National Sun Yat-sen University

Department of Electrical Engineering

INDEPENDENT STUDIES IN COMPUTER PRACTICE

計算機實作專題

期末成果報告

主題：

電池自動化分類機械手臂系統開發

Development of an Automated Battery Sorting Robotic Arm System

組員：

B103015006 胡庭翊　　**B103025010 盧奕睿**

B103015001 林佳明　　**B103015018 劉姵妤**

中華民國　一百一十三　年　六　月

電池自動化分類機械手臂系統開發

學生：胡庭翊、盧奕睿、劉姵妤、林佳明

摘要

摘要內文：

本專題希望透過電池分類系統的開發，結合機械手臂完成將廢電池夾取、自動化分類之功能。我們在既有的 YOLO 技術架構下將目標種類之電池標註、訓練，實行物件偵測，接著再透過 eye-to-hand 相機分辨電池。相機取得目標物資訊後，會再經由校正法將鏡頭拍攝到的圖像座標計算轉換成世界座標，最後透過 ROS 將座標傳送給機器手臂，完成夾取與放置於特定位置的動作。

關鍵詞：
YOLO、物件辨識、機器學習、分類、校正、電池、機械手臂、自動化

目錄

圖目錄

壹、前言

一、研究動機

現今的科技時代環境，工廠自動化與機器人技術快速發展，如何讓機器人可以將物品分類並可以準確地偵測與抓取成為更加靈活的生產工具，以協助業者省下許多繁複的人工成本，並運用在取替人類直接接觸有毒、有害的危險工作，增進人們工作安全性，亦是機器手臂協作發展的重要課題。

鑒於電池是工業發展與生活中相當普用的資材，它既具生產運用價值，但是廢棄電池也有汙染毒害問題，因此，本專題選擇各式常用電池作為訓練機器手臂夾取的材料，透過模型訓練機械手臂做電池的辨識與抓取，希望能為工廠自動化發展有所助益，也對維護環境工作安全提供貢獻。

二、研究問題

針對 TM Robot TM5-900 之電池夾取任務，利用 YOLOv5-obb (Oriented Bounding Boxes)演算法訓練辨別電池四端點及電池類別之模型，並利用 ROS 整合下列節點:

1. eye-to-hand 相機之即時相片資料
2. 前述訓練之模型及其回傳之四端點座標和電池類別
3. 電池圖片藉由機器學習模型辨識之像素座標與世界座標的轉換及標定邏輯
4. 手臂各節點之行為模式。

整合 TM Robot 公司提供之 ROS 驅動程式，可獲取所需資料或指令。利用驅動程式提供之指令集，蒐集相機擷取之圖片資料，並把圖片傳入已訓練之模型，獲取電池四端點座標集類別；模型回傳之電池四端點像素座標可藉由校正求得之內參和外參轉換成驅動程式能讀取之世界座標；最後利用驅動程式內建之指令集使手臂自預設位置移動至前述求得之電池位置，並完成夾取任務。本研究主要針對以下兩部分進行實作及最佳化探討:電池四端點像素座標及類別辨識模型訓練、像素座標轉換世界座標之校正法和內外參計算。

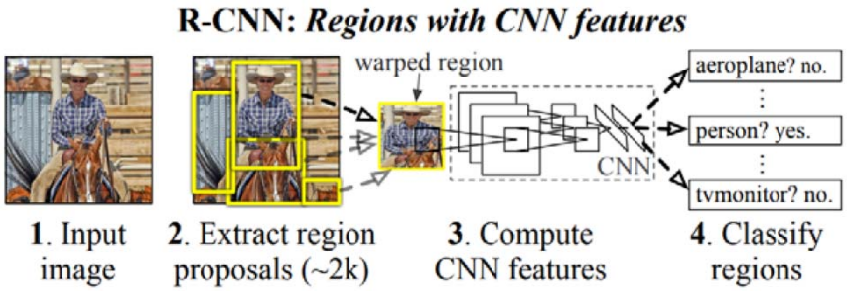
貳、正文

一、文獻回顧與探討

為達成本專題的目標，應先認識使用的工具，若能參考其他文獻與資料認識其背後工作原理，定能使計畫更具意義，執行起來更順遂。

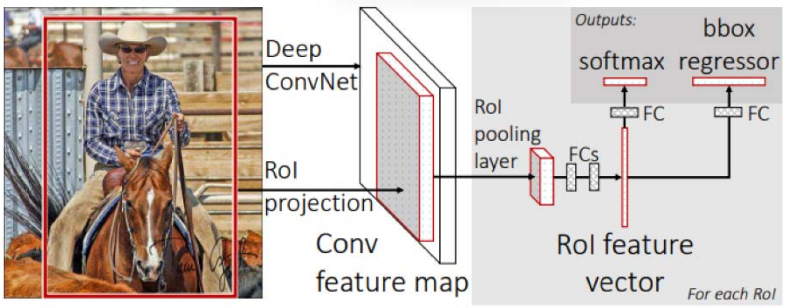
物件偵測是指在照片或影片等圖像內容中，以「框」標出有興趣的目標物件位置，並針對該物件進行分類並猜測機率。此技術可應用於工廠瑕疵檢測、人臉五官定位、醫療影像分析等各種智慧應用中。早期的物件偵測演算法以 two stage 為大宗，需要先用演算法生成樣本的候選框，再透過卷積神經網路做影像辨識，例如 R-CNN 便是 two stage 演算法的代表之一。

