Optimization on Trajectory of Stanford Manipulator based on Genetic Algorithm

Xi Han[1,a]

[1]*School of Mechanical Engineerin, University of Science and Technology Beijing, 100083 Beijing, China*

Abstract. The optimization of robot manipulator's trajectory has become a hot topic in academic and industrial fields. In this paper, a method for minimizing the moving distance of robot manipulators is presented. The Stanford Manipulator is used as the research object and the inverse kinematics model is established with Denavit-Hartenberg method. Base on the initial posture matrix, the inverse kinematics model is used to find the initial state of each joint. In accordance with the given beginning moment, cubic polynomial interpolation is applied to each joint variable and the positive kinematic model is used to calculate the moving distance of end effector. Genetic algorithm is used to optimize the sequential order of each joint and the time difference between different starting time of joints. Numerical applications involving a Stanford manipulator are presented.

1 Introduction

As modern industry advances towards intelligence and digitalization, the applications of industrial robots is becoming increasingly wider and deeper [1]. Robot manipulator is one of the most important devices among various industrial robots, which covers a diverse array of technologies from mechanical engineering, mathematical, to control theory, electronic engineering [2]. The optimization of trajectory plays a key role in the fulfillment of engineering tasks of robot manipulators, which has become a hot issue for academic and industrial research.

Atsushi et.al, who focused on the local navigation problem taking into consideration the dynamics of the robot, proposed a new technique with obstacle avoidance, called adaptive navigation. The advantages of the proposed navigation scheme are that less local information is required than in some other techniques and the navigation law is simpler and more flexible than ones using the artificial potential field methods [3]. This model has some limitations in expressing the relative motions between the end effector and the robot.

Ningyue et.al presented the ADRC controller in mobile manipulator system. A kind of dynamics decoupling control method is proposed to achieve synergy tracking between mobile platform and robotic arm. The effect of overall estimation and compensation ADRC control algorithm has on the disturbance of the model and manipulators tracking control strategy can be a good trajectory of the end-effector and mobile platforms separately controlled, and reduce the instability factor between the two coupling caused by the dynamics, reaching a higher position tracking accuracy [4]. In contrast to the previous research, this paper presented the structural relationship between the shape of manipulator and the robot body. They were not able to address the path planning problem in three-dimensional space, which is still a gap in actual industrial production. The robot mathematical model in three-dimensional space will be more complicated and contains more constrains.

Jia proposed a positive-inverse kinematics model for Stanford Manipulator, using cubic B-spline interpolation to achieve the trajectory planning of Stanford Manipulator joints [5]. But the article did not take into account the effect of the sequential order of the joint movements on the trajectory of the end effector.

Genetic algorithm (GA) was used by many researchers to solve the problem of robot path planning, which get the global optimal solution in a short time [6][7]. Dong combined the analytic method and genetic algorithm, obtaining a kinematic inverse solution with a redundant degree of freedom and achieving the optimal track planning and obstacle avoidance. The method has the characteristics of small calculation and strong adaptability [6].

Above all, the Stanford Manipulator is used as the research object in this paper, and the inverse kinematics model is established by Denavit-Hartenberg method. According to the initial posture matrix, the inverse kinematics model is used to find the initial state of each joint. In accordance with the given beginning moment, cubic polynomial interpolation is applied to each joint variable and the positive kinematic model is used to calculate the end effector path. Genetic algorithm is then used to optimize the sequential order of each joint and the time difference between different starting time of joints, with an objective function of minimizing the path of end effector.

[a] Corresponding author: hanxi1706@126.com

2 Mathematical Model of Stanford Manipulator

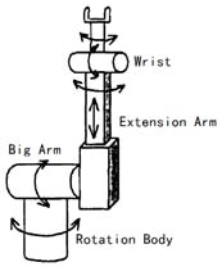

Fig 1. Model of Stanford Manipulator

0 shows that Stanford Manipulator consists of a Rotation Body, a Big Arm, an Extension Arm, Wrist and an End effector, which is made up of six joints (Five rotating joint and one locomotive joint). Each joint can express one degree of freedom. From bottom to top, the six kinematics parameters of Manipulator's joints are $\theta_1, \theta_2, d_3, \theta_4, \theta_5, \theta_6$. Obviously, end effector's position and attitude is related to the structure of Manipulator' connecting rod and kinematics parameters of Manipulator's joints. The matrix whi describes the end effector's position and attitude in reference coordinate system is expressed as follow:

$$T = [\vec{n} \quad \vec{o} \quad \vec{a} \quad \vec{p}] = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

Where $\vec{n}, \vec{o}, \vec{a}$ is the end effector's attitude, $\vec{p}$ is the end effector' position, p_x, p_y, p_z is the end effector's coordinate in the datum reference coordinate system. In order to make sure the matrix of effector's position and attitude T, the coordinate systems need to be set up at every rod. The relative position and attitude between these coordinate systems can be described by homogeneous transformation.

