Computing Algorithm of Robot Distributed 3D Graphic Simulation System

Junkai Cao a, Zhijun Gao, Zhonghua Han
Shenyang Jianzhu University, Shenyang, 110168, China
a Junkai@sjzu.edu.cn

Abstract. In view of the shortcomings of current multi-robot path planning strategies, such as high path coupling, long total path, and long waiting time for collision avoidance, as well as the resulting problems of low system robustness and low robot utilization, the paper proposes Multi-robot path planning particle swarm algorithm based on distributed three-dimensional space-time graphics and motion decomposition. We first generate a dynamic temporary obstacle in the time dimension according to the existing path set and the current robot position, and expand it together with the static obstacle into a three-dimensional search graphics space. Secondly, in the three-dimensional image space, the total path movement time is divided into three parameters: movement time, turning time and staying time in place, and using the conditional depth-first search strategy to calculate all the matching parameters from the starting node to the target node the required path collection. The experimental results show that the path of distributed 3D graphics planning by our proposed particle swarm algorithm has the advantages of total length, less running time, system collision-free, and high robustness, which solves the problem of multi-robot systems completing continuous random tasks.

Key words. Robot, 3D graphics simulation, particle swarm algorithm, path planning, spatio-temporal algorithm.

1. Introduction
With the development of science and technology, research in specific fields, such as space exploration, undersea exploration, etc., especially the exploration of outer space, will become the key research goal of various countries. In the process of space development, the importance of teleoperation of robots with increasing sex, it can allow humans to remotely control the robots working in the space environment on the ground. However, due to the transmission delay between the local control and the remote robot, which usually lasts several seconds, the operator is very It is difficult to directly control the robot in a closed loop based on the image returned before the transmission delay. An alternative method is to simulate the space robot and its working environment, and the operator directly controls the simulated robot, thereby generating a control sequence, so that the actual robot is at a certain level. The operation of the simulated robot is repeated after the time delay [1]. This paper proposes a particle swarm optimization algorithm for multi-robot path planning based on three-dimensional space-time images and motion decomposition. The proposed algorithm first uses the grid method to construct an image environment model, which is expanded to a time axis on its vertical coordinate axis to form a constrained
three-dimensional space-time image, which can store all robots’ planned paths and paths from the current moment to the maximum time scale. The parking position information of the robot when there is no task, and the position information of obstacles in the image.

2. Particle Swarm Algorithm

We take \( f(X) \) as the objective function of the minimization problem, then the current best position of particle \( i \) is determined by the following formula [2]:

\[
P(i+1) = \begin{cases} 
\min \{ f(X(t+1)) \} & \text{if } f(X(t+1)) < f(P(t)) \\
\max \{ f(X(t+1)) \} & \text{otherwise}
\end{cases} 
\]

(1)

We set the number of particles in the group as \( N \), and the best position experienced by all particles in the group as \( g(t) \), which is called the global best position, then

\[ x_j(t+1) = x_j(t) + \nu_j(t+1) \]

(2)

Comparing formulas (1) and (2), it can be seen that when the inertia weight \( w=1 \), the two formulas are the same, which indicates that the particle swarm algorithm with inertia weight is an extension of the basic particle swarm algorithm. The inertia weight \( w \) indicates to what extent the original velocity of the particle is preserved. In the literature, the inertia weight satisfies:

\[ w = 0.9 - \frac{t}{\text{MaxNumber}} \times 0.5 \]

(3)

Such an adaptive adjustment of the value of \( w \) can make the global search ability and the local search ability get a good balance, so that the search quickly converges to the global optimal solution. The formula of position weighting to unconstrained solution improved particle swarm algorithm is as follows:

\[ x_j(t+1) = x_j(t) + \nu_j(t+1) + k \times \frac{x_{oj} - x_j}{\| x_{oj} - x_j \|} \]

(4)

