Final Project

(control a mobile arm for pick and place application using visual servoing technique)

武敬祥、陳若桐、王奕勝、林昱宏

ABSTRACT

在這次的專題中，我們的目標是讓履帶車前進，並控制車上的 4R 手臂，使其末端插入目標物的洞裡。而我們會在車體上加裝視覺感測器，利用視覺辨識的技術先找到目標物(洞)，再利用上課所學的順逆運動學、順逆動力學等等，來控制 4R 手臂作動至理想地方。

硬體	目標功能	方法
手臂	順利作動，將手臂末端的圓柱插入目標物的洞裡	利用順逆運動學、順逆動力學推算4R手臂上每顆馬達所需動力等等參數
視覺感測器	辨識目標物位置，以利手臂作動至該處	利用visual sevoing技術，辨識黑白區域及深度
車體	使車體前進至方便手臂工作的位置	基礎arduino控制車體馬達

利用上述硬體與技術來達到專題的目標要求，並使各硬體間作動順利，車體能前進並讓手臂順利將末端圓柱插入目標物的洞裡。

1.INTRODUCTION

在學習了機器人動力與控制這門課以後，也迎來了我們的期末 project，而這次的期末 project 是要我們利用順向運動學與逆向運動學結合順逆向動力學去對機器手臂作應用。同時結合 matlab 和 solidworks 去模擬出機器人手臂的運動軌跡和各種物理參數，得到我們最終所需制定的機器手臂的長度和重量。

1.1 EXPERIMENTAL SETUP

-MG996R *4

-24vDC MOTOR *2

-H BRIDGE(L298N)

-ARDUINO UNO R3

-PCA9685

-降壓模組 *3

-25 V 鋰電池

-

1.2 Approach

OUTLINE:

