智能技术与系统综合实验
实验手册

实验 1：逆运动学控制

1.实验简介

舵机简介：

本次实验的 C5 智能小车机械臂的驱动电机为总线舵机，它与上节实验课用到的电机一样，通过一个 UART 口即可完成舵机角度的控制。总线舵机的通讯协议为：#idPpwmTtime!，例如：#000P1500T1000!

其中，#和!是协议中固定的格式。id 的范围是 000-254，必须为三位数，不足的位补 0，255 为广播 id，所有的设备都会响应这个指令。pwm 的范围是 0500-2500，必须为四位数，不足的位补 0。该总线舵机的可控角度范围为 270 度，通过 pwm 的值用来控制角度。当 pwm=1500 时，舵机旋转到 0 角度（开启机器人时手臂处于直立状态，此时所有的舵机角度都为 0 值）；当 pwm=0500 时，舵机角度旋转至-135 度；当 pwm=2500 时，舵机角度旋转至 135 度。time 代表时间，也必须为 4 位，范围为 0-9999，单位为 ms。

机械臂上各个舵机的 id 如下图所示：

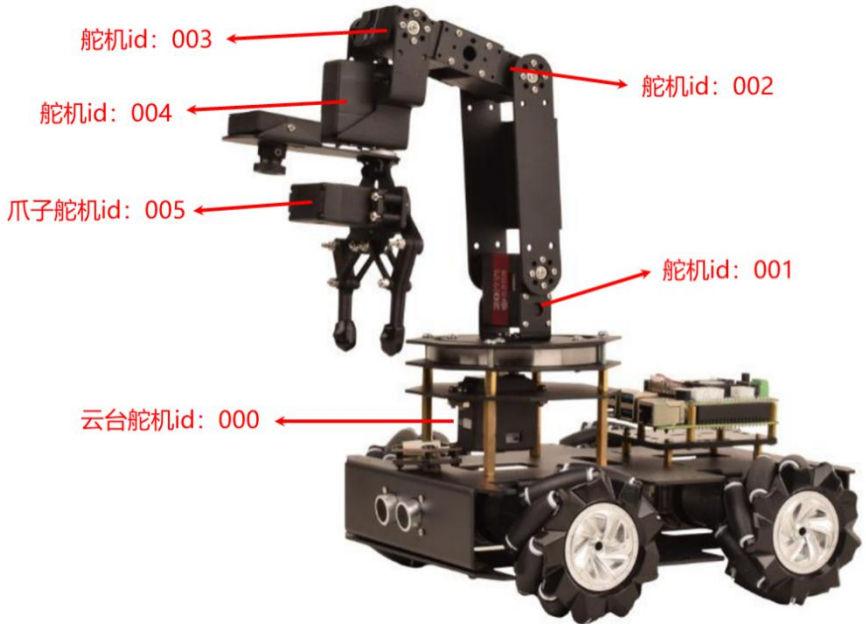

图 0 舵机 id

由于舵机安装的方向不一样，舵机旋转的正方向也不一样，而且初始位置也不尽相同，微调起来较为复杂。本次实验同学们所补充代码的部分不涉及舵机 pwm 值与实际角度之间的转换，以上内容作为了解。实际的调试在后文会有提及，将在获取的物块坐标数据中进行调整。

机械臂正逆运动学简介：

机器人运动学包括正向运动学和逆向运动学，正运动学分析是已知每个关节的角度，解算出末端执行器的位姿。而逆运动学研究的问题是，要求控制末端执行器到达某一位姿时，各关节应处于什么角度。一般正向运动学的解是唯一和容易获得的，而逆向运动学往往有多个解而且分析更为复杂。

2. 逆运动学分析

通常，机械臂的三维运动比较复杂，这里简化了模型便于理解，先去掉下方云台的旋转关节，这样就可以在二维的平面上进行运动学分析了。其中θ_1、θ_2、θ_3是各个关节的角度，对应机械臂上电机的旋转角度，是未知量。$P(x,y)$是末端执行器的位置，x和y是在 OXY 平面的坐标，是末端执行器与水平面的夹角。我们在这里使用几何法进行分析：