Comparing (3) and (4), it can be seen that there is an additional item \( k \times \frac{x_{oj} - x_j}{\| x_{oj} - x_j \|} \) in (4), which represents the component of the particle tending to the unconstrained solution, which is called the position weighted to the unconstrained solution item, where "\( x_{oj} \)" represents the \( j \) dimension of the particle the best position under unconstrained conditions. When the position of the \( j \) dimension of the particle is above the unconstrained optimal solution, the term is negative; when it is below the unconstrained optimal solution, the term is positive. It can be seen that adding this item can make the particles move to the unconstrained optimal solution faster [3]. The size of \( k \) determines the step length at which the particle position tends to the unconstrained solution. After this improvement, the convergence speed of the particle swarm can be accelerated. Another method of position weighting is to improve the optimal position of the group, that is, make the group optimal tend to the unconstrained solution faster, so that the particles in the entire group will tend to the global optimal solution faster. The specific formula is as follows:

\[ g(j) = g(j) + k \times \frac{x_{oj} - x_j}{\| x_{oj} - x_j \|} \]

(5)

The meaning and usage of \( k \times \frac{x_{oj} - x_j}{\| x_{oj} - x_j \|} \) is the same as (5). The speed evolution and position evolution formulas are the same as the basic particle swarm evolution formulas. The effect of the above methods may not be obvious when used alone, so they can be used in combination to optimize the performance of the particle swarm algorithm and accelerate the convergence speed of the algorithm.
3. Path planning algorithm of mobile robot based on 3D image

3.1. Description of the beam optimization problem
In the fields of science and engineering, the solution of many extreme value problems is often restricted by various realistic factors [4]. These constraints are usually described by a series of constraints. Solving extreme value problems with constraints is called a constrained optimization problem. Specifically, it can be represented by the following general form of nonlinear programming:

$$\min f(X) \quad X \in E^n$$
$$s.t. \quad g_i(X) \geq 0 \quad i=1,2,3,\ldots,m$$

(6)

Where $g_i(X)$ represents the constraint condition of the problem. Due to the existence of constraints, the solution of constrained extreme value problems is much more complicated and difficult than that of unconstrained extreme values. For the constrained minimum problem, not only must the objective function be continuously reduced in the iterative process, but also the feasibility of the solution must be paid attention to. In order to simplify the optimization process of constrained optimization problems, the method of converting constrained optimization problems into unconstrained optimization problems and non-linear programming problems into linear optimization problems can usually be used to simplify complex problems.

The path planning problem of mobile robots is a kind of optimization problem with constraints, which is to find an optimal path from the starting point to the target point under the condition that the robot does not collide with obstacles [5]. The constraint is that the robot path does not intersect with obstacles. To this end, it is necessary to find a feasible solution (a solution represents a path of the robot) that makes the objective function representing the path length obtain a minimum value. For the infeasible solution, the constrained optimization problem must be dealt with to turn it into an unconstrained optimization problem.

3.2. Approaches to constrained optimization problems
There are three commonly used methods to deal with constrained optimization problems: search space limitation method, feasible solution transformation method, penalty function method, etc. For the constrained optimization problem in this subject, we use the penalty function method to turn it into an unconstrained optimization problem. The basic idea of the penalty function method is: for an individual who has no corresponding feasible solution in the solution space, a penalty value is imposed when calculating its fitness, so as to increase the fitness value of the individual, so that the individual is not selected in the minimum optimization optimal. Specifically, the following formula can be used to adjust the fitness value of an individual:

$$F(X) = \begin{cases} f(X) \\ f(X) + P(X) \end{cases}$$

(7)

Where $f(X)$ is the original fitness function, $F(X)$ is the new fitness function after considering the penalty function, and $P(X)$ is the penalty function.