R-CNN [1]的運算方式為，在輸入的圖片中以 Selective Search 的方式先取出約 2 千個候選區域（region proposals），Selective Search 的理論概念是使用圖像分割（segmentation）後的結果，計算影像中所有相臨區域之間的相似性，然後把相似性高的區域合併起來。接著將這些區域使用卷積神經網路求得特徵集合（feature vector），再將其各自的特徵向量再丟入 SVM 分類器中去分類，並將候選區域送進 Bounding Box Regressor 中迴歸與 ground truth 的差異量，判斷當前區域所對應的實物是 background 還是所對應的物體類別，最後利用非極大抑制（non-maximal suppression）獲得最終的 bounding-box。



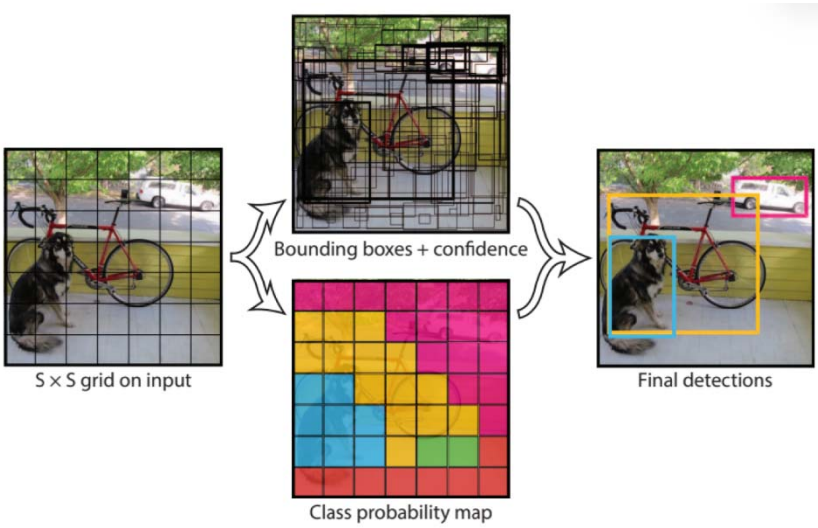
圖一、R-CNN 運作架構[1]

然而，R-CNN 將模型的訓練分成許多階段的方式非常沒有效率，再者在訓練時還需先把所有候選區域的特徵集合都存到硬碟上，不僅佔用硬體資源，也耗時許多，於是便有了 Fast R-CNN[2]與 Faster R-CNN[3] 等模型的出現。Fast R-CNN 與 Faster R-CNN 致力於利用運算共享以及神經網路來代替 Selective Search 找出候選區域，加速 R-CNN 上。Fast R-CNN 改變了候選區域的提取邏輯，將原本繁瑣的提取改為只對整張圖像全區域進行一次特徵提取，映射到卷積神經網路最後的 feature map 上才切出真正要拿來使用的 ROI(Region Of Interest)，因此加快了執行的速度；而 Faster R-CNN 則運用 RPN(Region Proposals Network)搭配 anchor box 的設計提前定義區域，使執行速度有更明顯的提升。



圖二、Fast R-CNN 運作架構[2]

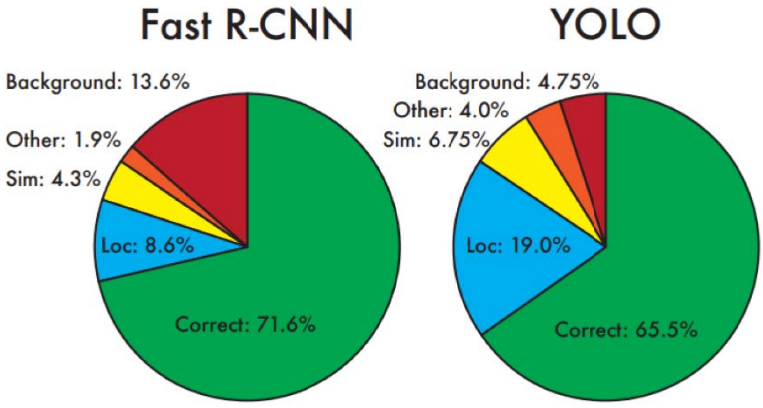
在 2016 年，J. Redmon 等人發表了一篇關於 YOLO，一個嶄新的物件偵測方式的論文[4]。與以往使用分類器進行的 two stage 物件偵測演算法不同，YOLO 將物件偵測視為一個迴歸任務（regression problem），在空間中分割出邊界框（Bounding Box）並計算出其類別的機率（associate class probability）。YOLO 僅需要一個卷積神經網路進行一次計算便能同時進行多邊界框與其類別機率的預測，其運算方式為將輸入圖像內容切成 s×s 個單位網格，再同時進行在每個網格內預測邊界框和各自的信賴指數（confidence score）以及針對每個網格進行類別機率預測，最後利用非極大抑制取得最終的偵測結果。



