期末專題 Report

[UEE3713] 機器人系統與應用設計實作 2020

**學號:** 106030009  **姓名:**葉蓁 **日期:** 2020/12/25

1. **目的:**

本次實驗任務目標為：

1. 讓機器人精準定位到夾取物件的座標位置點
2. 機器人能用深度相機辨識並找出目標夾取物之位置
3. 機器人能準確夾取物件，並根據其顏色，將其分類放置到分類盒指定位置



1. **設計說明:**

首先概述一下整體程式流程：

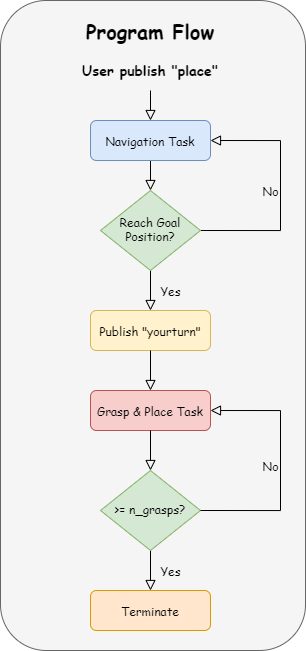
1. **Program Flow**

首先需要執行定位導航的機器人底盤運動控制，完成此任務後，機器人抵達定位點，發出 /yourturn Topic, 啟動 locobot.py 中的 callback function:

此時進入第二階段程式：夾取任務。

根據使用者(我們)輸入的參數: **--n\_grasps 決定執行幾次抓取任務。**

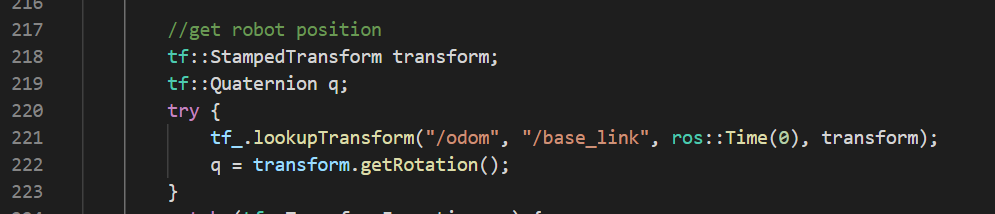
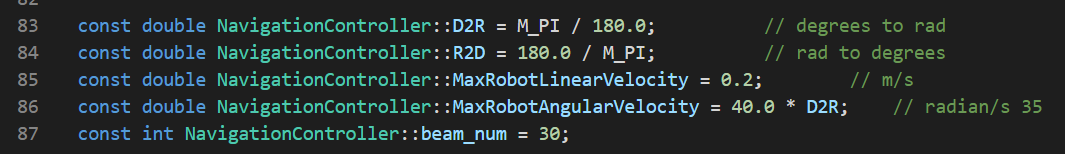
我們設定6次，若有抓取失敗，可以直接重新嘗試。



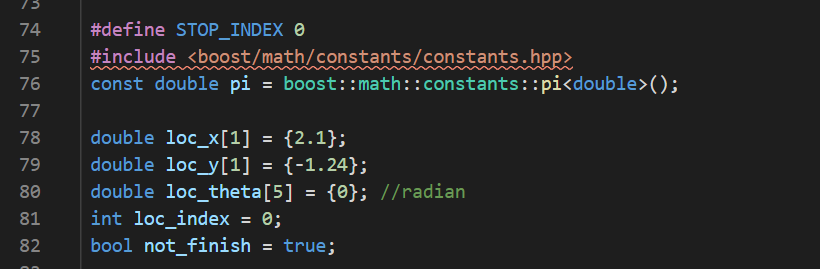
1. **Localization & Navigation**

底盤運動導航部分，我們使用 odometry 作為定位方式，仿照之前運動控制 assignment 01 的方式，透過位置點的傳輸，加上 ROS service 方式，將目標定位點傳給LoCoBot的navigation\_controller node。

navigation\_controller.cpp :