4. Path planning algorithm and simulation based on particle swarm optimization
We adopt the environmental modelling idea of regional division, and divide the robot’s working environment longitudinally according to actual needs. In order to facilitate calculation and research, here is divided into n longitudinally, and n vertical lines are represented by $L_1, L_2, \ldots, L_n$. The collision-free path of the mobile robot from the starting point S to the target point G must have an intersection with each vertical line, denoted as $P_1, P_2, \ldots, P_n$ in turn from left to right. Then the path planning can mean finding a suitable point $P_i$ on each vertical line $L_i$ within the range determined by the regional boundary to form a sequence of points (S $P_1, P_2, \ldots, P_n G$). Considering that the starting point and the target point have been fixed, the sequence ($P_1, P_2, \ldots, P_n$) can uniquely determine a movement path of the mobile robot [6]. If expressed by the coordinates of each point, a G matrix can be used to describe the path, and the
first row of the matrix is stored the abscissa of each point, the second line stores the ordinate of each point. Since the vertical line is equally divided, the abscissa of each point has a fixed arithmetic relationship, that is, the abscissa of each intersection point is known, and it is not necessary to show it. Then a vector \( X = [x_1, x_2, \ldots, x_n] \) of \( 1 \times N \) can represent a path (an individual) of the mobile robot, where \( x_i \) is the ordinate of point \( P_i \), and \( X \) is also called a solution to the path planning problem.

If \( x_j \) is regarded as an unknown variable that can move up and down on a vertical line, and obstacles are used as constraints, a function of the collision-free distance of a mobile robot from the starting point to the ending point can be obtained:

\[
f(X) = f_1 + \sum_{i=1}^{N} \sqrt{a^2 + (x_{i+1} - x_i)^2} + f_2
\]  

(8)

In the formula, \( f_1 \) represents the distance between \( P_1 \) and the starting point \( S \), \( f_2 \) represents the distance between \( P_n \) and the target point \( G \), and \( la = L_{SG}/n + 1 \) represents the distance between two adjacent vertical lines. In (8), only \( x_j \) is an unknown variable. As long as it is limited to the obstacle figure, the formula becomes a high-dimensional function, thus turning the mobile robot path planning problem into a constrained optimization problem. Without loss of generality, this constrained optimization problem can be expressed as:

\[
\min_{x \in \mathbb{R}, f(X) \neq 0} f(X) \quad i = 1, 2, \ldots, m
\]  

(9)

The penalty function can be introduced to transform the above problem into an unconstrained optimization problem

\[
F(X) = \begin{cases} 
\min f(X) & \text{if } f(X) = 0 \\
\min f(X) + P(X) & \text{otherwise}
\end{cases}
\]  

(10)

The \( F(X) \) obtained in this way is the objective function in the robot path planning problem, that is, the fitness function. For a feasible solution \( X \) (that is, the path indicated by \( X \) does not intersect with an obstacle), \( F(X) \) represents the length of the path that the mobile robot traverses from the starting point \( S \) to the target point \( G \); for the infeasible solution (that is, the path indicated by \( X \)) There is a position point in the obstacle, or the path passes through the obstacle) The penalty value can be added by using the penalty function technology to calculate its fitness value under unconstrained conditions, and then evaluate the individual optimality and the group optimality [7]. When using the particle swarm algorithm to plan the path of a mobile robot, the objective function for evaluating the best individual and the best group is \( F(X) \). The path of the robot is composed of the position information of the optimal particles in the group at different moments. The position information \( (x_1, x_2, \ldots, x_n) \) of each particle when passing through the vertical line \( L_i \) can determine a path. We call \( n \) the dimension of the path solution or particle dimension, which means that the particle takes values at \( n \) times in the planning space. Repeat the initialization of the particle swarm \( n \) times (where the \( i \)-th represents the initialization of the position of each particle at the moment when the vertical line \( L_i \) is reached), and initialize a position information for each dimension, so that \( N \) pieces of mobile robots can be selected. The path can be represented by a matrix of \( N \times n \):

\[
X = \begin{bmatrix}
x_{11}, x_{12}, \ldots, x_{1n} \\
x_{21}, x_{22}, \ldots, x_{2n} \\
\vdots \\
x_{n1}, x_{n2}, \ldots, x_{nn}
\end{bmatrix}
\]  

(11)