arm design

arm control

forward kinematics

inverse kinematics

`

outward dynamics

inward dynamics

coordinate transform

car control

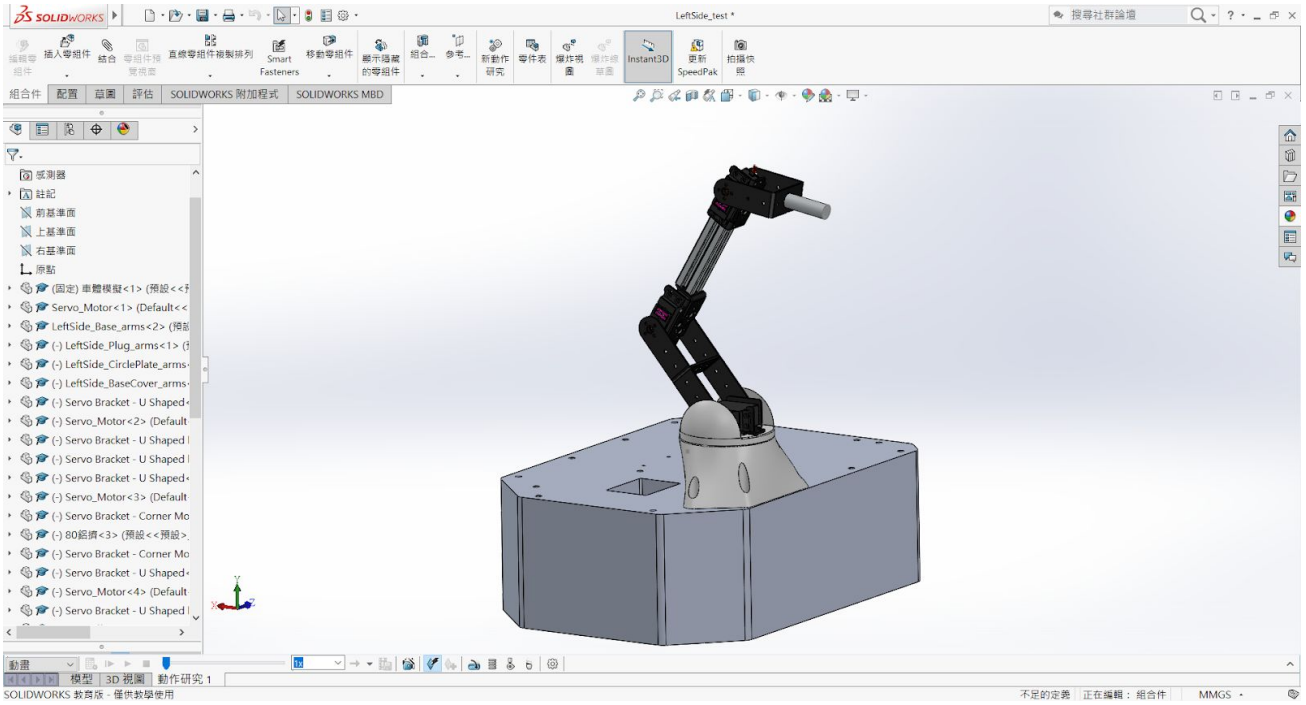
vision algorithm

find circle

flow chart

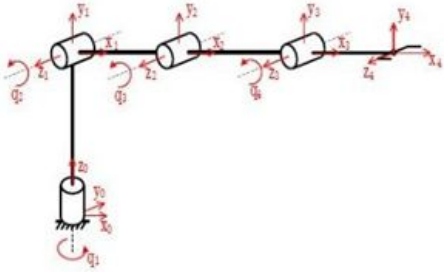
simulation

(1)Arm Design



(2)Arm control

Forward Kinematics



DH表

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	90°	0	23	θ_1
2	0	100.4	0	θ_2
3	0	170	0	θ_3
4	0	95	0	θ_4

Inverse Kinematics

Inverse kinematics solution

已知 X, Y, Z 和 $\phi = \theta_2 + \theta_3 + \theta_4$

$$\theta_1 = \tan^{-1}\left(\frac{Y}{X}\right)$$

計算

$$A = X - l_4 \cos\theta_1 \cos\phi$$
$$B = Y - l_4 \sin\theta_1 \cos\phi$$
$$C = Z - l_1 - l_4 \sin\phi$$

得出

$$\theta_3 = \cos^{-1}\frac{A^2 + B^2 + C^2 - l_2^2 - l_3^2}{2\, l_2\, l_3}$$

再計算

$$a = l_3 \sin\theta_3$$
$$b = l_2 + l_3 \cos\theta_3$$
$$c = Z - l_1 - l_4 \sin\phi$$
$$r = \sqrt{a^2 + b^2}$$

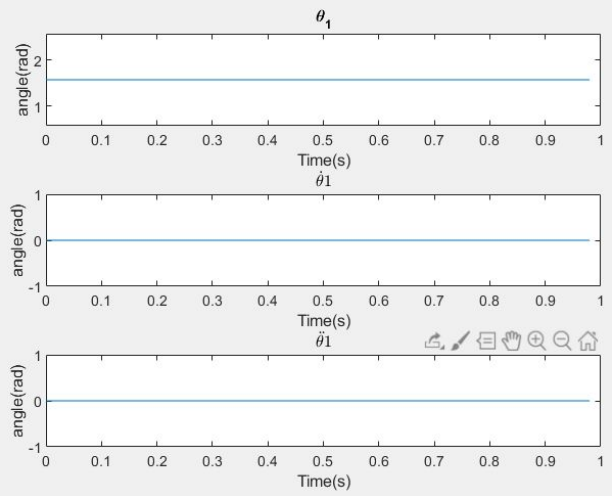
得出

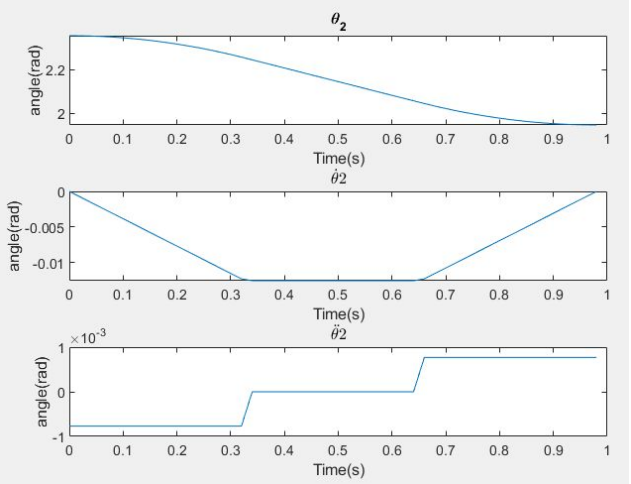
$$\theta_2 = \tan^{-1}\frac{c}{\sqrt{r^2 - c^2}_3} - \tan^{-1}\frac{a}{b}$$