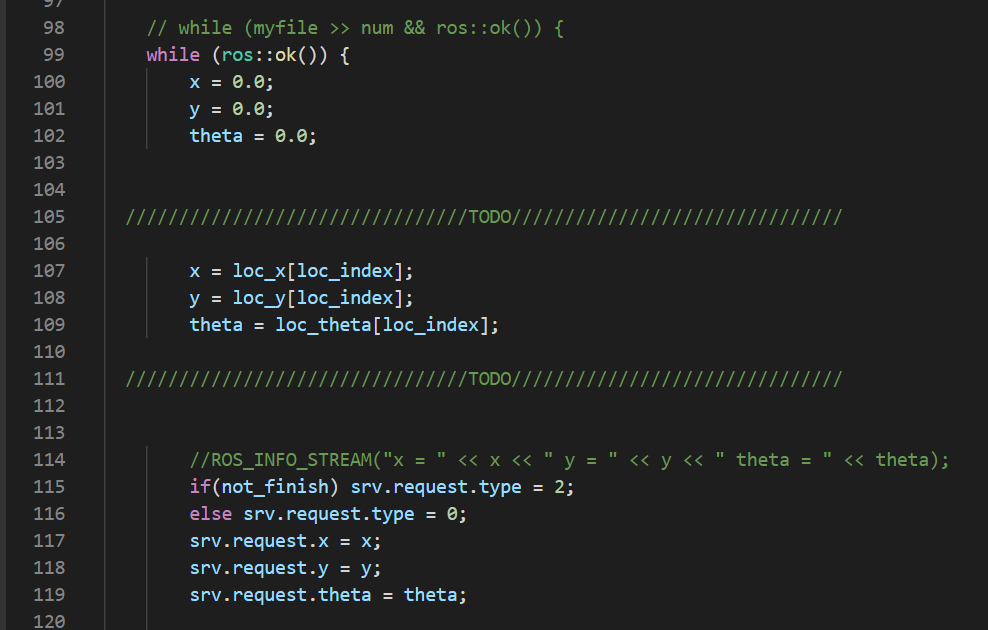
 

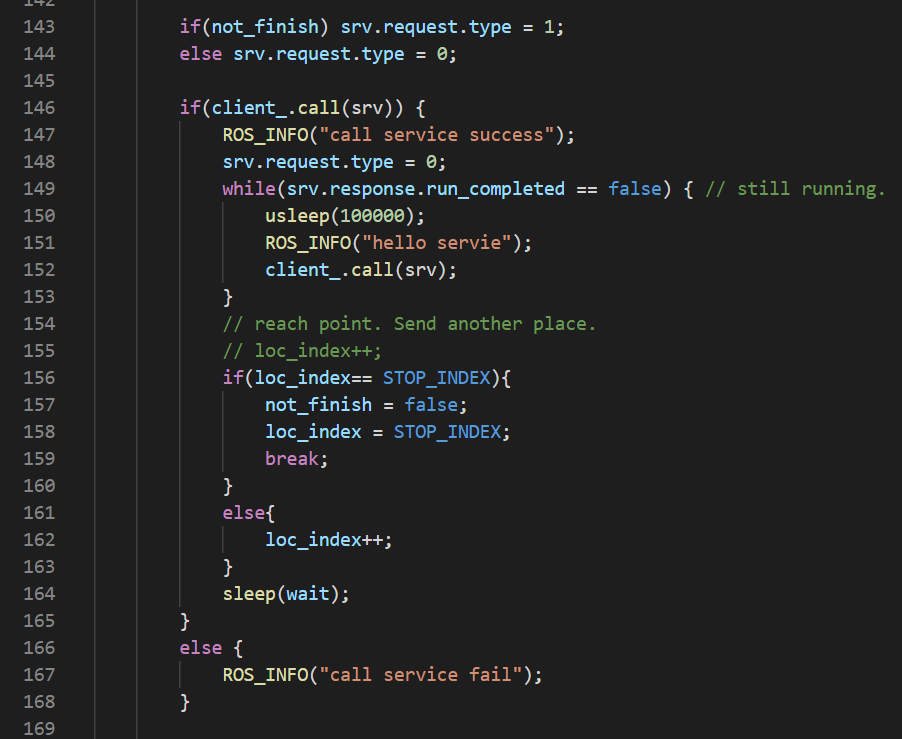
Command\_pub\_srv\_gui.cpp:



我們設定：**目標點 (x ,y ) = (2.1, -1.2) Theta (yaw) = 0.0 radian**

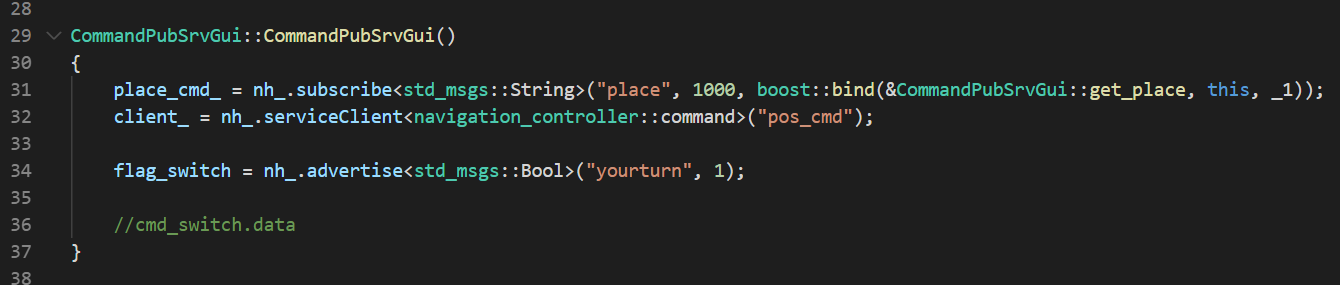
navigation\_controller node 會根據 odometry的計算，定位其位置，根據離目標點還有多遠，計算出當下需給馬達的速度命令，驅動馬達使機器人前進。



抵達定位點後，由 flag\_switch publish true到 /yourturn Topic，使locobot.py開始執行callback function。

下圖為：初始化 CommandPubSrvGui() 時其subscribe /place 並publish /yourturn的設定。 /place 就是我們使用者要 “開始”定位導航任務時，需要送的一個命令。(任意的 string 即可)





1. **Perception and Reasoning**

這步驟其實算是本次任務要能成功最關鍵的地方：

如何用影像辨識，找出物體的座標位置?

