Simulation Study on Robot Zero Angle Calibration Based on Cuckoo Search Algorithm

Zefan Cai 1, 2, Daoping Huang 2, *, Yiqi Liu 2

1 Department Electronic and Information Engineering
Shunde Polytechnic
Shunde 528300, China
2 Automation College
South China University of Technology
Guangzhou 510640, China

Abstract — The robot accuracy includes two aspects: absolute accuracy and repeated positioning accuracy. In general, the repeated positioning accuracy is very high, but the absolute accuracy must be calibrated to reach a high level. The robot absolute accuracy is influenced by the error of positioning, while the positioning error is controlled by many factors, such as manufacturing precision, installation process, kinematic parameters, sensor error, the zero position angle error of each joint, etc. When a robot is installed, the zero position error is difficult to eliminate manually. This paper tried to study how to calibrate the zero position angle error automatically based on the cuckoo search algorithm assumed that other factors were accurate. The simulation results show that the method is simple and accurate, and can be applied to the actual robot calibration.

Keywords - cuckoo search algorithm; robot; zero angle error calibration; absolute accuracy

I. INTRODUCTION

The robot accuracy includes two aspects: absolute accuracy and repeated positioning accuracy. Absolute accuracy stands for the ability of the robot to reach the set points in the working space, while repeated positioning accuracy is the ability of the robot to reach the teaching points again. In general, the repeated positioning accuracy is very high, can reach the level of 0.1mm ~ 0.01mm. Absolute accuracy is not high enough before calibration, just can reach the level of 0.1cm ~ 1cm. After calibration absolute accuracy can be greatly improved and can be equal to the repeated positioning accuracy. The robot absolute accuracy is affected by the error of positioning, while the positioning error is controlled by many factors, such as manufacturing precision, installation process, kinematic parameters, sensor error, the zero position angle error of each joint, etc. The zero position angle error is not only affected by manufacturing precision and installation process, but also by encoder error. When a robot is installed, the zero position error is difficult to eliminate manually. [1]

Recently there are two kinds of methods to improve absolute accuracy: [2]

(1) To improve the robot hardware, such as improving manufacturing accuracy of mechanical components, assembly and installation precision, stiffness of the robot joints, etc. This method will increase the cost of manufacturing, and professional workers must be employed in order to guarantee the assembly and the installation precision. This method can’t deal with the post positioning error.

(2) To introduce calibration technology of software. In this method, the robot's actual parameters are identified to improve the structure model of robot. It avoids pouring money and people. When there are some changes of the robot, the parameters can be identified again. The calibration technology has a very wide application prospect in improving the absolute accuracy of the robot, and has great research value.

There are a lot of people to research on the calibration technology. Several algorithms are introduced, such as Leveenberg-Marquarde [3], least square [4], maximum likelihood estimation [5], genetic algorithm [6], simulated annealing algorithm [7], etc. This paper tries to study how to calibrate the zero angle error based on the cuckoo search algorithm assumed that other factors are accurate. In former algorithms, a large number of location samples need be obtained in order to calibrate the robot. It is difficult to finish the implementation of the calibration. If the sample number is not enough, the results of the calibration will be poor. In the method described in this paper, only a few of samples need be obtained. It is successful to overcome the above shortcomings.

Cuckoo search algorithm (CS) [8-12] is a heuristic algorithm which is developed in 2009 by Xin-She Yang and Suash Deb. CS is based on the brood parasitism of some cuckoo species. In addition, this algorithm is enhanced by the so-called Lévy flights, rather than by simple isotropic random walks. Studies show that CS is potentially far more efficient than particle swarm optimization (PSO) and genetic algorithms. Because of its high efficiency, CS has caused the attention of scholars, and more and more papers about CS
have been published, and has been applied successfully in many fields. [11-16] However, there are no papers about application of robot calibration based on CS. This paper tries to study how to calibrate the zero angle error based on the cuckoo search algorithm, and achieves the desired results.

