國立中興大學電機資訊學院學士班
113 學年度專題報告

整合三維視覺與路徑規劃之機械手臂夾取系統

Robotic Arm Grasping System Integrating 3D Vision and Path Planning

指導教授:莊家峰 特聘教授兼系主任 Chia-Feng Juang

專題生:林佩萱 Pei-Hsuan Lin、黃俶靖 Chu-Jing Huang

中華民國一百一十四年六月

國立中興大學電機資訊學院學士班

專題報告

整合三維視覺與路徑規劃之機械手臂夾取系統

Robotic Arm Grasping System Integrating 3D Vision and Path Planning

專題題目說明與價值:

結合 YOLOv11 深度學習與 3D 視覺辨識的智能夾取系統，透過 Intel Realsense Depth Camera D435i 深度相機與 xArm-ESP32 機械手臂，辨識隨機擺放之不同形狀積木並準確放置至對應凹槽。系統整合影像分割、座標轉換、正逆向運動學與夾爪角度控制，實現隨機擺放物體下的穩定抓取，展示 AI 視覺與機械控制的深度整合。

自評貢獻(與過去前輩成果不同之處):

	過去前輩	本專題
辨識目標	顏色、數字、圖形	即時位置、形狀
控制器	ESP32	ESP32
機械手臂	幻爾科技 xArm-ESP32	幻爾科技 xArm-ESP32
相機鏡頭	RAPOO C200	Intel Realsense Depth Camera D435i
開啟鏡頭程式	OpenCV +Python	pyrealsense2 +Python
通訊協議	UART、SOCKET、MQTT	TCP

專題組員:

姓名	E-mail	負責項目說明	專題貢獻度
林佩萱	linl90050@gmail.com	手臂路徑規劃及動作組設計、座標重建、系統整合、正逆向運動學控制、通訊協議、系統測試	50%
黃俶靖	chujinghaung@gmail.com	訓練集資料收集、影像辨識模型訓練及調適、正逆向運動學控制、專題報告撰寫、系統測試	50%

指導教授簡述及簡評:

指導教授簽名:

合作者簽名:

中華民國 114 年 5 月 31 日

目錄

一、 摘要

本專題旨在透過機械手臂結合深度學習三維影像辨識技術，實現夾取不同形狀積木並將其移動到相應目標凹槽位置進行放置。首先，我們透過裝置在機械臂上的 Intel Realsense Depth Camera D435i 進行三維影像捕捉，並將其傳至電腦以 YOLOv11 語意分割模型進行積木與凹槽的形狀辨識，做深度學習運算與影像辨識處理，根據運算及辨識結果，ESP32 控制器指揮機械手臂執行夾取，被夾取物體為五種形狀的積木（方形、圓形、花形、星形 和三角形），機械手臂準確地將積木夾起後，將其移動至目標凹槽上方並放入凹槽中，目標凹槽亦具有五種與之相對應的形狀（方形、圓形、花 形、星形和三角形）。

在運動控制方面，我們採用了正向與逆向運動學技術進行路徑規劃，確保機械手臂能以高精度的姿態進行夾取、移動與放置操作。以此系統設計可實現即使積木與凹槽位置隨機改變，機械手臂依然能依據影像準確完成目標物的夾取，並將其放至相應凹槽的任務。

二、 研究動機

隨著自動化技術的發展，製造業與物流產業對於機械手臂的要求不僅限於基本的操作功能，還需要擁有更高的精確性、靈活性和適應能力，尤其是在處理形狀多樣、位置變化不定的物體時。

機械手臂在進行夾取與放置時，必須能夠準確的辨識物體形狀、大小以及位置。傳統的物體辨識方法通常依賴於固定的模型或傳感器，對環境變化的敏感度較低，無法應對物體隨機擺放的情況，從而限制了自動化系統的靈活性和準確度，為了解決這些問題，本計畫旨在利用深度學習技術，特別是基於深度相機進行三維影像捕捉的方式，提升機械手臂操作的準確性與適應性。

在物體識別方面，深度學習技術已被廣泛應用並取得了顯著的進展。本計畫選擇使用YOLOv11語意分割[1]模型進行被夾取物體與凹槽的識別。儘管該模型在影像處理方面為較佳的選擇，但面對形狀多樣、邊界複雜的物 體與凹槽時，如何提升識別準確性仍是一個挑戰。不同形狀與大小的物體 會導致模型識別的難度增加，尤其是在物體與凹槽邊界較為模糊或複雜時， 容易造成識別錯誤。因此，為了提升識別的精度，我們將進行多項優化措施。首先，需要對模型參數進行精細調整，

並利用數據擴增技術[2]生成更多樣化的訓練數據集，使得模型能夠學會識別各種變化中的物體形狀和尺寸。這些方法能夠有效應對不同物體形狀與凹槽之間的差異，並提高系統對背景干擾、物體遮擋等問題的魯棒性。

三、 研究目的

在深度學習模型識別精度提高的狀況下，物體的隨機擺放位置和環境變化依然對機械手臂的夾取精度構成挑戰。在真實應用中，物體的擺放位置通常是不確定的，這要求系統能夠根據實時的影像數據進行動態調整。為此，本計畫採用正向與逆向運動學技術來精確計算機械手臂的運動軌跡，並確保即便物體位置發生變動，機械手臂仍能精確地完成抓取與放置操作。透過深度學習模型，機械手臂能夠快速獲取物體的三維影像，並根據影像數據進行實時定位與動作調整。這樣一來，無論物體如何隨機放置，機械手臂都能在捕捉到的三維影像基礎上精確計算出物體的相對位置，並迅速調整運動軌跡，保證操作精度。

此外，在夾取過程中，物體的偏移或滑動也是常見的問題之一。為了應對這些挑戰，系統必須集成多種感測技術，用以提供即時反饋，並做出調整機械手臂的操作策略。例如在夾取物體時，若發現物體位置偏移，系統需根據感測器的回饋信息進行調整，從而確保夾取過程不會因物體的微小變動而失敗。這種動態調整機制不僅能夠提升系統的靈活性，還可以有效提高操作的成功率。

本計畫希望通過深度學習與運動學技術[3]的結合，解決物體識別與隨機位置變化的問題，顯著提升機械手臂在複雜環境中的操作精度。YOLOv11語意分割模型的優化，使得機械手臂能夠在多樣化形狀的物體與凹槽中精確識別，並利用深度學習進行動態調整，應對物體隨機放置所帶來的挑戰。而進一步結合正向與逆向運動學技術，則使得機械手臂能夠根據實時影像 進行路徑規劃，保證夾取與放置過程中的高精度操作。

四、 文獻回顧

本計畫參考了前幾屆學長姐的研究成果[4]，以深入瞭解影像辨識結合機械手臂為研究方向。實驗室的專題生以往使用的機械手臂為舊型的3D列印自製手臂，這是由之前的學長所留下供學弟妹使用的，但上一屆的學長姐發現當初機械手臂的設計並未考量到後續的維護，導致馬達燒壞時更換和拆裝變得困難，需要消耗大量的時間進行修繕，所以從上一屆開始，改為購買市面上的自學型機械手臂進行

二次開發，這雖然保證了基本硬體結構的穩定性，但也意味著資料量將會有不足的問題，需要重新摸索相關的硬體和軟體開發。

上一屆的學長姐使用RAP00 C200網路視訊攝影機輸入影像，並運用YOLOv5[5]進行機器學習，實現夾取指定顏色的方塊並將其移動至指定目標位置上方進行置。在此基礎上，我們的研究題目將聚焦於結合YOLOv11影像辨識技術與xARM-ESP32機械手臂控制，探索更精確的三維影像處理與高效的物體辨識方法，以提升不同形狀物件取放任務的準確性和效率。以下為計畫研究當中使用到的研究方法。

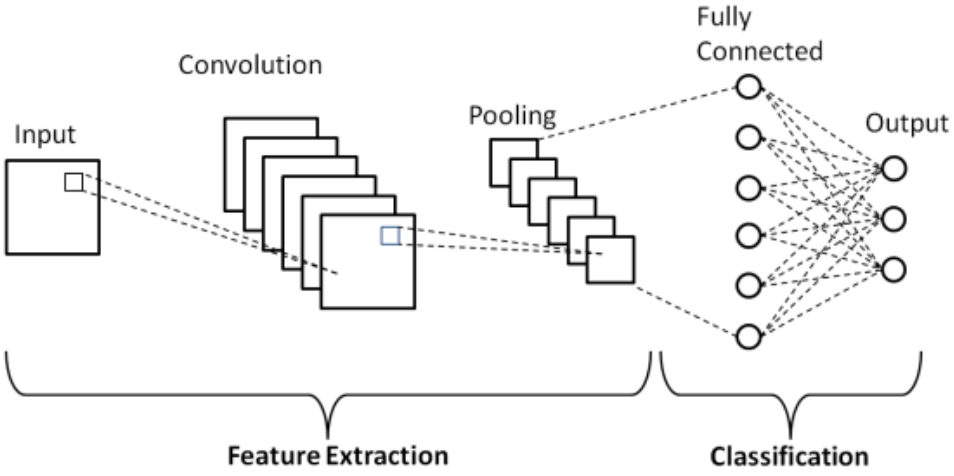
1. 機器學習：

機器學習[6]是人工智慧領域的分支，人工智慧是指由人製造出來的機器所表現出來的智慧，通常是指電腦模擬人類思維過程而產生人類行為的能力，主要在分析資料以制定決策和預測。而機器學習旨在使計算機系統能夠通過數據學習改進其性能，無明確的程式碼指令。機器學習包括多種學習模式，並結合不同的演算法，根據資料特性與目標需求，可選擇監督式、非監督式、半監督式或強化式學 習等四種模式，每種模式可應用一種或多種演算法技術，具體取決於資料集的類型和期望的結果。在本計畫中，我們需採用監督式學 習模式，透過YOLOv11語意分割模型對積木與凹槽的形狀進行分類與檢測，根據辨識結果實現積木的自動夾取與放置。此模型將結合機器學習的核心能力，包括模式識別與結果預測，透過高效分類及邊界框檢測，支持實現目標物準確定位。機器學習技術的應用，使系統能夠快速處理來自深度相機的數據，確保機械手臂能即時完成 對積木的動作規劃與執行。

2. 深度學習：

深度學習[7]是機器學習的子集合，專注於利用具有多層結構的人工神經網絡來學習數據的高層次表示。這種機器學習之所以稱之為深度，是因為包含許多層神經網路，以及大量複雜且離散的數據。在本計畫中，深度學習技術將被用於精確辨識積木與凹槽的形狀特徵， 從而提高機械手臂執行任務的準確性。我們採用了YOLOv11模型進 行目標檢測與語意分割。YOLO 模型是一種基於卷積神經網路(Convolutional Neural Network ,CNN)的高效目標檢測模型，CNN[8]由卷積層（Convolution Layer）、池化層（Pooling Layer）、全連接層（Fully Connected Layer）所構成，能提取圖像局部與全局特徵並進行模式分類。CNN架構圖如圖一表示。在計畫應用中，深度學習的核心優勢在於能通過大量訓練數據進一步提升系統的辨識能力，即 使在複雜背景下亦能穩定運作，為機械手臂的動作規劃提供高可靠的數據支持。此外，YOLOv11特有的增強模型結構（如C3k2塊、 SPPF模塊）有效提升了特徵提取的精度和效率，並結合GPU訓

練大幅加快運算速度，確保系統在即時應用場景中的高效性。

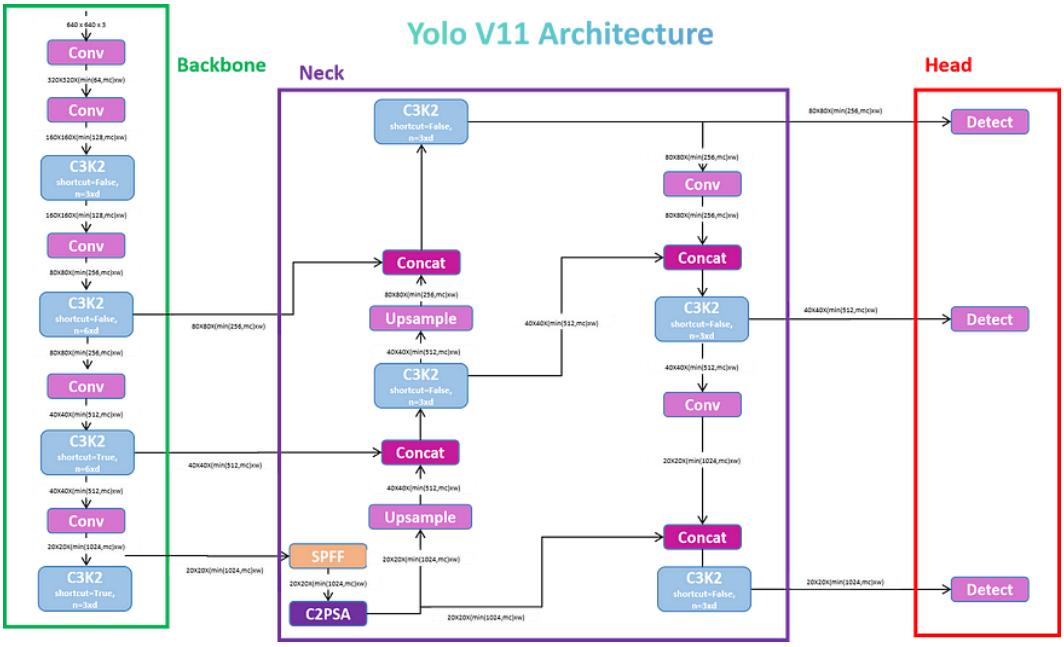


圖一、CNN架構圖

3. Yolov11:

YOLO（You Only Look Once）是一種高效的目標檢測模型，通過單次卷積神經網路運算實現物件的邊界框座標與分類概率預測，擁有高速度與高準確度的特性。與傳統的目標檢測系統不同，YOLO不依 賴多階段的分類器操作，也不需要繁瑣的修正和預測過程，相反， 透過單次的CNN運算（one-stage detection），直接預測圖片中物件的邊界框座標與分類機率，實現物件識別與定位，這種設計大幅提高了檢測的速度和效率。