可惜我們沒有足夠的時間去研究更好的演算法，

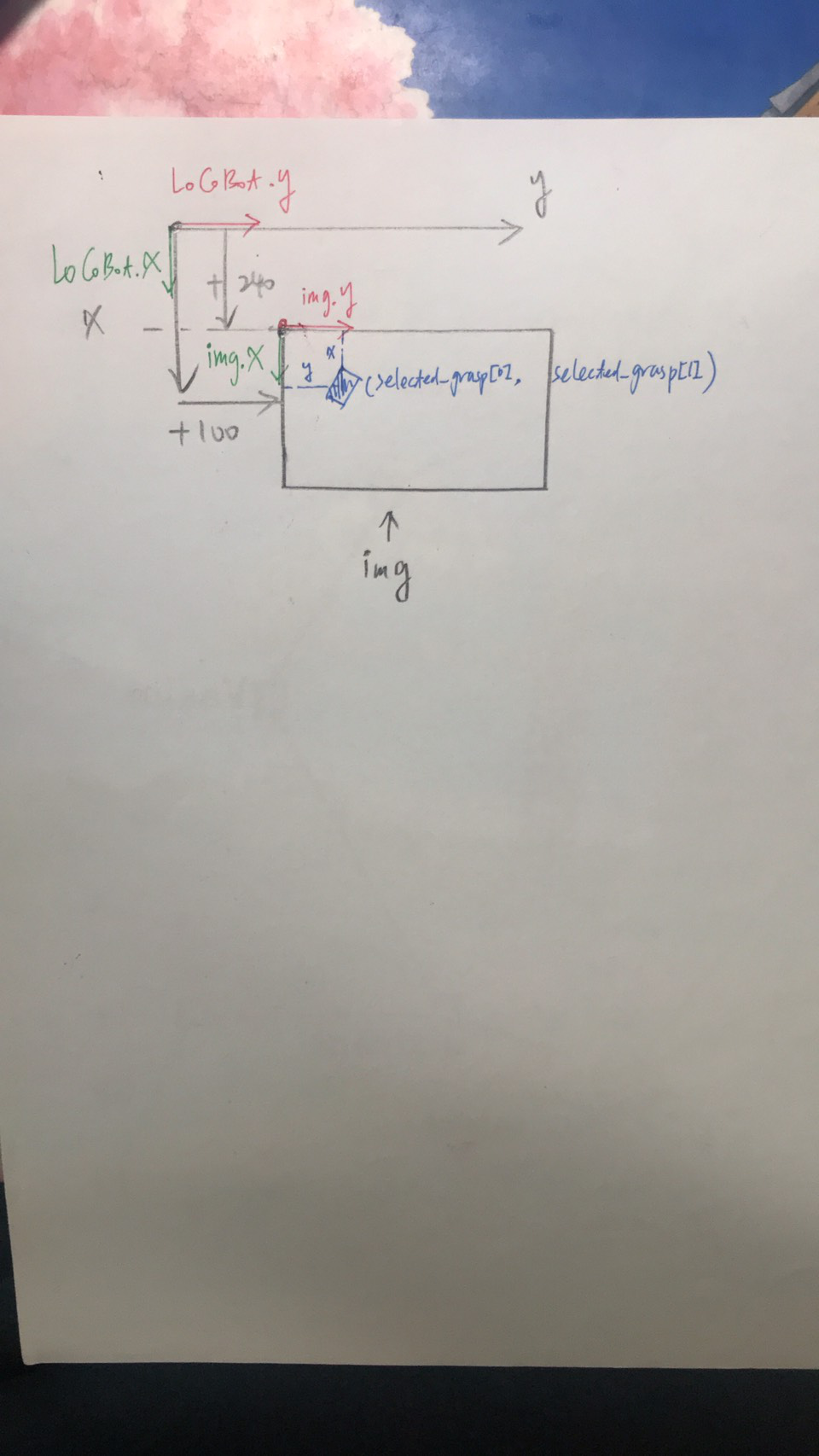
我們直接使用 助教範例提供的 compute\_grasp() 去 predict 出目前相片中他認為 特徵點最明顯的位置。

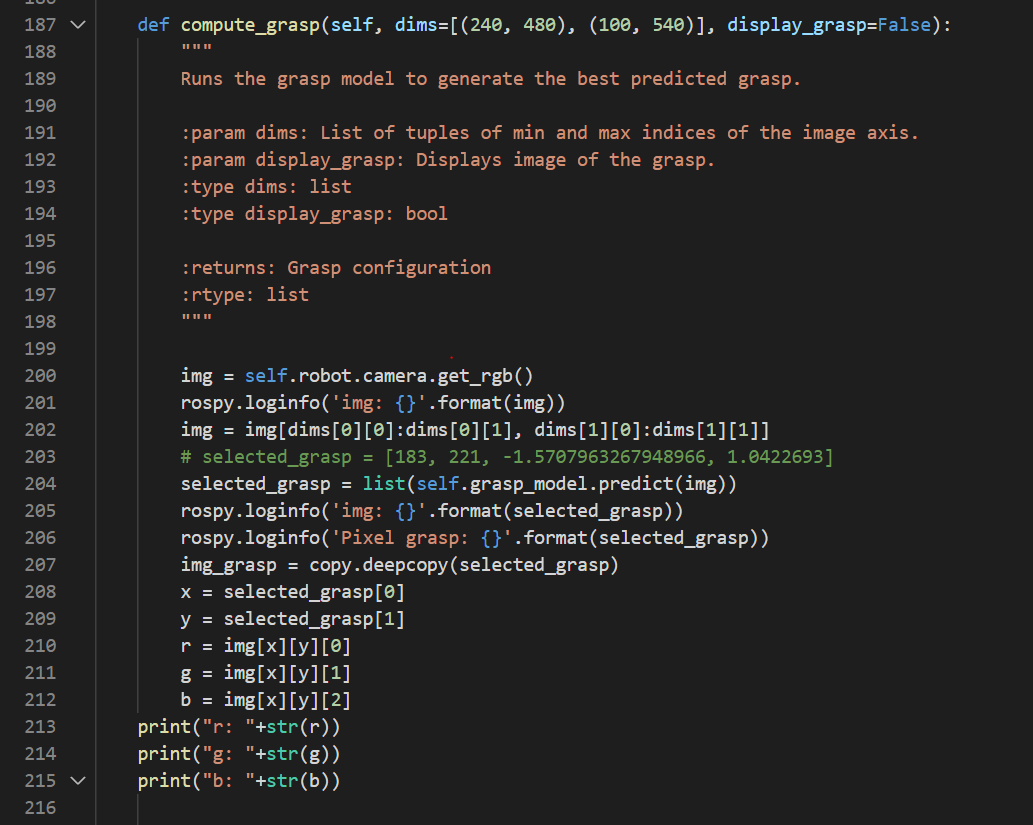
由於我們發現 selected\_graph 回傳的是一個 1維的vector, 有四個element，分別代表: **[ x, y, z, theta]**