In this passage, we set up six coordinate systems on the Stanford Manipulator, and describe their relationship by matrix. As usual, we define Matrix A as homogeneous transformation among a rod and its adjacent member. A tandem manipulator is connected by a series of rods with a drive joint. A manipulator with n degrees of freedom has n rods and n joints. The foundation rod is called rod 0. For Stanford Manipulator, it is not considered into the six rods. The rod 1 and rod 0 are connected by joint 1. The rod 2 and rod 1 are connected by joint 2, and so on. There is no joint at the end of rod 6.

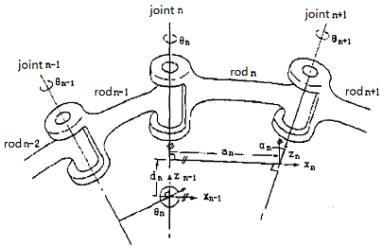

Fig 2. Coordinate transformation of adjacent connecting rod

0 shows the relationship of the adjacent coordinate system n-1and n. Firstly, the x_{n-1} axis is rotated by θ_n angle around the z_{n-1} axis. Secondly, the x_{n-1} axis moves d_n distance along the z_{n-1} axis. Thirdly, the z_{n-1} axis moves a_n distance along the rotating x_{n-1} axis, that is, x_n axis. Finally, the z_{n-1} axis is twisted by α_n angle around the x_n axis. As a result, the Matrix A is presented as follows,

$$\mathrm{A}_n = \begin{bmatrix} cos\theta_n & -sin\theta_n & 0 & 0 \\ sin\theta_n & cos\theta_n & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_n \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\alpha_n & -sin\alpha_n & 0 \\ 0 & sin\alpha_n & cos\alpha_n & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} cos\theta_n & -sin\theta_n & sin\theta_n sin\alpha_n & a_n cos\theta_n \\ sin\theta_n & cos\theta_n & -cos\theta_n sin\alpha_n & a_n sin\theta_n \\ 0 & sin\alpha_n & cos\alpha_n & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

According to the structure of Stanford Manipulator, the link motion parameters is shown in 0.

Table 1 Stanford Manipulator's the link motion parameters

Joint Number	Joint Parameter	Torsional angle α	Offset
1	θ_1	-90°	0
2	θ_2	90°	d_2
3	d_3	0°	d_3
4	θ_4	-90°	0
5	θ_5	90°	0
6	θ_6	0°	0

Therefore, when the reference coordinate system is transformed to the base coordinate system, the Matrix of manipulator end effector's position and attitude can be represented as:

$$T_6 = A_1 \cdot A_2 \cdot A_3 \cdot A_4 \cdot A_5 \cdot A_6 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

To facilitate writing and reading, we denote $\sin\theta_i = s_i, \cos\theta_i = c_i$,

$$n_x = c_1[c_2(c_4c_5c_6 - s_4s_6) - s_2s_5c_6] - s_1(s_4c_5c_6 + c_4s_6) \tag{4}$$

$$n_y = s_1[c_2(c_4c_5c_6 - s_4s_6) - s_2s_5c_6] + c_1(s_4c_5c_6 + c_4s_6) \tag{5}$$

$$n_z = -s_2(c_4c_5c_6 - s_4s_6) - c_2s_5c_6 \tag{6}$$

$$o_x = c_1[-c_2(c_4c_5s_6 + s_4c_6) + s_2s_5s_6] + s_1(s_4c_5s_6 - c_4c_6) \tag{7}$$

$$o_y = s_1[-c_2(c_4c_5s_6 + s_4c_6) + s_2s_5s_6] - c_1(s_4c_5s_6 - c_4c_6) \tag{8}$$

$$o_z = s_2c_4c_5s_6 + c_2s_5s_6 + s_2s_4c_6 \tag{9}$$

$$a_x = c_1(c_2s_4s_5 + s_2c_5) - s_1s_4s_5 \tag{10}$$

$$a_y = s_1(c_2s_4s_5 + s_2c_5) + c_1s_4s_5 \tag{11}$$

$$a_z = -s_2c_4s_5 + c_2c_5 \tag{12}$$

$$p_x = c_1s_2d_3 - s_1d_2 \tag{13}$$

$$p_y = s_1s_2d_3 + c_1d_2 \tag{14}$$

$$p_z = c_2d_3 \tag{15}$$

Equations (4)-(15) is the Stanford Manipulator's

forward kinematics model, in which the position and attitude matrix T can be obtained according to the joint parameters. We can also know that the position of end effector relative to base coordinate system p_x, p_y, p_z only depends on the joint parameters θ_1, θ_2, d_3.

