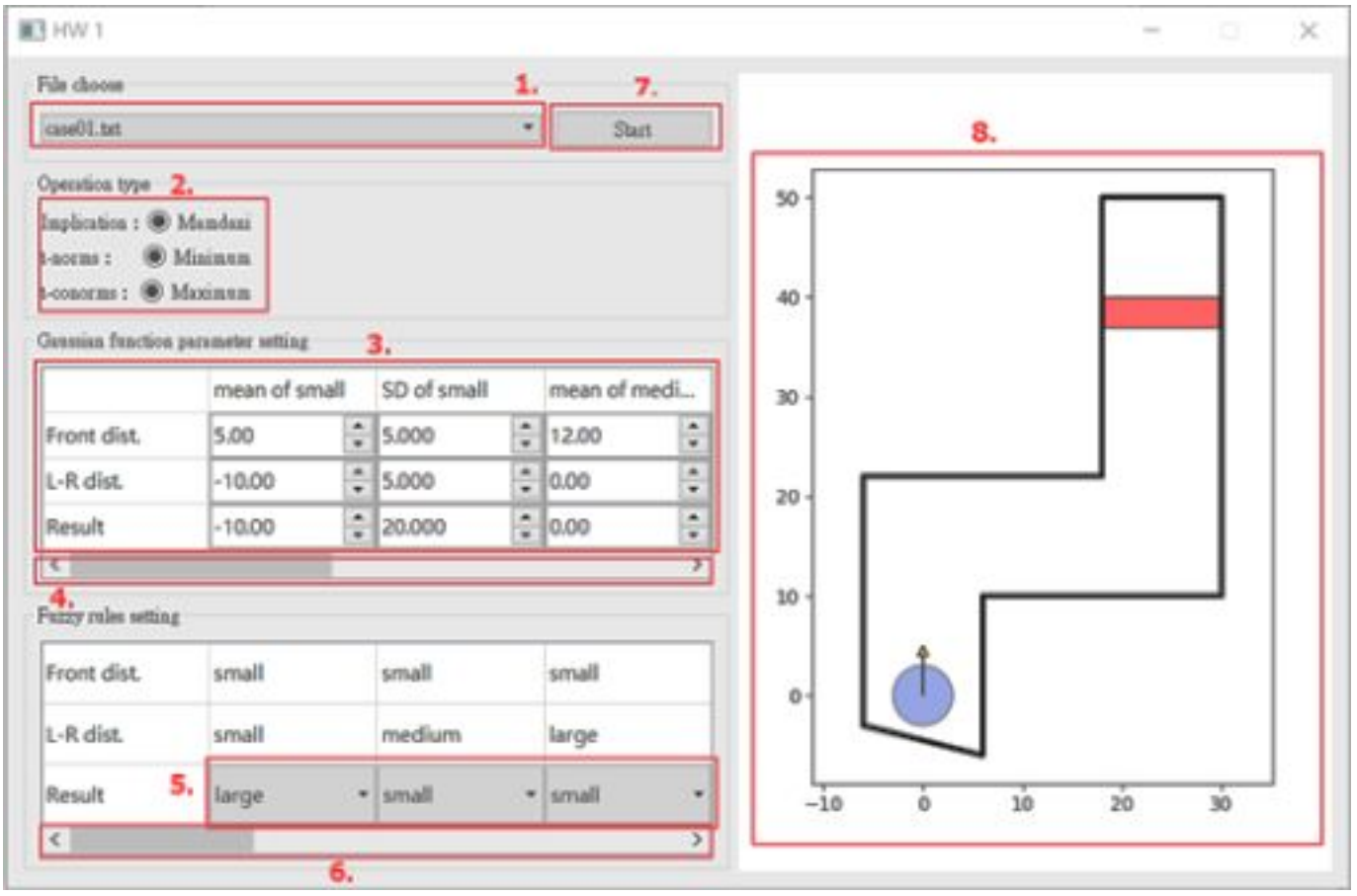
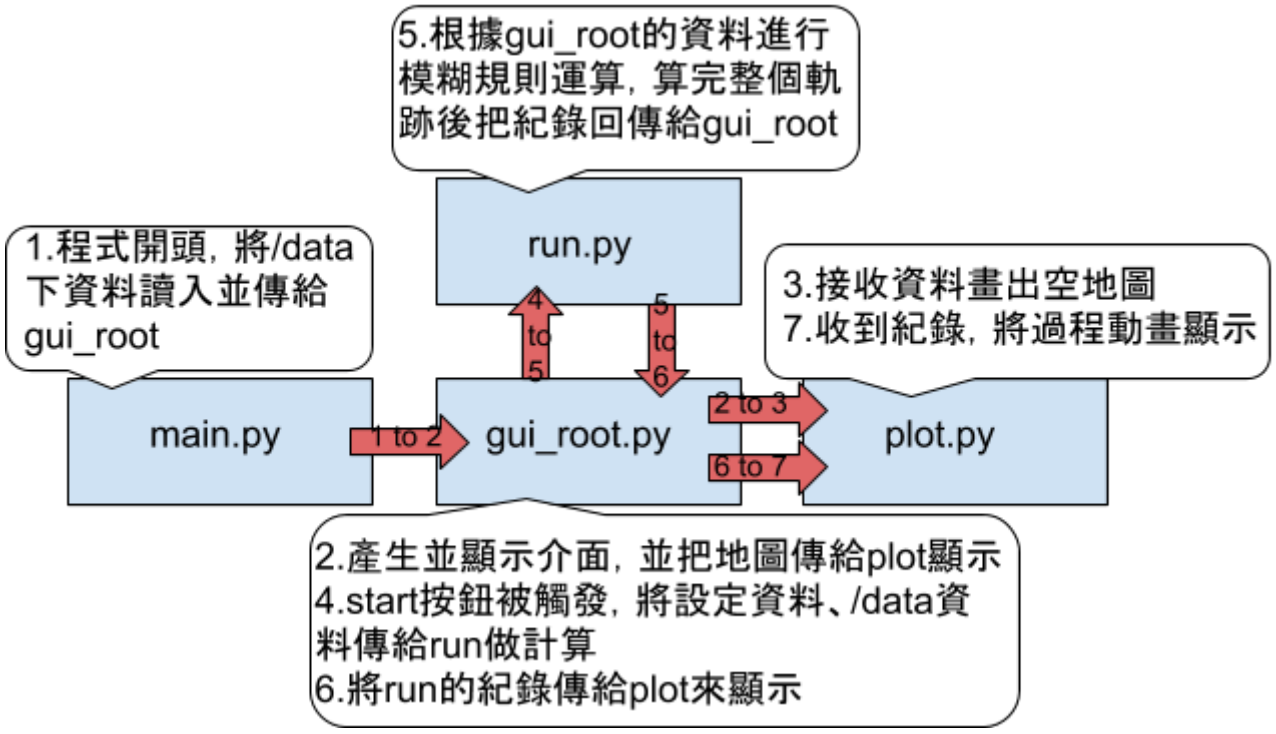
A. 程式介面與操作說明