Among them, \( X_i = [x_{i1}, x_{i2}, \ldots, x_{in}] \) represents the \( i \)-th individual, which is a path solution of the mobile robot, which is represented by the position information of the particles in \( n \) dimensions. Another way of initialization is to first initialize a path of a mobile robot, that is, a particle solves the position
information in n dimensions on the path, and then loops N times to generate N paths. After the path is generated, it is necessary to evaluate the adaptability of the individual, and select the best individual and the best group [8]. Because it is initialization, the individual has no historical value, so the fitness value of the individual is the optimal fitness value of the individual, and the position of the individual is the optimal position of the individual. Fig. 1 and Fig. 2 respectively show the optimal one among the N paths generated after initialization in a random mode and a fixed mode. When the program is run multiple times, the path in Figure 1 changes randomly, while the path in Figure 2 is always fixed, and its length is 127.9590.

Figure 1. A path of a mobile robot generated by random initialization

Figure 2. A mobile robot path generated by fixed initialization
For the two path generation methods of fixed initialization and random initialization, various particle swarm evolution algorithms are used to optimize the path. The results are shown in Table 1.

Table 1. Various algorithms optimize the fixed initialization path (Pm=0.54 is the probability of mutation)

|                | Basic particle swarm algorithm | Inertial weighted particle swarm |
|----------------|-------------------------------|----------------------------------|
| t              | 100                           | 100                              |
| G              | 102.963                       | 102.941                          |
| Position-weighted improved particle swarm | Add genetic variation to improve particle swarm | Pm=0.54 |
| t              | 100                           | 100                              |
| G              | 103.530                       | 103.555                          |

5. Conclusion
This paper analyses the components of the movement time of a single robot to complete a specific atomic task, and divides it into computer solution time, path movement time, and uncertain time, and expands it into a time axis in the vertical direction of the two-dimensional raster image. The expansion of the three-dimensional image into a three-dimensional space-time map effectively puts the collision avoidance in the path planning on the first layer, reducing the path conflict problem. Through specific analysis of the path movement time, it is divided into turning time, stop time, path running time parameters, and then use the conditional DFS path search to preferentially use the low-cost parameters, DFS can find it in the shortest possible time an optimal path.

References
[1] Huang, Q., Lan, J., & Li, X. Robotic arm based automatic ultrasound scanning for three-dimensional imaging. IEEE Transactions on Industrial Informatics, 15(2) (2018) 1173-1182.
[2] Bedaka, A. K., Vidal, J., & Lin, C. Y. Automatic robot path integration using three-dimensional vision and offline programming. The International Journal of Advanced Manufacturing Technology, 102(5) (2019) 1935-1950.
[3] Sun, B., Zhu, D., & Yang, S. X. An optimized fuzzy control algorithm for three-dimensional AUV path planning. International Journal of Fuzzy Systems, 20(2) (2018) 597-610.
[4] Li, X., & Yu, D. Study on an optimal path planning for a robot based on an improved ANT colony algorithm. Automatic Control and Computer Sciences, 53(3) (2019) 236-243.
[5] Pandey, P., Shukla, A., & Tiwari, R. Three-dimensional path planning for unmanned aerial vehicles using glowworm swarm optimization algorithm. International Journal of System Assurance Engineering and Management, 9(4) (2018) 836-852.
[6] Huang, Q., Wu, B., Lan, J., & Li, X. Fully automatic three-dimensional ultrasound imaging based on conventional B-scan. IEEE transactions on biomedical circuits and systems, 12(2) (2018) 426-436.
[7] Che, G., Liu, L., & Yu, Z. An improved ant colony optimization algorithm based on particle swarm optimization algorithm for path planning of autonomous underwater vehicle. Journal of Ambient Intelligence and Humanized Computing, 11(8) (2020) 3349-3354.
[8] Yu, Z., He, Y., Xu, Y., & Chen, H. Vision-based deviation extraction for three-dimensional control in robotic welding with steel sheet. The International Journal of Advanced Manufacturing Technology, 95(9) (2018) 4449-4458.