圖三、YOLO 運作架構[4]

與 two stage 物件偵測需要先用演算法生成樣本的候選框，再透過卷積神經網路做影像辨識不同，YOLO 為 one stage 的物件偵測，物件類別與位置偵測以及物件辨識一步到位，雖然精準度會略低於如 R-CNN 與其變體的 two stage 演算法，但 YOLO 提升的速度以及其降低背景誤判機率的特性，在現實考量上非常划算，亦更適合本專題的需求。再加上 YOLO 是開源且普遍被使用的模型，本身非常容易打造並且可以直接在整張圖像上訓練，其便成為了此專題中物件偵測

系統的理想選擇。



圖四、與 Fast R-CNN 相比，YOLO 大幅降低背景誤判的機率[4]

為了捕捉工作平台上的物件進行偵測，本計畫欲使用眼在外（eye-to-hand）方法，也就是將相機固定於機器手臂外，再透過校正取得機器人座標與相機座標之間的關係。在 1999 年的一篇論文[5]提出一種利用平面的棋盤方格進行校正的方法，接著在 2000 年，同一作者又發布了一篇論文[6]將此方法完整化，變作實用的嶄新校正方法。此校正法首先需採集以相機拍攝的二維方格棋盤各個位姿之圖片，以計算求得相機本身的內參（Intrinsic），也就是相機內部的各個參數，接著再放置校正用的棋盤方格於工作平面上並拍攝一張照片，透過照片中校正棋盤的位姿計算求得相機的外參（Extrinsic），也就是相機本身擺放相對於世界座標系旋轉平移的向量。

世界座標系為使用者定義的三維世界座標系，用以描述目標物在真實世界的位置，相機座標系（Camera coordinate system）為以相機為原點的座標系，幫助溝通世界座標系與圖像及像素座標系，兩者的單位皆為公尺(m)。圖像座標系（Image coordinate system）和像素座標系（Pixel coordinate system）都成像在平面上，只是各自的原點和單位不同：圖像座標系的原點在相機光軸與成像平面的焦點，單位為公釐(mm)；而像素座標系的原點在圖像左上角，像點（0，0）處，單位為pixel。世界座標系可以透過乘上外參矩陣與內參矩陣得出其像素座標系，同理，藉由將像素座標乘上內參的反矩陣與外參的反矩陣，我們便可得知其在世界座標系中的座標。最後再將此座標系平移為以機器人為中心的座標系，便能提供機器手臂夾取目標物的座標。這種方法的優點在於操作簡單，而且精度較高，可以滿足大部分場合。就算手臂與攝影機的相對位置改變了，重新校正所需的時間也不需要太長，而且操作上的難度不高，就算沒有基礎的人也能輕鬆完成。如果是攝影機移動了，那只需要重新拍一張棋盤校正板的照片就可以運作了。如果是手臂移動了，那也只需要重新偵測手臂原點與棋盤校正板之間的距離就能繼續運作。

求得目標物座標後，為了實現軟硬體之間的溝通，我們便需要使用到 Robot

Operation System (ROS)。ROS 是一個專為機器人軟體開發所設計出來的開源作業系統框架[7][8]，為一種中介軟體提供類似作業系統的功能。ROS 可視作一個軟體框架，提供軟硬體溝通的基礎架構，並且也支援多種程式語言，降低了機器人的開發門檻，便於開發者專注於演算法與應用的研究。同時它也擁有眾多強力的開發工具，其功能模組都被封裝於獨立套件包（package）中，便於開發者共享，這使得ROS得以有效簡化開發機器人平台的複雜任務建立與穩定版本控制。另外 ROS 也整合了許多第三方常見開發工具，幫助開發者快速完成機器人應用的建立、設計與多機整合。

