MDPI

Article

Using Particle Filter to Solve the Problem of Symmetric Multiple Solutions in Inverse Kinematics of Manipulator

Chien-Lin Chiang [1], Chang-Chen Hsieh [2], Mao-Hung Yang [3], I-Long Lin [4], and Yi-Yuan Chiang [5,*]

[1] Tatung University; f2267505@gmail.com
[2] Vanung University; superlongtw@gmail.com
[3] Vanung University; 0309.kevin@gmail.com
[4] Tatung University; cyberpaul@ttu.edu.tw
[5] Vanung University; yychiang.vnu@gmail.com
* Correspondence: yychiang.vnu@gmail.com; Tel.: +886-925-688317

Abstract: In the inverse kinematics of robotic arm, the most frequently encountered problem is multiple solutions for the poses of the arm. The multiple solutions often appear symmetrically. In practical applications, a set of solutions must be selected from the multiple solutions as the pose of the robot arm. The symmetric multiple solutions are all reasonable poses reachable by the robot arm. However, in the process of arm movement, if the solution is not selected properly, the robotic arm will rapidly change from a solution to a solution in two adjacent time samples. The posture of the symmetric solution sampled at the previous time will cause the mechanical arm to change the posture rapidly and cause damage to the mechanism, and even cause the mechanism to oscillate greatly. To avoid the interference of symmetric multiple solutions, we adopt the non-parametric Bayesian filter (that is, particle filter) based on the Monte Carlo method to process the robotic arm instead of the traditional inverse kinematics method using algebraic or geometric methods. The particle filter uses many random sample points and the design of importance weights to make the sample points converge to the optimal solution in an iterative process. This paper shows how to use the design of importance weights to prevent the oscillation problem of symmetrical multiple solutions in adjacent time sampling during the movement of the manipulator.

Keywords: mechanical arm; inverse kinematics; multiple solutions; symmetry

1. Introduction

In today's society, after entering the industrial era 4.0, robotic arms have become an indispensable and important piece of equipment in many heavy industries today. In many applications, robotic arm has its important contribution, especially in many semi-automatic or even fully automatic factories. The application field is also very wide, automotive, plastics, electronics, chemical, machinery, medical and many other fields, are often seen in the robot arm related applications, especially automated assembly, transport, cutting and welding. The problem of multiple solutions is often encountered in the inverse kinematics of mechanical arms. The multiple solutions often appear symmetrically. In practical application, a set of solutions must be selected from the multiple solutions as the pose of the robotic arm. Although the symmetric multiple solutions are reasonable poses for the robotic arm, but in the process of arm movement, if the solution is not selected properly, it will make the robotic arm change the pose from a certain solution to the symmetric solution of the previous time in two nearby time samples, which will make the robotic arm change the pose quickly and cause damage to the mechanism, or even cause the mechanism to vibrate significantly. In order to avoid the interference of symmetric multiple solutions, we abandon the traditional algebraic or geometric method of inverse kinematics, and adopt the Monte Carlo-based nonparametric Bayesian filter (also known as particle filter) to deal with the problem of robot arm inverse kinematics. The particle filter is designed with a

Citation: Lastname, F.; Lastname, F.; Lastname, F. Title. *Journal Not Specified* **2022**, *1*, 0. https://doi.org/

Received:
Accepted:
Published:

large number of random sample points and importance weights to allow the sample points to converge to the best solution in an iterative process. In this paper, we illustrate how to use the importance weight design to avoid the oscillation problem of multiple solutions of symmetry in the adjacent time samples during the robot motion. In the process of industrial automation, we believe that robotics-related industries, including robotic arms, are sure to shine in the future. We also hope to improve the efficiency and quality of robotic arms through this study.