[x, y, z] 表示物體的 絕對位置座標，單位: 公尺(m)

這邊theta指的是 yaw 角，單位是 radian，繞著z軸轉多少度。

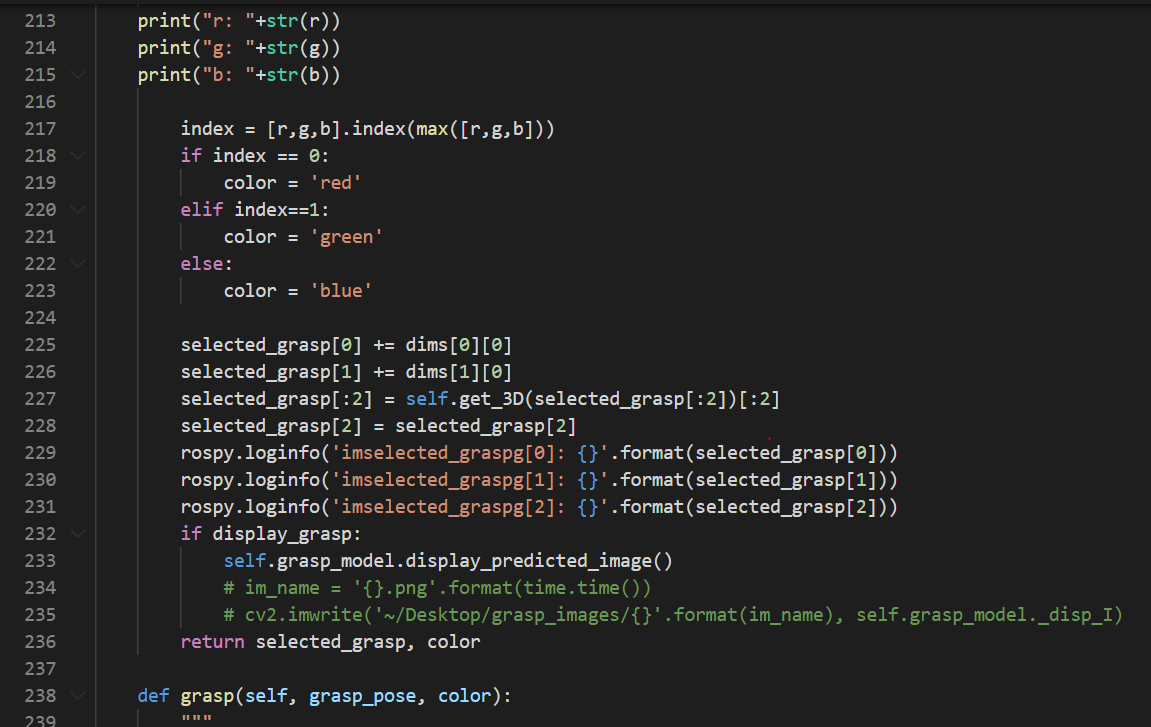
我們從 selected graph取得 x, y 座標位置，加上相機左上角點的起始座標位置(offset)，**得到以手臂座標為基準點，物件的真正座標。**





接著我們從 img[x][y] 這個 entry, 再去分析R, G, B 三個 channels 的值。

若 R 最大，則判斷他是紅色物件，若 G最大，判斷為綠色物件。若 B最大，判斷為藍色物件。



接著我們會把 grab\_pose 和 color當參數回傳到callback function, 再傳給 grasp() 函式，使其得以判斷 “現在是夾哪個顏色的物件，要放置到哪”