YOLOv11[9]引入了C3k2塊（內核大小為2的跨階段部分）、SPPF組件（空間金字塔池-快速）和C2PSA組件（具有並行空間注意力的卷積塊），這些組件有助於透過多種方式提高模型效能，例如增強特徵提取。本計畫中，YOLOv11模型被應用於積木與凹槽的形狀辨識，透過其優化後的實例分割功能，準確分割不同形狀的積木並匹配至 相應的凹槽位置。此外，YOLOv11的GPU優化與延遲減少技術，使模型能在高效能運算環境中運作，大幅縮短辨識時間。圖像處理方面，YOLOv11附帶了邊界模型、實例分割（-seg）、姿態估計（-pose）、定向邊界框（-obb）和分類（-cls）。YOLOv11在不同模型 尺寸（從奈米到超大）上的多功能性，滿足從邊緣設備到高效能運 算環境的各種應用需求，其架構圖如圖二表示。與YOLOv10相比， YOLOv11通過減少模型參數實現速度的進一步提升，同時保持高辨 識精度，支持在各種應用場景中實現目標物的即時檢測與操作。



圖二、yolov11架構圖

4. 視覺引導機械手臂：

在機械手臂的智能夾取研究中，視覺引導技術的應用能有效提升機械手臂的適應性與抓取精度。在Research and Implementation of Robotic Arm Positioning and Grasping Based on Visual Guidance[10]中，學者提出了一種基於ROS平台的視覺引導機械手臂定位與抓取系統，結合Hikrobot 2D視 覺傳感器與Siasun GCR5機械手臂，透過相機標定、手眼標定及深度學習方法實現目標物體的識別與定位。該研究採用YOLOv3進行物件識別，並透過MoveIt與RRT-connect演算法規劃機械手臂的運動軌跡。

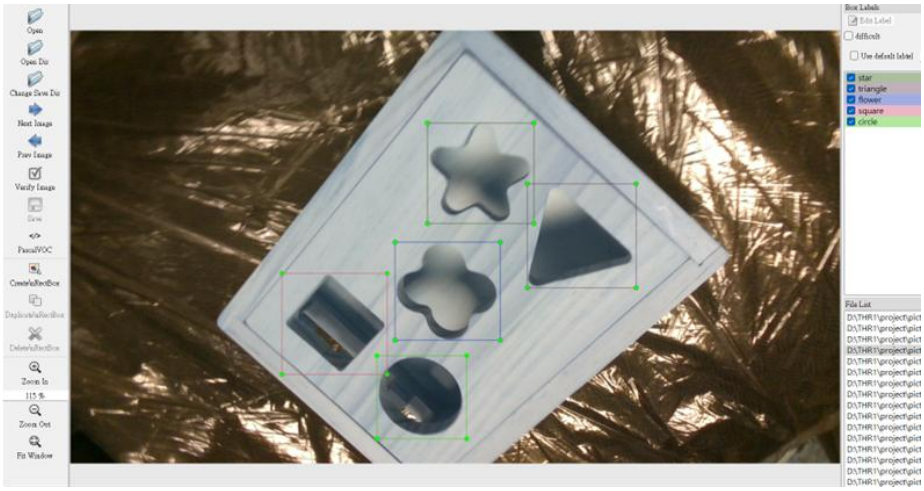
文章[10]主要採用2D視覺系統進行抓取，雖然可有效提升機械手臂在動態環境下的適應性，但對於三維物體的深度資訊掌握較為有限。本計畫則需進一步利用三維視覺辨識技術，結合深度攝影機，提升對物體形狀與空間位置的理解，使機械手臂能夠在更複雜的場景中執行精確抓取。

五、 研究方法與步驟

本計畫使用xArm-ESP32 機械手臂套件與Intel Realsense Depth Camera D435i 深度攝影機實現影像接收與夾取放置任務。以下為製作說明:

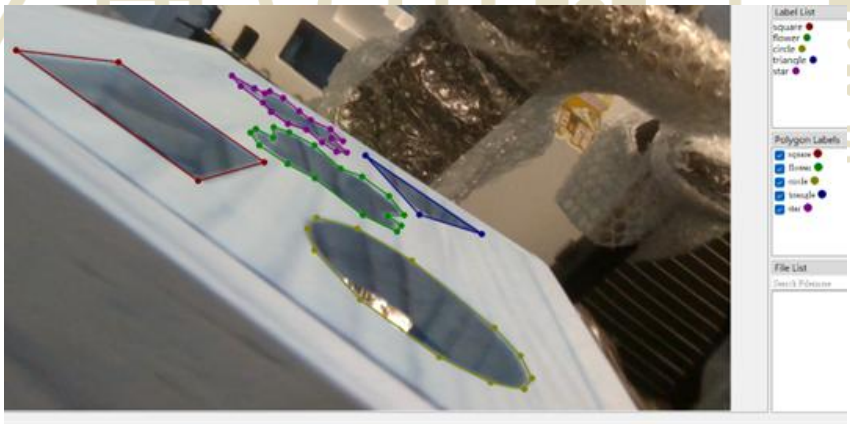
1. 物體分割：

- 相機拍攝各角度、姿態積木及凹槽(方形、圓形、花形、星形、 三角形)，使用Labelimg做實例分割手動標註，圖三，後做自動標註，使用google colab進行照片訓練。

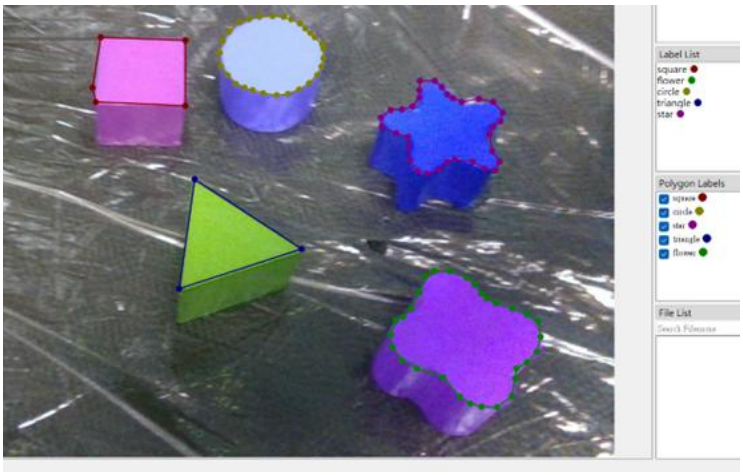


圖三、凹槽實例分割標註

- 使用Labelme做語意分割手動標註，並用YOLOv11語意分割模型訓練五個積木以及凹槽的形狀，找到每個形狀積木的遮罩，並標出每個遮罩距離最遠的兩個點（未來的機械手臂夾取點)， 手動標註如圖四、圖五。



圖四、凹槽語意分割標註



圖五、積木語意分割標住

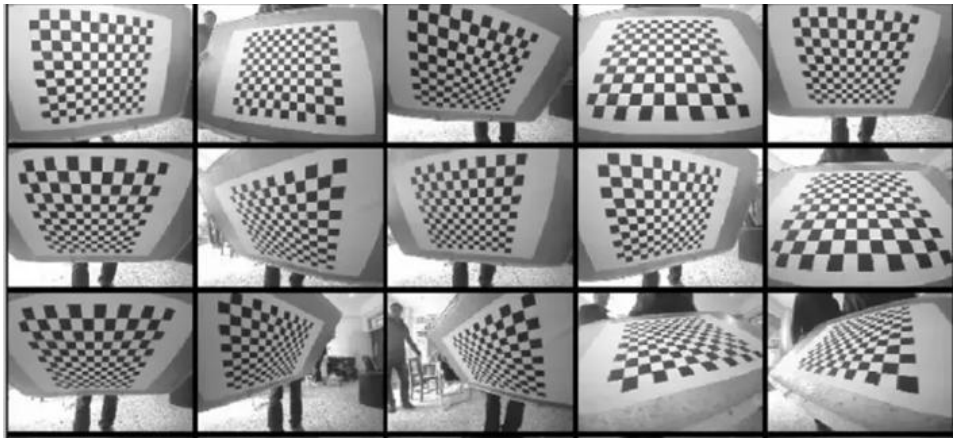
2. 相機標定:

- 將深度攝影機架設在機械手臂末端關節，使機械手臂移動後可準確找到夾取與放置的位置，並定義機械手臂基座高度。
- 做相機標定，在圖像測量過程和機器視覺應用中，為了確定物體表面某點的三維幾何位置與其在圖像中的對應點之間的轉換關係，需要建立相機成像的幾何模型[11]，這些幾何模型參數就是相機參數，包含相機的內參(像素座標至相機座標的轉換）和外參(相機座標至世界座標的轉換)。

  本計畫將採用張正友博士提出的張氏標定法，透過平面棋盤格進行標定，計算出相機的內外參數，並進一步估計畸變係數以優化成像結果。張氏標定法使用二維方格組成的標定版進行標定，採集標定版不同位置、姿勢的圖片，提取圖片中角點的像素座標，通過單應矩陣計算出相機的內外參數值，並利用非線性最小二乘法估計畸變係數，後使用極大似然估計法優化參數。

  此方法介於攝影標定法與自標定法之間，克服了攝影標定法需要的高精度三維標定物的缺點，並且解決了自標定法魯棒性差的問題。

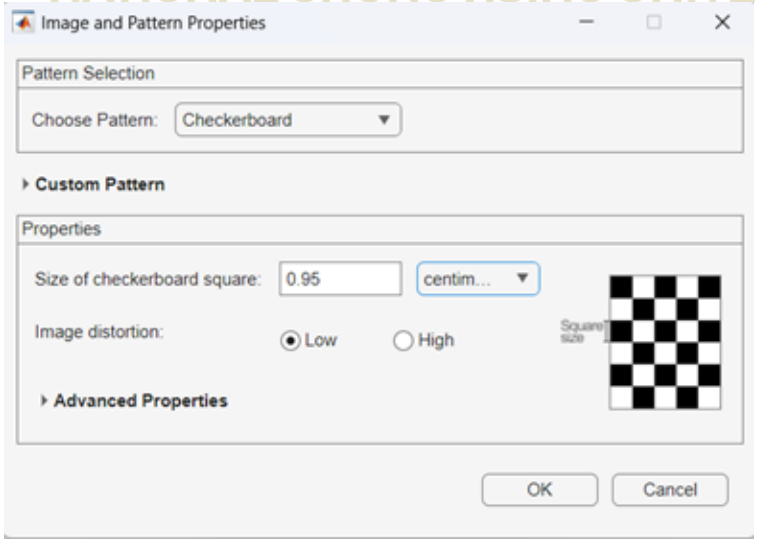
- 選擇棋盤作為標定物，因為其平面結構比起三維物體更容易進行數據處理。然而，平面棋盤相較於三維物體會喪失部分空間信息，因此需要通過改變棋盤的方位進行多次拍攝，從而獲取更豐富的坐標數據。如圖六所示，相機從不同方向拍攝同一個棋盤圖像，以提供更多的幾何信息。



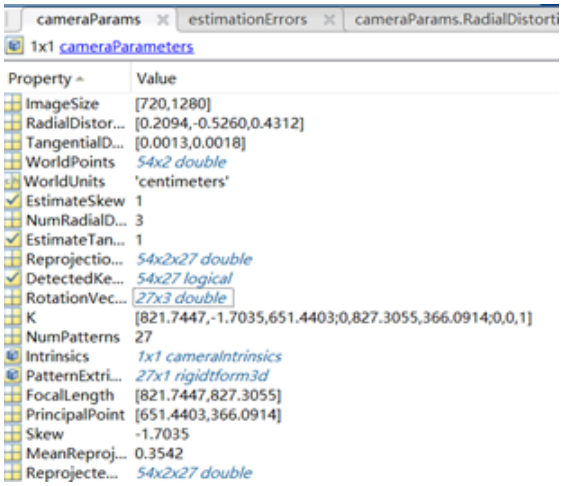
圖六、棋盤格

- 取得照片後將其匯入matlab獲取相機內參，確定工作範圍做座標軸轉換，matlab設定方式如圖七、圖八，結果預計如圖九。

圖七、選定Camera Calibrato



圖八、設定棋盤格初始邊長



圖九、所得之內參

3. 手臂控制:

- 至xArm-ESP32機械手臂官網（幻爾科技官網）下載軟體，包括Python編譯器和PC上位機(機械手臂參數控制軟體)，軟體如圖十、圖十一，軟體介面如圖十二、圖十三。

Python編譯器控制步驟：

(1) 將xArm-ESP32接上筆電。

(2) 在介面中選擇連接控制板，圖標由橘轉綠即為連接成功。

(3) 撰寫程式碼：在本地項目新建文件並撰寫程式碼，儲存後在界面選擇下載，下載成功程式碼名稱由綠轉紫。

(4) 燒錄：在介面選擇運行，運行圖標由橘轉綠。

(5) 若要要修改程式碼，需在每一次修改後重新儲存、下載、運行。

PC上位機使用：

(1) 將xArm-ESP32接上筆電

(2) 界面中選擇COM端口連接

(3) 介面左側可直接調整機械臂各關節參數

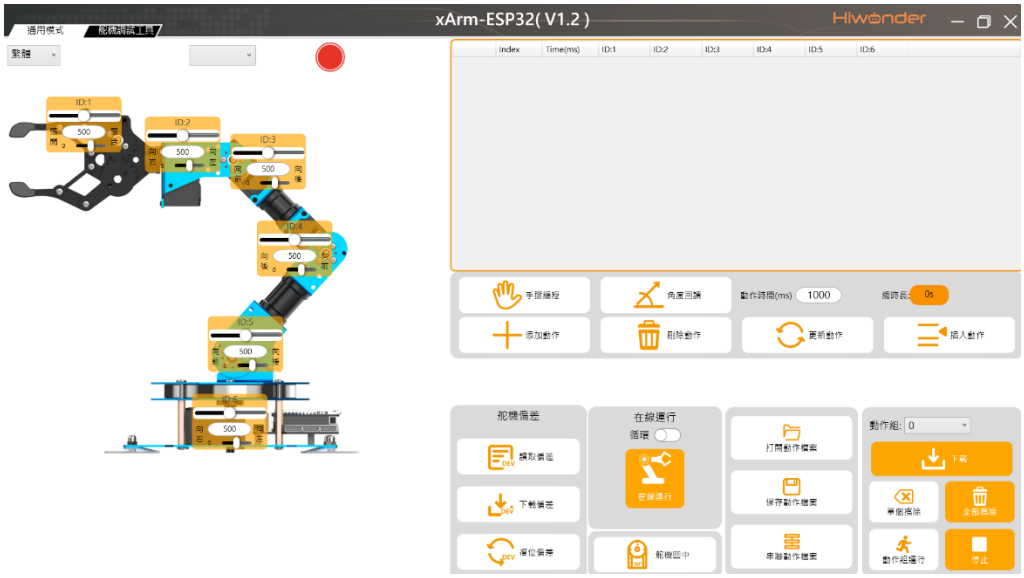
(4) 介面右側可除錯動作，摒除存到控制器

(5) 控制端配置控制指令並執行

圖十、燒錄及編譯軟體

圖十一、PC上位機(手動調整機械臂參數軟體)