2. Background Knowledge: Inverse Kinematics

The purpose of inverse kinematics is to investigate how the end point of a robotic arm reaches a specific position and attitude in space, i.e., given the end point of a bar (Link) The spatial position of the end point of the link in the absolute coordinate system is derived from the equation to find the rotation angle of each joint. In the process of solving, it is necessary to consider whether the end point Whether the end point is within the reach of the robot arm (Figure 2-16, 17), the geometric limitation of the arm itself, and whether there is any obstacle in the arm movement. If there are multiple solutions, the solution closest to the current state is chosen [2].

In forward kinematics, we use the known to find the In inverse kinematics, we use the known to find . Because the inverse kinematics will encounter multiple solutions to the problem, the solution method will be relatively more complicated than the forward one [9].

The solution methods are broadly classified as follows: analytical (geometric, algebraic, etc.) and numerical methods [3][4].

2.1. Geometric Method

The geometric methods could be used to solve not only 3D robot arms, but also 2D planar robots. For 3D cases, it just need to separate the 3D space geometry into 2D planar geometry as shown in Figure 1.

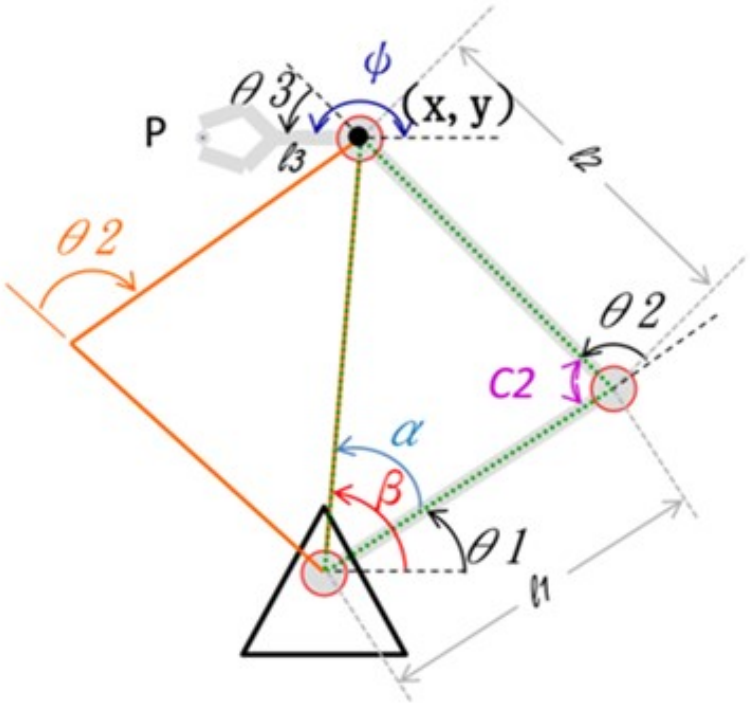

Figure 1. The schematic of RRR type robot arm.

The end point of link2 is (x, y). Use the cosine theorem to find θ_2. The angle could be plotted into two solutions: the green triangle and the orange one. Take Figure 1 as an example, using the geometric method, the space is separated into planar geometry. Since l_1

and l_2 are the lengths of Link1 and Link2, respectively; we could use the cosine theorem to find the $c2$ angle. Therefore, we have the floolwing equations:

$$x^2 + y^2 = l_1^2 + l_2^2 - 2l_1 l_2 \cos(\pi - \theta_2) \tag{1}$$

$$\cos(\theta_2) = \frac{x^2 + y^2 - l_1^2 - l_2^2}{2l_1 l_2} \tag{2}$$

Then, θ_2 along with θ_1 and θ_3 could be solved by inverse trigonometric functions.

2.2. Algebraic Method

The algebraic method is to directly combine the items in the pose matrix $T_0, T_1, \ldots, T_n$, and to define the intermediate variables (which is a combination of joint parameters). for the purpose of constructing a one-dimensional high-order equation. All joint angles can be find by solving this equation. We use the same example as shown in Figure 1, the pose matrix of the end point is

$$
\begin{aligned}
{}^0_3T &= \begin{bmatrix} \cos(\theta_1+\theta_2+\theta_3) & -\sin(\theta_1+\theta_2+\theta_3) & 0 & l_1\cos\theta_1 + l_2\cos(\theta_1+\theta_2) \\ \sin(\theta_1+\theta_2+\theta_3) & \cos(\theta_1+\theta_2+\theta_3) & 0 & l_1\sin\theta_1 + l_2\sin(\theta_1+\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} && (3) \\
&= \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 & x \\ \sin(\phi) & \cos(\phi) & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. && (4)
\end{aligned}
$$

By (3) and (4), all angles could be solved by simple algebraic computations.

