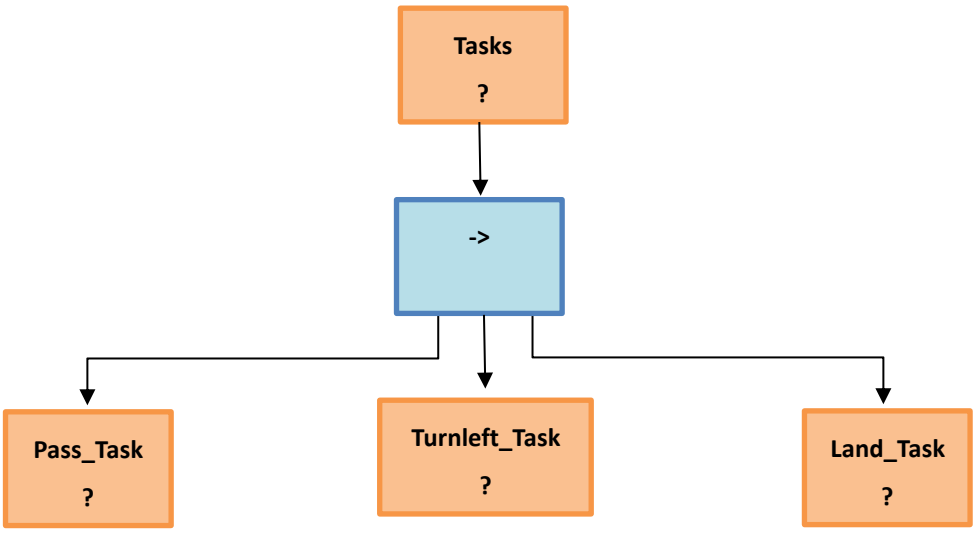
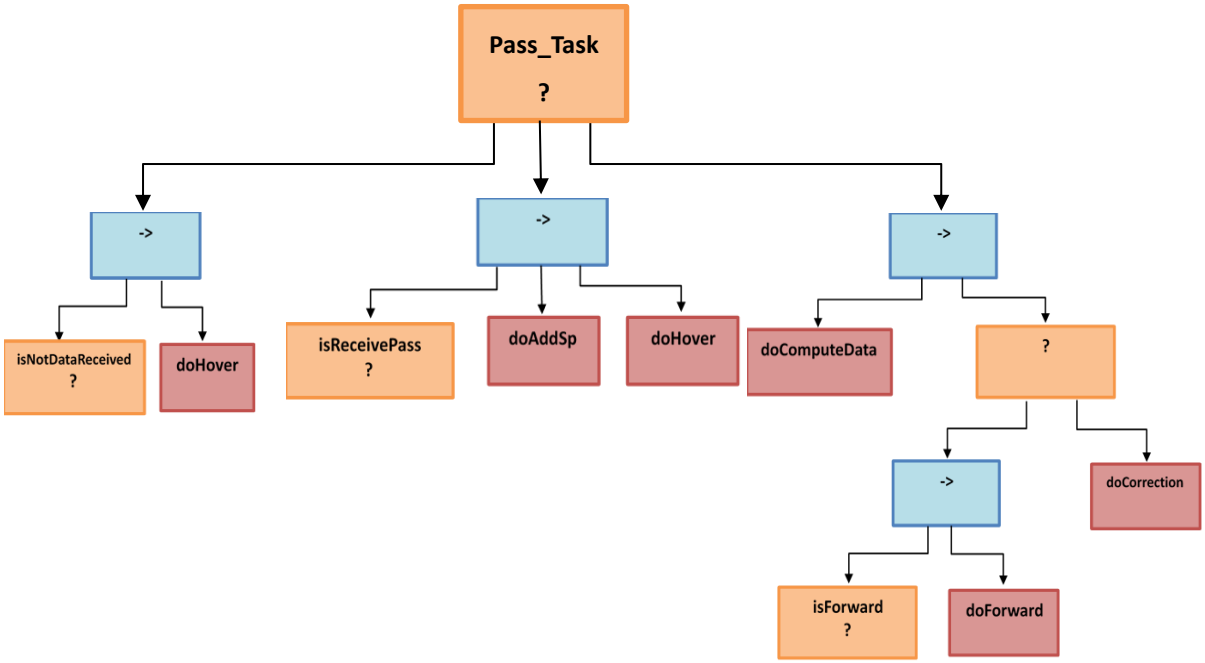
技術學習成果報告