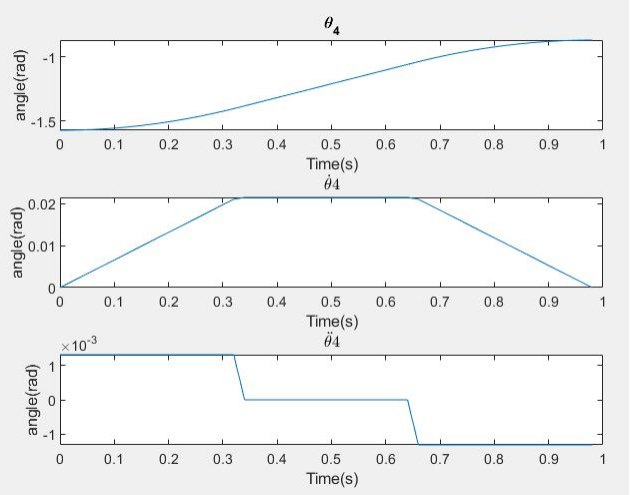
最後可知

$$\theta_4 = \phi - \theta_2 - \theta_3$$

Kinematics simulation







Outward dynamics

$$ {}^{1}\dot{v}_1 = \begin{bmatrix} gs_1 \\ gc_1 \\ 0 \end{bmatrix} \quad {}^{1}\dot{v}_{c_1} = \begin{bmatrix} -L_1\dot{\theta}_1^2 + gs_1 \\ L_1\ddot{\theta}_1 + gc_1 \\ 0 \end{bmatrix} \quad {}^{1}F_1 = m_1 * \begin{bmatrix} -L_1\dot{\theta}_1^2 + gs_1 \\ L_1\ddot{\theta}_1 + gc_1 \\ 0 \end{bmatrix} $$

$$ {}^{2}\dot{v}_2 = \begin{bmatrix} gs_{12} \\ gc_{12} \\ 0 \end{bmatrix} \quad {}^{1}\dot{v}_{c_1} = \begin{bmatrix} -L_2\dot{\theta}_2^2 + gs_{12} \\ L_2\ddot{\theta}_2 + gc_{12} \\ 0 \end{bmatrix} \quad {}^{2}F_2 = m_2 * \begin{bmatrix} -L_2\dot{\theta}_2^2 + gs_{12} \\ L_2\ddot{\theta}_2 + gc_{12} \\ 0 \end{bmatrix} $$

$$ {}^{3}\dot{v}_3 = \begin{bmatrix} gs_{123} \\ gc_{123} \\ 0 \end{bmatrix} \quad {}^{1}\dot{v}_{c_1} = \begin{bmatrix} -L_3\dot{\theta}_3^2 + gs_{123} \\ L_3\ddot{\theta}_3 + gc_{123} \\ 0 \end{bmatrix} \quad {}^{3}F_3 = m_3 * \begin{bmatrix} -L_3\dot{\theta}_3^2 + gs_{123} \\ L_3\ddot{\theta}_3 + gc_{123} \\ 0 \end{bmatrix} $$

$$ {}^{1}\omega_1 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix} \quad {}^{1}\dot{\omega}_1 = \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta}_1 \end{bmatrix} $$

$$ {}^{2}\omega_2 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_2 \end{bmatrix} \quad {}^{2}\dot{\omega}_2 = \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta}_2 \end{bmatrix} $$

$$ {}^{3}\omega_3 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_3 \end{bmatrix} \quad {}^{3}\dot{\omega}_3 = \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta}_3 \end{bmatrix} $$

Inward dynamics

$$^{1}n_1 =$$

$$\begin{bmatrix} 0 \\ 0 \\ \frac{L_3 m_3 g c_{123}}{2} + L_2 m_2 g c_{12} + m_3 g L_2 (s_{123} s_3 + c_{123} c_3) + m_1 L_1^2 \ddot{\theta}_1 + m_1 L_1 g c_1 + L_1 m_3 g s_2 (c_3 s_{123} - s_3 c_{123}) + L_1 m_2 g s_{12} s_2 + L_1 m_3 g c_2 (s_{123} s_3 + c_{123} c_3) + L_1 m_2 g c_{12} c_2 \end{bmatrix}$$