In industrial applications, we often design the position of end effector p_x, p_y, p_z and solve the corresponding joint parameters θ_1, θ_2, d_3. This is called inverse kinematics. For Stanford Manipulator, the inverse kinematics analytical solution is as follows,

$$\theta_1 = tan^{-1}\left(\frac{p_x}{p_y}\right) - tan^{-1}\left(\frac{d_2}{\sqrt{p_x^2 + p_y^2 - d_2^2}}\right) \quad (16)$$

$$\theta_2 = \tan^{-1}\frac{c_1 p_x + s_1 p_y}{p_z} \quad (17)$$

$$d_3 = s_2\left(c_1 p_x + s_1 p_y\right) + c_2 p_z \quad (18)$$

$$\theta_4 = tan^{-1}\frac{-s_1 a_x + c_1 a_y}{c_2\left(c_1 a_x + s_1 a_y\right) - s_2 a_z} \quad (19)$$

$$\theta_5 = tan^{-1}\frac{c_4\left[c_2\left(c_1 a_x + s_1 a_y\right) - s_2 a_z\right] + s_4[-s_1 a}{s_2\left(c_1 a_x + s_1 a_y\right) + c_2 a_z} \quad (20)$$

$$\theta_6 = tan^{-1}\frac{-c_5\{c_4[c_2(c_1 o_x + s_1 o_y) - s_2 o_z] + s_4[-s_1 o_x + c_1 o_y]\} + s_5[s_2(c_1 o_x + s_1 o_y) - c_2 o_z]}{-s_4[c_2(c_1 o_x + s_1 o_y) - s_2 o_z] + c_4(-s_1 o_x + c_1 o_y)} \quad (21)$$

3 Motion trajectory planning

From Chapter 0 we can know that the parameters of manipulator's joints $[\theta_1, \theta_2, d_3]$ can determine the position of end effector $[p_x, p_y, p_z]$. And the trajectory of end effector will be determined by the position $[p_x, p_y, p_z]$ at all moments. In trajectory planning, we solve the initial values and final values of joints' parameters by inverse kinematics according to the given corresponding initial position $[p_{x0}, p_{y0}, p_{z0}]$ and end position $[p_{xf}, p_{yf}, p_{zf}]$. Besides, the time slot from the former to the latter t_f should be specified according to the task requirements. Then, each joint parameter will be performed by polynomial interpolation operation to get smooth function $\theta(t)$. Finally, we get position-time function $p(t)$, which can accurately indicate the end effectors trajectory, by positive kinematics algebra for the smooth function $\theta(t)$.

According to mechanical knowledge, manipulator's movement should be smooth, because uneven movement will aggravate the wear of mechanical parts and cause manipulator's vibration and shock. Hence, the described trajectory function must be continuous. Its first derivative (Velocity) and even the two derivative (Acceleration) should be continuous too.

In this passage, the parameters of manipulator's joints are processed with three polynomial interpolation. So there are four constraint conditions of the joints' parameters as follows,

$$\theta(0) = \theta_0 \quad (22)$$

$$\theta(t_f) = \theta_f \quad (23)$$

$$\dot{\theta}(0) = \dot{\theta}(t_f) = 0 \quad (24)$$

The function that satisfies all of the above constraints is shown as follows,

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \quad (25)$$

The constraint is replaced by a function, and get undetermined coefficient as follows,

$$\begin{cases} a_0 = \theta_0 \\ a_1 = 0 \\ a_2 = \dfrac{3}{t_f^2}(\theta_f - \theta_0) \\ a_3 = -\dfrac{2}{t_f^3}(\theta_f - \theta_0) \end{cases} \quad (26)$$

Obviously, it is completely unnecessary to make all parameters of Manipulator's joints start changing at the same time. In the other words, there are many kinds of orders sequences between different parameter variations. With a result the trajectory of end effector will be also different. Suppose intermediate moment $t = t_m$ is inserted between starting time $t = 0$ and finishing time $t = t_f$, a part of parameters of Manipulator's joints begin changing at $t = 0$, and the others begin changing at $t = t_m$. All parameters stop at $t = t_f$ and the end effector reach the final position.