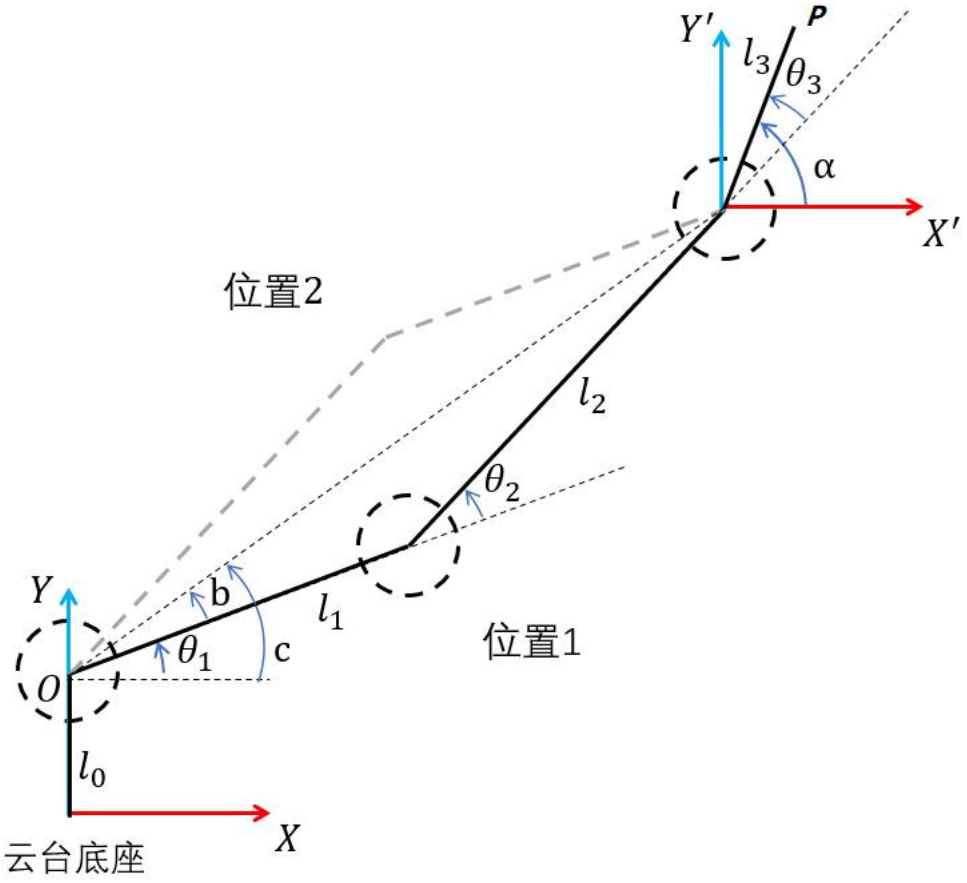

图 1. 几何法机械臂分析图

根据上图，我们可以列出下面方程：

$$x = l_0cos(\theta_1) + l_1cos(\theta_1 + \theta_2) + l_2cos(\theta_1 + \theta_2 + \theta_3) \quad (1)$$

$$y = l_0sin(\theta_1) + l_1sin(\theta_1 + \theta_2) + l_2sin(\theta_1 + \theta_2 + \theta_3) \quad (2)$$

$$\alpha = \theta_1 + \theta_2 + \theta_3 \quad (3)$$

下面对上述方程组进行简化，把公式（3）代入（2）和（1）得：

$$x = l_0cos(\theta_1) + l_1cos(\theta_1 + \theta_2) + l_2cos(\alpha) \quad (4)$$

$$y = l_0sin(\theta_1) + l_1sin(\theta_1 + \theta_2) + l_2sin(\alpha) \quad (5)$$