II. DENAVIT-HARTENBERG PARAMETERS EXPRESSION OF MULTI-JOINT ROBOT

There are several expressions of robot pose, such as Euler angle, rotation matrix and four element method, etc. The most common one is the rotation matrix, and Denavit-Hartenberg (D-H) parameters expression is the most common of the rotation matrix method. D-H is developed by Denavita and Hartenberg in 1995. [17]

In D-H, the D-H parameters of the ith link must be determined firstly. There are 4 parameters: link length $a_i$, link twist $\alpha_i$, link offset $d_i$ and joint angle $\theta_i$. Then homogeneous coordinate transformation matrix $T_i$ will be calculated with D-H parameters. $T_i$ is a primary coordinate transformation matrix, which describes the relative translation and rotation between the link coordinate systems.

$$\begin{bmatrix}
\cos \theta_i & -\sin \theta_i \cos \alpha_i & \sin \theta_i \sin \alpha_i & a_i \cos \theta_i \\
\sin \theta_i & \cos \theta_i \cos \alpha_i & -\cos \theta_i \sin \alpha_i & a_i \sin \theta_i \\
0 & \sin \alpha_i & \cos \alpha_i & d_i \\
0 & 0 & 0 & 1
\end{bmatrix}$$

(1)

For rotating joints, $a_i$, $\alpha_i$ and $d_i$ are constant, while $\theta_i$ is variable. Instead for sliding joints, $a_i$, $\alpha_i$ and $\theta_i$ are constant, while $d_i$ is variable.

For a 6 links multi-joint robot, the terminal pose matrix $T$ is a relative translation between the hand (the 6th joint) and base coordinate system. $T$ is as follow: [17]

$$T = T_6 T_5 T_4 T_3 T_2 T_1$$

(2)

Assuming the position vector of the hand coordinate origin in the base coordinate is $p$, while the direction vectors are $n$, $o$ and $a$, $T$ can be described by a 4x4 matrix as follow:

$$\begin{bmatrix}
n_i & a_i & a_s & p_x \\
n_x & a_x & a_s & p_y \\
n_z & a_z & a_s & p_z \\
0 & 0 & 0 & 1
\end{bmatrix}$$

(3)

Fig. 1 is the three-dimensional (3D) simulation diagram of a low-cost 6 joints robot, and Fig. 2 is a schematic diagram of the robot and the corresponding coordinate system.

$$\begin{align}
p_i &= c_1 \left[ -s_3 \left( d_4 c_5 s_6 - d_5 s_4 \right) + s_2 \left( d_4 c_3 + d_3 \right) + a_1 + a_2 c_2 \right] \\
& \quad -s_1 \left[ d_3 s_5 c_6 + d_4 c_5 \right], \\
p_x &= s_3 \left( d_4 c_5 s_6 - d_5 s_4 \right) + s_2 \left( d_4 c_3 + d_3 \right) + a_1 + a_2 c_2 \\
& \quad + c_1 \left[ d_3 s_5 c_6 + d_4 c_5 \right], \\
p_z &= -s_2 \left( d_4 c_5 s_6 - d_5 s_4 \right) - c_2 \left( d_4 c_3 + d_3 \right) + a_2 s_2, \\
s_i &= c_1 \left[ c_{23} \left( c_4 c_5 c_6 - s_4 s_6 \right) + s_3 s_5 c_6 \right] + s_i \left( s_4 c_5 c_6 + c_4 s_6 \right), \\
s_x &= c_{23} \left( c_4 c_5 c_6 - s_4 s_6 \right) + s_3 s_5 c_6 - c_i \left( s_4 c_5 c_6 + c_4 s_6 \right), \\
s_z &= s_2 \left( c_4 c_5 c_6 - s_4 s_6 \right) - c_2 \left( d_4 c_3 + d_3 \right), \\
a_s &= s_1 \left[ c_{23} \left( c_4 c_5 c_6 - s_4 s_6 \right) + s_3 s_5 c_6 \right] - s_i \left( s_4 c_5 c_6 - c_4 s_6 \right), \\
a_x &= s_2 \left( c_4 c_5 c_6 - s_4 s_6 \right) + c_2 \left( d_4 c_3 + d_3 \right), \\
a_z &= c_1 \left[ c_2 c_5 s_6 + s_4 c_6 \right] + s_2 s_5 c_6, \\
a_y &= -s_1 \left( c_2 c_5 s_6 + s_4 c_6 \right) + c_2 \left( d_4 c_3 + d_3 \right), \\
a_y &= -s_1 \left( c_2 c_5 s_6 + s_4 c_6 \right) - c_2 \left( d_4 c_3 + d_3 \right) \\
a_z &= -s_2 \left( c_4 c_5 c_6 - s_4 s_6 \right) - c_2 \left( d_4 c_3 + d_3 \right)
\end{align}$$