圖十二、燒錄及編譯介面



圖十三、機械手臂關節參數調整介面

- 正向運動學：實現正向運動學[12]，透過機械手臂各個關節的角度計算機械手臂末端執行器（夾爪）的位置。以往專題生的連桿變換是基於DH參數（Denavit-Hartenberg Matrix），計算每個關節的相對變換，建立機械手臂的正向運動學模型，並逐層累積轉換矩陣，從基座座標系（Base Frame）開始，一直到末端執行器的座標系，最終得到末端的位置（Position）與姿態（Orientation）。

  首先，先定義DH參數法中的重要矩陣和變數。而在DH解法中，每一個關節i相對於前一個關節i-1之間的變換，可以用四個DH參數描述，分別是關節旋轉角度（Joint Angle）θ_i、連桿長度（Link Length）a_i、連桿偏移量（Link Offset）di和關節扭轉角（Twist Angle）α_i，基於這四個參數，我們定義從坐標系i-1到坐標系i的齊次變換矩陣T_i^{i-1}，如圖十四，而把各關節的變換矩陣依次相乘， 即可獲得正向運動學方程式。

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

圖十四、變換矩陣

- 逆向運動學：逆向運動學[13]主要用於計算機械手臂或其他多關節系統中各關節參數（如角度或位置），使末端執行器（夾爪）達到特定的目標位置和姿態。

  與正向運動學（由關節參數推導末端位置）相反，存在的挑戰在於解的多

樣性、一致性和效率問題，因為可能存在多解、無解或奇異解。解法通常分為解析法（推導公式）、數值法（優化算法）和學習法（機器學習）。

本計畫將應用幾何方法求解逆向運動學問題，通過已知目標座標及總角度推算各關節的運動參數，實現機械手臂的控制。為提升解算效率與穩定性，解法結合了解析法與數值法，確保在多解、奇異解等情況下的可靠運作。在運用過程中，我們使用幾何方法進行推算，已知目標座標（X,Y,Z）和總角度Φ(=$\theta_1+\theta_2+\theta_3$)，目標為求出各關節角度$\theta$的具體數值，並以度數表示，解法如圖十五。

公式:

(1) 垂直距離（調整z值以除去基座高度）：z = z – baseHeight

(2) 水平距離：r = $\sqrt{x^2+y^2}$

(3) 總距離（目標與原點的3D距離）：

total_distance = $\sqrt{\text{horizontal\_length}^2+z^2}$

(4) 延伸距離分段（可調經驗值）：d_1 = 0.17*D、d_2 = 0.83*D

(5) 中繼高度：h_1 = $\sqrt{{L_1}^2+{d_1}^2}$

(6) 總高度補償：h_2 = h_1 + baseHeight、h_3 = h_2 – cubeHalfHeight

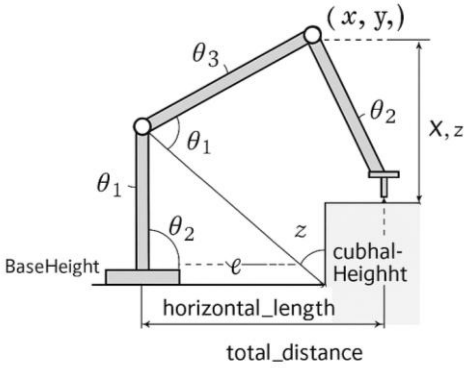
(7) 斜邊長：hyp_1 = $\sqrt{{d_2}^2+{d_3}^2}$

(8) 計算θ_0：

$\theta_0 = \tan^{-1}\frac{y}{x}$

(9) 計算θ_1、θ_2、θ_3：

- $\theta_1 = 180° - \cos^{-1}(\frac{{L_1}^2+{d_1}^2-{h_1}^2}{2\times L_1\times d_1})$
- $\angle_1 = \cos^{-1}(\frac{{L_1}^2+{d_1}^2-{h_1}^2}{2\times L_1\times d_1})$
- $\angle_1 = \cos^{-1}(\frac{{hyp_1}^2+{h_3}^2-{d_2}^2}{2\times hyp_1\times h_3})$
- $\angle_1 = \cos^{-1}(\frac{{L_2}^2+{hyp_1}^2-{L_3}^2}{2\times L_2\times hyp_1})$
- $\theta_2 = 180° - \angle_1 - \angle_2 - \angle_3$
- $\theta_3 = 180° - \cos^{-1}\left(\frac{{L_3}^2+{L_2}^2-{hyp_1}^2}{2\times L_3\times L_2}\right)$



圖十五、以目標座標與總角度求出各關節角度

4. 座標轉換:

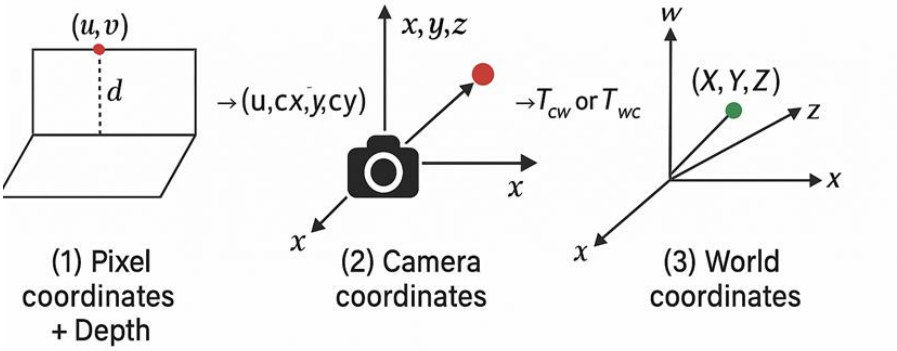
確認世界座標使用vscode打開深度攝影機，記錄深度攝影機的有效量測距離。量測積木高度和攝影機與桌面的當前距離，接著研讀xArm-ESP32機械手臂資料，確認機器人座標，利用深度相機數值進行座標係轉換，計算出該點相對圖片中心的相對距離

座標系分為世界座標系、相機座標系、圖片座標系和像素座標系，世界座標系(World Coordinate System)為用戶定義的三維坐標系，用於表示目標物在真實世界中的位置，其單位為公尺；相機座標系(Camera Coordinate System)是以相機為基準建立的坐標系，用於從相機視角描述物體的位置，起到連接世界坐標系與圖像像素坐標系的橋樑作用，單位同樣為公尺；圖片座標系(Image Coordinate System)用於描述物體從相機坐標系投影到圖像平面的過程，幫助理解成像後的透射關係。像素座標系（Pixel Coordinate System）是以像素為單位的2D座標系，常用於存取影像中的像素點，因單位為像素，不能直接反映真實世界的長度。

座標轉換時，四個座標系之間的轉換順序可定義為從二維到三維的映射過程，這個過程描述了如何將物體在照片中的像素位置，依次轉換為圖像平面上的幾何位置，然後再映射到相機視角的三維空間，最終確定在真實世界中的位置。

在進行三維重建、視覺定位或物體追蹤等電腦視覺應用時，理解並正確實作各種座標系之間的轉換是極為關鍵的步驟。實際上，從相機所擷取的二維圖像資訊到三維空間中的物體位置估算，需經過一系列座標系的轉換。

其轉換順序為：像素座標系 → 圖像座標系 → 相機座標系 → 世界座標系，如圖十六。



圖十六、座標轉換

透過此一轉換流程，可將二維圖像上的像素點，結合深度資訊與相機內外參數，推算出該點在世界空間中的三維位置，並進一步應用於空間定位、3D 建模、機器人導航等領域。

- 像素座標 + 深度資訊 → 相機座標

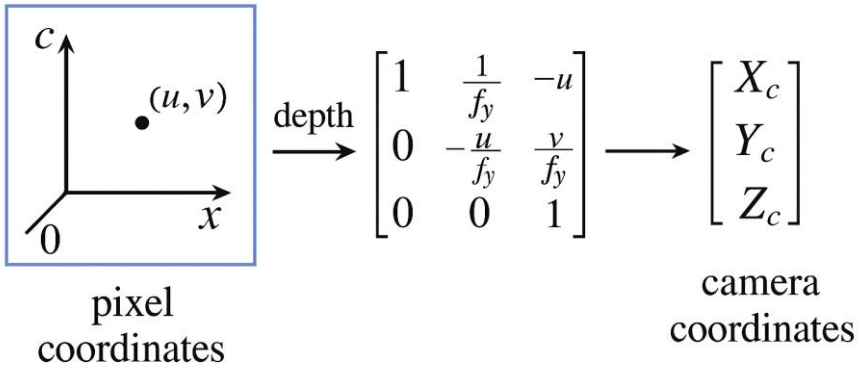
  假設相機內參（f_x, f_y, c_x, c_y）如下：焦距 f_x, f_y，主點 c_x, c_y（圖像中心）。給定像素點 (u, v) 及其深度 d，可轉換為相機座標系下的三維點，如圖十七：

  $x_c = ((u - c_x) * d) / f_x$

  $y_c = ((v - c_y) * d) / f_y$

  $z_c = d$

  因為圖像縮放和 RealSense 的深度單位規範，深度及長度值原為 2 倍距離，最終要再除以 2 才是實際值。



圖十七、像素座標 + 深度資訊 → 相機座標

- 相機座標 → 世界座標

  透過相機的位姿矩陣T_{cw}或其反矩陣T_{wc}，可將點從相機座標轉換到世界座標。其為一個4x4齊次變換矩陣T：

$$T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$$

  其中，R是3x3的旋轉矩陣，t是3x1平移向量，T可由SLAM、相機標定或系統估算得到。四元數可轉換為R，平移直接帶入，如圖十八。

$$\begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} \quad T_{wc} = \quad T_{wc} \quad = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}$$

$$T_{wc} = \begin{bmatrix} R_{11} & R_{12} & t_1 & t_2 & 0 \\ R_{21} & R_{22} & t_2 & t_3 & 0 \\ \vdots & \vdots & \vdots & \vdots & 1 \\ R_{31} & R_{32} & t_3 & t_3 & 1 \end{bmatrix}$$

圖十八、相機座標轉世界座標公式

5. 夾取與放置:

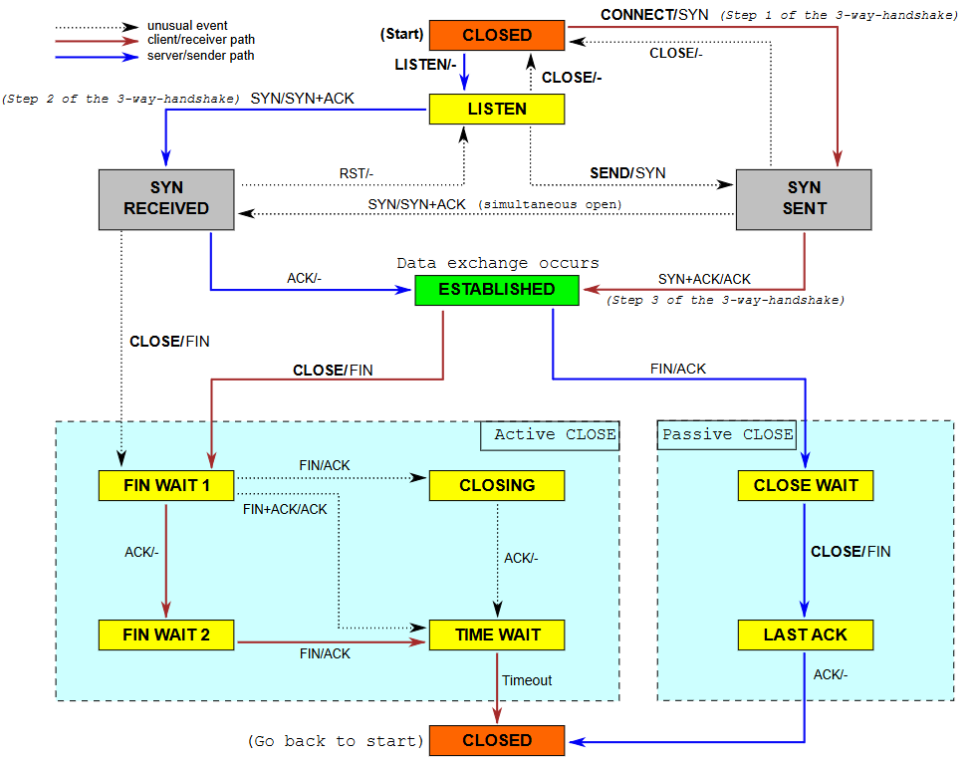
- 將正向運動學所得到的位置資訊與分割物體時所得之目標點與照片中心

相對距離相加，得到實際機械手臂需要到達的位置，接著運用逆向運動學[13]，取得到達目標座標所需的各關節參數，機械手臂按照參數動作，將夾爪移動到被夾取目標上方。

- 機械手臂執行器（夾爪）到達夾取目標上方後，旋轉夾爪使其對應遮罩最遠兩點，垂直放下夾爪夾取目標積木，做第一階段夾取。
- 利用深度攝影機尋找凹槽，取得分割物體時所得之影像中心點與凹槽中心點之相對距離，與機械手臂當前相對座標相加，得到目標座標，接著運用逆向運動學取得到達目標座標所需的各關節參數，機械手臂按照參數動作，將夾爪移動到目標位置（凹槽中心）上方，旋轉夾爪使其對應步驟一所得之凹槽遮罩最遠兩點，垂直放下夾爪並放下目標積木，完成積木放置。