3. Particle filter and application in inverse kinematics

3.1. Particle filter

Particle filters are commonly used for robot positioning and tracking applications. In this paper, we use the particles in the particle filter to represent the pose of the robotic arm and the position of the particles to represent the end position of the robotic arm. Thus, each particle can be considered as a robot arm state, and each joint angle is embedded in the particle parameters [5]. A group of particles is used to track the position of the robotic arm branch, and the attitude of the particles is averaged to find the estimated attitude. In this way, the inverse kinematics of the robotic arm can be efficiently calculated.

For a particle filter, a particle is a pair with n variable of $p = (v_1, v_2, \ldots, v_n)$. where v_i is a certain state of the robot. For example, the joint angle and end position of a robot arm; the position coordinates, speed, acceleration, and direction of a self-driving car, etc. Simply put, the particle set here is a set of hundreds, thousands, or tens of thousands of particles, which is what we call the particle filter summary.

Particle filter is a kind of Bayesian filters, specifically, nonparametric Bayesian filters. The Bayesian filter is an algorithm for estimating the state of a robot based on Bayes' theorem in the theory of probability [6]. Before going deeper into the theory, let us first explain the robot's environment. In this paper, we refer to the robot as a multi-joint robot arm. The state of the robot refers to the angle of each joint of the robot arm and the coordinates of the end position of the arm. Observations of sensors, in a broad sense, refer to the readings of any sensors mounted on the robot. In the study of the inverse kinematics of the robot arm, we assume that the current states of each joint, i.e., the current angles, can be read by the internal encoder of the motor. The end position of the robot arm can be indirectly calculated from the angle of each joint and the link length, and is therefore observable. The control variavle is a variable that can be controlled on the robot. Changing the control variables will result in a change in the robot's state. The control variables of the robot arm are the joints of each controllable angle.

3.2. Design of particle filter in inverse kinematics

The main purpose of this section is to use the particle filter to deal with the inverse kinematics of the robot arm. The so-called inverse kinematics is to find an attitude under the condition that the end coordinates of the robot arm are known, so that the end coordinates of the robot arm under the attitude can meet the condition. The attitude of the robot arm is determined by all the joint angles together. Therefore, in more detail, the so-called inverse kinematics is to derive the angle of each joint from the known terminal coordinates, so that the terminal coordinates of the robot arm can meet the required conditions under the attitude.

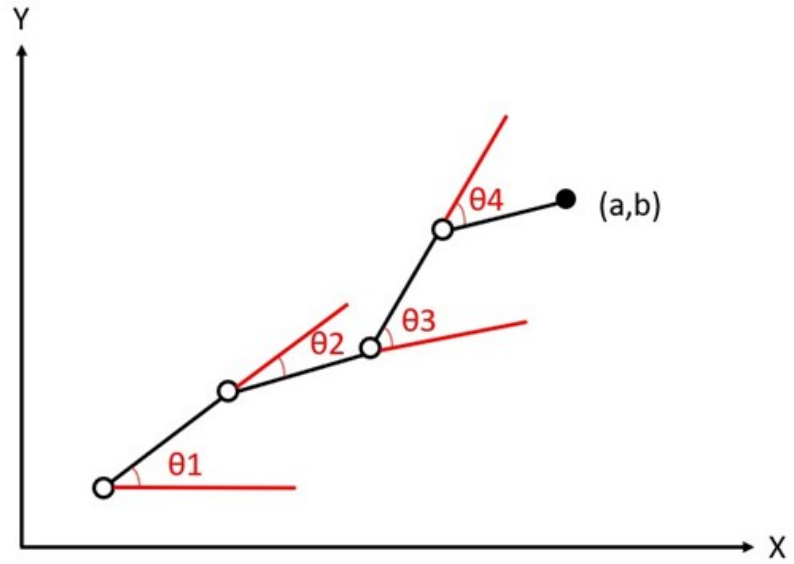

Figure 2. Four joints robot arm.