(4)

Where $s_i = \sin(\theta_i)$, $c_i = \cos(\theta_i)$, $s_j = \sin(\theta_j + \theta_i)$, $c_j = \cos(\theta_j + \theta_i)$. 

Figure 1. 6 joints robot 3D simulation diagram

Figure 2. 6 joints robot schematic diagram

DOI 10.5013/IJSSST.a.17.09.16 16.2 ISSN: 1473-804x online, 1473-8031 print
According to Eq. (1) and (4), since all the 6 joints of the robot are rotary joint, the variables of D-H parameters are \( \theta_i (i = 1 \sim 6) \). If all the \( \theta_i \) are known, it is easy to get \( T \) with Eq. (4). When a robot is assembled, \( a_i, d_i \) and \( \alpha_i \) are fixed, and we assume they are accurate, while \( \theta_i \) are variables which can be determined by angle encoder. Since angle encoder always has zero error, \( \theta_i \) in Eq. (1) and (4) must be replaced by \( \theta_i' \), where \( \theta_i' = \theta_i + \Delta \theta_i \), and \( \Delta \theta_i \) is zero error modifier. In order to improve positioning accuracy, \( \Delta \theta_i \) must be found. This paper will introduce how to get \( \Delta \theta_i \) based on CS.

III. CUCKOO SEARCH ALGORITHM

When XinShe Yang and Suash Deb described CS, they defined three rules as follows: [9-10]

Rule 1: Each cuckoo lays one egg at a time, and dumps it in a randomly chosen nest;

Rule 2: The best nests with high-quality eggs will be carried over to the next generations;

Rule 3: The number of available host nests is fixed, and the egg laid by a cuckoo is discovered by the host bird with a probability. In this case, the host bird can either get rid of the egg, or simply abandon the nest and build a completely new nest.

Based on these three rules, the concrete steps of CS are as follows:

1. Initialization. Randomly generate an initial population of \( n \) nests at the position, \( X_0 = (x_0^0, x_0^1, \cdots, x_0^n) \), then evaluate their fitness values \( F_0 \) and find the current global Best one.

2. Update. The positions to the new positions \( X_i = (x_i^0, x_i^1, \cdots, x_i^n) \) by Levy flights, then evaluate their fitness values \( F_i \) and find the current global value. Record the global value and its corresponding position.

3. Selection. Draw a random number \( r \) from a uniform distribution \([0, 1]\). Update \( X_i \) if \( r > p_a \). Then evaluate their fitness values \( F_i \) again and find the current global value. Record the global value and its corresponding position.

4. Judgement. If the stopping criterion is met, then the best global position is found so far. Otherwise, return to step (2).

In the above steps, CS algorithm not only uses the Levy flights search method, but also introduces the elite retention strategy. In step (3), a second best solution may be selected, if there is no elite retention strategy, the best solution may be lost. A good combination of local search and global search is obtained in this algorithm. The selection step has increased the diversity of positions. In the case of sufficient number of iterations, CS can converge to the global optimal solution in probability 1. [9]

In search step, Levy flights use the random walk strategy shown as Eq. (5), which is in order to strengthen global search ability. [19]

\[
X_{i+1} = X_i + \alpha \odot \text{Lev}y(\beta)
\]  