6. 操作介面與遠端控制:

- 設計操作介面，顯示電腦影像辨識所偵測到的物件類別、座標，提供使用者控制機械手臂的功能。
- 透過TCP(Transmission Control Protocol)協議實現電腦與機械手臂控制器之間的數據傳輸，不再受制於導線，使使用者能夠遠端控制機械手臂，並確保遠端控制的穩定性與可靠性，如圖十九。



圖十九、TCP協定狀態圖

六、 系統架構

1. 軟體搭建

- Matlab2024：

  matlab官網：https://www.mathworks.com/products/matlab.html

- Intel RealSense Viewer：

  Intel RealSense SDK 2.0官網：https://www.intelrealsense.com/sdk-2/

- Visual Studio：

  官網：https://visualstudio.microsoft.com/zh-hant/downloads/

- Python 3.9.13：

  Python官網：https://www.python.org/downloads/

- OpenCV

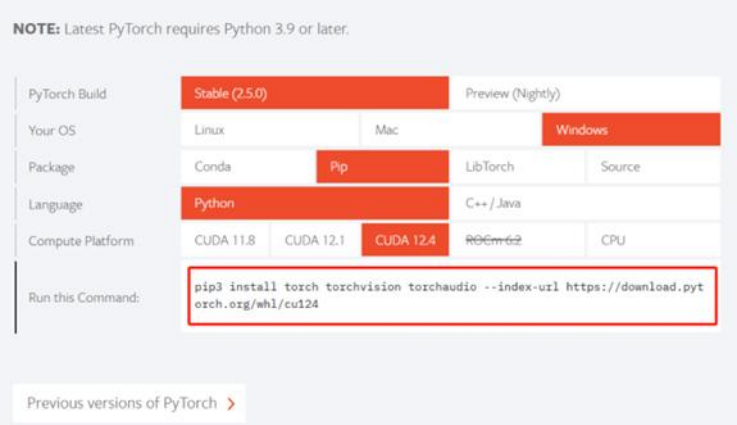
  官網：https://pypi.org/project/opencv-python/

- CUDA：

  NVIDIA DEVELOP：https://developer.nvidia.com/cuda-toolkit

- PyTorch，如圖二十：

  官網：https://pytorch.org/



圖二十、PyTorch設定

- Python Editor 1.0.3
- Yolov11專案：

  GitHub 下載：https://github.com/

- Labelme安裝

- xArm-V2.6

2. 硬體設備：

- Intel Realsense Depth Camera D435i
  因其加入IMU（慣性測量單元），可使攝影機在任何情況下移動時都能提升深度感知能力。這為基礎的 SLAM（同步定位與建圖）與追蹤應用開啟了可能性，能夠更好地對齊點雲資料。它也提升了機器人與無人機對環境的理解能力。使用D435i時，Intel RealSense SDK 2.0 會提供時間戳記對齊的IMU 數據，這些數據可以與高品質的深度資料同步對應。

圖二十一、Depth Camera D435i

- xArm-ESP32
  我們選擇xArm-ESP32作為我們的六軸機械手臂（包括5個自由度及夾爪），其為鋁合金材質，提供了微控制器的各種基礎操作Python庫文件，各種傳感器的庫文件，以及上位機，如圖二十二。

圖二十二、xArm-ESP32機械手臂

- 伺服馬達：
  本專題機械臂六關節總共使用三種型號伺服馬達，分別是LX-15D，如圖二十三、LX-225，如圖二十四、1號總線伺服馬達，各關節馬達分配如圖二十五。

圖二十三、LX-15D

圖二十四、LX-225

Servo ID	Model	position	Installation Note	Note
No.1	Specific servo gripper	Hand	Servo wires have been connected	-
No.2	LX-15D	Wrist	You just need to install the driving servo horn manually and do not need to install the auxiliary servo horn	Set the limited angle to 240°.
No.3		arm	Both driving and auxiliary servos have been installed.	-
No.4				
No.5	LX-225			
No.6	LX-15D	Button	Both driving and auxiliary servos have been installed.	Set the limited angle to 320°.

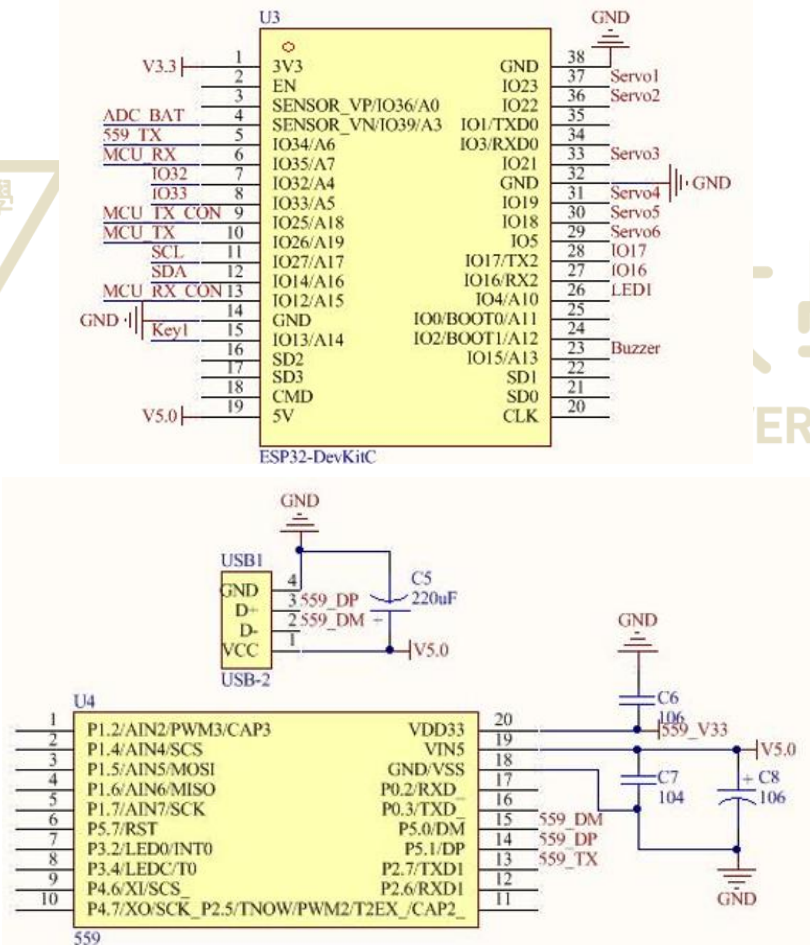
圖二十五、各關節馬達分配

- ESP32控制器

  本專題使用ESP32控制器，如圖二十六，ESP32核心板和多功能擴展板組合而成，除了設有USB Type-C接口，支援USB和UART開發使用外，還內置了蜂鳴器、LED等多個電子元件，方便用戶可以直接在上面連接其它傳感器和

執行模塊進行二次開發，且其帶有的WiFi和藍牙功能，方便用戶開發無線數據傳輸。ESP32原理如圖二十七。

圖二十六、ESP32控制器



圖二十七、ESP32原理圖

3. 硬體架構及維修

- 本專題使用 Intel Realsense Depth Camera D435i 深度攝影機進行畫面擷取，將影像傳回個人電腦，並利用 YOLOv11 訓練好的模型進行影像分析，判

斷有哪些被夾取物並告訴使用者，使用者選擇後影像分析其中點和最遠兩點，將資料傳至控制器，使機械臂移動到該位置附近，接著深度攝影機再次擷取畫面，影像分析後再次將資歷傳至控制器並移動，重複此步驟直到機械臂夾爪與被夾取物中心點距離小於 0.2 公分。

為能避免在夾取過程中移動到積木導致夾取失敗，我們將深度攝影機安裝在機械臂上，使機械臂每移動一次，都可擷取一張即時影像畫面。

- 機械臂維修：

本專題在機械臂實作與長時間操作過程中，為確保系統穩定運作，曾經歷並排除多項硬體與連接問題。以下列出本專題在執行過程中常見的維修狀況與處理方式：

(1) 馬達燒壞關節鎖死：

當機械臂關節 LED 閃爍紅光或關節伺服馬達冒煙時，表示此關節出現問題，需先一一排查是否是程式碼的問題導致關節鎖死，或是設備短路、斷度導致。

方法一：購買相同馬達，將機械臂關節拆卸後自行更換，更換時需注意接角，切勿接錯方向。缺點為安裝難度較高，具有失敗可能性。

方法二：若機械臂仍在保固期間，可先詢問原公司維修辦法並送修，缺點為因本專題使用之機械臂原公司工程師皆不在台灣，所以會導致工程師估價以及維修時間較長。

(2) 訊號線扯斷：

方法一：購買相同款式之訊號線（線頭上部分寬度、下部分寬度、高度、長度皆需精確量測），自行拆卸機械臂後組裝，組裝時需注意機械臂關節墊片位置，原基礎關節數值為 500，各關節數值皆為 500 時，機械臂成直立狀，若組裝後非此狀態，則表示墊片方向錯誤，需重新拆卸關節進行校正。訊號線如圖二十八，訊號線端子如圖二十九。

圖二十八、訊號線

圖二十九、訊號線端子

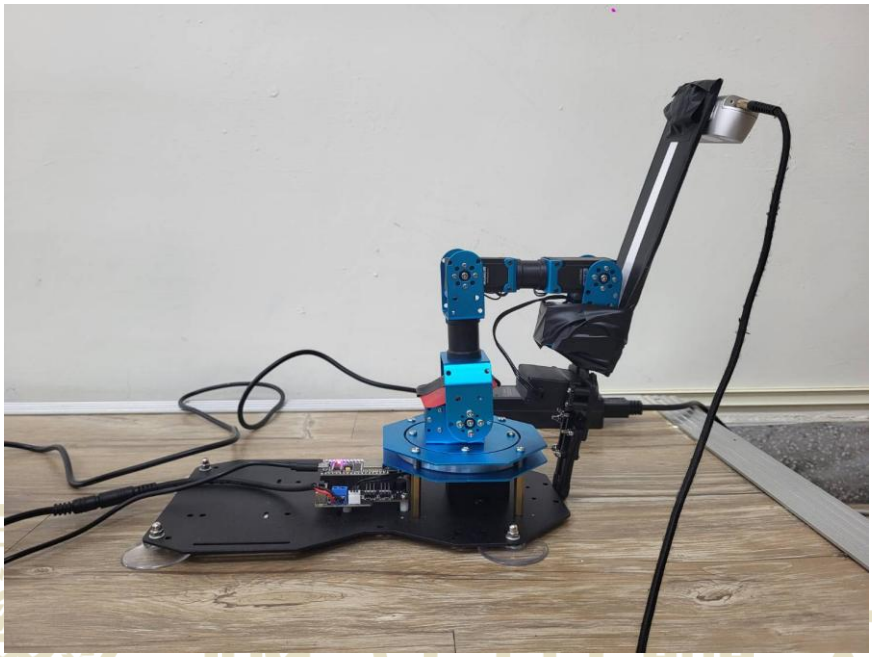
方法二：同馬達燒壞關節鎖死之方法二，若機械臂仍在保固期間，可先詢問原公司維修辦法並送修。

七、 任務介紹

本研究旨在開發一套整合三維視覺辨識與運動規劃技術的機械手臂智能夾取系統，能夠：

1. 辨識不同形狀的積木（方形、圓形、花形、星形、三角形）與其對應的凹槽。
2. 利用YOLOv11深度學習語意分割模型辨識物體並定位夾取點。
3. 透過Intel RealSense D435i深度相機捕捉三維影像，獲得空間座標資訊。
4. 應用正向與逆向運動學，計算機械手臂動作軌跡並完成高精度夾取與放置。
5. 搭配ESP32控制模組控制xArm機械手臂進行動作。

6. 設計TCP通訊的遠端操作介面，實現電腦與機械手臂間的資料傳輸與遠端控制。任務執行期間，即使積木或凹槽位置改變，系統也能藉由即時影像分析與座標轉換，自動完成對位與移動操作。運動控制部分整合正向與逆向運動學，提升夾取與放置的精準度與穩定性。

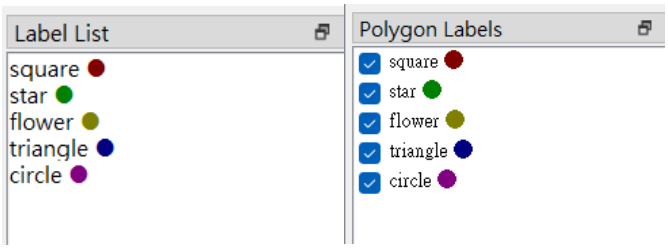


圖三十、實作狀態

八、 模型訓練

1. 標註照片

- 進行影像辨識前需先訓練自己的數據集，我們使用Intel Realsense Depth Camera D435i深度攝影機拍攝照片，並使用labelme標註圖片，將所看到的積木和凹槽分為五個類別，分別是square、circle、flower，star、triangle。