In Figure 2, we have marked the joint angles. The angle of each joint is calculated from the extension of the previous bar to the angle of the bar of the joint. The first joint is calculated from the horizontal line to the angle of the first bar, since there is no previous bar. The attitude of the robot arm is defined as . The so-called example in the particle filter, as applied to the inverse kinematics of the robot arm, refers to a specific posture. As we described in the previous section, particle filters are generally used in robot positioning and tracking problems. In inverse kinematics, we have to invert the pose of a robot arm from a known terminal coordinate, which is actually very similar to the concept of localization and tracking. We can consider the process of to find the appropriate attitude of the robot arm as a localization problem, that is, to find out the position of the attitude. On the other hand, we can also consider this problem as a tracking problem, where the target pose is the state we want to track [11].

After understanding how to correspond the inverse kinematics with the particle filter, we will explain the details of the particle filter algorithm.

3.2.1. Step 1: Random particle generation

The first step of the algorithm is to randomly generate particles. Suppose we want to generate particles, these particles represent different poses of the robot arm. The following symbols are used to represent particles.

$$(\theta_1^{(1)}, \theta_2^{(1)}, \theta_3^{(1)}, \theta_4^{(1)}) \tag{5}$$

$$(\theta_1^{(2)}, \theta_2^{(2)}, \theta_3^{(2)}, \theta_4^{(2)})$$
$$\vdots$$
$$(\theta_1^{(n)}, \theta_2^{(n)}, \theta_3^{(n)}, \theta_4^{(n)})$$

The lower number represents the number of joint angles, while the upper number represents the number of particles.

3.2.2. Step 2: Calculate importance weights

After the random particle generation, the next step is to calculate the weight of each particle. The weight of the particle represents the good or bad of the example, so in order to calculate the weight, we need to find the error of the particle first. To calculate the error, we have to use the robot arm's forward kinematics to obtain a terminal coordinate from the particle's joint assembly, $(x^{(i)}, y^{(i)})$, and then use the distance between the calculated terminal coordinates and the actual terminal coordinates (x_0, y_0). The distance between the calculated terminal coordinates and the actual terminal coordinates is used as the error function. After the error is obtained, the weight can be further calculated. In the following, we explain the above steps in detail.

Each particle represents a certain attitude from which an end coordinate can be calculated using the forward kinematics. The forward kinematics can be considered as a function (FK). Then,

$$(x^{(i)}, y^{(i)}) = FK(\theta_1^{(i)}, \theta_2^{(i)}, \theta_3^{(i)}, \theta_4^{(i)})$$

The distance is calculated using the usual Euclidean distance, $d^{(i)}$, represents the distance from the first particle to (x_0, y_0). The distance,

$$d^{(i)} = \sqrt{(x^{(i)} - x_0)^2 + (y^{(i)} - y_0)^2} \tag{6}$$

Once we have the distance, we can convert the distance to a weight in the following way.

$$w^{(i)} = \frac{1}{1 + d^{(i)}}$$

In order to deepen the difference between good and bad particles, the above weights can be taken as times. As for how many times, it must be decided in the experiment.

3.2.3. Step 3: Resampling

The weight of each particle calculated in the previous step will be used as the basis for resampling. We convert the weights into probabilities, and then use the probabilities to resample the particles. Therefore, the higher the weight of a particle, the more often it will be selected after resampling. First, each weight must go through a formalization process before it can be converted into a probability. Define

$$W = \Sigma_{i=1}^{n} w^{(n)}.$$

Then,

$$\overline{w}^{(i)} = \frac{w^{(i)}}{W} = p^{(i)} \tag{7}$$

The probability here will be the probability that the particle will be selected after resampling. In order to keep the number of particles from decreasing due to re-sampling, we add a perturbation error to each particle when re-sampling.