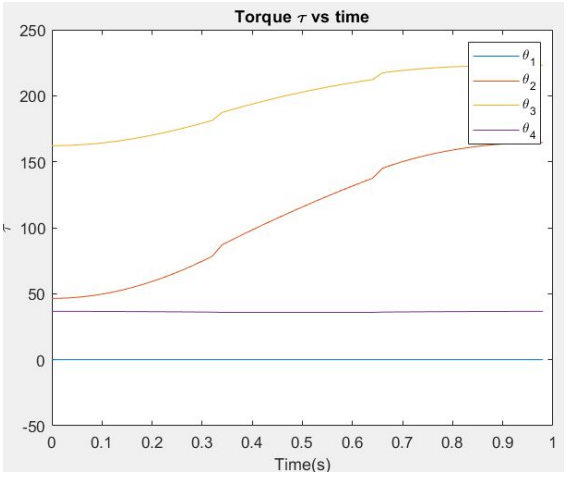
$$^{2}n_2 =$$

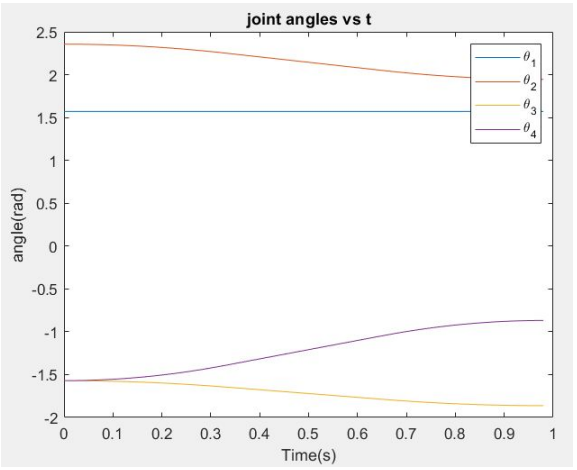
$$\begin{bmatrix} 0 \\ 0 \\ \frac{L_3 m_3 g c_{123}}{2} + L_2^2 m_2 \ddot{\theta}_2 + L_2 m_2 g c_{12} + m_3 g L_2 (s_{123} s_3 + c_{123} c_3) \end{bmatrix}$$

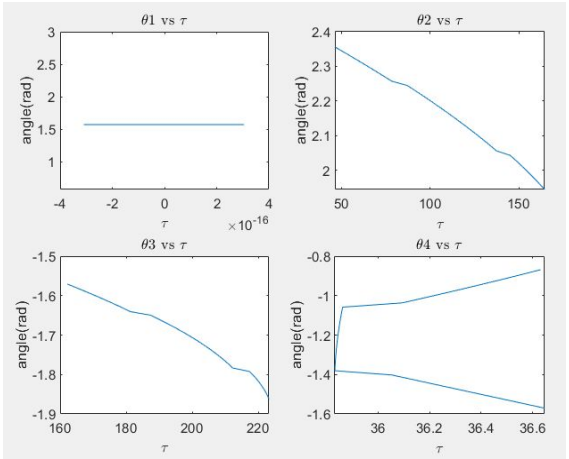
$$^{3}n_3 =$$

$$\begin{bmatrix} 0 \\ 0 \\ \frac{m_3 L_3^2 \ddot{\theta}_3}{2} + \frac{L_3 m_3 g c_{123}}{2} \end{bmatrix}$$