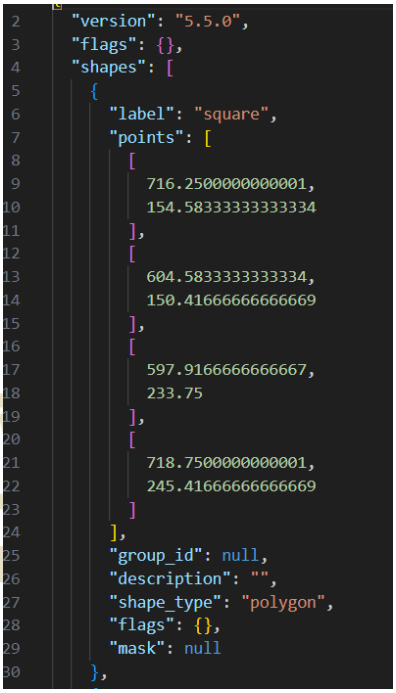


- Labelme使用方式：

(1) 點選左上角 「Open Dir」 開啟欲標註的影像資料夾（建議圖檔統一格式，如 .jpg 或 .png）

(2) 開始標註：點選「Create Polygon」，在影像上依序點選輪廓邊界，點選最後一點時，雙擊或按 Enter 鍵完成，輸入標籤名稱（label）並按 Enter 確認。

(3) 標註完成後，會在圖片相同位置產生一個 .json 檔案，其包含影像資訊（大小、路徑）和各個標註物件（標籤、點座標、類型），如圖三十一。



圖三十一、範例.json檔案

- 因 LabelMe 原生是 .json，需要透過工具轉換為 YOLO 格式，例如 labelme2yolo 或自行撰寫 Python 腳本進行轉換。

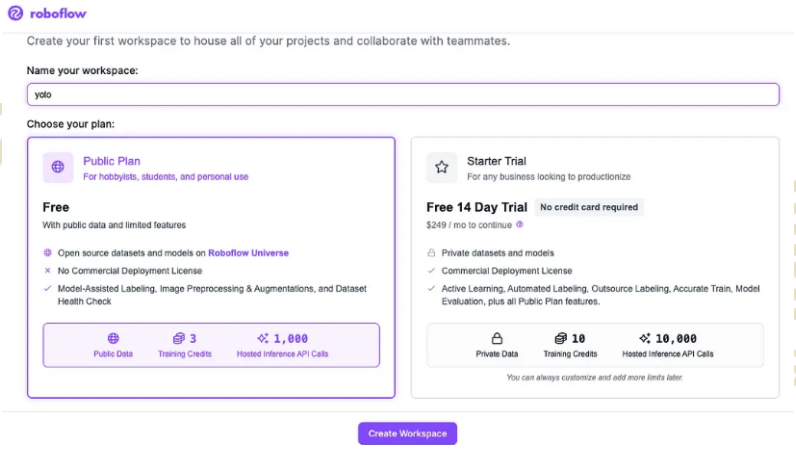
2. Roboflow

Roboflow 是一個專為電腦視覺專案設計的整合平台，主要提供資料集建立、標註、轉換、增強、訓練、部署等功能，是許多使用 YOLO、Detectron2、TensorFlow、PyTorch 等框架進行影像辨識開發者常用的工具。

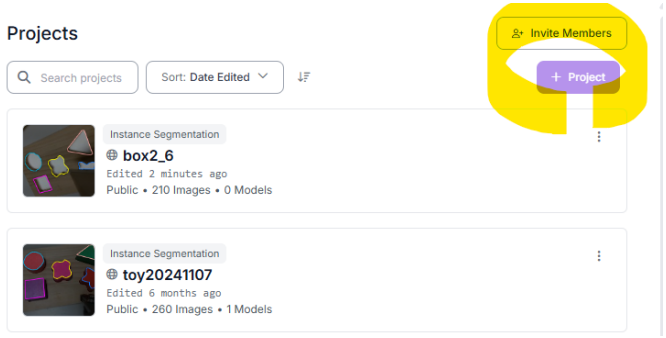
- 本專題主要利用Robolfow做相片預處理和資料增強的部分， 預處理是在模型訓練前，對輸入圖片做的基本標準化處理，確保圖片品質一致，排除模型無法收斂的非必要因素。目的是統一尺寸（避免 tensor shape mismatch）、去除雜訊（提升收斂穩定度）、灰階轉換（適合某些簡化任務）、壓縮資料大小（加快訓練速度），而Roboflow支援的預處理操作有以下：

(1) Resize（重新調整大小）：將所有圖片轉成固定解析度。
(2) Grayscale（轉灰階）：把 RGB 轉成單通道灰階圖。
(3) Auto-orient（自動調整方向）：根據 Exif 資訊轉正圖片方向。
(4) Strip Metadata（移除EXIF）：清除檔案中的 GPS、設備資訊等。
(5) Smart Crop（智慧裁剪）：對包含標註物件的區域裁切、放大。

- 資料增強則是在訓練集資料有限時，人工模擬多樣場景或視角變化，讓模型學習更具泛化能力，減少overfitting，其目的為增加樣本多樣性（角度、亮度、大小、遮擋等）、提升模型在真實環境的容錯率（不怕小變化就預測錯誤）、模擬目標出現在不同位置、方向、環境的情況，例如，翻轉、亮度調整、模糊、雜訊、遮擋、斜切等。
- Roboflow使用步驟：
  (1) 註冊帳號並建立新專案，如圖三十二、圖三十三

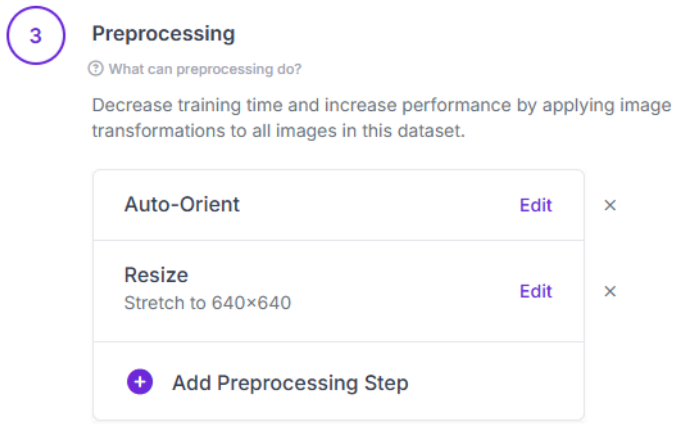


圖三十二、Roboflow註冊帳號

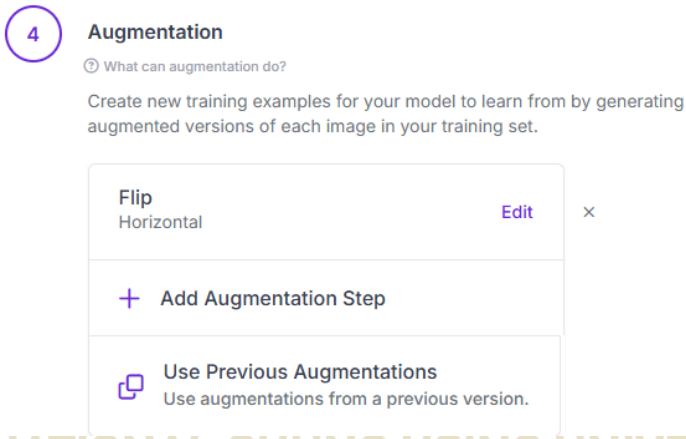


圖三十三、建立新專案

(2) 上傳數據後選擇影像前處理和資料增強的項目，如圖三十四、圖三十五。最後取得照片集。



圖三十四、影像前處理項目



圖三十五、資料增強項目

3. Yolov11

在完成 Roboflow 的影像前處理與資料增強後，我們將增強後的資料集下載成 YOLO 格式，接著在本地進行 YOLOv11 的模型訓練。

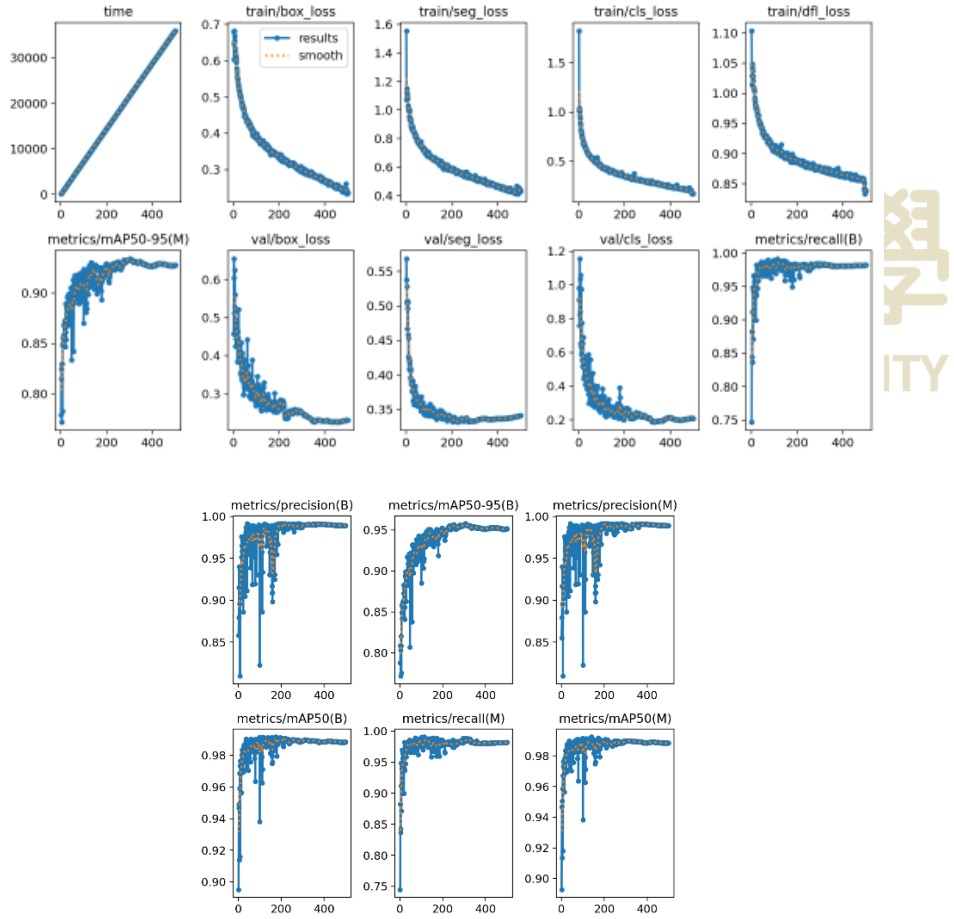
我們選用的是Ultralytics最新版本的YOLOv11，這個版本相較於YOLOv8在精準率與速度上做了最佳化，同時也保有良好的模型擴充彈性與訓練效率。

- 訓練流程如下：
  (1) 環境建置：我們先在本機使用 Python 建立虛擬環境，安裝對應的 PyTorch CUDA 版本與 YOLOv11 套件，確保整體 GPU 加速與模組相容。
  (2) 資料結構整理：匯入後的資料依 YOLO 標準格式整理，包含 images/train、images/val、labels/train、labels/val 四個目錄，並建立對應的 data.yaml 檔來定義類別名稱與路徑。
  (3) 啟動訓練：使用 yolo detect train 指令，搭配自訂參數設定：
  (4) 觀察訓練結果：訓練完成後，模型權重會輸出在

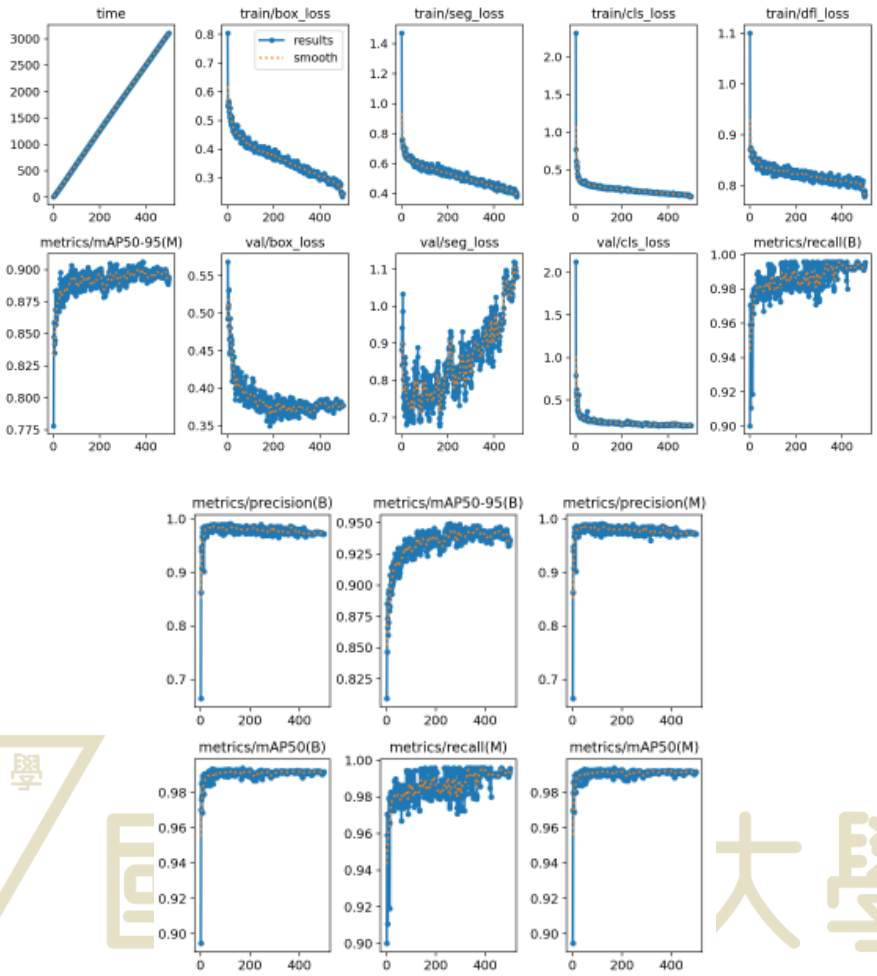
runs/detect/exp/weights/best.pt，可用於後續推論。並搭配 results.png、confusion_matrix.png 等輸出，觀察模型的準確率、損失下降曲線與混淆矩陣。

- 選擇YOLOv11 的原因：
  (1) 效能提升：相較 YOLOv5/YOLOv8，YOLOv11 對推論速度與參數數量做了優化。
  (2) Ultralytics 原生支援：能與新版 yolo 指令整合無縫操作，訓練流程標準化。
  (3) 方便部署：支援匯出成 ONNX 與 TorchScript，可用於各種設備與邊緣部署。
- 訓練結果：