3.2.4. Step 4: Step 4: Solve the new pose and return to Step 2

Finally, all the resampled particles are applied to the forward kinematics to obtain a group of end coordinates. This group of end coordinates is averaged to get an estimate of the current end coordinates of the robot arm. After the calculation is completed, return to Step 2 for the second estimate.

4. Experimental Results

4.1. Simulator design

In order to test the effectiveness of the method proposed in this paper, we designed a software robotic arm simulator. This design is based on an object-oriented design approach. Any planar robotic arm structure can be generated by the `RobotArm` class. The `RobotArm` category is composed of the `Link` class. We use the bottom-up approach to design the `Link` class first, and its attributes are:

- `length`: the length of the link
- `joint_x`: The joint of this bar (link) Coordinate
- `joint_y`: the joint of this bar (link) Coordinate
- `angle`: The angle of the joint
- `end_x`: Terminal coordinates
- `end_y`: terminal coordinates

This class has a member function `moveto()` to move the bar. The class `RobotArm` can be defined using the above Link class. If we take a four-rod planar robot arm, the Python language defines it as follows.

```
class RobotArm:
    def __init__(self, x, y, theta1, theta2, theta3, theta4, L1, L2, L3,
                         L4):
        self.link1 = Link(x, y, theta1, L1, 'black',20)
        self.link2 = Link(self.link1.end_x, self.link1.end_y,
                                    theta1+theta2, L2, 'blue',15)
        self.link3 = Link(self.link2.end_x, self.link2.end_y,
                                    theta1+theta2+theta3, L3, 'green',10)
        self.link4 = Link(self.link3.end_x, self.link3.end_y,
                                    theta1+theta2+theta3+theta4, L4, 'red',5)
    def moveto(self, theta1, theta2, theta3, theta4, penState=True):
        self.link1.moveto(self.link1.joint_x, self.link1.joint_y,
                                     theta1, False)
        self.link2.moveto(self.link1.end_x, self.link1.end_y,
                                     theta1+theta2, False)
        self.link3.moveto(self.link2.end_x, self.link2.end_y,
                                     theta1+theta2+theta3, False)
        self.link4.moveto(self.link3.end_x, self.link3.end_y,
                                     theta1+theta2+theta3+theta4, penState)
```

Then we explain how to test this simulated arm. First, we create the arm and set its starting position. Assume that the base is $(0, 0)$ and the length of each bar is 100 then the end coordinates are $(400, 0)$.

```
myarm = RobotArm(-200,-200, 0,0,0,0,0, 100,100,100,100)
```