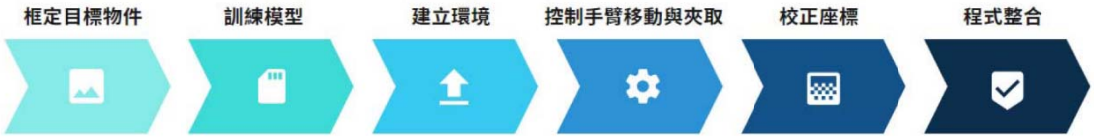
ROS 主要依靠中心化 P2P（Centralised Peer to peer）實作。中心化 P2P 的概念為使用一個中心服務器（Server）連結其他節點（Node）的訊息資料並對請求做出回應；而節點負責處理與發布訊息資料，具有 index 便於找到絕對地址。在 P2P 架構中，節點指的是程序（Process），一個機器人控制系統通常包含多個節點；Master 是一個特殊的節點，其作為 Server，負責查找節點並傳遞資料與溝通；訊息（Message）是節點之間傳遞的資料，為一種資料結構；Topic 是 message 發布時的標籤；發佈者（Publisher）為發布消息的節點，而訂閱者（Subscriber）則為接收消息的節點，節點會透過 topic 找尋需要接收的訊息，同一種 topic 的訊息可以由不同的節點發布或接收，因此存在複數個發佈者和訂閱者，或是一個節點同時為發佈者與訂閱者都是有可能的；套件包（package）為一個 ROS 軟體的基本單位，其包含許多節點與相關函式庫、資料集（data set）、配置文件（configuration file）或其他和專案有關的檔案。透過 ROS 我們便能實作機器人的操作。

而在實作方面，本專題使用實驗室內的達明機器人公司的 TM5 常規負載機器人[9]作為實作。

圖五、TM5 常規負載機器人[9]

二、研究方法及步驟

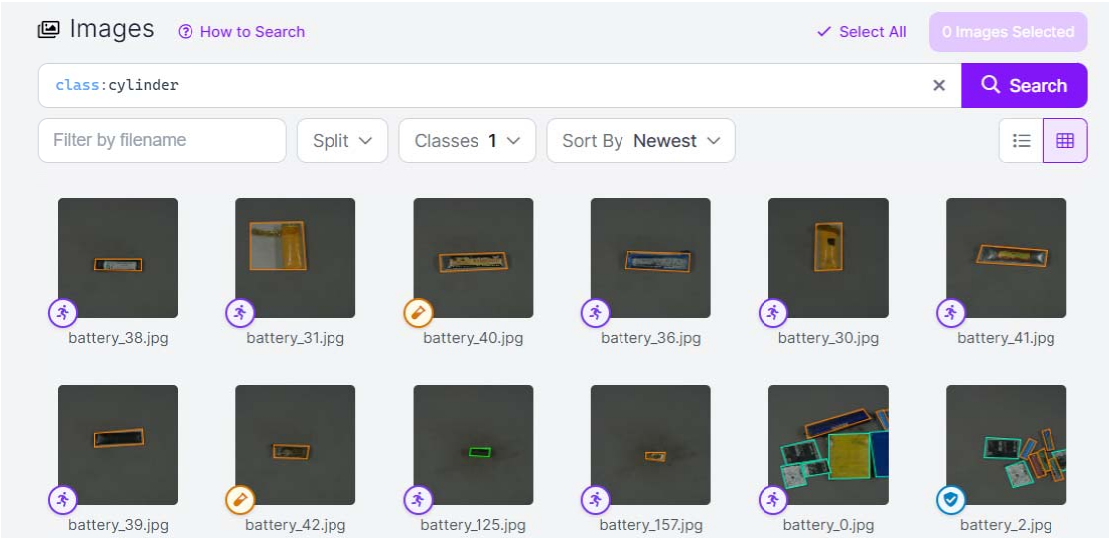
本專題的步驟流程如圖六所示，主要分為「框定目標物件」、「訓練模型」、「建立環境」、「控制手臂移動與夾取」、「校正座標」、「程式整合」。以下將對各流程一一介紹。



圖六、計畫流程圖

(一) 框定目標物件

首先，為了使模型可以較精確的辨識電池，我們蒐集了很多從廢電池回收廠拍攝的廢電池影像，利用線上影像管理工具 Roboflow 進行圖片框定與標示。在框定電池的時候，採用多邊形的標記框來使畫框旋轉，貼合電池影像的邊緣以達到更好的電池辨識效果，並且依據 circle、cylinder、package、phone、square 共五種類別進行標註完成人工分類。當所有的圖片都標註完成，並且確定各類別的數量分布平均，再加上諸如亮度調整或角度旋轉等 data augmentation 之後，我們就可以把影像匯集成資料集並選擇以 YOLOv5 Oriented Bounding Box 的形式匯出資料用以進行模型訓練。



圖七、利用 roboflow 標記的電池資料集

(二) 訓練模型

環境建立我們採用[10]docs 檔案中 install.md 的作法。首先我們需要先建立一個虛擬環境並啟用它，使後續的操作如安裝套件可以在這個環境當中進行。

1. 建立虛擬環境的指令:

$ conda create –n Py39_Torch1.10_cu11.3 python=3.9 -y

Py39_Torch1.10_cu11.3 是為這個新建立的虛擬環境取的名字

2. 啟用該環境指令:

$ source activate Py39_Torch1.10_cu11.3

接著就可以安裝 Pytorch 和 torchvision 套件並檢查版本，使用指令:

$ pip3 install torch==1.10.1+cu113 torchvision==0.11.2+cu113 torchaudio==0.10.1+cu113 -f https://download.pytorch.org/whl/cu113/torch_stable.html

我們也可以利用指令 **nvcc -V** 來查看 cuda 的版本:

$ nvcc -V
$ python
>>> import torch
>>> torch.cuda.is_available
>>> exit()

在新創的自訂資料夾中複製儲存庫:

$ git clone https://github.com/hukaixuan19970627/yolov5_obb.git

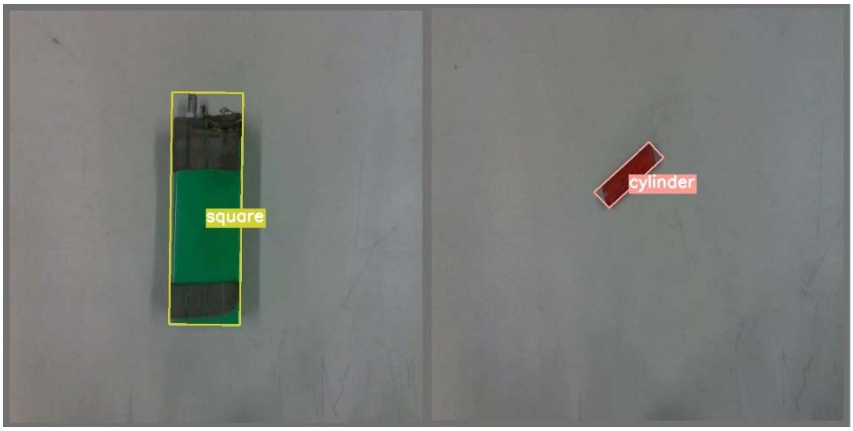
進入剛剛生成的 yolov5_obb 資料夾:

$ cd yolov5_obb

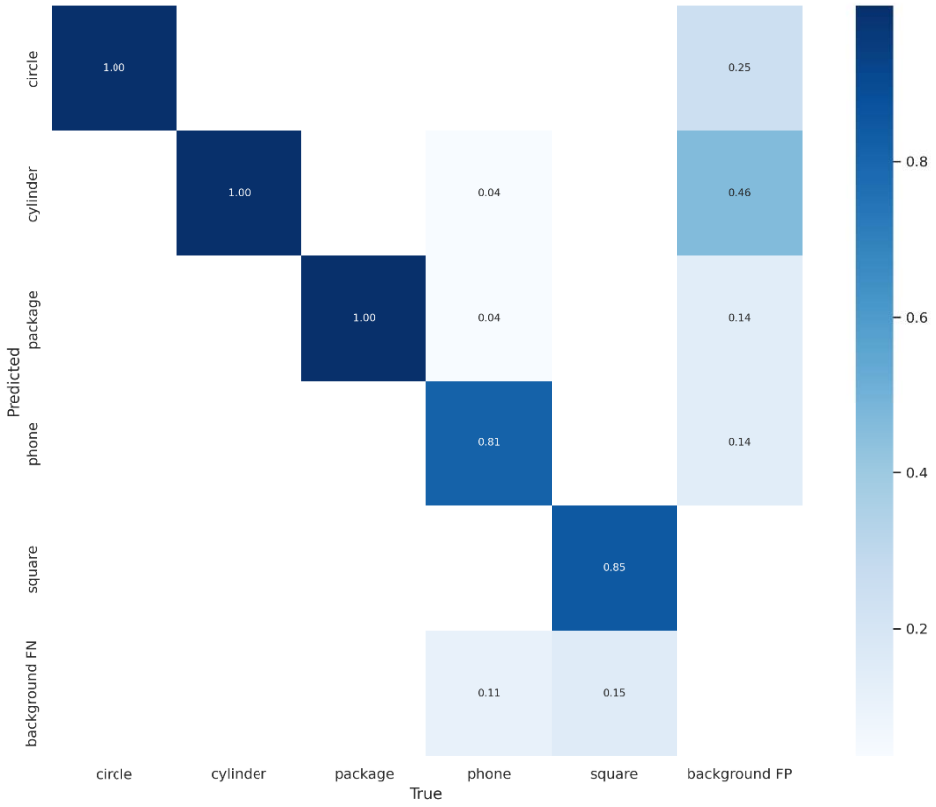
安裝 yolov5_obb:

$ pip install -r requirements.txt
$ cd utils/nms_rotated
$ python setup.py develop