Where \( X_i \) represents \( i \)th solution in \( t \)th generation; \( \alpha \) is the step size used to control the range of random search. A bigger \( \alpha \) is better for global search and a smaller is better for local search. In the whole search, \( \beta \) should be from big to small. In general, \( \alpha \) is controlled by search range, \( \alpha = O(L/10) \), where \( L \) is the search range. If Eq. (5) changes to \( X_{i+1} = X_i + \alpha \), then Levy flights will change to standard random walks. As Levy flights have infinite mean and variance, CS can explore the search space more efficiently than algorithms using standard random walks. This advantage, combined with both local search capabilities and guaranteed global convergence, makes CS very efficient.

Paper [10] adopted the step size shown as follow:

\[
\alpha = \alpha_0 (X_i - X_{\text{best}})
\]  

Where, \( \alpha_0 \) is constant which is 0.01 in the most time; \( X_{\text{best}} \) is the current global optimal solution.

In Eq. (5) \( \odot \) represents point multiplication; \( \text{Lev}y(\beta) \) obeys Levy probability distribution as follow:

\[
\text{Lev}y(\beta) \sqcup u = r^{-\beta}, 0 < \beta \leq 2
\]  

Paper [10] adopted Eq. (8) to calculate \( \text{Lev}y(\beta) \).

\[
\text{Lev}y(\beta) \sqcup u = r^{-\beta}, \beta = 1.5
\]  

Where \( u \) and \( v \) obey the standard normal distribution. \( \phi \) is as follow:

\[
\phi = \left[ \frac{\Gamma(1+\beta)\times\sin(\pi\times\beta/2)}{\Gamma(\frac{1+\beta}{2})\times\beta\times2^{(\beta-1)/2}} \right]^{1/\beta}
\]  

According to Eq. (5)~(9), the update equation in Levy flights is shown as Eq. (10).

\[
X_{i+1} = X_i + \alpha_0 \odot \text{Lev}y(\beta)(X_i - X_{\text{best}})
\]  

In selection step, after some bad positions are discarded, the same number of new positions will be produced by Eq. (11).

\[
X_{i+1} = X_i + r(X_i - X_{i,k})
\]  

Where \( r \) is a scaling factor, which is a random number in uniform distribution \([0, 1]\). \( X_{i,j} \) and \( X_{i,k} \) are two solutions in \( t \)th generation.

IV. FIND ROBOT ZERO ANGLE ERRORS WITH CS

The steps to find robot zero angle errors \( \Delta \theta_i \) with CS are as follow:

1. Drive the robot to any \( N \) positions with the controller, then get these \( N \) positions’ coordinates with some equipments, read and record the corresponding angles \( \theta_i \) in controller, where \( j = 1 \sim N, i = 1 \sim 6 \).
(2) Consider the 6 expected zero angle errors as the cuckoos’ position. At the begin, initialize all the nests’ positions, in other words, each cuckoo has 6 random zero angle errors $\Delta \theta_m$, where $m$ is the number of the cuckoo, $i = 1 \sim 6$.

(3) Calculate theoretical values of these N positions with $\theta_i$ in step (1) and $\Delta \theta_m$ in step (2).

(4) Calculate the differences of actual values and theoretical values for each position, as shown in Eq. (12) or Eq. (13).

$$F_j = \sum_{n=1}^{3} |T_{act} - T_{theo}|$$

$$F_j = \sum_{n=1}^{3} \sum_{i=1}^{4} |T_{act} - T_{theo}|$$

Where $T$ is the actual pose matrix, $T_{act}$ is the element of $T$; $T_i$ is the calculation pose matrix, $T_{theo}$ is the element of $T_i$. The difference between Eq. (12) and (13) is that Eq. (12) is concerned only at the distance of the robot terminal position, while Eq. (13) pays attention to the distance of the robot terminal position and direction at the same time.

(5) Calculate the average value $F_j$ of $F_i$, and consider $F_j$ as the cuckoo’s fitness. Save the current best fitness and position.

(6) Update the positions with Eq. (10).

(7) Calculate fitness again, and save the current best fitness and position.