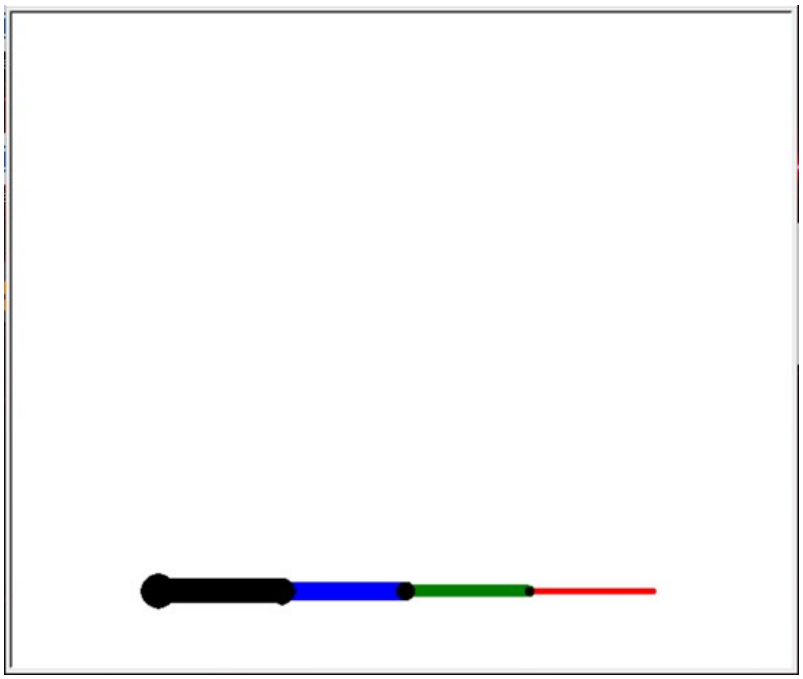

Figure 3. Moves the terminal to the $(0, 400)$ position.

4.2. Experimental results of simulation

In Figure 3, the robot arm has four links (Link1 Link4). Assuming that the robot arm moves along a straight line, the attitude of each link of the robot arm is calculated by using the particle filter. We can see the performance of the robot arm by the experimental results as follows. When the robot arm uses the particle filter and the algorithm combined with a large number of random sample points and importance weights, it is obvious that the operation process of the arm is relatively smooth and smooth, and no oscillation occurs.

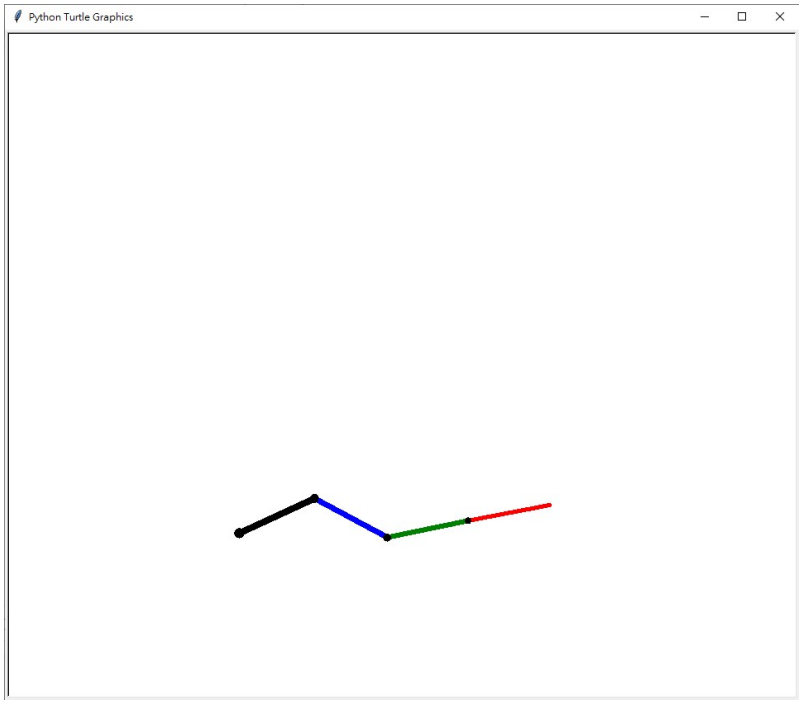

Figure 4. Arm start point operation.