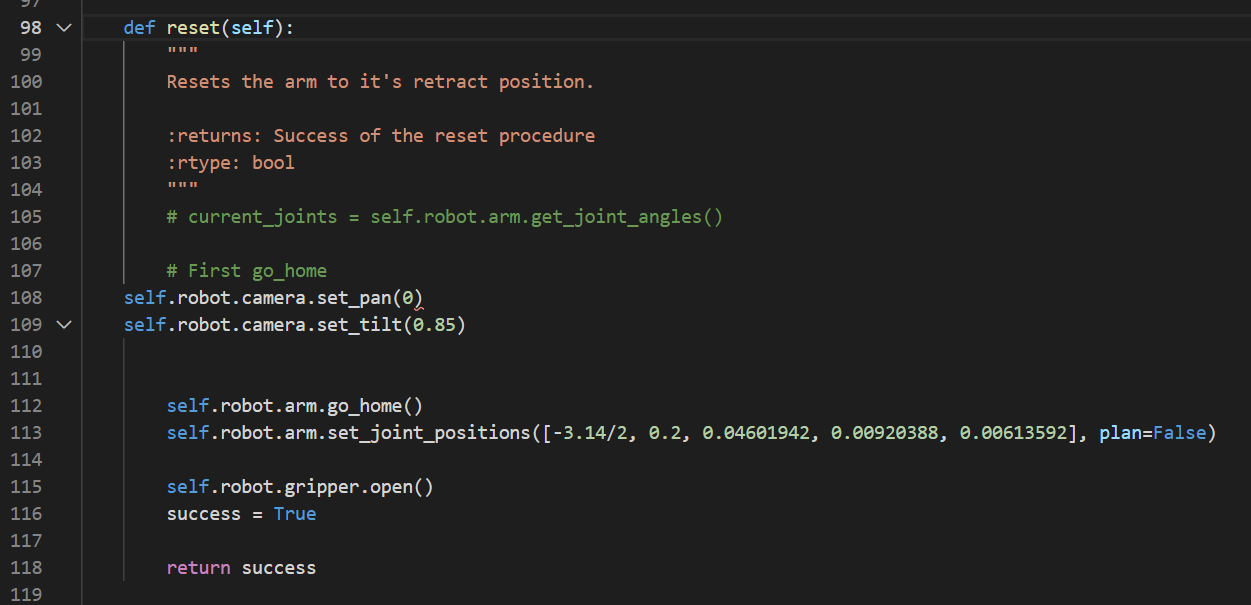
1. **Manipulation**

首先在每次夾取前，需要將手臂 reset到不會遮住camera的位置。

我們設定為右方位置。

透過量測　current\_joints 並trial and error後，我們使用 F.K. 設定馬達轉角。

程式碼 def reset() :



Reset我們做三件事情：

* 1. 讓camera 回到最適合的角度 (set\_pan(0) 和 set\_tilt(0.85) )
  2. 讓手臂先 go\_home() 再用F.K.轉到右邊避免遮住相機視線
  3. 讓手臂的gripper 打開

**手臂控制路徑規劃部分：**

由於 compute\_grasp幫我們計算出“欲夾取物件之姿態”，適合使用Inverse Kinematics 方式(已知目標點，計算手臂各軸馬達所需之轉角)。

我們將手臂夾取分為以下幾個步驟：

1. 移動到 “欲夾取物件”的正上方
2. 往下移動到 “欲夾取物件”的z座標
3. 進行夾取動作 (關閉夾爪)
4. 垂直向上先移動到一個安全的高度
5. 移動到 “欲放置盒子內位置”的正上方
6. 垂直向下移動到合適的高度
7. 打開夾爪，放置物件
8. 向上移動，回到 reset位置

我們將移動任務分開為這麼多個步驟的原因，是因為就算給定兩點A, B ，若單純用 I.K. 計算而不限制條件，手臂會計算 “其認為之最佳路徑”，過程中可能有Z座標或其他非預期之移動，導致撞到盒子或其他物件。舉例：假設A, B 兩點的Z座標同樣高度，但距離相對較遠，手臂不一定會只轉動第1軸的馬達，他可能搭配其他軸的馬達，因此過程中夾爪 (end effector)的 z座標可能有上下位移而打到東西的可能。

避免此問題有幾個方法：

1. 同樣用 I.K.計算，但多設幾個路徑必須經過點。
2. 使用 F. K . 來直接寫死馬達絕對轉角。

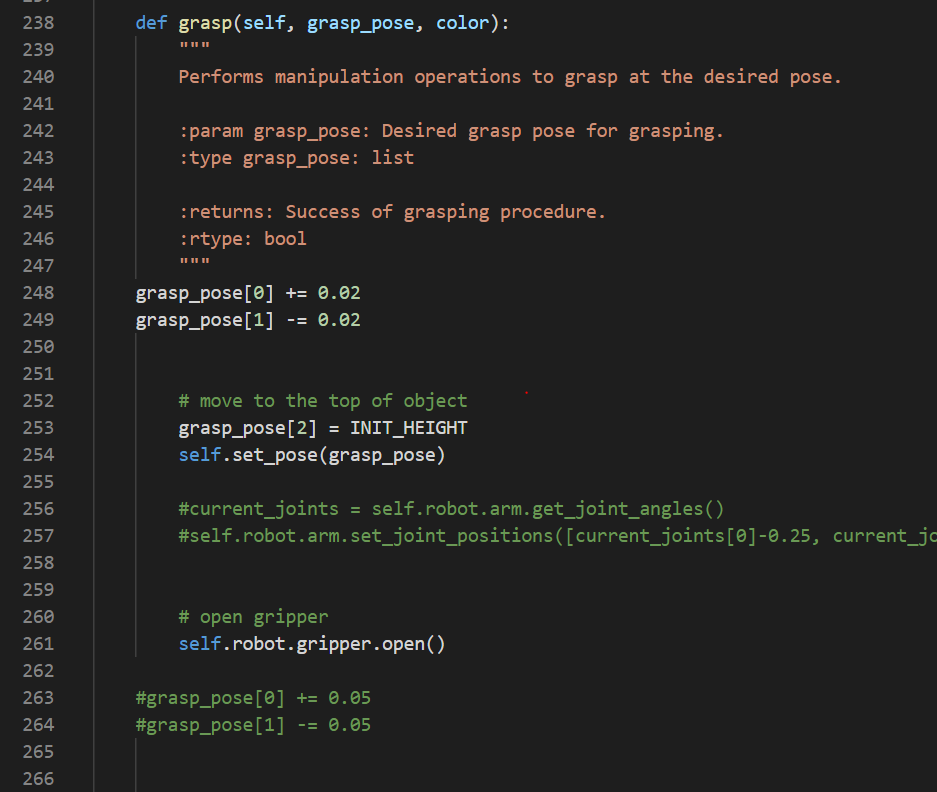
使用第一個方法的缺點：需要多寫幾個中間點(看起來較冗長)