So there are serval constraint conditions of the joints' parameters which begin changing at $t = t_m$ as follows,

When $t < t_m$,

$$\theta(t) \equiv \theta_0 \quad (27)$$

When $t > t_m$, the function that satisfies all of the above constraints as (25),

The constraint conditions (22)-(24)change to (28)-(30) as follows,

$$\theta(t_m) = \theta_0 \quad (28)$$

$$\theta(t_f) = \theta_f \quad (29)$$

$$\dot{\theta}(t_m) = \dot{\theta}(t_f) = 0 \quad (30)$$

The undetermined coefficients are calculatedas follows,

$$\begin{cases} a_0 = \dfrac{\theta_0 t_f^3 - 3\theta_0 t_f^2 t_m + 3\theta_f t_f t_m^2 - \theta_f t}{(t_f - t_m)^3} \\ a_1 = \dfrac{6 t_m t_f (\theta_0 - \theta_f)}{(t_f - t_m)^3} \\ a_2 = -\dfrac{3(t_m + t_f)(\theta_0 - \theta_f)}{(t_f - t_m)^3} \\ a_3 = \dfrac{2(\theta_0 - \theta_f)}{(t_f - t_m)^3} \end{cases} \quad (31)$$

As a result, The position-time function of Manipulator's end effector $p(t)$ is a piecewise function. The movement distance of end effector is

$$L = \int_0^{t_m} \sqrt{\dot{p}_{x1}^2(t) + \dot{p}_{y1}^2(t) + \dot{p}_{z1}^2(t)}\,dt + \int_{t_m}^{t_f} \sqrt{\dot{p}_{x2}^2(t) + \dot{p}_{y2}^2(t) + \dot{p}_{z2}^2(t)}\,dt \quad (32)$$

In this passage, the trajectory optimization objective is to find the middle moment between joint 2 and joint 3 to minimize the movement distance of end effector when the initial position $[p_{x0}, p_{y0}, p_{z0}]$ and end position $[p_{xf}, p_{yf}, p_{zf}]$ is given. Now, the concrete trajectory planning requirements for Stanford Manipulator is given as follows. The joint parameter of rotation body θ_1 will always start changing at $t = 0$, but the joint parameter of big arm θ_2 and the joint parameter of extension arm d_3 will start changing at $t = t_2$ and $t = t_3$,respectively. The three joint parameters must stop at $t = t_f$, at the same time the end effector must arrive the final position.

The solution is to get the best combination of middle moment t_2, t_3 , which can make the distance of end effector from start position to final position the shortest.

4 The Proposed Genetic Algorithm

In this passage, genetic Algorithm(GA) is used for global search to find the shortest distance of end effector L and the corresponding middle moment t_2, t_3 . Genetic Algorithm consists of the following functions, initialization parameters, fitness function, ranking strategy, selection, crossover and mutation probabilities. The main body of GA is shown in 0

Table 2 The main body of GA for trajectory

```
Input:
int generation_size;
int pop_size;
int chromo_size;
double cross_rate;
double mutate_rate;
Output:
initialization(pop_size,chromo_size);
for i=1 To generation_size
fitness(pop_size,chromo_size);
rank(pop_size, chromo_size);
selection(pop_size, chromo_size);
crossover(pop_size,chromo_size,cross_rate);
mutation(pop_size,chromo_size,mutate_rate) ;
end for
```

Initialization function is shown in 0. It should be noticed that the column length of population (pop) should be two times that of chromosome length (chromo_size), because there are two independent variables. pop(i, j) is equal to 0 or 1each with a fifty percent. In this case, individuals composed of arbitrary middle moment t_2, t_3 is represented as a binary code whose length is equal to two times the length of chrom_size.