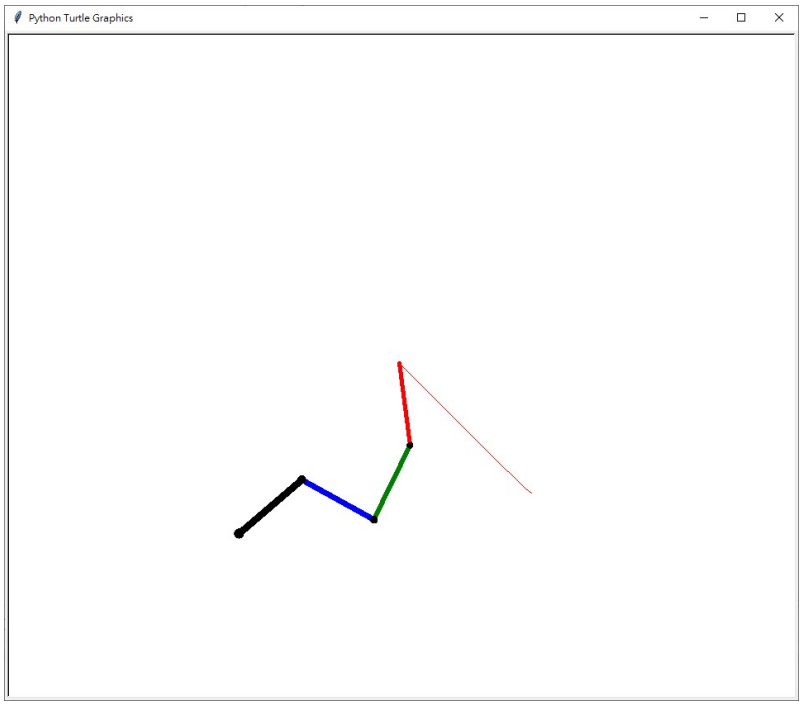

Figure 5. Arm operation to the center point.

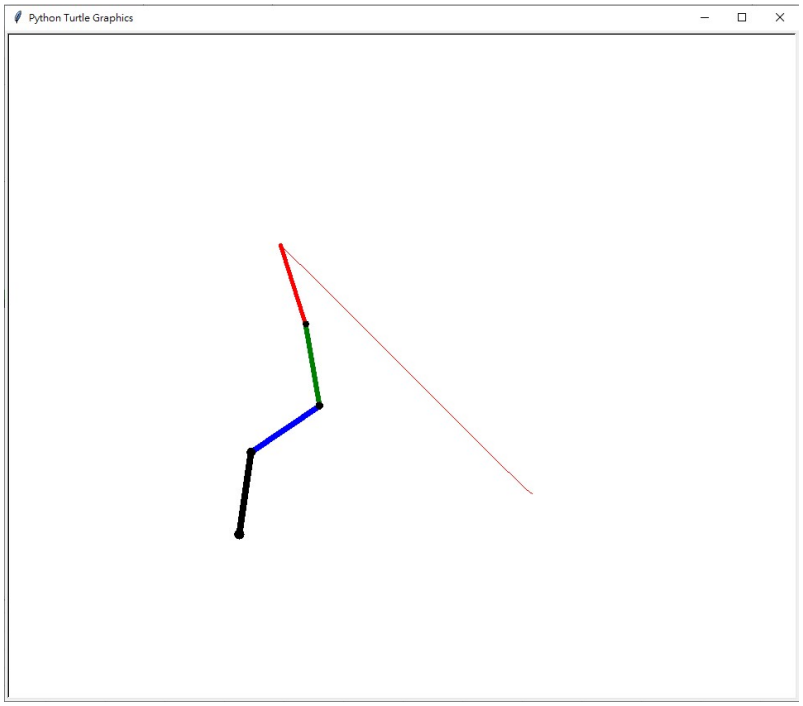

Figure 6. Arm operation to the end.