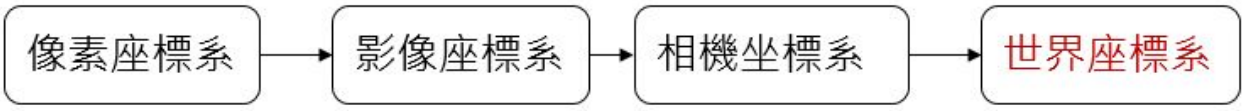
Dynamics Simulation

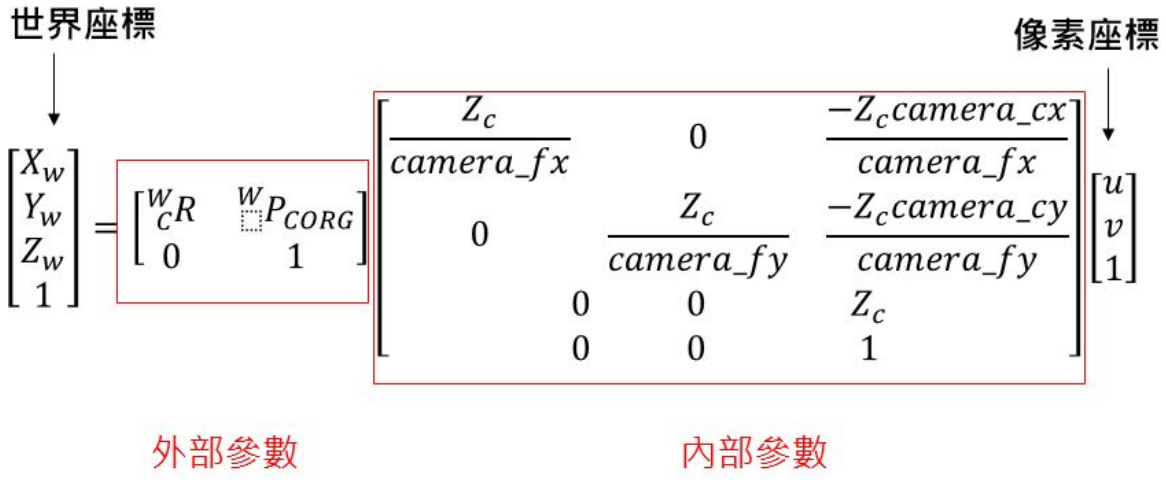






(3)Coordinate Transform





$$\begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = \begin{bmatrix} {}^{W}_{C}R & {}^{W}P_{CORG} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{Z_c}{camera\_fx} & 0 & \frac{-Z_c camera\_cx}{camera\_fx} \\ 0 & \frac{Z_c}{camera\_fy} & \frac{-Z_c camera\_cy}{camera\_fy} \\ 0 & 0 & Z_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

內部參數(intrinsic parameter):

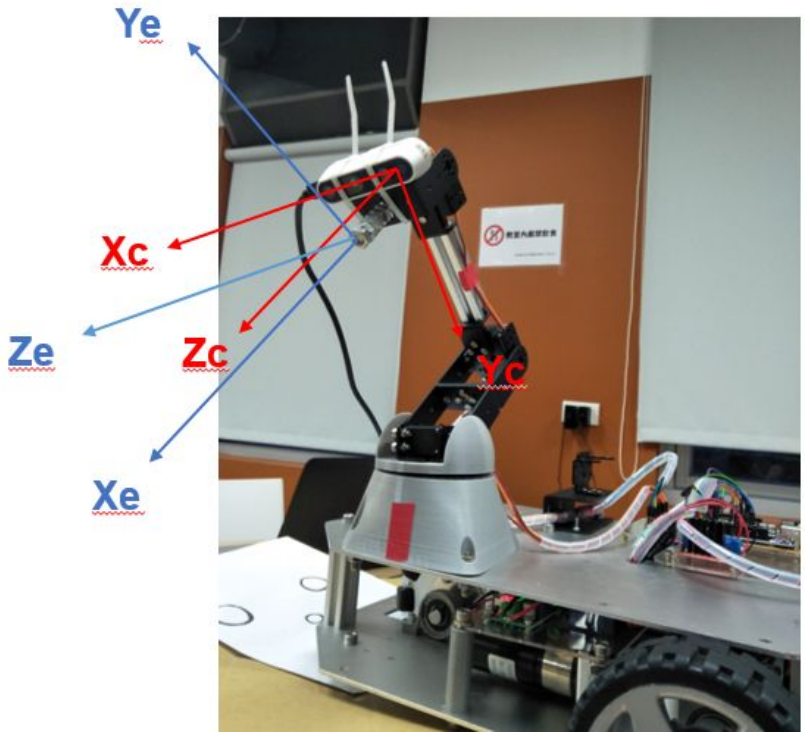
Realsense d435i

```
camera_cx = 321.798
camera_cy = 239.607
camera_fx = 615.899
camera_fy = 616.468
```