Table 3 initialization function for randomly generating middle moment t_2, t_3

```
Function initialization(pop_size,chromo_size)
Input:
int pop_size;
int chromo_size;
int array[2] pop(pop_size, chromo_size);
Output:
for i=1 To pop_size
    for j=1 To 2×chromo_size
        pop(i, j) = round(rand);
    end for
end for
```

Fitness function is used to calculate population fitness (fitness_value) as 0. Each individual should be divided into two parts and converted into decimal numbers. Then, middle moment t_2, t_3 is based on the corresponding decimal numbers in the $[0, t_0]$ range. Upper limit t_0 is different from final joints stopping time t_f. The wear and vibration will be intensified if the two numbers are too close. The first half of each solution is transformed into the moment when parameters of joint 2 begins changing t_2, while the second half of each solution is transformed into the moment when parameters of joint 3 begins changing t_3. In the end of this function, t_2, t_3 is substituted into calculations to get the distance of end effector L. To facilitate the arrangement of fitness_value from small to large, fitness_value is defined to be equal to the reciprocal of L.

Table 4 fitness function for calculate fitness_value

```
function fitness(pop_size, chromo_size)
Input:
int pop_size;
int chromo_size;
int array[2] pop(pop_size, chromo_size);
double array t2(pop_size);
double array t3(pop_size);
double t0;
double array fitness_value (pop_size);
Output:
for i=1 To pop_size
    for j = 1 To chromo_size
        if pop(i , j) == 1
            t2(i) = t2(i) + 2^(j−1);
        end if
    end for
    for j = chromo_size+1 To 2×chromo_size
        if pop(i, j) == 1
            t3(i) = t3(i) + 2^(j−chromo_size−1);
        end if
    end for
    t2 (i) = t2 (i) × t0/(2^(chromo_size−1));
    t3 (i) = t3 (i) × t0/(2^(chromo_size−1));
    fitness_value=1/L(t2, t3);
end for
```

The process of computing $L(t_2, t_3)$ will vary with the size of the t_2, t_3. Therefore, we need to discuss the situation separately as Figure 3.

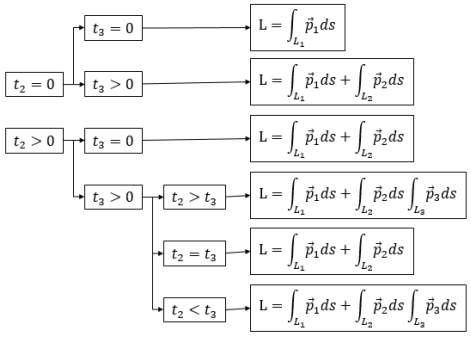

Fig 3. Multiple possibilities for computing the distance of end effector L

This paper only illustrates the process of calculating the distance of end effector L when $t_3 > t_2 > 0$. Obviously, it needs to be divided into three sections.

When $0 < t < t_2$,

$$\begin{cases} a_{10} = \theta_{10} \\ a_{11} = 0 \\ a_{12} = \dfrac{3}{t_f^2}(\theta_{1f} - \theta_{10}) \\ a_{13} = -\dfrac{2}{t_f^3}(\theta_{1f} - \theta_{10}) \end{cases} \quad (33)$$

$$\theta_1(t) = a_{10} + a_{11}t + a_{12}t^2 + a_{13}t^3 \quad (34)$$

$$\theta_2(t) = \theta_{20} \quad (35)$$

$$d_3(t) = d_{30} \quad (36)$$

$$p_{x1}(t) = \cos[\theta_1(t)]sin\theta_{20}d_{30} - \sin[\theta_1(t)]d_2 \quad (37)$$

$$p_{y1}(t) = \sin[\theta_1(t)]sin\theta_{20}d_{30} + \cos[\theta_1(t)]d_2 \quad (38)$$

$$p_{z1}(t) = cos\theta_{20}d_{30} \quad (39)$$

$$L_1 = \int_{L_1} \vec{p}_1 ds = \int_0^{t_2} \sqrt{\dot{p}_{x1}^2(t) + \dot{p}_{y1}^2(t) + \dot{p}_{z1}^2(t)}dt \quad (40)$$

When $t_2 < t < t_3$, $\theta_1(t)$ stands for Formula (34), $d_3(t)$ stands for Formula(36). The relationship between the remaining parameters and time as follows.