使用前述完成框定標示之dataset，利用 yolov5-obb[10]演算法，調整諸如 batch 大小等超參數後執行 train.py 之模型訓練函式，生成 best.pt 之模型，即為後續用於辨識電池四端點像素座標及類別之權重。yolov5-obb 也提供 detect.py 函式，使使用者可以利用先前建立之權重，並給予新的圖片源，得以辨識當前圖片由權重中定義的邏輯預測之結果，並利用此結果資料結合自定義之函數，回傳所需資料。本實作於 detect.py 檔中，寫入重心計算、長短邊計算、像素座標轉換世界座標之公式，便可利用由權重預測出的電池四端點像素座標後，套入公式計算得知所需資訊。



圖八、利用 best.pt 模型辨識 test data 資料集中的電池圖片成果



圖九、所訓練出之 best.pt 權重的成果分析(confusion matrix)

由圖九可發現，所訓練出之權重在當給予之電池圖片類別為 circle, cylinder, package 時，皆有 100%的正確率，而當電池類別為 phone 和 square 時則有將近 15%-19%的機率會辨識為其他類別。

(三) 建立環境(ROS workspace and configuration setup)

由於此專題中存在手臂動作控制、手臂內建相機資料處理及 yolo 模型和手臂原廠提供之 ROS driver 之間的溝通，我們利用 ROS 完成之間的整合任務。

首先，由於接下來主要程式新增與改動會在 detect.py 檔案裡修改，因此必須建置一個 ROS workspace 並於原始碼中加入 yolov5_obb, tm_ros_driver[11] package, 後續的座標轉換、標定、長短邊與重心計算邏輯將加至 yolov5_obb > decect.py 中，並於標頭加入:

$ import rospy

$ from tm_msgs.msg import *

$ from tm_msgs.srv import *

令此檔案成為 ROS master 能夠辨識的節點(node)，並有權限取用 tm_ros_driver 中的發布者(publisher)所發布的所有資料型態之主題(topic)內容訊息(message)(例:機器手臂及時狀態、eye-to-hand 相機取得之圖片)和 server 所發布的 service。

(四) 控制手臂移動與夾取

使用程式控制手臂移動位置與夾爪夾取。先藉由 YOLO 即時偵測辨識出目標物的外型框架後，將框架四個邊角的座標值計算求得平均，即可獲得目標物的中心點座標，也就是協作機器人的夾具需抵達的目標位置。至於夾具之角度則會透過計算外型框架的長短邊求得，利用畢氏定理計算各點間的距離，即可判斷出長短邊結果。最後，將夾具移動至中心點座標並將夾爪兩側對齊框架之長邊即為夾取點。

得知夾取點後，本專題使用 ROS 為平台，傳輸目標資訊給協作機器人。ROS 提供了兩種網路通訊方式，分別為 TCP（Transmission Control Protocol）協議與 UDP（User Datagram Protocol）協議。UDP 的優點是速度快，但數據是否會被接收以及數據傳輸的順序無法被保證，相對的，雖然 TCP 速度較慢，但他具有安全可靠的優點，也能夠檢查與糾正錯誤，使數據確實且按照順序的被接收。為確保數據傳輸的準確性，本專題使用 TM ROS Driver[11]上的範例程式碼「send_script」，透過 TCP 協議進行數據的傳輸，藉由傳送座標控制手臂移動到目標位置。

當程式被呼叫時，會先建立一個傳輸座標用的節點（node），然後當發現傳輸座標的服務（service）後，創建一個服務客戶端（client）請求服務調用並輸入請求數據，請求的數據為欲移動至的座標。接著數據會透過 TM ROS Driver 內的 Service Server 提供的運動指令，對手臂接收端節點「Listen node」下指令移動到指定位置，而若過程中出現問題，便將問題回報列印並終止程式運作。

控制手臂夾爪夾取的方式與移動手臂相仿，參考 TM ROS Driver 上的範例程式碼「set_io」，對 Listen node 發送數位或類比輸出值的指令。當程式被呼叫時，先透過指令將控制手臂夾爪的端子（pin）設為低電位以打開夾爪，接著等待兩秒，再將該端子設為高電位闔上夾爪，同樣的，若過程出錯程式會將問題回報列印並終止運作。

(五) 校正座標

列印出方格長寬已知的棋盤方格，並選定相機從各個角度拍攝之，因為棋盤是二維的物體，所以不同角度的照片能獲得更豐富的座標訊息。其原理是將各棋盤角點列成未知數並利用不同照片中角點之間的關係進行最佳化的動作，得出我們所需的矩陣。因為主要的角點只有四個角落與中心點，所以理論上只需要 5 個不同角度的照片就能找出答案，但使用越多照片也就能解更多的角點，結果也就越精確。這些複雜的計算並不需要我們親自手算，參考 GitHub 上的程式碼[12]，計算求得該相機內參(圖十)。求得內參後將內參更新至程式中，再固定相機，拍攝同棋盤方格一起的工作平台，運用與求取內參同樣的方法求得外參。其中外部參數是由 3*3 的旋轉矩陣(rotation matrix)(圖十一)和 3*1 的變換矩陣(translation matrix)(圖十二)組合而成。將內參與外參相乘得到轉移矩陣 H，給予任一個待轉換的二維像素座標（u，v），其與世界座標系（x，y，z）之間的關係如圖十三：

$$A = \begin{bmatrix} \alpha & \gamma & u0 \\ 0 & \beta & v0 \\ 0 & 0 & 1 \end{bmatrix} \quad R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad T = \begin{bmatrix} \mathrm{T}_1 \\ \mathrm{T}_2 \\ \mathrm{T}_3 \end{bmatrix}$$