外部參數(extrinsic parameter):

外部參數=手臂順向Transform matrix * Rotation matrix

Rotation matrix:使相機坐標系與手臂end_effeter 的方向一致



(Xc,Yc,Zc):相機坐標系

(Xe,Ye,Ze):end_effecter坐標系

Rotation matrix = Rx(pi)*Ry(-pi/2)

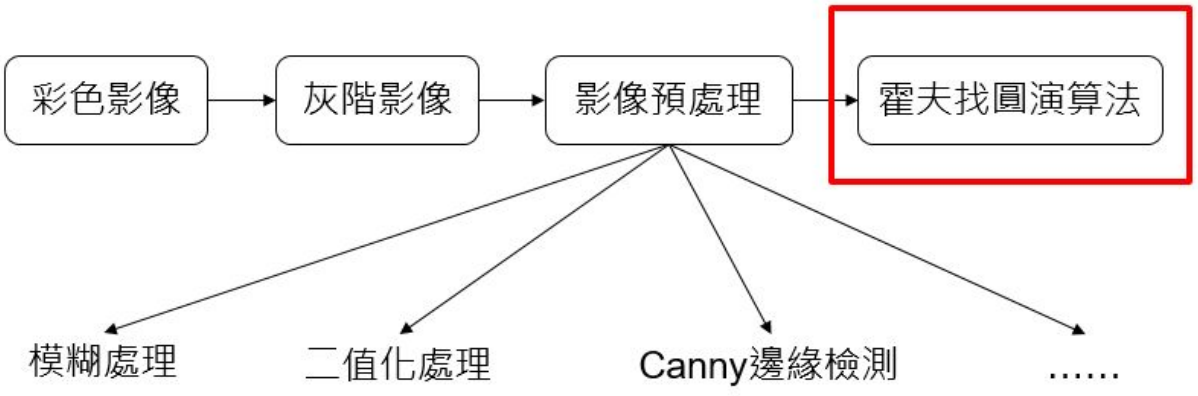
(4)Car Control

pseudo code:

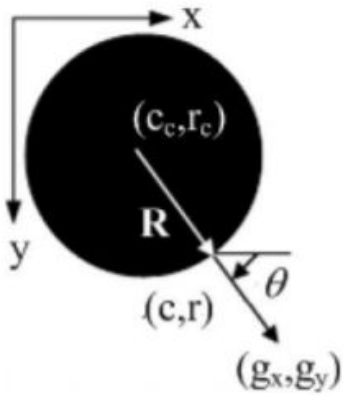
```
目標物在車體右邊
k = 0.30        #gain
if 找到三個圓:
    將圓從大排到小
    if (相機中心-圓中心) < 30mm:
        velocity = 0
    else :
        velocity = (相機中心-圓中心) * k

    if velocity > 0:
        go backward
    else:
        go forward
else:
    跑很快
```

```
k = 0.5
if np.size(coordinate,0) == 3:
    coordinate = rad_sort(coordinate)
    if -coordinate[0][0] < 10:
        vel = 0
    else :
        vel = -coordinate[0][0] * k

    if vel < 0:
        vel = -vel
        velocity = str(vel) + "," + "1\n"
    else:
        velocity = str(vel) + "," + "0\n"

    ser.write(velocity.encode())
else:
    ser.write(("50,0\n").encode())
```

(5)Vision Algorithm



Hough for circle algorithm:



$$(c-c_c)^2+(r-r_c)^2=R^2$$

$$c_c=c-R\cos\theta$$

$$r_c=r-R\sin\theta$$

$$\cos\theta=\frac{g_x}{\sqrt{g_x^2+g_y^2}}$$

$$\sin\theta=\frac{g_y}{\sqrt{g_x^2+g_y^2}}$$

gx : gradient in x axis

gy : gradient in y axis

Implementation

(1)初始化一個三維累加器accu=(X_C , Y_C , R)，分別代表圓心X座標，圓心Y 座標及圓半徑R

(2)利用 sobel filter 取出圓形的gx ,gy

(3)給定圓半徑R的最大Rmax及最小值Rmin

(4)利用下式:

$$X_C = x - Rcos\theta$$

$$Y_C = y - Rsin\theta$$

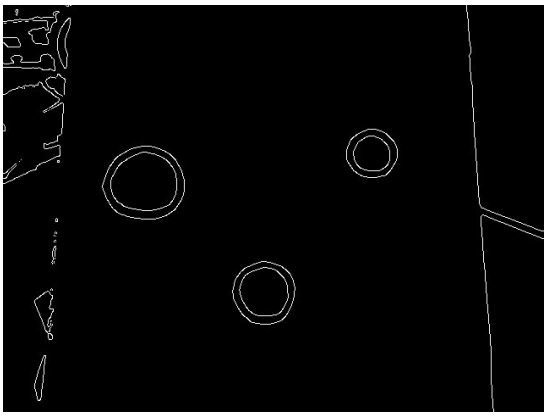
對每一個R及圓心pixel座標(x,y)求出圓心(Xc,Yc)

(5)將累加器accu(X_C , Y_C , R)+1

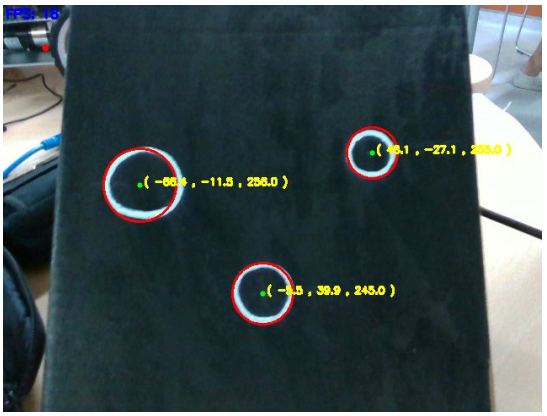
(6)得到accu最大的值及求得圓的圓心及半徑

實際拍攝:

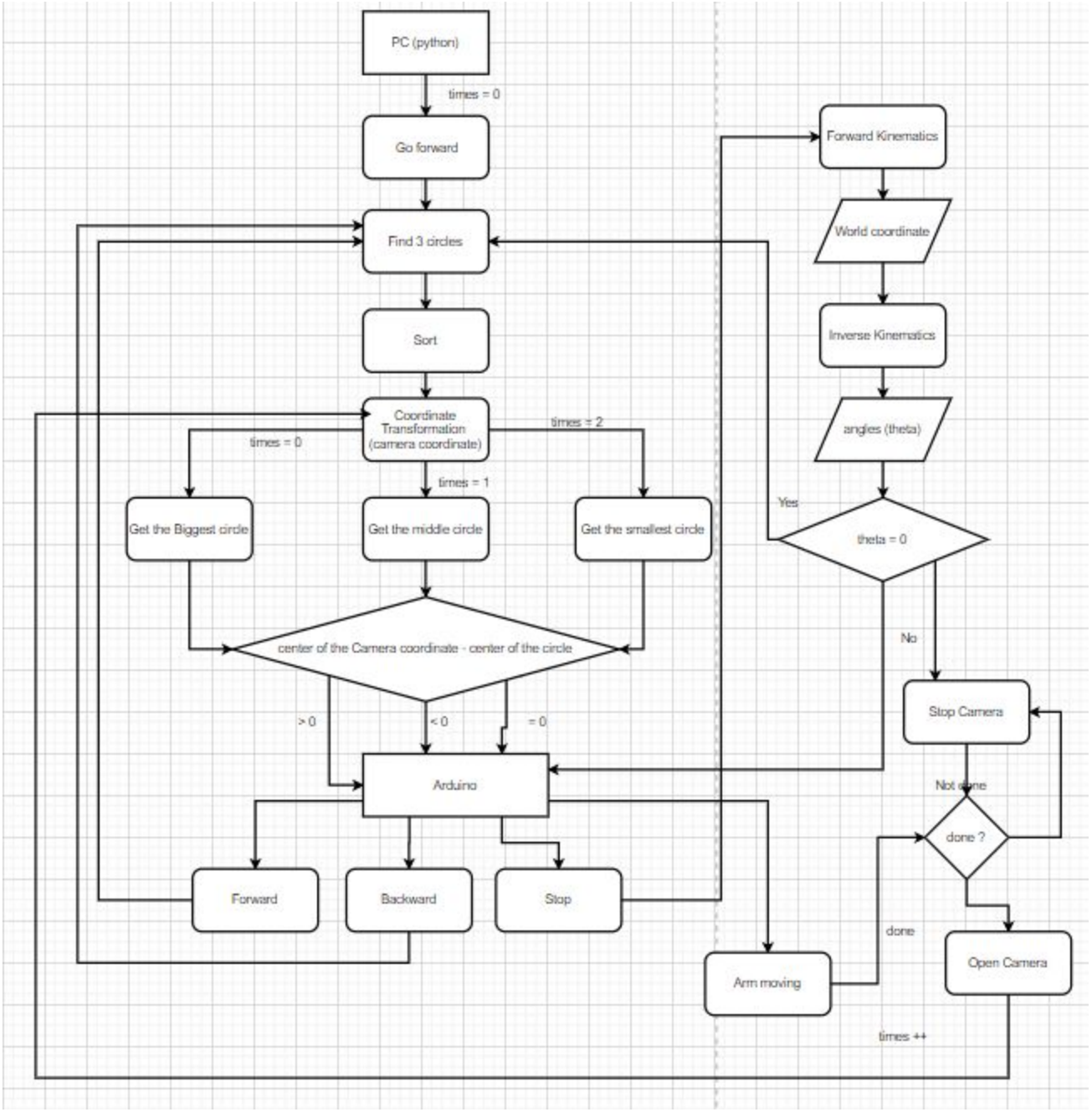
edge image



rgb image

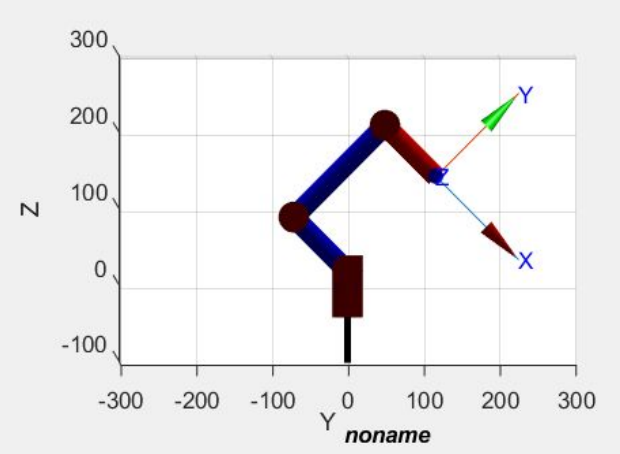


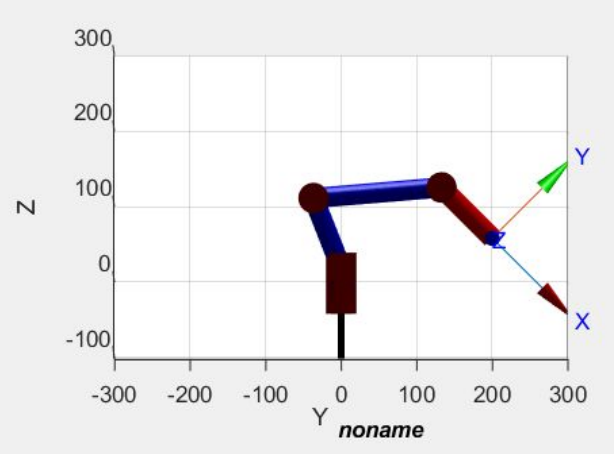
(5)Flow Chart


PC (python)
times = 0
Go forward
Find 3 circles
Sort
Coordinate Transformation (camera coordinate)
times = 0
times = 1
times = 2
Get the Biggest circle
Get the middle circle
Get the smallest circle
center of the Camera coordinate - center of the circle
> 0
< 0
= 0
Arduino
Forward
Backward
Stop
Forward Kinematics
World coordinate
Inverse Kinematics
angles (theta)
Yes
theta = 0
No
Stop Camera
Not done
done ?
done
Open Camera
Arm moving
times ++

1.3 Simulation

initial position **final position**





- solidwork simulation:

https://drive.google.com/file/d/1zjSCiypEF NablboHPWApGBV0MS0Uq0G0/view

-Trajectory simulation video:

https://drive.google.com/file/d/1MfFIZxiK4l 62xk2KEm2N1igUXctGE2Kb/view?usp=s haring