訓練好的結果儲存至以設定好的資料夾中，以其中兩次訓練為例，圖三十六為第一次訓練、圖三十七為第二次訓練結果。



圖三十六、第一次訓練結果



圖三十七、第二次訓練結果

分析兩次訓練解果發現：

(1) Loss：

第一次的Loss初始值為平穩且快速下降，val loss部分略有波動但整體趨勢穩定，表示偵測穩定且邊緣清楚。

第二次的Loss初始值高（>2）、val loss 波動大，表示偵測點位不穩定，中心點誤差偏大

(2) mAP與Precision / Recall：

第一次的mAP50-95可達0.92~0.93，高度可靠，且precision 與 recall 都維持在0.98~0.99左右，代表模型穩定學到邊界框與語意標註，且泛化性佳。第二次的初期 precision 僅約 0.66，mAP 也僅 0.77~0.78，且Recall 過高可能代表模型過度傾向召回，導致 false positive 增加，綜合來看模型還沒學到足夠的特徵，可能會出現「抓得到大概形狀但定位不準」的情況。

分析兩次訓練解果的差異：因第二次資料標註品質差、樣本數少、分布不均導致，最終解決方法：

(1) 優先使用預訓練模型：model=yolov11n.pt 並微調

(2) 分批訓練：用warmup（先100 epoch再繼續resume）方式訓練

(3) 優化資料標註：減少標註誤差、中心點標註偏移、小物體位置模糊等問題。

(4) 適當調整lr：降低初始學習率如 lr0=0.001

4. Yolov8、yolov11模型

本專題中，我們以 YOLOv11 為物件偵測主體模型進行訓練與應用，但由於 YOLOv8 亦為目前常見的實務模型之一，故我們進一步就兩者在模型結構、效能與部署彈性進行對比分析。透過這項比較，可幫助釐清 YOLOv11 在本專案任務中是否具有更佳的實用性與表現，也為後續模型選擇提供依據。

- 架構差異：
  YOLOv8 使用 PAN-FPN 架構進行特徵融合，模型結構相對簡潔，適合快速訓練與部署。而 YOLOv11 則採用改良版 BiFPN（Bidirectional Feature Pyramid Network）以及輕量上下文編碼模組，能更有效處理不同尺寸與不同背景下的物件，提升對小物件或遮擋物件的辨識能力。此外，YOLOv11 在 Head 部分也整合了多任務支援，可以同時進行分類、偵測、分割甚至姿態預測，且推論效率相對更高。

- 訓練與效能：
  在相同訓練條件下（相同資料集、相同 epoch 數），YOLOv11 的 mAP50 準確率普遍高於 YOLOv8，特別是在小模型（如 YOLOv11-nano）或低資源環境中效果更加明顯。根據官方與社群測試數據顯示，YOLOv11 相比 YOLOv8 約提升 2～4% 的準確率，且推論速度略快。

- 選擇原因：
  考量到本專題需要在準確率與即時偵測間取得平衡，並具備一定的部署彈性，因此我們最終選擇使用 YOLOv11 作為訓練主體模型。其在準確度、架構先進性與部署便利性上，皆較 YOLOv8 更符合本專題需求。

九、 程式碼說明

1. 手臂控制程式碼：

- 移動（all_to_test.py）：

  模組與函式介紹:

  (1) run_action_group(...)：從 action_group 中逐一執行各伺服馬達的動作。使用 bus_servo.run(id, position, time) 控制目標馬達移動至指定位置，並以time.sleep()等待最大執行時間。

  (2) position_to_angle(position, middle_angle, flip)：將總線伺服器回傳的位置值（0~1000）轉換為實際角度（單位：度）。

  (3) read_motor_positions(...)：讀取編號 6、5、4、3（基座、肩膀、手肘、手腕）的馬達位置，回傳值會限制在 0~1000 範圍內，若讀取錯誤則返回四個 None。

  (4) calculate_angles(x, y, z)：根據期望的 XYZ 世界座標，反解出四個馬達應達成的角度。以逆向運動學計算 θ_0（基座）、θ_1（肩部）、θ_2（手肘）、θ_3（手腕）等角度。需考慮臂長、基座高度、Z 軸補償、空間範圍，且含有大量三角函數與餘弦定理計算。

  (5) AngleConvert(angle, middle_angle, flip)：將角度值轉為對應的伺服器位置值（0~1000）。利用 (angle - middle_angle) 轉換為實際 position，若 flip，則回傳 1000 - position。

  (6) ArmControl(target_coordinate, run_time, bus_servo)：移動機械臂到指定三維座標位置。使用 calculate_angles 計算反解角度，利用 AngleConvert 將角度轉為位置，然後送給六顆伺服器，若反解失敗（例如超出工作範圍），則回傳 False。

  (7) move_arm(dx, dy, bus_servo=None)：將末端夾爪根據相對位移 (dx, dy) 進行移動，為主控制流程。

  步驟一：讀取目前位置

  步驟二：加上 dx, dy 得到目標位置

  步驟三：執行 ArmControl（逆向運動學 + 伺服驅動）

  步驟四：再次讀取新位置，反推現在的 XYZ

- 夾取（all_to_testa.py）：

  模組與函式介紹:

  (1) AngleConvert(...)：角度 ➜ 馬達控制值（0–1000），當 angle == middle_angle 時，控制值為 500，計算公式：p = 500 + (angle -

middle_angle) * 25 / 6，若 flip == True，將最終值反轉成 1000 - p（方向反轉用）。

(2) calculate_movement(x, y, dx, dy, theta_deg)：給定原始點 (x, y) 和偏移量 (dx, dy)，再加上一個基準旋轉角 θ，計算最終旋轉後的新座標。

(3) position_to_angle(...)：馬達控制值 ➜ 角度（與 AngleConvert 反向），給定 position（0–1000）與中心角，還原對應的角度，支援 flip=True 時進行方向調整。

(4) read_motor_positions(...)：讀取馬達 ID6（基座）、ID5（肩部）、ID4（手肘）、ID3（手腕）的目前控制值（0 – 1000），加上範圍保護（0~1000），若失敗，回傳四個None。

(5) calculate_angles(x, y, z)：反向運動學 IK 計算，將目標空間座標 (x, y, z) 轉換為四個關節的角度（θ0~θ3），臂長 L1, L2, L3，基座高度 baseHeight，運用餘弦定理與投影幾何解出每個角度，若超出最大長度，則丟出錯誤（out of range）。

(6) ArmControl(target_coordinate, run_time, bus_servo)：將 XYZ 座標轉換成對應的馬達控制值，並執行移動。

(7) runall_to_test2(specified_angle, shape, run_time=2500, bus_servo=None)：為主要流程控制器：自動完成「夾取任務」的所有步驟。

步驟一：讀取當前末端夾爪位置。

步驟二：計算目標座標（offset 位移 dx=30, dy=26，Z=25）。

步驟三：執行機械臂移動到目標位置。

步驟四：打開夾爪並旋轉夾爪角度。

步驟五：下降至夾取位置並關閉夾爪。

步驟六：回到預設位置 (X=0, Y=90, Z=20) 。

- 放下（all_to_testb.py）：

模組與函式介紹:

除主程式外，其餘模組皆與夾取時相同。

(1) runall_to_test3(...)：主流程控制器。

步驟一：決定夾爪旋轉角度，若形狀為s2或f，表示為斜角抓取。

步驟二：讀取目前末端執行器位置（正向運動學）。

步驟三：計算目標抓取點。

步驟四：移動到目標點（Z=25）。

步驟五：移動到抓取點（Z = -10）並關夾爪。

步驟六：返回預設位置並歸零夾爪旋轉。

- 連接機械臂與PC（boot3.py）：
  主要基於 Wi-Fi 的 TCP Socket Server，用於接收遠端指令，並根據指令格式自動控制機械臂完成不同的動作流程。是整個機械臂系統的通訊入口與控制中樞。

  功能：

  (1) 連接 Wi-Fi
  (2) 開啟 TCP Socket Server
  (3) 等待並接收兩個參數的指令
  (4) 根據小數位數與數值特徵選擇控制動作
  (5) 呼叫不同的機械臂控制模組（move_arm、runall_to_test2、runall_to_test3）

  模組與函式介紹：

  (1) __init__()：初始化類別，設定屬性 received_data 用來儲存接收到的指令內容。
  (2) connect_wifi()：連接 Wi-Fi，開啟無線網路介面，最多等待 20 秒完成連線，回傳 wlan 物件供後續使用。
  (3) start_socket_server()：核心伺服器流程，使用 Wi-Fi 的 IP 開啟 TCP Server，監聽 port 1000，不斷接受 client 連線，每次接收到資料後執行 self.process_received_data() 處理邏輯。
  (4) process_received_data()：訊息解析與動作選擇邏輯。
  (5) main() 函式：建立伺服器物件並啟動 TCP 通訊流程。
- 正向運動學副函式（ForwardKinematics）：

  實作了以 DH 參數為基礎的機械手臂正向運動學（Forward Kinematics, FK）系統，可根據 4 軸角度 theta0～theta3 推算出末端執行器（夾爪）在三維空間中的座標 (x, y, z)。

  模組與函式介紹：

  (1) dh_transform(alpha_deg, a, d, theta_deg)：產生單一DH參數下的4x4轉換矩陣如下。

  | cosθ -sinθ 0 a |
  | sinθ·cosα cosθ·cosα -sinα -d·sinα |
  | sinθ·sinα cosθ·sinα cosα d·cosα |
  | 0 0 0 1 |

  (2) matmul_4x4(m1, m2)：將兩個 4x4 矩陣相乘，標準三層迴圈矩陣乘法，

回傳新矩陣結果 res[i][j] = Σ m1[i][k] * m2[k][j]。

(3) forward_kinematics(theta0, theta1, theta2, theta3, d, l1, l2, l3)；依序串接每段 DH 矩陣，得到總轉換矩陣。

(4) calculate_position(theta0, theta1, theta2, theta3)：主呼叫介面，計算末端執行器位置，使用固定幾何參數：d = 65（底座到第一關節的垂直高度）、l1 = 101（第一段手臂長度）、l2 = 95（第二段手臂長度）、l3 = 165（第三段手臂長度）。呼叫 forward_kinematics(...)，得到最終 (x, y, z) 並回傳。

- 逆向運動學副函式（inverse_kinematics）：

以幾何法解四軸機械手臂的逆向運動學（Inverse Kinematics, IK），輸入一個期望的三維空間座標 (x, y, z)，輸出需要的四個關節角度 θ0, θ1, θ2, θ3，讓末端執行器能移動到該位置。

步驟：

(1) 參數初始化:

```
L1 = 101      # 第一段手臂長度
L2 = 95       # 第二段手臂長度
L3 = 165      # 第三段手臂長度（夾爪或末端工具）
baseHeight = 65  # 機械臂基座高度
cubeHalfHeight = z  # 預設末端物件高度等於 z（等效物體厚度）
```

(2) 調整 Z 軸，扣除基座高度：讓後續計算以「肩部為原點」為基準點。

(3) 計算與目標點的距離：
horizontal_length 是底座平面到目標點 XY 的水平距離。
total_distance 是機械臂末端實際要延伸的總距離（3D 空間）。

(4) 檢查是否超出機械臂最大長度：檢查可達性，若超出臂長總合，則無法到達該點。

(5) 計算 θ0（基座旋轉角）：這是平面上的偏轉角，讓機械臂轉向目標位置。

(6) 將整體距離分為兩段比例長度：將總距離分為 17% 與 83%，能分段解決 θ_1、θ_2、θ_3 的幾何問題。

(7) 計算 θ1（肩關節）：使用餘弦定理解出肩部需仰角多高，配合距離 distance_project_part1。

(8) 推導 θ3（手腕角度）：先解出最遠那段的長度（斜邊 hyp1），再用餘弦定理求出關節 3 的彎曲角度 θ_3。

(9) 計算 θ2（手肘角）三段餘弦定理補角加總：θ2 是中間關節，角度取決於三段夾角的相減。

(10) 回傳角度結果：所有角度單位皆為度（degree），四個值代表。

θ0：基座轉角

θ1：肩部仰角

θ2：肘部彎曲角

θ3：手腕角度

2. 物件偵測程式碼：

- 電腦端夾取（conformity.py）：

(1) 主要功能：

- 利用 Intel RealSense 相機 ＋YOLO 模型辨識與追蹤積木。
- 根據積木座標計算相對位移。
- 將結果經由 TCP 傳送給機械手臂，控制其動作。

(2) 模組與函式初始化：

- 匯入必要模組。
- socket 用來 TCP 通訊。
- pyrealsense2 控制 D435i 相機。
- YOLO 使用 Ultralytics 的模型進行積木辨識。

(3) TCP連線與傳送指令：

- 連接機械手臂的伺服器 IP 和 port。
- connect_to_server()：建立 TCP 連線。
- send_to_arm(message)：將訊息轉為字串送出，並接收回傳訊息。

(4) 通用功能函式區塊：

- find_farthest_points(contour)：給定遮罩的邊界，找出距離最遠的兩點 → 用來估算物體的寬與方向。
- compute_average_depth(depth_values)：過濾掉 0 深度後計算平均。
- compute_average_depth(depth_values)：將積木名稱轉為指定的代碼（如 circle → 1111）。