圖十、內部參數　　圖十一、旋轉矩陣　　圖十二、變換矩陣

$$\begin{pmatrix} \mathrm{x} \\ \mathrm{y} \\ 1 \end{pmatrix} = \mathrm{ZH}^{-1} \begin{pmatrix} \mathrm{u} \\ \mathrm{v} \\ 1 \end{pmatrix} = \mathrm{Z} \begin{bmatrix} \mathrm{H}_{11} & \mathrm{H}_{12} & \mathrm{H}_{13} \\ \mathrm{H}_{21} & \mathrm{H}_{22} & \mathrm{H}_{23} \\ \mathrm{H}_{31} & \mathrm{H}_{32} & \mathrm{H}_{33} \end{bmatrix} \begin{pmatrix} \mathrm{u} \\ \mathrm{v} \\ \mathrm{y} \end{pmatrix}$$

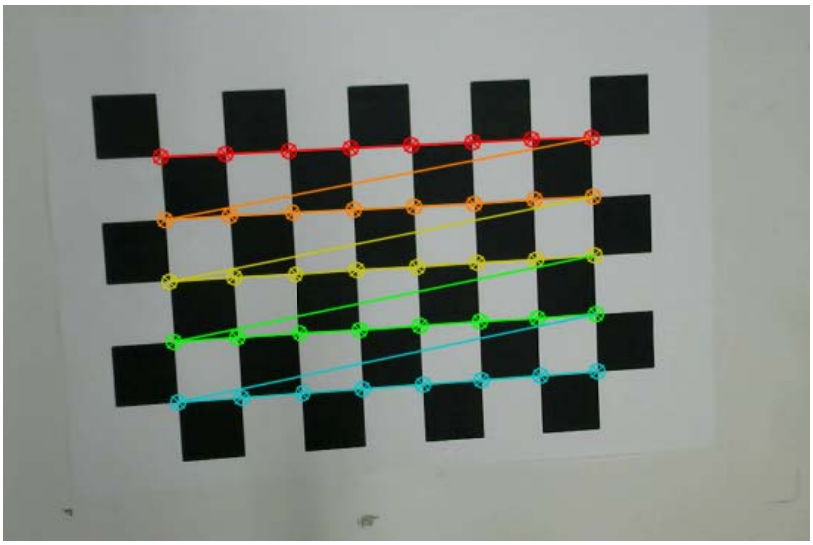
圖十三、從像素座標轉換至世界座標

其中，Z 為尺度因子，可透過以下式子將之消去：

$$\begin{cases} x = \dfrac{H_{11}\mathrm{u} + \mathrm{H}_{12}\mathrm{v} + \mathrm{H}_{13}}{\mathrm{H}_{31}\mathrm{u} + \mathrm{H}_{32}\mathrm{v} + \mathrm{H}_{33}} \\ \mathrm{y} = \dfrac{\mathrm{H}_{21}\mathrm{u} + \mathrm{H}_{22}\mathrm{v} + \mathrm{H}_{23}}{\mathrm{H}_{31}\mathrm{u} + \mathrm{H}_{32}\mathrm{v} + \mathrm{H}_{33}} \end{cases}$$

圖十四、消去尺度因子

最後求得的便是校正後的真實世界座標，將此座標單位轉換並加上手臂本身到像素座標（0，0）處所需的平移量，即為手臂待移動至的目標座標。



圖十五、棋盤校正示意圖

(六) 程式整合與改善

參考 YOLO 的物件偵測程式「detect.py」[13]，將之修改加入上述控制手臂的程式並整合，測試執行，執行成功再於 TM5 機器人上實際操作。

機器手臂的操作流程圖十三，先開啟手臂，接著在終端（terminal）進入 ROS 的資料夾啟動 roscore，再開啟一個終端連結到手臂的 IP 位址，然後打開達明機器人提供的 GUI（Graphical User Interface）介面確認手臂的連線狀況，最後進入虛擬環境中執行整合的程式。

圖十六、系統執行流程

參、結論

一、研究結果

在辨識電池種類與外框上我們的系統理論上已經能很好的完成，並且計算出目標物的中心座標與夾取角度，然而經過測試，尚有一些類別判斷錯誤的問題，例如 cylinder 會被辨識成 square，以及桌上污漬、雜物會被判斷成電池。我們覺得或許在資料集的採用、data augmentation，乃至於到超參數、信賴指數等可控參數以及 yolo 版本與影像預處理都是可嘗試調整的方向。

