an obtuse angle with piecewise polynomial curves instead of polar polynomial. Finally, it can generate a global smooth path, which was shown in Figure 7. The control period is 100ms, the moving speed is 30mm/s, and path follows the simulation results are shown in Figure 7.

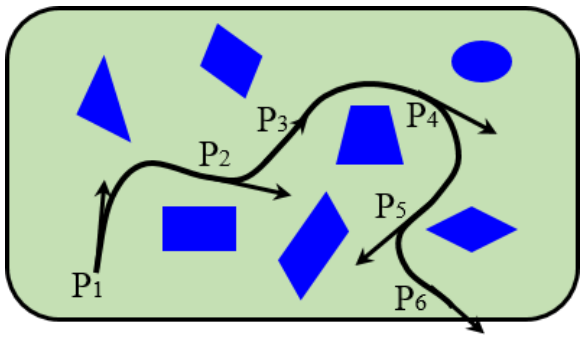


Fig.7 Path follows the simulation results

V. CONCLUSION

In this paper, we study the path planning of different grid scale obstacles by using the improved algorithm. The improved algorithm is introduced to optimize the key points, and the path is simplified as a key point. In the end, the path is generated by the position, slope and curvature. Its calculation is simple, and it can greatly improve the efficiency of the robot when encountering obstacles.

ACKNOWLEDGEMENTS

This work is supported by Scientific Research Fund Project of Yunnan Education Department (2019J0758)

REFERENCES

[1] Yinka-Banjo C O , Agwogie U . Mobile Robot Path Planning in an Obstacle-free Static Environment using Multiple Optimization Algorithms. Nigerian Journal of Technological Development, 2020, 17(3):165-173.

[2] Zheng Y, Yan B , Ma C , et al. Research on obstacle detection and path planning based on visual navigation for mobile robot. Journal of Physics, 2020, 1601(6):062044 (10pp).

[3] Zhong X , Tian J , Hu H , et al. Hybrid Path Planning Based on Safe A* Algorithm and Adaptive Window Approach for Mobile Robot in Large-Scale Dynamic Environment. Journal of Intelligent & Robotic Systems, 2020, 99(1):65-77.

[4] Sun Y , Zhang C , Sun P , et al. Safe and Smooth Motion Planning for MecanumWheeled Robot Using Improved RRT and Cubic Spline. Arabian Journal for Science and Engineering. Section A, Sciences, 2020, 45(4):3075-3090.

[5] Ould Mohamed Mohamed Vall, Modeling and Networked Control of Two-rigid Link Robot Arm, WSEAS Transactions on Systems and Control, Volume 15, 2020, Art. #39, pp. 375-382.

[6] Lyubomira Miteva, Kamen Delchev, Kaloyan Yovchev, Evgeniy Krastev, Design of Biped Robot with Anthropomorphic Gait, WSEAS Transactions on Systems and Control, Volume 14, 2019, Art. #33, pp. 257-263.

[7] Van Den Berg J, Abbeel P, Goldberg K. LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information. The International Journal of Robotics Research, 2019, 30(7): 895-913.

[8] Liao C , Liao Y , Xie J . Obstacle Avoidance Trajectory Planning of Loading Robot Based on Improved RRT Algorithm. International Core Journal of Engineering, 2020, 6(5):209-213.

[9] Atia M G B , El-Hussieny H , Salah O . A Supervisory-Based Collaborative Obstacle-Guided Path Refinement Algorithm for Path Planning in Wide Terrains. IEEE Access, 2020,170: 257-266.

[10] Das S K, Dutta A K, Debnath S K. OperativeCriticalPointBug algorithm-local path planning of mobile robot avoiding obstacles. Indonesian Journal of Electrical Engineering and Computer Science, 2020, 18(3):1646.

[11] Asgari M , Foghahayee H N . State Dependent Riccati Equation (SDRE) controller design for moving obstacle avoidance in mobile robot. SN Applied Sciences, 2020, 2(11), pp. 47-55.

[12] Jung J W , Park J S , Kang T W , et al. Mobile Robot Path Planning Using a Laser Range Finder for Environments with Transparent Obstacles. Applied Sciences, 2020, 10(8), pp.2799.

[13] Ajeil F H , Ibraheem I K , Azar A T , et al. Autonomous navigation and obstacle avoidance of an omnidirectional mobile robot using swarm optimization and sensors deployment. International Journal of Advanced Robotic Systems, 2020, 17(3):172988142092949.

[14] Martyshkin A I . Motion Planning Algorithm for a Mobile Robot with a Smart Machine Vision System. Nexo Revista Científica, 2021, 33(2), pp.651-671.

[15] Persson S M, Sharf I. Sampling-based A* algorithm for robot path-planning. The International Journal of Robotics Research, 2019, 33(13), pp. 1683-1708.

[16] Badmos T A , Omolaye P O , Mebawondu J , et al. Robot Path Planning Performance Evaluation of a Dynamic Environment. IOSR Journal of Electronics and Communication Engineering, 2020, 13(6), pp.19-26.

[17] Delgado R, Choi B W. Practical high curvature path planning algorithm in joint space. Electronics Letters, 2015, 51(6): 469-471.

[18] Wang L L , Pan L X . Research on SBMPC Algorithm for Path Planning of Rescue and Detection Robot. Discrete Dynamics in Nature and Society, 2020, 2020(10):1-11.

[19] Shih B Y, Chang H, Chen C Y. RETRACTED: Path planning for autonomous robots–a comprehensive analysis by a greedy algorithm. Journal of Vibration and Control, 2019, 19(1): 130-142.

[20] Islam M R , Protik P , Das S , et al. Mobile robot path planning with obstacle avoidance using chemical reaction optimization. Soft Computing, 2021(10), pp.1-28.

[21] Zhang X , Lai J , Xu D , et al. 2D Lidar-Based SLAM and Path Planning for Indoor Rescue Using Mobile Robots. Journal of Advanced Transportation, 2020, 2020(3), pp. 1-14.

[22] Liu L S , Lin J F , Yao J X , et al. Path Planning for Smart Car Based on Dijkstra Algorithm and Dynamic Window Approach. Wireless Communications and Mobile Computing, 2021, 2021(4), pp.1-12.

[23] Chen J , Zhang R , Han W , et al. Path Planning for Autonomous Vehicle Based on a Two-Layered Planning Model in Complex Environment. Journal of Advanced Transportation, 2020, 2020(9), pp.1-14.

[24] Liu Z , Liu H , Lu Z , et al. A Dynamic Fusion Pathfinding Algorithm Using Delaunay Triangulation and Improved A-star for Mobile Robots (January 2021). IEEE Access, 2021, PP(99):1-1.

[25] Fethi, Matoui, Boumedyen, et al. Contribution to the path planning of a multi-robot system: centralized architecture. Intelligent Service Robotics, 2020, 13(1), pp. 147-158.