$$\begin{cases} a_{20} = \dfrac{\theta_{20}t_f^3 - 3\theta_{20}t_f^2t_2 + 3\theta_{2f}t_ft_2^2 - \theta_{2f}}{(t_f - t_2)^3} \\ a_{21} = \dfrac{6t_2t_f(\theta_{20} - \theta_{2f})}{(t_f - t_2)^3} \\ a_{22} = -\dfrac{3(t_2 + t_f)(\theta_{20} - \theta_{2f})}{(t_f - t_2)^3} \\ a_{23} = \dfrac{2(\theta_{20} - \theta_{2f})}{(t_f - t_2)^3} \end{cases} \quad (41)$$

$$\theta_2(t) = a_{20} + a_{21}t + a_{22}t^2 + a_{23}t^3 \quad (42)$$

$$p_{x2}(t) = \cos[\theta_1(t)]\sin[\theta_2(t)]d_{30} - \sin[\theta_1(t)]d_2 \quad (43)$$

$$p_{y2}(t) = \sin[\theta_1(t)]sin[\theta_2(t)]d_{30} + \cos[\theta_1(t)]d_2 \quad (44)$$

$$p_{z2}(t) = cos[\theta_2(t)]d_{30} \quad (45)$$

$$L_2 = \int_{L_2} \vec{p}_2 ds = \int_{t_2}^{t_3} \sqrt{\dot{p}_{x2}^2(t) + \dot{p}_{y2}^2(t) + \dot{p}_{z2}^2(t)}dt \quad (46)$$

When $t_3 < t < t_f$, $\theta_1(t)$ stands for Formula (34), $\theta_2(t)$ stands for Formula(42). The relationship between the remaining parameters and time as follows.

$$\begin{cases} a_{30} = \dfrac{d_{30}t_f^3 - 3d_{30}t_f^2t_3 + 3d_{3f}t_ft_3^2 - d_{3f}t_3^3}{(t_f - t_3)^3} \\ a_{31} = \dfrac{6t_3t_f(d_{30} - d_{3f})}{(t_f - t_3)^3} \\ a_{32} = -\dfrac{3(t_3 + t_f)(d_{30} - d_{3f})}{(t_f - t_3)^3} \\ a_{33} = \dfrac{2(d_{30} - d_{3f})}{(t_f - t_3)^3} \end{cases} \quad (47)$$

$$d_3(t) = a_{30} + a_{31}t + a_{32}t^2 + a_{33}t^3 \quad (48)$$

$$p_{x3}(t) = \cos[\theta_1(t)]\sin[\theta_2(t)]d_3(t) - \sin[\theta_1(t)]d_2 \quad (49)$$

$$p_{y3}(t) = \sin[\theta_1(t)]sin[\theta_2(t)]d_3(t) + \cos[\theta_1(t)]d_2 \quad (50)$$

$$p_{z3}(t) = cos[\theta_2(t)]d_3(t)$$

$$p_{z2}(t) = cos[\theta_2(t)]d_{30} \quad (51)$$

$$L_3 = \int_{L_3} \vec{p}_3 ds = \int_{t_3}^{t_f} \sqrt{\dot{p}_{x3}^2(t) + \dot{p}_{y3}^2(t) + \dot{p}_{z3}^2(t)}dt \quad (52)$$

Finally, the distance of end effector is calculated as follows:

$$L = L_1 + L_2 + L_3 \quad (53)$$

Ranking function sorts individuals in pop from small to large in order to facilitate selection function operation. Selection function is used to screen a new generation of populations. According to the ranking results of the pop, the probability that the N individual is selected is $P(n)$

$$P(n) = [\sum_n^{i=1} fitness(i)] \Big/ [\sum_{pop\_size}^{i=1} fitness(j)] \quad (54)$$

Thus, individuals with greater fitness are more likely to be selected into the next generation of individuals. At the same time, elite mechanism is utilized that individuals with the highest fitness per generation are automatically reserved for the next generation.

Crossover function selects two individuals in the new species group selected by selection function as parent individuals, and takes any point from the two extracted individuals as intersections. If their crossover probability (cross_rate) allows the two parent individuals to cross, they start at that point until all parts after that point are cross interchanged as 0.

Table 5 crossover function

```
function crossover(pop_size, chromo_size,
cross_rate)
Input:
int pop_size;
int chromo_size;
double cross_rate;
int cross_pos;
int array[2] pop(pop_size, chromo_size);
Output:
for i=1 To pop_size Step 2
    if(rand < cross_rate)
        cross_pos = round(rand×2×chromo_size);
        if or (cross_pos == 0, cross_pos == 1)
            continue;
        end if
        for j=cross_pos To 2×chromo_size
            pop(i, j) ↔ pop(i+1,j);
        end for
    end if
end for
```

Similar to B, A should first give a point that needs to mutate, and then substitute the other alleles for the gene at that point, which is usually a simple reverse operation as 0.