至於在校正方面，在預期中經過座標的校正與轉換後我們應該要能精確無比的夾取桌面上的電池，但是因為在參數的取得上存在不穩定性，導致了計算出來的座標也是不穩定的，再者，當我們使用不同設備進行運作的時候誤差也不盡相同，雖然我們在寒假的時候有將手臂的功能完善，校正後的手臂也能準確無翻動

的夾取目標物，然而我們的方法重現後存在誤差並難以再現。因此雖然目前在 try and error 後能夠勉強夾取到目標物，但是我們不認為這符合我們的期待，因此我們必須首先必須要改進我們獲取參數的方法並增加其穩定性，或者更根本的改換演算法，如此一來才能使我們的系統在實務上更具實用性。

二、未來展望

透過本次的計算機專題，我們對機器學習領悟有了實務上的理解和經驗，雖然是最入門的 YOLO 演算法，但從建立虛擬機、安裝 torch、conda 一步一步慢慢摸索的過程中，也發現計算機工程更廣闊的應用，希望日後能在此領域有更多的學習與實作經驗。原本在題目發想的過程中，計畫訓練判斷電池夾取點的模型作為手臂夾取邏輯的依據，但持續更換嘗試訓練出來的成效依舊不佳，於是我們最後保留成效良好的初始模型，並利用四端點的像素座標計算得出所需資訊，在嘗試又失敗的的過程中找到折衷且最有效益的方式。

至於座標校正與夾取，我們深刻的體會到理論與實際應用上的差別，尤其是硬體設備上的影響，不同的鏡頭、手臂所得出的結果都不盡相同，也時常有預期外的錯誤發生，當執行發生錯誤時，對於找尋錯誤的源頭是個艱難的過程。在理論上正確的情況下發生錯誤，也讓我們去思考實際上有可能遇到問題的地方並嘗試解決，或者作出細微的調整。這也讓我們累積起相關的經驗，當我們以後遇到相同的問題時能夠更有效率的解決，另外，操作機械手臂於我們而言是一個非常難得可貴的經驗，雖然對 ROS 的功能與操作尚未熟練，但經過此次專題，我們也看到了機械手臂可以延伸研究的諸多可能性。

肆、參考資料

[1] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 2014, pp. 580-587, doi: 10.1109/CVPR.2014.81. keywords: {Proposals;Feature extraction;Training;Visualization;Object detection;Vectors;Support vector machines},

[2] R. Girshick, "Fast R-CNN," 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 2015, pp. 1440-1448, doi: 10.1109/ICCV.2015.169. keywords: {Training;Proposals;Feature extraction;Object detection;Pipelines;Computer architecture;Open source software},

[3] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137-1149, 1 June 2017, doi: 10.1109/TPAMI.2016.2577031.
keywords: {Proposals;Object detection;Convolutional codes;Feature extraction;Search problems;Detectors;Training;Object detection;region proposal;convolutional neural network},

[4] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 779-788, doi: 10.1109/CVPR.2016.91. keywords: {Computer architecture;Microprocessors;Object detection;Training;Real-time systems;Neural networks;Pipelines},

[5] Zhengyou Zhang, "Flexible camera calibration by viewing a plane from unknown orientations," Proceedings of the Seventh IEEE International Conference on Computer Vision, Kerkyra, Greece, 1999, pp. 666-673 vol.1, doi: 10.1109/ICCV.1999.791289. keywords: {Cameras;Calibration;Computer vision;Layout;Lenses;Nonlinear distortion;Computer simulation;Testing;Voltage control;Robustness},

[6] Z. Zhang, "A flexible new technique for camera calibration," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 11, pp. 1330-1334, Nov. 2000, doi: 10.1109/34.888718.
keywords: {Cameras;Calibration;Computer vision;Layout;Lenses;Nonlinear distortion;Closed-form solution;Maximum likelihood estimation;Computer simulation;Testing},

[7] Quigley, Morgan, et al. "ROS: an open-source Robot Operating System." ICRA workshop on open source software. Vol. 3. No. 3.2. 2009.

[8]ROS Wiki
http://wiki.ros.org/

[9]達明機器人公司
https://www.tm-robot.com/zh-hant/

[10]胡凱旋,Yolov5 for Oriented Object Detection
https://github.com/hukaixuan19970627/yolov5_obb

[11]TM ROS Driver
https://github.com/TechmanRobotInc/tmr_ros1/tree/melodic

[12] Calibration_ZhangZhengyou_Method
https://github.com/zhiyuanyou/Calibration-ZhangZhengyou-Method

[13] ultralytics/yolov5
https://github.com/ultralytics/yolov5/blob/master/detect.py