为了方便计算，令：

$$m = l_2cos(\alpha) - x \quad (6)$$

$$n = l_2sin(\alpha) - y \quad (7)$$

化简公式（4）和（5）得：

$$l_1^2 = (l_0 cos(\theta_1) + m)^2 + (l_0 sin(\theta_1) + n)^2 \tag{8}$$

展开可得：

$$l_1^2 - l_0^2 - m^2 - n^2 = 2ml_0 cos(\theta_1) + 2nl_0 sin(\theta_1) \tag{9}$$

通过计算得：

$$sin(\theta_1 + \varphi) = \frac{l_1^2 - l_0^2 - m^2 - n^2}{2\sqrt{ml_0^2 + nl_0^2}} \tag{10}$$

其中：

$$\varphi = atan\left(\frac{m}{n}\right) \tag{11}$$

解得：

$$\theta_1 = asin\left(\frac{l_1^2 - l_0^2 - m^2 - n^2}{2\sqrt{ml_0^2 + nl_0^2}}\right) - \varphi \tag{12}$$

3. 源码分析及实验操作

补充 kinematics.py 实现逆运动学求解，根据逆运动学分析代码可以解出舵机的角度，需要补充的代码在 kinematics_analysis 函数中。

变量定义

l_0	底盘到第二个舵机中心轴的距离
l_1	第二个舵机到第三个舵机的距离
l_2	第三个舵机到第四个舵机的距离
l_3	第四个舵机到机械臂(闭合后)最高点的距离

代码实现

Ⅰ. 已知 x y 求解坐标的斜边 q

$$q = \sqrt{x^2 + y^2}$$

Ⅱ. 在机械臂投影到斜边平面后，计算$y_1 + y_2$、$z_1 + z_2$

$$y = y_1 + y_2 = q - l_3 \times cos(\alpha \times pi/180)$$

$$z = z_1 + z_2 = z - l_0 - l_3 \times sin(\alpha \times pi/180)$$

注意，传入 kinematics_analysis 函数的α是角度，这里需要转为弧度。计算出两个值后需要判断是否超出范围，函数中的判断代码为：

```
1. if(z < -l0) :
2.     return 1;
3. if(sqrt(y*y + z*z) > (l1+l2)) :
4.     return 2
```

Ⅲ. 计算l_2末端与 0 点连线的水平夹角和l_2末端与 0 点连线的l_1夹角

$$c = \text{acos}\,(y/\sqrt{y^2 + z^2})$$

$$b = (y^2 + z^2 + {l_1}^2 - {l_2}^2)/(2 \times l_1 \times \sqrt{y^2 + z^2})$$

Ⅳ. 根据第Ⅲ步计算得到的两夹角以及$z_1 + z_2$值的正负，计算 1 号舵机的弧度再转化为角度，并判断是否越界

函数中的计算代码为：

```
if (z < 0) :
    zf_flag = -1
else :
    zf_flag = 1
theta1 = c * zf_flag + acos(b);      #计算 1 号舵机的弧度
theta1 = theta1 * 180.0 / pi;            #转化为角度
if(theta1 > 180.0 or theta1 < 0.0) :
    return 4
```

Ⅴ. 计算 2 号舵机的弧度再转化为角度，并判断是否越界，计算公式为：

$$\theta 2 = acos\left(\frac{{l_1}^2 + {l_2}^2 - y^2 - z^2}{2 \times l_1 \times l_2}\right)$$

$$\theta 2 = 180 - \theta 2 \times 180/\pi$$

Ⅵ. 计算 3 号舵机的角度，并判断是否越界，计算公式为：

$$\theta 3 = \alpha - \theta 1 + \theta 2$$

4. 实验小结

本实验实现了机械臂的逆运动学及相应的关节电机控制。为后续结合 openCV 图像处理做自动识别夹取物体，奠定了基础。

实验 2：机械臂目标抓取

1.实验简介

1.1 opencv 简介

OpenCV（OpenSourceCaptureVision）是一个免费的计算机视觉库，可通过处理图像和视频来完成各种任务，比如显示摄像头输入的信号以及让机器人识别现实生活中的物体。OpenCV 提供了完整的 Python 接口，在我们提供的镜像系统中已经集成了 Python3.7 和 python-opencv 库文件。