Table 6 mutation function

```
function mutation(pop_size, chromo_size, mutate_rate)
```

```
Input:
int pop_size;
int chromo_size;
double mutate_rate;
int mutate_pos;
int array[2] pop(pop_size, chromo_size);
Output:
for i=1 To pop_size
    if rand < mutate_rate
        mutate_pos = round(rand×2×chromo_size);
        if mutate_pos == 0
            continue;
        end if
        pop(i,mutate_pos) = 1 - pop(i, mutate_pos);
    end if
end for
```

5 Simulation Results

All experiments were performed on MATLAB R2016a(9.0.0.341360), The operating system of the computer is 64 bit Windows10 family Chinese version, CPU is Intel(R)Core(TM)i5-4200H CPU @ 2.80GHz, RAM capacity is 4.00GB.

The 6 nodes are selected randomly in the three-dimensional space as 0 to verify the universality of the algorithm.

Table 7 Stanford Manipulator's position and joints parameters at different nodes

Node sequence	p_x	p_y	p_z	θ_1	θ_2	d_3
0	150	-100	120	-1.5708	0.6947	156.205
1	-100	150	80	-1.9656	-0.8961	128.063
2	200	100	50	-0.2717	1.2780	173.205
3	300	100	100	-0.1725	1.2259	295.804
4	-200	320	-80	-1.4210	1.3437	-355.387
5	-250	-340	-60	-1.3002	1.4198	-398.998

The algorithm parameters are set as follows,pop_size = 30, chromo_size = 10, generation_size = 300, cross_rate = 0.8, mutate_rate = 0.02. The un-optimized path indicates the distance at which all joint parameters move at the same time from the t=0 moment. Ten node sequences are used for optimization among five nodes as 0.

Table 8 experimental results in different node sequences

Node sequences	Un-optimized distance	Optimized distance	Optimum percentage	Generation of best individual	Code time
0-1	179.987	179.209	0.43%	139	97s
0-2	229.755	223.821	2.58%	137	96s
0-3	371.456	340.236	8.40%	215	92s
0-4	544.034	542.320	0.32%	218	94s
0-5	610.924	606.129	0.78%	223	99s
1-2	225.479	176.215	21.85%	141	108s
1-3	383.143	240.443	37.24%	92	113s
2-3	112.031	112.011	0.02%	295	104s
3-4	525.231	525.142	0.02%	61	84s
4-5	82.508	82.471	0.04%	245	111s

The experiments results show that the movement distance is optimized greatly by the algorithm, which is reduced by 37% compared to un-optimized condition. When the precision is set as 1/1000, the code runs from one and a half minutes to two minutes. Figure4. The comparison of un-optimized path and optimized path of Node 1-3is shown in Figure 4.

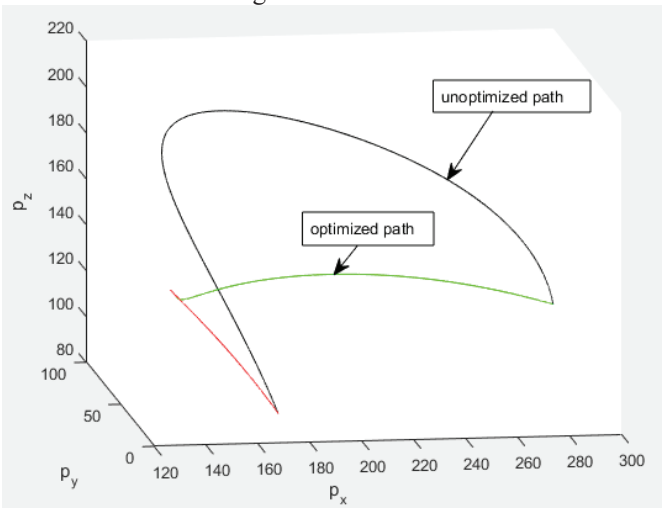

Fig 4. The comparison of un-optimized path and optimized path of Node 1-3

To investigate the influence of these parameters on the performance of the algorithm, each parameter is set in four levels as shown in 0. There is no doubt that huge computational efforts have to be performed if the algorithm is implemented under all possible combinations. To overcome this drawback, the Taguchi method of design of experiment (DOE) is used in this study, which uses orthogonal arrays to decrease the number of experiments [10]. According to the number of parameters and the number of factor levels, the orthogonal array $L_{16}(4^3)$ in 0 is selected for executing the experiments.