第二個方法缺點為：需要先tune過各個位置對應之大概轉角。

綜合考量後，**我們最後選擇第一個方**法**：多設幾個中間經過點。**

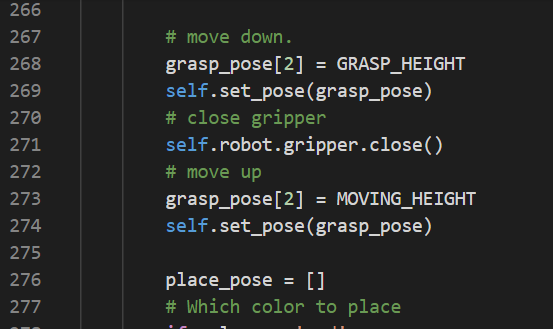
**從程式碼實現上述設計：**

1. 我們觀察我們手臂會偏離目標座標左下各一點點，因此我們加了一小段offset，讓他往右上各2公分偏移，藉此我們手臂能夠更準確落在物體正上方。
2. 讓手臂移動到物體正上方 “INIT\_HEIGHT”處，並讓夾爪打開



1. 接著執行一系列動作：

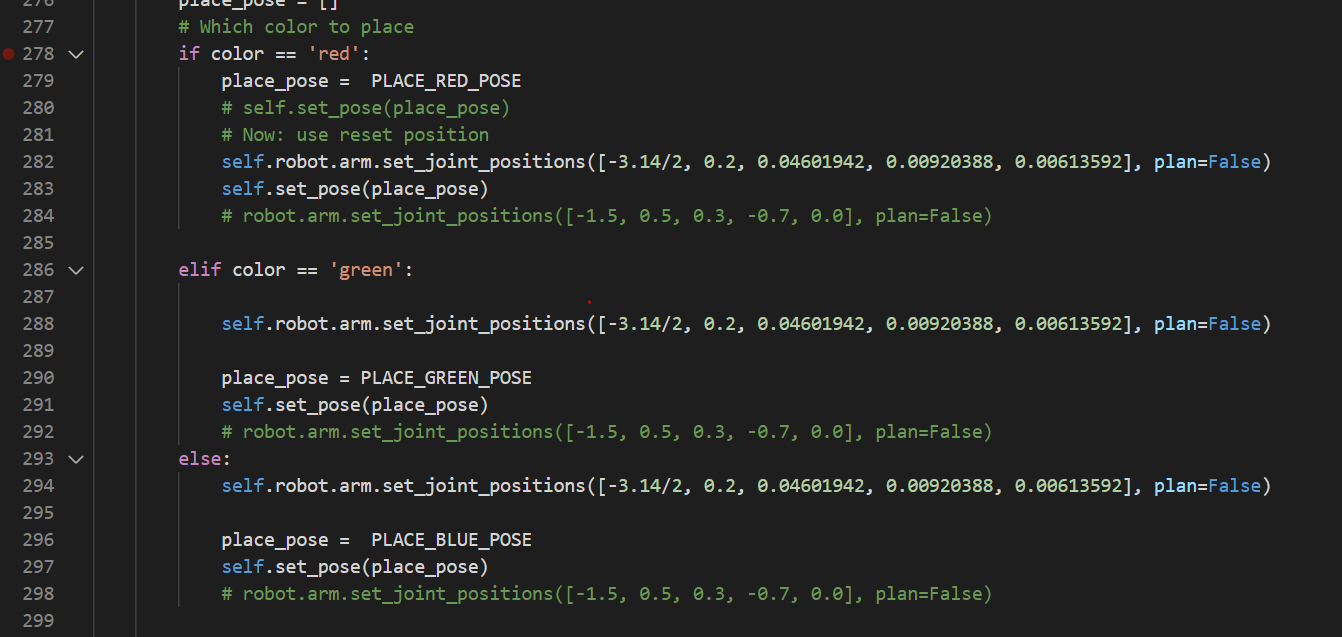
往下到欲夾取的高度 “GRASP\_HEIGHT”，關閉夾爪(夾取)，往上到 “欲移動的高度”(MOVING\_HEIGHT)。