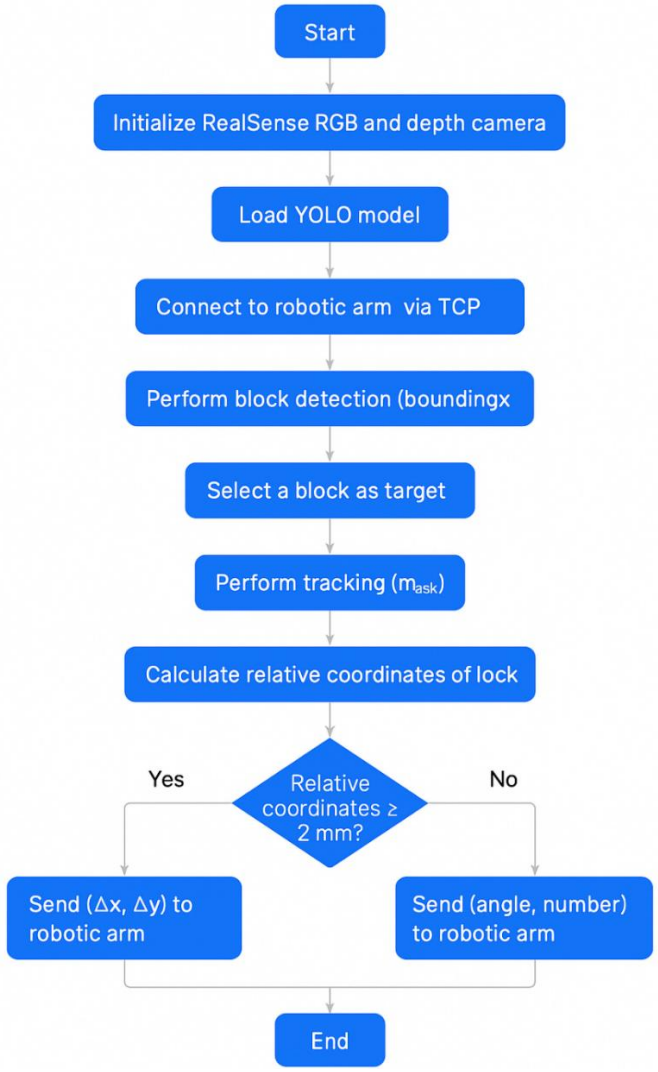
(5) 主流程 run_a111()：

- 相機初始化與深度內參取得：取得相機內參（fx, fy, ppx, ppy）與 depth_scale。
- 載入YOLO模型：同一個模型路徑，同時用來做邊界框檢測（model_box）與遮罩辨識（model_mask）。
- 與機械手臂連線（connect_to_server()）。
- 進行積木偵測 (邊界框)：利用YOLO偵測積木框，解析bbox資訊並記錄積木名稱、座標與中心點，並畫出邊框、中心點、積木名

稱，等待使用者選擇一個積木編號作為目標。

- 進行追蹤（5 秒收集資料）：再次使用YOLO，但這次拿遮罩進行更細緻的追蹤，對應積木名稱，確認遮罩輪廓，找最遠兩點fp1, fp2，計算角度。
- 深度 + 座標計算：透過 depth_image[y, x] * depth_scale 取得深度資訊，透過像素轉世界座標公式轉換，並進一步計算出積木相對於畫面中心的相對座標 (Δx, Δy)。
- 傳送控制訊息給機械手臂：若相對座標大於 2 公釐，代表要移動手臂 → 傳送 XY，若位置準確無誤，傳送角度與積木代碼 → 表示可以進行夾取。
- 回傳 5 個關鍵值作為結果。

流程圖，如圖三十八：



圖三十八、流程圖

- 電腦端放置（all_to_tast3.py）：

  這段程式碼整合了 RealSense D435 相機、YOLO 模型與 TCP 傳輸，完成「積木辨識 → 精細追蹤 → 世界座標轉換 → 傳送給機械手臂」的完整流程。

  (1) 模組與參數設定：
  - socket：建立TCP傳輸。
  - pyrealsense2：控制RealSense相機。
  - cv2：OpenCV：影像處理與顯示。
  - YOLO：使用Ultralytics的YOLO模型進行目標檢測。

  (2) 函式定義：
  - connect_to_server() / send_to_arm(message)：建立並維持TCP連線，用於傳送「角度」或「XY 移動量」指令給機械臂。
  - find_farthest_points(contour)：傳入遮罩輪廓 contour，找出輪廓中距離最遠的兩點fp1和fp2，用來判斷積木方向、寬度，以及計算角度。
  - compute_average_depth(depth_values)：計算去除0值後的平均深度，避免無效點干擾座標推估。
  - get_block_number(block_name)：將積木的類別名稱（如 "circle"）轉成對應代碼（如 1111），用來傳送給機械手臂辨識用途。

  (3) 主流程 run_all1()：
  - 相機初始化：同時開啟 深度與彩色影像串流，擷取內參：fx, fy, ppx, ppy，取得深度比例 depth_scale 用來從 depth map 換算實際距離（m）。
  - 載入 YOLO 模型：邊界框 (box) 與 遮罩 (mask) 分開處理，但實際為同一模型。
  - 積木初步偵測（邊界框）：使用YOLO模型進行推論，取出邊界框bbox=(xmin, ymin, xmax, ymax)、類別ID與名稱block_name，計算中心點 (cx, cy) 並加入 block_list，並顯示圖像標註，等待使用者選擇一個積木進入追蹤。
  - 遮罩精細追蹤（5 秒內）：根據 selected_block["name"] 選出同類別遮罩，重新計算 fp1, fp2 最遠兩點，並從其中間取出中心點 candidate_center。
  - 追蹤資料濾波與最佳解選擇：將5秒內每一幀的中心點深度（collected_depths_center）與目標點深度（collected_depths_object）。
  - 像素轉世界座標轉換公式：

    a. 畫面中心世界座標（基準點）：

    Xc = (cx - ppx) * Z / (fx * 2)

Yc = (cy - ppy) * Z / (fy * 2)

Zc = Z

b. 積木世界座標：

Xo = (ox - ppx) * Z / (fx * 2)

Yo = (oy - ppy) * Z / (fy * 2)

Zo = Z

c. 相對座標（以中心為基準）：

rel_x = Xo - Xc

rel_y = Yc - Yo　# 注意 Y 軸修正方向

rel_z = Zo - Zc

- 控制邏輯分支：

a. 若 |rel_x| > 3 or |rel_y| > 3：表示位置偏差大，傳送 XY 移動資訊給手臂，並 sleep(15) 再追蹤一次。

b. 若位置偏差已小於 3 mm：表示已就定位，傳送角度與積木代碼讓手臂執行動作（夾取等）。

- 最終輸出：

return (

screen_world_coords,　# 畫面中心座標

object_world_coords,　# 物件中心座標

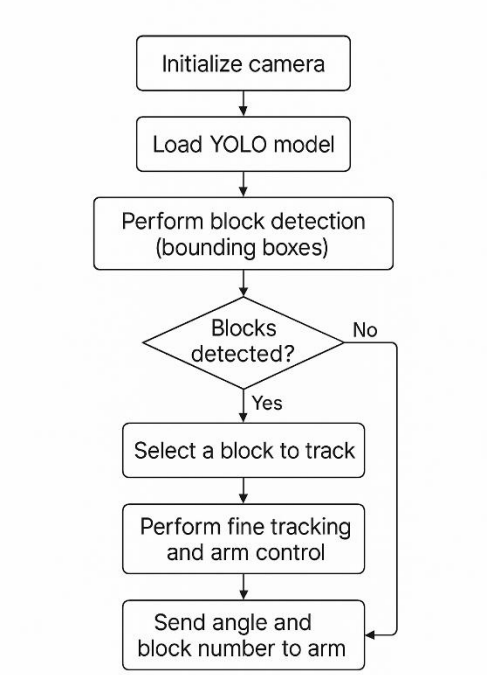
relative_world_coords, # 相對 XYZ

angle_final,　　# 角度

selected_block["name"] # 積木名稱

)

(4) 流程圖，如圖三十九：



圖三十九、流程圖

十、 程式整合

本專題系統整合了多項模組與演算法，實現從積木辨識、座標轉換、夾取操作到物品放置的完整流程。系統主體以 Python 實作，並以模組化架構設計，涵蓋以下幾大部分。

1. 系統架構

本系統共分為以下四個主要子程式：

- boots3.py：主流程，負責控制整體流程執行與狀態轉換。
- all_to_test.py：微調控制與誤差補償，處理目標對準與角度調整。
- all_to_testa.py：負責計算移動後的正向運動學與深度修正值。
- all_to_testb.py：放置流程控制，包含箱子辨識與物體放置。

2. 程式流程整合圖

整體邏輯自相機啟動後，依序經過 YOLO 模型辨識、精細追蹤、座標轉換、誤差判斷與補償、夾取與放置等動作，完成一筆操作循環，如圖四十。

圖四十、流程圖

3. 模組整合細節說明

- 主流程控制（boots3.py）：
  (1) 使用RealSense D435i相機取得彩色與深度影像。
  (2) 呼叫YOLOv11模型辨識積木位置與類型。
  (3) 使用遮罩追蹤方法尋找積木最遠兩點，計算其中心與旋轉角度。
  (4) 呼叫相對座標轉換函數取得物件與畫面中心之相對位置差。
- 微調流程與誤差補償（all_to_test.py）：
  (1) 當相對位移大於2 mm時，自動進入此模組。
  (2) 計算需補償的XY相對移動量，並傳送至機械手臂進行微調。
  (3) 移動後再次擷取影像並重新比對是否已成功對準。
  (4) 若仍未成功，則重複補償流程直到對準為止。
- 移動與運動學補正（all_to_testa.py）：
  (1) 根據實際移動後機械手臂位置，執行正向運動學計算出末端座標。
  (2) 進行與目標位置的深度校正，使後續補償更精準。
- 放置流程（all_to_testb.py）：

(1) 使用YOLO模型辨識箱子位置與角度。
(2) 重複與toy相同的追蹤與誤差判斷流程。
(3) 執行放置動作：夾爪打開 → 移動至箱子中心上方 → 下壓 → 放開 → 收回至初始位姿。

4. 通訊協定與硬體整合

- 使用 TCP socket 與機械手臂控制器（ESP32）通訊。
- 傳輸格式分為兩類：
  (1) 微調移動："ΔX ΔY"（相對 XY 位移）。
  (2) 夾取或放置："角度 積木編號"（旋轉角度、積木類型）。

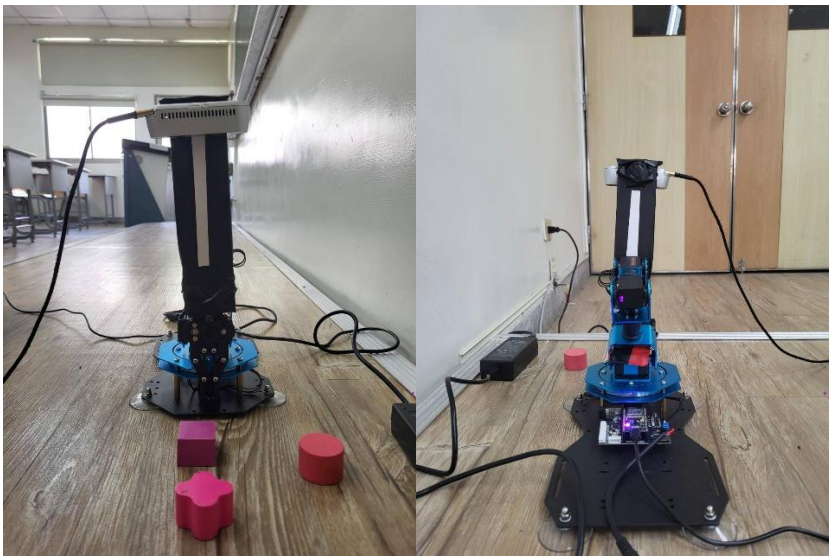
5. 模組整合特色與優化設計

- 模組清晰、職責分明：每個腳本處理一個核心任務，方便維護與測試。
- 追蹤精準、誤差補償機制：利用最遠兩點法與座標轉換強化空間定位。
- 自動補償迴圈：保證最終定位誤差小於2 ㎜，提高作業成功率。
- 可擴充設計：可隨需求替換模型與擴展更多辨識物件種類。

十一、 成果展示

1. 執行程式：
   影像端：conformity.py、all_to_tast3.py
   手臂端：boot3.py
2. 操作步驟：

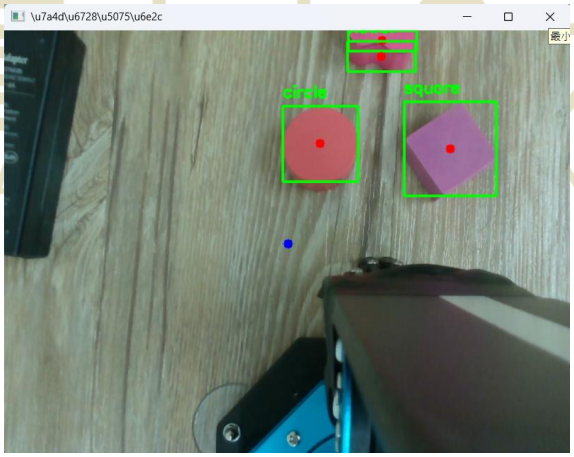
step1. 架設操作台，將 Intel RealSense D435i 相機固定於機械手臂上，使其能完整俯視夾取與放置區域，如圖四十一。



圖四十一、操作台

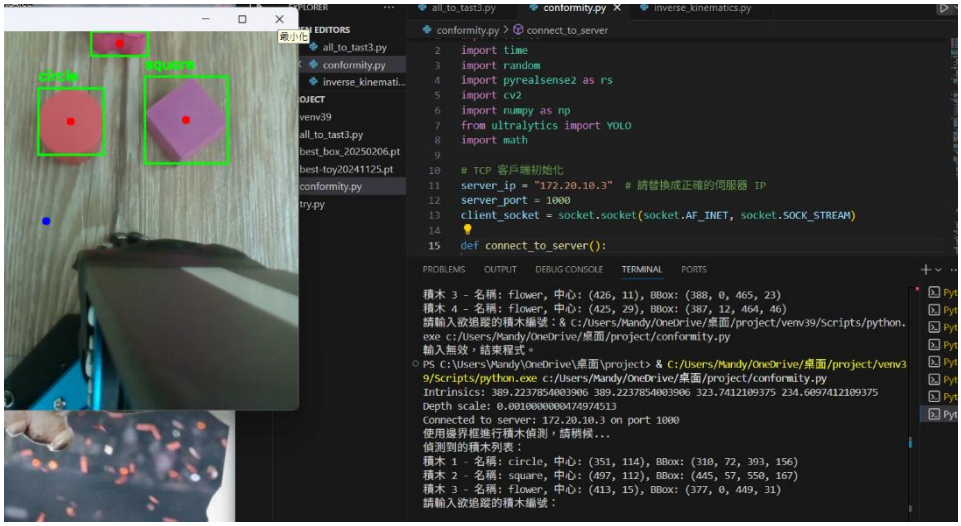
step2. 使用USB線連接pc與機械臂、攝影機，確保機械手臂與影像擷取功能皆已連線準備完畢。

step3. 啟動主程式conformity.py：系統自動啟動 RealSense 相機，鏡頭開啟後，開始偵測影像中的積木類型，如圖四十二。



圖四十二、偵測影像畫面

step4. 介面輸出畫面中所有的積木名稱以及邊界框，如圖四十三，使用者選擇欲辨識的模型類別(積木)，選擇後使用遮罩模型計算最遠兩點與中心位置，並推算物體角度，。