Table 9 Combinations of parameter values

Parameters	Factor level			
	1	2	3	4
Pop_size(p_s)	20	30	40	50
Cross_rate(c_r)	0.5	0.6	0.7	0.8
Mutate_rate(m_r)	0.02	0.03	0.05	0.1

Table 10 The orthogonal array $L_{16}(4^3)$ and experimental results

Test Number	Factor Level			Average results		
	p_s	c_r	m_r	Code time	Generation of best individual	The optimization distance
1	1	1	1	82s	141	176.211
2	1	2	2	80s	80	176.233
3	1	3	3	83s	68	176.211
4	1	4	4	79s	57	176.233
5	2	1	2	119s	87	176.220
6	2	2	1	120s	30	176.211
7	2	3	4	119s	106	176.211
8	2	4	3	122s	162	176.233
9	3	1	3	130s	115	176.211
10	3	2	4	146s	236	176.214
11	3	3	1	119s	79	176.211
12	3	4	2	132s	179	176.233
13	4	1	4	126s	28	176.211
14	4	2	3	58s	220	176.211
15	4	3	2	58s	140	176.211
16	4	4	1	163s	136	176.214

In different factor lever, the running time of the algorithm is between one and a half to three minutes, and the convergence of the L is stable and has no obvious fluctuation. As shown in 0, the algorithm converges to the

50th generation 50 times.

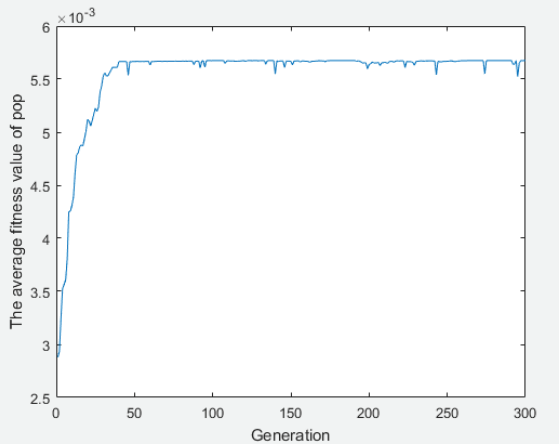

Fig 5. Convergence curve when factor level is 111

6 Conclusion

In this paper, an optimization scheme for joint motion time series is proposed on the basis of the Stanford Manipulator's mathematical model. Under the specified conditions, the minimum path of the end effector from the prescribed start point to the end point is obtained by applying the GA optimization method and performing three polynomial interpolation of the joint variables.

Genetic algorithm has good repeatability and stable solution, and is not easy to generate local optimal solution. However, the optimization effect of the algorithm is related to the predetermined position. The reason is that all joints can only reach the finishing line at the same time. In the future work, more complex joint time series constraints can be taken into consideration to provide a good optimization effect for the first and last positions.

Although the problem of obstacle avoidance is not considered in this paper, the algorithm can set up a barrier space, and then sift out all the individuals entering the barrier space in the process of population selecting. This provides some feasible direction for further research.

Refiences

1. R. Brooks, "Flesh and Machines," Pantheons Books, New York, 2002.
2. John J. Craig, "Introduction to Robotics Mechanics and Control", China Machine Press, Beijing, 2006.6.
3. Atsushi Fujimori, Peter N. Nikiforuk, and Madan M, Gupta, Adaptive Navigation of Mobile Robots with Obstacle Avoidance, IEEE Transactions on robotics and automation, 1997.8.
4. Ningyue, Liyan, and Liukeping, The Research of Mobile Manipulator Trajectory Tracking Cooperative Control based on the ADRC
5. JIA Wei-ping, The Track Project Research of Stanford Manipulator's Moving Route, Robot Technology, 2005
6. DONG Yun, YANG Tao, LI Wen. Algorithm Based on Analytical Method and Genetic Algorithm for Inverse Kinematics of Redundant Manipulator, 2012, 29(3):239-243
7. SHEN Xiao-ning, Li Sheng, GUO Yu, CHEN Qing-wei, HU Wei-li. Multi-objective Genetic Algorithm for Inverse Kinematics Problem of Redundant Manipulator[J]. Journal of System Simulation, 2008, 20(2): 399-403
8. JIA Wei-ping, The Research of Stanford Manipulator Kinematics and Compulator Simulation [D]
9. CAO Dao-you, Research and Applications Based on Improved Genetic Algorithm [D]
10. Montgomery, D.C. (2008). Design and analysis of experiments. John Wiley & Sons.