■ 控制框架設計與分析

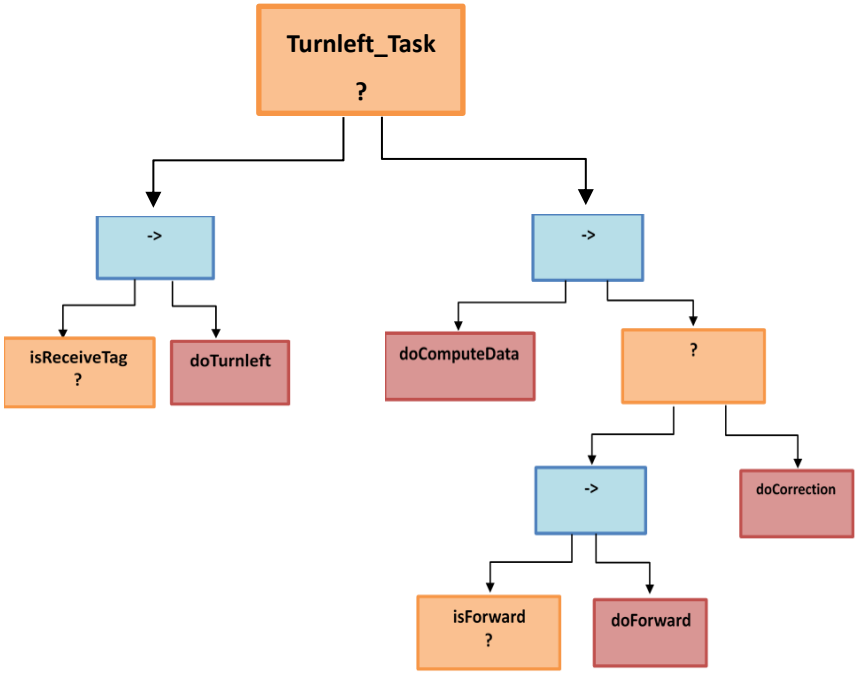
- 使用 Behavior Tree 設計無人機控制邏輯框架，共分為三大任務:



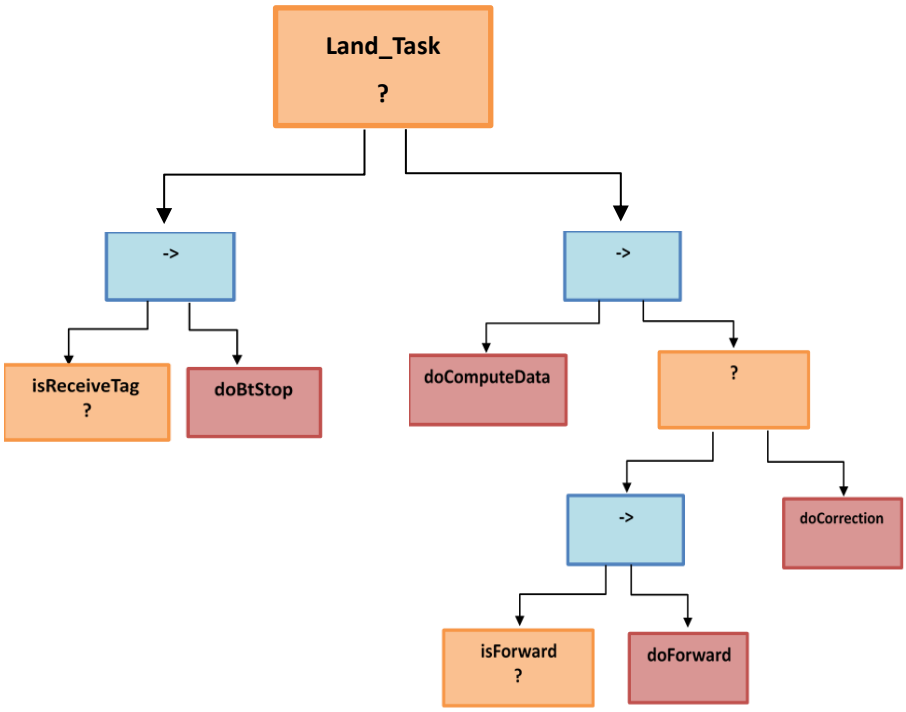
1）偵測紅框並快速通過



2）偵測到 AprilTag 向左轉



3）偵測到 AprilTag 降落



■ 程式模組結構說明

1） detect.py（camera 影像辨識）

✧ 顯示鏡頭中心點(綠色) / 目標中心點(紅色)