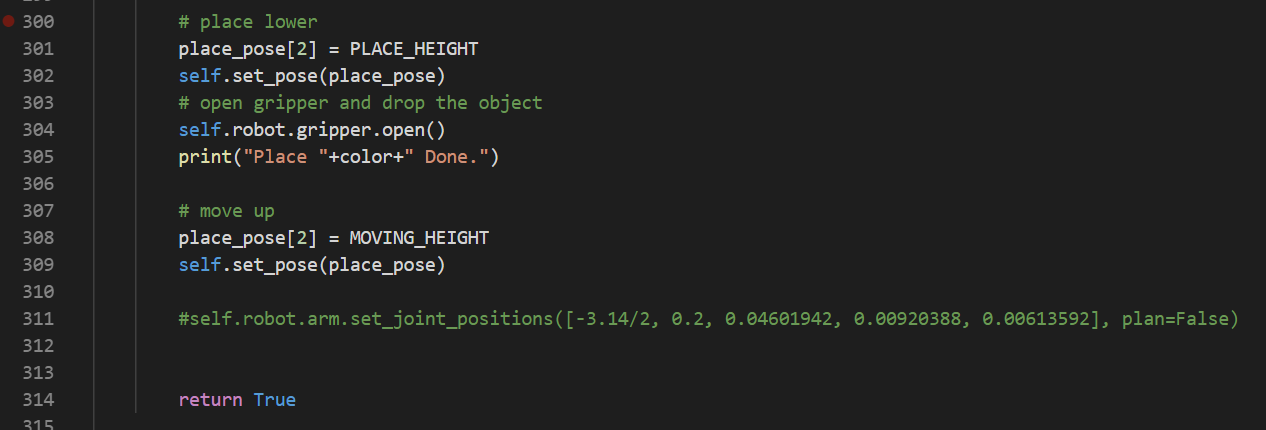
1. 接著根據不同顏色(傳進此函式的參數color)，決定放置點在哪。

如果是 ‘red’，要放到 PLACE\_RED\_POSE。其他以此類推。

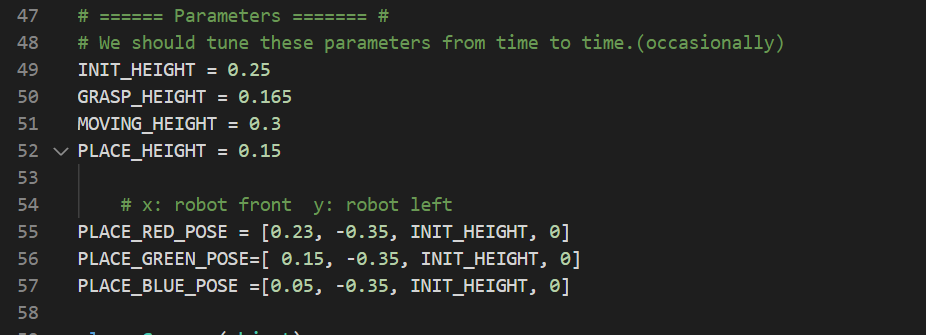
注意我們多使用一個 F.K. 讓手臂真正放下前，先回到盒子正上方，壁面移動時Z座標有往下的動作而卡到地板或其他障礙物的風險。



1. 移動到盒子上方後：首先往下到 PLACE\_HEIGHT(欲放開物體的高度，低一點避免物體彈開)，接著鬆開夾爪放置物體。然後先往上移動回到MOVING\_HEIGHT，再return回到下一次 grasp任務迴圈(會先reset)。回到MOVING\_HEIGHT的用意一樣是避免打到東西。



另外因為有許多參數須要透過實驗 再加上 fine tuning, 我們使用參數化將多個需要常常改到的參數拉到最上面，宣告成全域變數，方便使用者改動。



1. **結果**
2. demo :

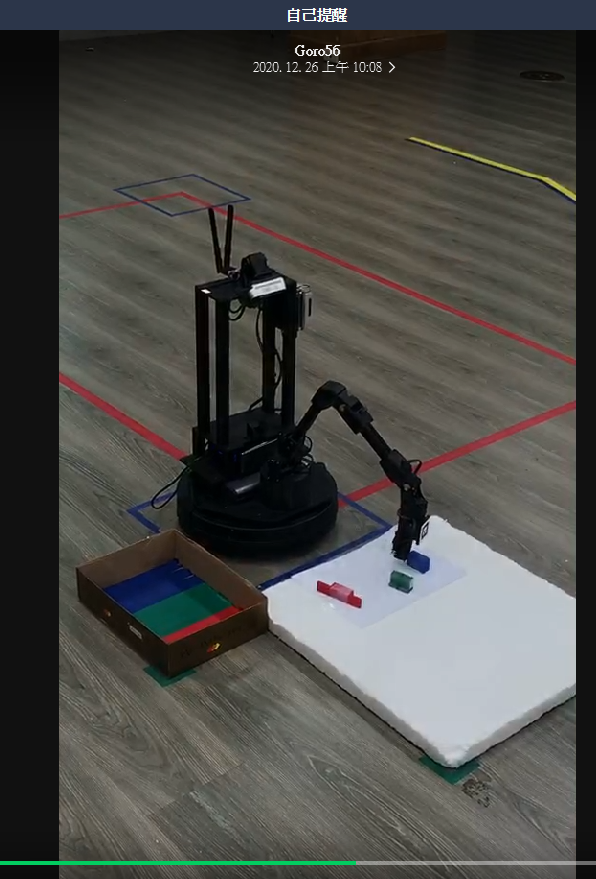
定位導航有些歪掉，導致 camera只照到藍色和綠色物件。

夾取任務：

**藍色方塊**：成功夾取，但因為 LoCoBot本身離盒子太遠，放置的位置(寫死)不夠遠，因此無法放入盒內。

**綠色方塊**：夾取失敗。

**紅色方塊**：因相機沒照到，辨識失敗。



第二次 demo:

定位導航有調整回來，進入藍色框框成功。

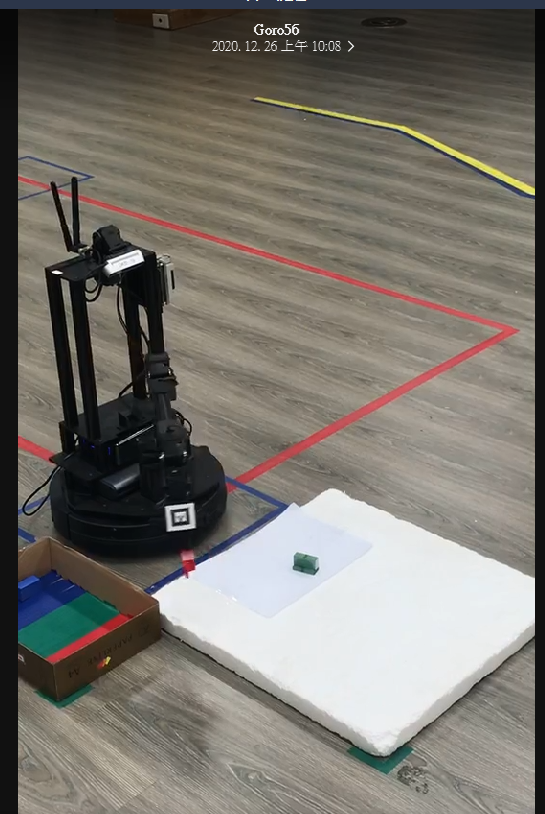
夾取任務：

**藍色方塊**：成功夾取，成功放置

**紅色方塊**：成功夾取，但因判斷的點位置，b channel值大於r channel(中心點判斷錯誤)，所以放置失敗(放到藍色盒子內)

**綠色方塊**：夾取失敗，兩次夾取都因中心點太低，沒能夾到綠色方塊。

中心點判斷錯誤的原因，或許可以使用更高的　n\_sample數，用時間換取準確度。



1. **討論:**

過程中遇到許多問題：

首先是 “理解程式碼”的部分：

1. 如何取得r,g,b channels的值?

透過 print(img), print(selected\_graph)後觀察發現：

從 selected\_graph[0],[1]拿到 x, y 座標位置

再從img[x][y][0], [1], [2]拿到 r, g, b 三個值!

1. 如何使用I.K.和F.K.的API?

可以先用 get\_current\_joints() 讀取現在轉角，作為F.K.設定的參考。

另外當天還有其他技術問題：

1. 搜不到 locobot的wifi hotspot

解法：打開 ~/.bashrc, 把 export ROS master : ….10.42.0.2 (或.1)全部註解掉，並勾選enable wifi(若接螢幕使用gui介面)，記得要重新source bashrc (. ~/.bashrc)然後重新開一個 terminal (因原先的 termina的環境變數已經設好)

1. LoCoBot校正 camera 的第三個 script無法運行

原因是因為需要 升級 Pytorch的版本，但因升級完後還需要降級才能使用這次任務需用到的api功能，先考慮不使用這個方法。

1. Segmentation Fault

可能因為 ctrl + C 沒有完全終止掉曾經開啟的 process，導致同樣的 process被開了很多遍，占用系統資源，導致cpu出錯，memory存取出現問題。

解法：使用kill %1 或 kill %2, %3 等去kill掉所有尚未終止的processes。

1. Bad Callback : locobot.py cannot connect to move\_group
2. Run Time Error

不知道為甚麼計算出的 z 會是0，導致 Raise RunTimeError，在locobot.py的get3D\_camera() 函式中的 get\_Z()中。

因觀察：三個物件的 z 值大概是 0.6左右，解法: 在裡面把 z設為0.6，暫時解掉莫名的 run time error產生。

1. 從 visual studio code編輯器複製程式碼，或是從記事本等編輯器複製，python會讀出一些奇怪的亂碼導致編譯失敗。

因為我們的機器從最初的 10 號中間換過6號，導致最後要再使用10號時須要做程式碼搬移的動作。

最後解法：使用scp 從local 端 直接 copy 整個 folder到遠端LoCoBot路徑上

(scp -r ~/navigation\_controller [locobot@10.42.0.1:~low\_cost\_ws\_my/src/.../navigation\_controller](mailto:locobot@10.42.0.1:~low_cost_ws_my/src/.../navigation_controller))

**心得：**

* 1. 首先謝謝三位老師精心準備這學期的課程，真的收穫滿滿!

還有各個助教用心準備的教材，以及不厭其煩地為我們解惑，解決許多Error和debug，非常感謝助教! 真的無限感激!

謝謝老師給我們機會親自動手實作，撰寫程式並實現在 LoCoBot機器人平台上，千載難逢的機會與永生難忘的寶貴經驗!

* 1. 針對期末專題競賽，Demo完畢才發現：

單純以完成這次的 task 的話，在grasp 其實不用知道現在是夾甚麼顏色，因為特定顏色的方塊會放在固定的位置，只需知道現在grasp\_pose 的 y座標大概落在哪個區域，即可判斷現在夾取的是哪個物件，就不會有上述 ”因為框框政中心點是在白色紙，導致他判斷成錯的r, g, b channel 物件”的問題了~~ QQ 可惜太慢想到。

不過以長久之計，以及要 general 和 robust的話，不該用這種偷吃步的方法哈哈，理想的優化方法還是要專注在 object detection 的演算法，以及夾爪判定位置的準確度，以及手臂運動控制的定位精度。

附上我們共同撰寫的 hackmd筆記：

<https://hackmd.io/@lovelyrachelhsia/PyRobot>

Merry Christmas & Happy New Year!

