

Lesson 1 What is Inverse Kinematics?

1. Robot Kinematics

The robot kinematics can be divided into forward kinematics and inverse kinematics. Forward kinematics refers to the position information of end effector can be obtained when then joint angle and linkages parameters are known. For inverse kinematics, given the robot's end-effector positions, the joint position can be calculated. (the end-effector refers to the top part of the gripper framed in the below red box)



In practical, the desired position where the end effector will reach to can not be known if using forward kinematics to control robot arm. Therefore, inverse kinematics is usually used to control robot arm because it can calculate the joint angle, given the robot's end-effector positions.

2. Solution to Inverse Kinematics

Geometric and algebraic solutions are usually used in inverse kinematics.

The geometric solution is used here. Given the linkage length (the distance between two servos), set the end-effector coordinate (X, Y and Z) and the pitch angel, and then calculate the joint angle according to the trigonometric function.

The code path of the detailed calculation process:

ArmPi_mini\ArmlK\ArmMoveIK.py

```

16 class ArmlK:
17     servo3Range = (500, 2500.0, 0, 180.0) # pulse width, angle
18     servo4Range = (500, 2500.0, 0, 180.0)
19     servo5Range = (500, 2500.0, 0, 180.0)
20     servo6Range = (500, 2500.0, 0, 180.0)
21
22     def __init__(self):
23         self.setServoRange()
24
25     def setServoRange(self, servo3_Range=servo3Range, servo4_Range=servo4Range, servo5_Range=servo5Range,
26         servo6_Range=servo6Range):
27         # adapt to different servo
28         self.servo3Range = servo3_Range
29         self.servo4Range = servo4_Range
30         self.servo5Range = servo5_Range
31         self.servo6Range = servo6_Range
32         self.servo3Param = (self.servo3Range[1] - self.servo3Range[0]) / (self.servo3Range[3] - self.servo3Range[2])
33         self.servo4Param = (self.servo4Range[1] - self.servo4Range[0]) / (self.servo4Range[3] - self.servo4Range[2])
34         self.servo5Param = (self.servo5Range[1] - self.servo5Range[0]) / (self.servo5Range[3] - self.servo5Range[2])
35         self.servo6Param = (self.servo6Range[1] - self.servo6Range[0]) / (self.servo6Range[3] - self.servo6Range[2])
36
37     def transformAngleAdaptArm(self, theta3, theta4, theta5, theta6):
38         # convert the angle calculated by inverse kinematics into pulse width corresponding to servo
39         servo3 = int(round(theta3 * self.servo3Param + (self.servo3Range[1] + self.servo3Range[0])/2))
40         if servo3 > self.servo3Range[1] or servo3 < self.servo3Range[0]:
41             logger.info('servo3(%) exceed the range(%, %s)', servo3, self.servo3Range[0], self.servo3Range[1])
42             return False
43
44         servo4 = int(round(theta4 * self.servo4Param + (self.servo4Range[1] + self.servo4Range[0])/2))
45         if servo4 > self.servo4Range[1] or servo4 < self.servo4Range[0]:
46             logger.info('servo4(%) exceed the range(%, %s)', servo4, self.servo4Range[0], self.servo4Range[1])
47             return False
48
49         servo5 = int(round((self.servo5Range[1] + self.servo5Range[0])/2 + (90.0 - theta5) * self.servo5Param))
50         if servo5 > ((self.servo5Range[1] + self.servo5Range[0])/2 + 90 * self.servo5Param) or servo5 < ((self.servo5Range[1]
51         + self.servo5Range[0])/2 - 90 * self.servo5Param):
52             logger.info('servo5(%) exceed the range(%, %s)', servo5, self.servo5Range[0], self.servo5Range[1])
53             return False

```