- 紅框偵測:

```python
def detect_frame(image):
    hsv_img = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    # 定義 HSV 值域的紅色上下界(需要定義兩次)
    lr0, ur0 = np.array([0, 70, 0]),  np.array([5, 255, 255]) # 定義 HSV 下界 & 上界
    lr1, ur1 = np.array([175, 70, 0]), np.array([180, 255, 255]) # 定義 HSV 下界 & 上界
    mask0, mask1 = cv2.inRange(hsv_img, lr0, ur0), cv2.inRange(hsv_img, lr1, ur1) # 建立遮罩1, 遮罩2

    mask = cv2.bitwise_or(mask0, mask1) # 將兩組遮罩進行or運算合併

    # 對找出的HSV mask 進行輪廓搜尋: findContours
    contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

    # max function 在遇上空陣列[]會報錯, 導致程式終止,
    # 如要避免此情況可以在執行max()前透過continue來避免
    if len(contours) == 0:
        print("not found")
        return None, None, None, None

    max_contour = max(contours, key = cv2.contourArea)

    # 使用boundingRect()找出可以包覆max_contour的最小矩形(平行軸)
    x, y, w, h = cv2.boundingRect(max_contour) # 取得座標與長寬尺寸

    cv2.rectangle(image, (x, y), (x + w, y + h), (0, 0, 255), 2) # 將包覆矩形繪製在影像上
    cv2.circle(image, (int(x+w/2), int(y+h/2)), 5, (0,0,255), -1) # 繪製方形的中心點
    cv2.circle(image, (480, 200), 5, (0,255,0), -1) # 繪製鏡頭的中心點
    return x+w/2, y+h/2, w*h
```

- AprilTag 偵測:

```python
def detect_apriltag(image):
    # 建立 aprilTag Detector
    detector = apriltag.Detector()
    # detector接收灰階圖像進行偵測, 這裡將轉換成gray格式
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    # 使用 aprilTag Detector 進行 aprilTag偵測
    result = detector.detect(gray_image)

    # 複製一份 原始影像image 作為 show_image
    show_image = image.copy()
    # 當 沒有偵測到 aprilTag, 在視窗上顯示 "not found"
    # 當 有偵測到 aprilTag, 取出其中心點 center 座標
    if len(result) == 0:
        print("not found")
        return  0, 0, 0
    else:
        # result 為 1 list, 故透過 [0] 取出第一個 aprilTag
        center = result[0].center
        corner = result[0].corners
        # 透過左下角與右下角計算寬 w
        w = math.sqrt( (corner[0][0] - corner[1][0])**2 + (corner[0][1] - corner[1][1])**2 )

        # 透過右下角與右上角計算寬 h
        h = math.sqrt( (corner[2][0] - corner[1][0])**2 + (corner[2][1] - corner[1][1])**2 )

    cv2.circle(image, (int(center[0]),int(center[1])) , 5, (0,0,255), -1)
    cv2.polylines(image,[np.int32(corner)], True, (0,0,255), 2, cv2.LINE_AA)
    cv2.circle(image, (480, 200), 5, (0,255,0), -1) # 繪製鏡頭的中心點
    return center[0], center[1], w*h
```

- 偵測目標條件判斷:

i. current_task = 0

透過 ROS Publisher，將**紅框**的訊息發布至 topic:

/target_point 上給控制程式，調整飛行以過框。

```
# 當 start_detect 為 True
if start_detect:
    if current_task == 0: # 當前偵測對象為方框
        frame_data = detect_frame(image)
        # 確保 x 坐標不是 None , 代表已經成功偵測到方框
        if frame_data[0] is not None:
            # (w * h)/(960*720) 為方形面積/總畫面面積, 兩者相除後得到 方形面積在總畫面中的占比, 透過str()將float轉成 string
            ratio = frame_data[2] / (960*720.0)
            # 當 mask面積 / 畫面總面積 (960 * 720) 大於等於 門檻值 0.35, 表示無人機已經夠接近框, 可以進行加速過框行為
            if ratio >= TARGET_RATIO:
                # 透過 point_pub 發布 Float64MultiArray訊息, 將 框的 中心點x, 中心點y, 1發佈出去, 1表示占比已超過門檻值
                point_pub.publish(Float64MultiArray(data = [frame_data[0], frame_data[1], 1]))
                print("Start fly through the frame")
                # 可加速通過框後就不再發佈訊息, 將 start_detect 改為 False
                #start_detect = False
                current_task = 1

            # 無人機還不夠接近框, 可以繼續進行校正與向前
            else:
                # 透過 point_pub 發布 Float64MultiArray訊息, 將 框的 中心點x, 中心點y, 0發佈出去, 0表示占比未超過門檻值
                point_pub.publish(Float64MultiArray(data = [frame_data[0], frame_data[1], 0]))
```

ii. current_task = 1

透過 ROS Publisher，將 **AprilTag** 的訊息發布至 topic:

/target_ap 上給控制程式，使其轉向或降落。

```
elif current_task == 1: # 當前偵測對象為 apriltag
    apriltag_data = detect_apriltag(image) # 得到 x, y, w, h
    print(apriltag_data[2])

    # 表示無人機已經位於apriltag前正確距離, 可以進行下降過框行為
    if apriltag_data[2] >= 20000:
            # 透過 point_pub 發布 Float64MultiArray訊息, 將apriltag的 中心點x, 中心點y, 1(表示占比已超過門檻值)發佈出去
            ap_pub.publish(Float64MultiArray(data = [apriltag_data[0], apriltag_data[1], 1]))
            print("Start Turn left or Go down")
            # 可加速通過框後就不再發佈訊息, 將 start_detect 改為 False
            #start_detect = False

    # 無人機還不夠接近框, 可以繼續進行校正與向前
    else:
            # 透過 point_pub 發布 Float64MultiArray訊息, 將apriltag的  中心點x, 中心點y, 0(表示占比未超過門檻值)發佈出去
            ap_pub.publish(Float64MultiArray(data = [apriltag_data[0], apriltag_data[1], 0]))
```

2） BT_test.py（邏輯控制飛行）

- Behavior tree

```
self.tree = (

    (self.isNotDataReceived >> self.doHover)
| (self.isReceivePass >> self.doAddSp >> self.doHover)
>> ((self.isReceiveTag  >> self.doTurnleft >> self.isReceiveTag >> self.doBtStop) | (self.doComputeData >> (self.isForward >> self.doForward) | (self.doCorrection)))
| (self.doComputeData >> (self.isForward >> self.doForward) | (self.doCorrection))

)
```

➢ 此方法後來發現無法執行到 doBtStop，偵測到 AprilTag 只會不斷左轉。

- 新增關於 AprilTag 條件判斷和動作:

```python
@action
def doTurnleft(self):
    print("action: turnleft")
    PI = 3.1415926535897

    msg = Twist()
    angular_speed = 90*2*PI/360
    msg.angular.z = -abs(angular_speed)
    t1.controler.move(msg, 1.4)
```

```python
@condition
def isReceiveTag(self):
    print("condition: isReceiveTag")
    return t1.state.canTag == 1
```

■ 系統運作結果

- Demo 1: 僅測試 AprilTag 部分，可以正常偵測且執行。

  https://youtu.be/jF381qoQyIc

- Demo 2: 在課堂上最終展示時訊號不穩，僅完成過框與轉向。

  https://youtu.be/_0dp4KAprCg

■ 心得經驗

從第一階段編寫劇本飛行，到第二階段影像偵測過框，最後第三階段整合展示，過程中遇到許多問題，例如在過紅框時，每台無人機飛行穩定度與狀態不一，有時候會撞到邊框，經調整面積占比值，以及加速向前通行速度調慢、持續時間拉長後，有加以改善。另外在整合 camera 偵測 AprilTag 和紅框也花費了一些時間，最終可以藉由傳值(current_task)，分別處理不同偵測目標，並顯示鏡頭及目標物中心點，方便查看飛行調整過程。最後的展示成果雖然無法如願完成所有任務，但在不斷測試中，學習到如何去控制邏輯程式和調整數值，以達到較好的飛行穩定度。