(8) Draw a random number $r$ from a uniform distribution $[0,1]$. Update the positions if $r > p_a$.

(9) Calculate fitness once again, and save the current best fitness and position.

(10) If the stopping criterion is met, then the best global position is found so far. Otherwise, return to step (3).

V. SIMULATION AND VERIFICATION

In simulation, 6 small random zero angle errors $\Delta \theta_i$ are generated from a uniform distribution $[-0.1, 0.1]$, they are $[-0.1336, -0.1856, 0.0116, 0.1728, -0.0688, -0.1301]$ (Unit: º). $N$ of joint thetas $\theta_i$ are generated from a uniform distribution $[-180, 180]$ (Unit: º). Then the pose matrices are calculated with Eq. (4) based on $\Delta \theta$ and $\theta_i$, which are considered as the robot pose matrices.

In the CS, the dimension is 6, $\alpha_0 = 0.01$, $\beta = 1.5$, $p_a = 0.25$, the population size is 35, iterations is 500. The simulation results with different position numbers $N$ are shown in table 1 and 2, where $\Delta \theta_i(i = 1 \sim 6)$ are zero angle errors searched in CS, $F$ is the average distance between each pair of theoretical positions and actual positions before calibration, while $F'$ is the averages after calibration. $F''/F$ is the multiple of the position distance errors before and after calibration.

The reasons of parameters selection are as follow:

Since there are 6 parameters to be calibrated, so the dimension is set to 6.

The values of these three parameters, $\alpha_0$, $\beta$, and $p_a$, are the recommended values from the literature which are obtained after a large number of experiments.

The global optimal value is easier to obtained if the population size is bigger, but bigger population size needs more memory and time; after experimental verification, size 35 is good for accuracy, memory and time;

The choice of iteration number is determined by the accuracy of the calibration.

According to table 1, when $N$ changes from 1 to 5, $F'' / F$ decrease rapidly, while $N$ from 6 to 10, $F'' / F$ has fluctuated. The value of $F'' / F$ is smaller when $N = 5$ than $N = 6$. From the start of $N = 2$, the search value of the first 5 zero errors are close to the actual values, while the 6th is random. The reason why the 6th is random is that the 6th joint is a revolute joint, its revolute angle $\theta_i$ just affects its direction but not affects its position, and Eq. (12) can only represents the error of position. In the algorithm with Eq. (12), the optimal value of the 6th joint zero angle error can’t be searched.

The characteristic of the table 2 is basically similar to table 1. The only difference is that from the start of $N = 2$ all the zero angle errors can be accurately searched in table 2 because Eq. (13) can represent the error of position and direction at the same time. Eq. (12) and Eq. (13) each have their advantages and disadvantages. When using Eq. (12), we just need a three coordinate tester to get the robot terminal position coordinate, and it is simple and cheap, but the 6th zero angle error can’t be searched.

When using Eq. (13), we need a double theodolite tester to get the robot terminal position coordinate and direction, and it is complex and expensive, but all the zero angle errors can be searched. Regardless of the situation, if the accuracy is not particularly high, then $N = 2$ is OK, otherwise $N = 5$ is better.
In this algorithm, the fitness curve throughout the iteration is shown in Fig. 3. According to the figure, the search value is close to the optimal value at the 100th iteration. In the first 100 iterations, search values quickly close to the optimal values, and then converges slowly to the optimal value after 100 times.

![Algorithm evolution curve](image)

Figure 3. Algorithm evolution curve

According to table 1, table 2 and Fig. 3, if we want the algorithm quickly converge, then set N to 2, iteration number set to 150.

VI. CONCLUSION

The absolute accuracy of the robot is affected by many aspects. The joints’ zero angle errors are one of them. With the cuckoo search algorithm introduced in this paper, the optimal values of the zero angle errors can be found quickly. In this algorithm, just a few position messages need to be obtained. It is easy to operate. This method has good generality. It not only can be applied to the robot calibration with revolute joints, but also can be applied to the robot calibration with sliding joints.

REFERENCES