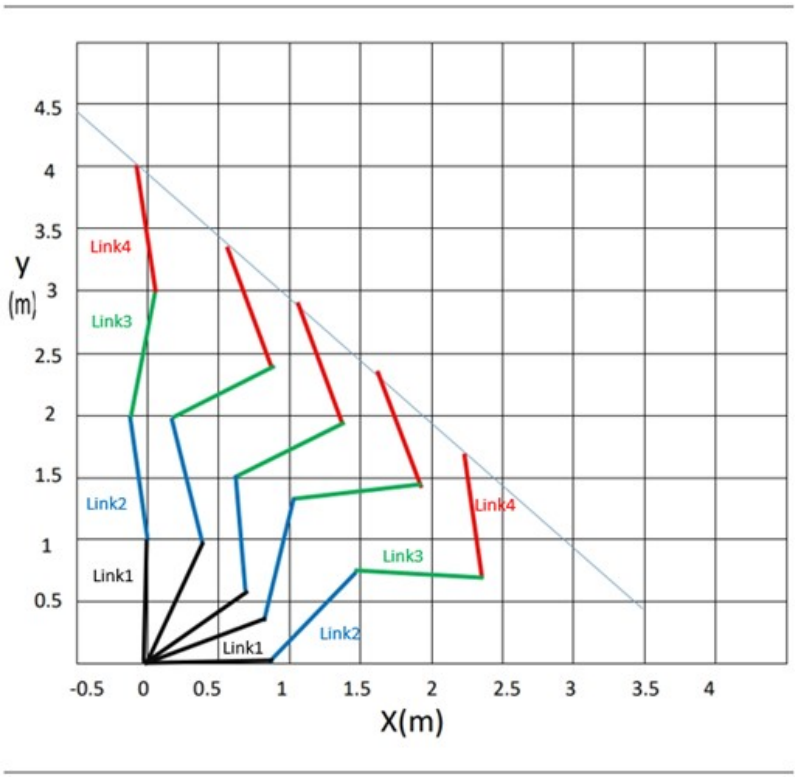

Figure 7. Movement of the arm in different positions.

5. Conclusions

A common problem encountered in the inverse kinematics of mechanical arms is the problem of multiple solutions. The multiple solutions often appear symmetrically. In order to avoid the interference of symmetric multiple solutions, we abandon the traditional algebraic or geometric inverse kinematics approach and adopt the Monte Carlo-based non-parametric Bayesian filter (also known as particle filter) to deal with the inverse kinematics problem of robotic arm. The particle filter is designed with a large number of random sample points and importance weights to allow the sample points to converge to the best solution in an iterative process. This paper illustrates how to use the importance weight design to avoid the oscillation problem of symmetric multiple solutions in the adjacent time samples during the robot motion. Future goals for robotic arms operating in three dimensions are bound to be needed for future research work, and we will be working toward solving the problem of robotic arms in three dimensions.

References

1. Goldenberg, A.A.; Benhabib, B.; Fenton, R.G. A complete generalized solution to the inverse kinematics of robots. *IEEE Journal on Robotics and Automation* **1985**, *1.1*, 14–20.
2. D'Souza, A.; Vijayakumar, S.; Schaal, S. Learning inverse kinematics. *Proceedings 2001 IEEE/RSJ International Conference on Expanding the Societal Role of Robotics in the Next Millennium (Cat. No. 01CH37180)* **2001**, *1*.
3. Kostic, D. et al. Modeling and identification for high-performance robot control: an RRR-robotic arm case study. *IEEE Transactions on Control Systems Technology* **2004**, *12.6*, 904–919.
4. Hou, Z. Geometric method for global stability of discrete population models. *Discrete and Continuous Dynamical Systems-B* **2020**.
5. Falgout, R.D. An introduction to algebraic multigrid. *Computing in Science and Engineering* **2006**, *8*.
6. Jouin, M. et al. Particle filter-based prognostics: review, discussion and perspectives. *mechanical systems and signal processing* **2016**, *72-73*,2–31.
7. Mochnac, J.; Marchevsky, S.; Kocan, P. Bayesian filtering techniques: Kalman and extended Kalman filter basics. *2009 19th International Conference Radioelektronika* **2009**.
8. Labbe, R.R. *Kalman and bayesian filters in python*,2014; pp. 43–63.
9. Birgé, L.; Massart, P. Gaussian model selection. *Journal of the European Mathematical Society* **2001**, *3.3*, 203–268.

10. Jiang, Z. et al. A new kind of accurate calibration method for robotic kinematic parameters based on the extended Kalman and particle filter algorithm. *IEEE Transactions on Industrial Electronics* **2017**, *65*.
11. Junior, J. et al. FRPSO: Inverse kinematics using fully resampled particle swarm optimization. *2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE)* **2018**.