OpenCV 于 1999 年由加里·布拉德斯基在英特尔启动，第一次发布于 2000 年。瓦迪姆·皮萨列夫斯基与加里·布拉德斯基一起管理英特尔的俄罗斯软件 OpenCV 团队。2005 年，OpenCV 在斯坦利上使用，斯坦利是赢得 2005 年 DARPA 大挑战赛的车型。后来，在柳树车库的支持下，该项目继续积极发展，加里·布拉德斯基和瓦迪姆·皮萨列夫斯基领导了该项目。OpenCV 现在支持与计算机视觉和机器学习相关的多种算法，并且正在日益扩展。OpenCV 支持各种编程语言，如 C++、Python、Java 等，可在不同的平台上使用，包括 Windows、Linux、OSX、安卓和 iOS。基于 CUDA 和 OpenCL 的高速 GPU 操作的接口也在积极开发中。OpenCV-Python 是 OpenCV 的 PythonAPI，结合了 OpenCVC++API 和 Python 语言的最佳品质。阅读 stack.py 中使用 OpenCV 进行图像处理的部分，参考注释学习处理图像信息实现目标抓取。

1.2 实验内容

阅读 stack.py 中使用 OpenCV 进行图像处理的部分，参考注释学习处理图像信息实现目标抓取。

2.机械臂码垛

2.1 实验目标

本实验完成机械臂码垛功能的实现。代码运行逻辑为：首先将导入的摄像头库模块实例化，并打开摄像头，获得摄像头实时图像并进行缩放、滤波、边缘检测等处理确定所识别物体的位置信息得到相对坐标，之后判断数据稳定后开启搬运功能。实验代码已给出图像识别进行目标信息获取及定位的部分，需要补充完成搬运功能流程中各电机的控制。

2.2 相关知识学习

1. 机械臂逆运动学控制参考实验 1 内容。
2. 颜色识别工作原理，参考实验提供库文件中相关函数。

2.3 工作原理

2.3.1 夹取部分

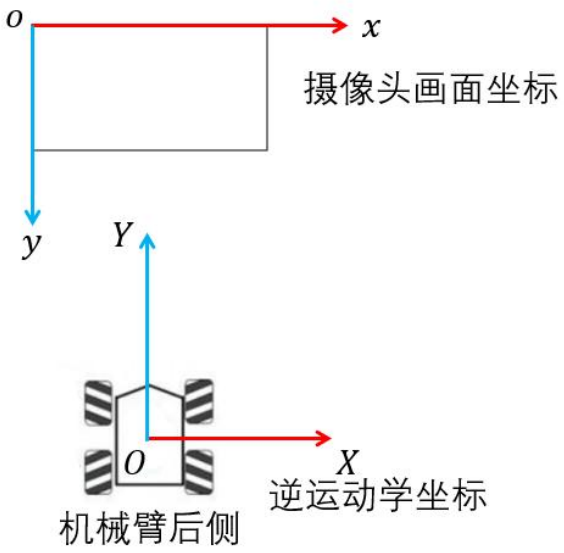

图 2. 坐标关系示意图

若小车编号为 33-37 或是运行时报错“连接被拒绝的”，请在 Camera.py 文件中将以下代码：

相关代码

```
cv2.Videocapture("http://127.0.0.1:8080/?action=stream?dummy=param.mjpg")
```

替换为：

相关代码

```
cv2.Videocapture(0)
```

在运行实验代码后，首先启动摄像头模块，之后机械臂会先运行到预设的初始位置，此时运动学坐标和摄像头画面坐标存在对应变换关系。实验代码中已经给出了摄像头画面坐标到运动学坐标的对应变换，因此在通过颜色识别获取物体在画面上的坐标后，根据公式换算为逆运动学坐标，就可以调用逆运动学函数控制机械臂移动到对应位置。