圖四十三、積木名稱以及邊界框

step5. 計算世界座標與相對座標誤差：系統根據內參與深度值進行像素→相機→世界座標轉換，並計算相對畫面中心的 XY 誤差。

step6. 若誤差大於閾值，啟動補償流程 all_to_test.py：系統執行逆向運動學補償，控制手臂向目標相對方向微調。微調後再次擷取影像重新驗證誤差。

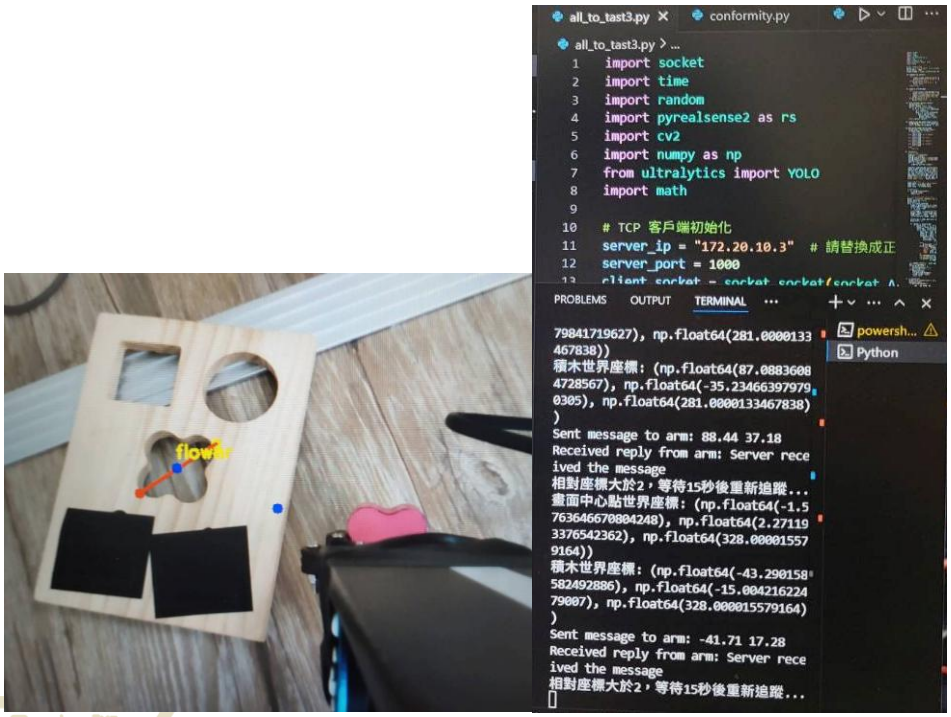
step7. 若誤差修正成功（小於 2mm），啟動夾取程序 all_to_testa.py：手臂依序執行：夾爪打開 → 移動至積木上方（Z=20） → 下壓至 Z=-10 完成夾取 → 移動回中繼點 (X=0, Y=90, Z=20) → 夾爪夾合，如圖四十四。

圖四十四、夾取程序

step8. 啟動放置流程 all_to_test3.py：系統重新使用 YOLOv11 模型辨識凹槽類別，並要求使用者輸入放置目標編號。

step9. 使用者選擇目標凹槽後，系統將使用遮罩模型計算最遠兩點與中心位置，

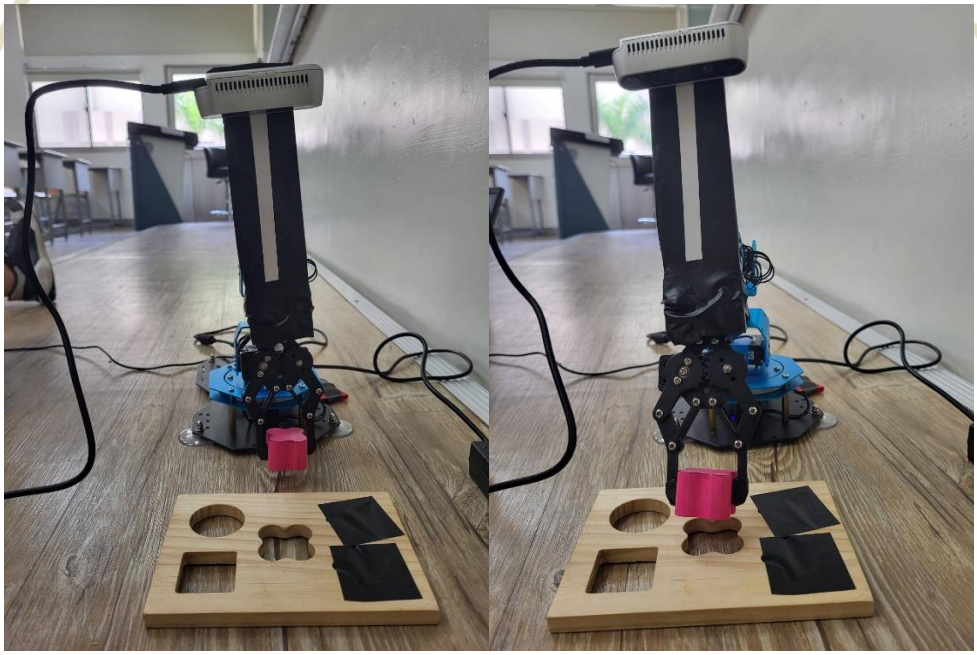
推算物體角度，並計算世界座標與相對座標誤差，如圖四十五。

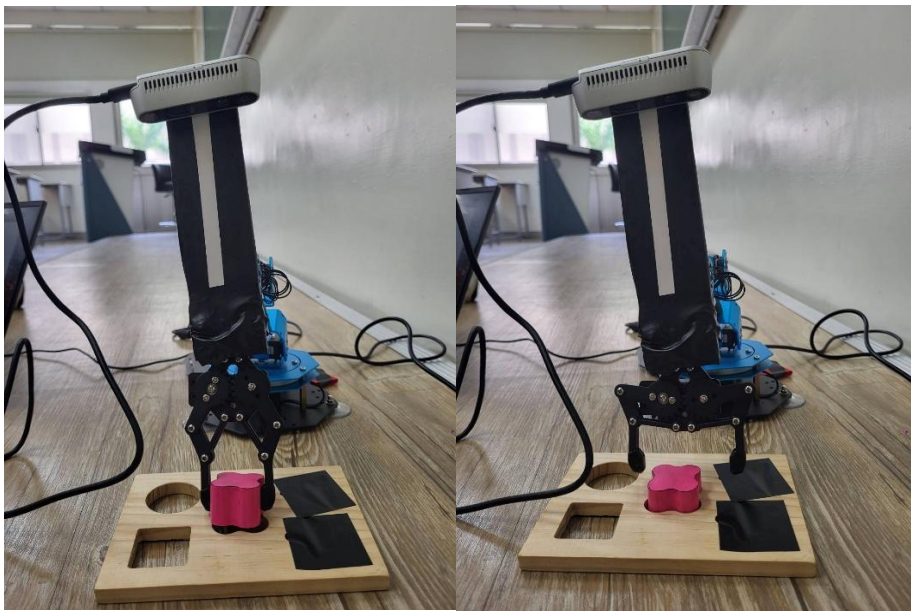


圖四十五、最遠兩點與中心位置

step10. 若誤差大於閾值，啟動 all_to_test.py 補償

step11. 若成功則執行放置流程：夾爪打開 → 下壓至 Z=-7 完成放置 → 返回中繼點（X=0, Y=90, Z=20） → 夾爪夾合，如圖四十六。





圖四十六、放置流程

step12. 若需進行下一筆作業可重新輸入積木與放置目標，否則按 q 結束程式

十二、 結論與未來展望

本專題整合YOLOv11深度學習模型與Intel Realsense D435i深度相機，實現基於三維視覺與運動規劃的機械手臂智能夾取系統。系統可即時辨識並夾取隨機擺放的五種不同形狀積木，並放置至對應凹槽，達成自動化之智能作業流程。

在正逆向運動學方面，本專題以幾何法結合解析法完成精準位置與角度推算，並實現世界座標轉換模組，有效將像素座標結合深度資訊轉換為可控制的真實三維空間座標。

然而，實作過程中亦發現多項限制與潛在風險。首先，在YOLOv11模型應用方面，雖經資料增強與精調訓練，實際準確率約落在73%左右，尚有優化空間。為避免模型誤判導致操作失誤，建議展示時應提前移除非目標積木，以減少干擾，並在模型首次偵測失敗時自動重啟辨識流程以提升成功率。

其次，部分硬體限制亦對系統穩定性造成影響，尤以馬達訊號過載與讀值異常為甚。在特定設備（如學姊所使用之機械臂）上尤為明顯，推測為內部記憶體溢位或封包阻塞導致訊號讀取失敗，後續建議深入檢視其訊號清除機制與錯誤復原策略，並加入過載保護與重試機制。

另外，D435i深度相機偶爾出現讀不到深度值的情況，建議可透過微調蜂巢板傾角、改變積木位置或導入邊界追蹤輔助修正，提高深度擷取成功率。同時也建議導入「夾取前動態追蹤」策略，在手臂靠近積木時同步修正座標與姿態，以減少誤差累積導

致夾取偏差。

針對目前 X 軸單次移動誤差較大的問題，建議未來可加入「回零校正」或「單點補正」機制，例如將機械臂先歸位至各軸值500（預設中心），再一次性移動至目標點以降低偏差，惟須注意此方式可能存在中途卡頓或路徑遮蔽風險，應進行安全評估後方能應用。

綜上所述，本專題已建構一個具備三維感知、即時辨識能力的智慧型夾取系統，展示機器視覺與運動控制整合之實用潛力。未來若能進一步導入多物件追蹤、自動誤差修正、模型優化訓練與更強韌的錯誤處理機制，將可提升系統整體效能與商用價值，朝向工業4.0與智慧製造之應用邁進。

十三、 參考資料

[1] Luhao He,Yongzhang Zhou ORCID,Lei LiuORCID & Jianhua Ma. (2024). Research and Application of YOLOv11-Based Object Segmentation in Intelligent Recognition at Construction Sites. Buildings. 14(12), 3777

[2] Alhassan Mumuni & Fuseini Mumuni. (2022). Data augmentation: A comprehensive survey of modern approaches. Array. 16, 100285

[3] Moa Eriksson, Elias Euler, Cedric Linder, Urban Eriksson & Nadaraj Govender. (2022) The Variation of University Physics Students' Experience of Plus and Minus Signs in 1D Vector-kinematics Revisited, African Journal of Research in Mathematics. Science and Technology Education. 26:1, 63-76, DOI: 10.1080/18117295.2022.2091327

[4] Yi-Xuan Ku, Hsin-Jui Kuo.（2025, January）.YOLOV8-object-detection-and-six-axis-robotic-arm-. Github.

[5] JM. Karthi, V Muthulakshmi, R Priscilla, P Praveen & K Vanisri. (2021). Evolution of YOLO-V5 Algorithm for Object Detection: Automated Detection of Library Books and Performace validation of Dataset. 2021 International Conference on Innovative Computing, Intelligent Communication and Smart

Electrical Systems (ICSES). DOI：10.1109/icses52305.2021.9633834

[6] John D. Kelleher, Brian Mac Namee & Aoife D'Arcy. (2015). Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies (1th ed.).

[7] AYann LeCun, Yoshua Bengio & Geoffrey Hinton. (2015). Deep learning. Nature volume 521, 436–444

[8] Laith Alzubaidi, Jinglan Zhang, Amjad J. Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, J. Santamaría, Mohammed A. Fadhel, Muthana Al-Amidie & Laith Farhan. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. Journal of Big Data. 8, 53

[9] Rahima Khanam, Muhammad Hussain. (2024). YOLOv11: An Overview of the Key Architectural Enhancements. arXiv. 2410, 17725

[10] Qiang Liu, Xinwei Wang, Fengli Liu and Li Mu. (2024). Research and Implementation of Robotic Arm Positioning and Grasping Based on Visual Guidance. Journal of Physics: Conference Series 2785 (2024) 012017. doi:10.1088/1742-6596/2785/1/012017

[11] Kang Liao, Lang Nie, Shujuan Huang, Chunyu Lin, Jing Zhang, Yao Zhao, Moncef Gabbouj & Dacheng Tao. (2023). Deep Learning for Camera Calibration and Beyond: A Survey. arXiv. 2303,10559

[12] J. A. Medrano-Hermosillo, R. Lozoya-Ponce, J. Ramírez-Quintana & R. Baray-Arana. Forward Kinematics Analysis of 6-DoF Articulated Robot using Screw Theory and Geometric Algebra. 2022 XXIV Robotics Mexican Congress (COMRob). Mineral de la Reforma/State of Hidalgo, Mexico, 2022, pp. 1-6, doi: 10.1109/COMRob57154.2022.9962312.

[13] A. D'Souza, S. Vijayakumar & S. Schaal. Learning inverse kinematics. Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and

Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180), Maui, HI, USA, 2001, pp. 298-303 vol.1, doi: 10.1109/IROS.2001.973374.

十四、 附件(程式碼)

本專題所使用的程式碼可掃描以下QRcode獲取。