相关代码

```
kms_x = (c_x-width/2)*30/c_h
kms_y = 160-((c_y-hight/2)*30/c_h)
```

式中，c_x、c_y 为图像中目标识别框的角点坐标，width、hight 为图像的长宽像素数，计算出的 kms_x、kms_y 为目标的运动学坐标。

2.3.2 码垛部分

相关代码

```
#移动到码跺位置
elif clamp_step == 6:
    kinematics_move(-200+count*3,50,20+count*30,2000)
    time.sleep(3)
    clamp_step = 7
```

通过记录 count 变量判断识别到木块的次数，再提升对应高度（Z 轴的值）。需要注意，代码中颜色识别部分只定义了红、蓝、绿三种颜色，因此操作时不要使用黄色木块，避免出现识别不准的问题。

2.4 实验内容

补充 stack.py 中机械臂运动控制的代码，实现码垛操作。

代码实现

```
#张开爪子
if clamp_step == 1:
    Str = "#000P{0:0>4d}T{0:0>4d}!#004P{1:0>4d}T{2:0>4d}!#005P1200T{2:0
>4d}!".format(servo_yuntai,servo_zhuazi, 1000)
    myUart.uart_send_str(Str)
    time.sleep(2)
    clamp_step = 2
#机械臂运行到目标位置
elif clamp_step == 2:
    kinematics_move(kms_x-10,kms_y+10,15,2000)
    #请同学们自行调整机械臂抓取偏差，参考范围为±50
    time.sleep(2)
    clamp_step = 3
#闭合爪子
elif clamp_step == 3:
    #参考第一步张开爪子，控制 005 号电机 pwm 速度 1900，执行 1000ms
    time.sleep(2)
    clamp_step = 4
#抬起机械臂
elif clamp_step == 4:
    #控制 000 号电机 pwm 速度 1500，执行 2500ms
    # 001 号电机 pwm 速度 2100，执行 2000ms
    # 002 号电机 pwm 速度 2300，执行 2000ms
    # 003 号电机 pwm 速度 1000，执行 2000ms
    # 004 号电机 pwm 速度 1500，执行 2000ms
    # 005 号电机 pwm 速度 1900，执行 2000ms
    time.sleep(3)
    clamp_step = 5
#旋转机械臂
elif clamp_step == 5:
    myUart.uart_send_str('#000P{:0>4d}T1000!'.format(int(1500-2000.0 *
-90/ 270.0)))
    time.sleep(1)
    clamp_step = 6
#移动到码跺位置
elif clamp_step == 6:
    kinematics_move(-200+count*3,50,20+count*30,2000)
```

```
    time.sleep(3)
    clamp_step = 7
#张开爪子
elif clamp_step == 7:
    # 005 号电机 pwm 速度 1200，执行 1000ms
    time.sleep(1)
    clamp_step = 8
#抬起机械臂
elif clamp_step == 8:
    # 001 号电机 pwm 速度 2100，执行 1500ms
    # 002 号电机 pwm 速度 2300，执行 1500ms
    # 003 号电机 pwm 速度 1000，执行 1500ms
    # 004 号电机 pwm 速度 1500，执行 1500ms
    time.sleep(2)
    clamp_step = 9
#旋转到中间位置
elif clamp_step == 9:
    # 000 号电机 pwm 速度 1500，执行 1000ms
    time.sleep(1)
    clamp_step = 10
#回到初始位置
elif clamp_step == 10:
    clamp_step = 0
    count += 1
    if count > 3:
        count = 0
    time.sleep(2)
    myUart.uart_send_str('{#000P1500T1000!#001P1500T1000!#002P2000T1000!
#003P0750T1000!#004P1500T1000!#005P1500T1000!}')
    time.sleep(1)
    eEvent.clear() #线程旗标标志为假
    time.sleep(2)
    Running = True #开启图像处理
    time.sleep(2)
```

2.5 具体实验目标：

根据红色注释补充代码中 clamp_step=3、4、7、8、9 中的电机控制代码，根据每台车实际情况调节 clamp_step=2 中的偏差，使机械臂成功完成码垛功能。

具体代码在完成前述实验后，联系任课老师和助教。验收完毕后，请在收回实验小车前删除实验文件。