1. 選擇輸入軌道檔案。選擇後區域8會立刻更新。讀取檔案位置為執行檔案的同個目錄下的「data」資料夾中所有.txt檔。
2. 顯示模糊蘊含、模糊交集、模糊聯集的運算方式，若之後需要擴充也可以直接增加按鈕在此區域，供使用者選擇。
3. 設定每個模糊變數的歸屬函數的參數(高斯函數的mean、standard deviation)，預設是實驗出可以到達終點的參數。另外此處措辭集中的small用的高斯函數是單調遞減函數，小於mean部分歸屬程度都為1；措辭集中的medium用的是正常高斯函數；措辭集中的large用的高斯函數是單調遞增函數，大於mean部分歸屬程度都為1。
4. 橫移軸用來調整表格橫向位置
5. 設定每個模糊規則的後鑑部，有large、medium、small可以選擇。
6. 橫移軸用來調整表格橫向位置。
7. 當參數、規則都設定好了之後，就可以按下此鍵，開始進行運算。
8. 當start被按下並計算完，此處就會畫出車子移動過程與軌跡。若撞到牆車子會變成紅色並停止，不會輸出記錄檔；若抵達終點，車子會變成綠色並停止，然後會在執行檔案的同個目錄下直接輸出紀錄檔，train4D.txt、train6D.txt。

B. 程式碼說明

1. 本次作業程式碼總共分為四個檔案，main.py、gui_root.py、run.py、plot.py，分別用來讀取檔案、建立介面、執行計算(包含模糊系統相關計算、碰撞處理)、畫出地圖並將計算結果顯示在地圖中，運算簡易流程如下。



2. library部分主要引入四個， pyqt5 用來構築GUI介面（按鈕、數值輸入、選項），shapely、descartes用來處理碰撞、抵達終點問題，matplotlib 則是處理輸出影像（右側顯示區）
3. 首先介紹main.py中較關鍵的部分，其中主要功能就是把讀入檔案傳給gui_root.py。

```
def main():
...(表示省略不重要程式碼)
    #把讀入檔案傳給gui_root
    gui_root = GuiRoot(read_file())

def read_file():
    """用來讀檔的function"""

    road_map = namedtuple('road_map', ['start', 'x', 'y'])
...
    #path為儲存/data目錄下所有檔案絕對路徑的list
    paths = (join(datapath, f) for f in folderfiles if isfile(join(datapath, f)))
    for idx, content in enumerate(list(map(lambda path: open(path, 'r'), paths))):
        i = 0
        #把讀入檔案分為3個list來儲存起點、x、y位置
        for line in content:
            if i == 0:
                dataset[folderfiles[idx]] =
road_map(list(map(float,line.split(','))),[],[])
            else:
                dataset[folderfiles[idx]].x.append(float(line.split(',')[0]))
                dataset[folderfiles[idx]].y.append(float(line.split(',')[1]))
            i += 1
    return dataset
```

4. 接下來介紹用來產生介面的gui_root.py，他和各個程式都有關聯，首先接收main.py的軌道資料，並傳給plot.py來顯示初始狀態，然後顯示出剩餘介面；當你參數、規則設定好後，他會把設定讀入並傳給一個thread 來執行run.py；當計算完畢後，他會把計算紀錄檔傳給plot.py來顯示出過程與軌跡。

```
class GuiRoot(QWidget):
    """Root of gui."""

    def __init__(self, dataset):
        """Create GUI root with datasets dict"""
...
        #視窗基本設定
        self.setFixedSize(1000, 620)
        self.center()
        self.setWindowTitle('HW 1')
        self.show()
...
        #呼叫下面function來建立各個區塊介面
        self.file_run_creation(self.datalist)
        self.operation_type_creation()
        self.fuzzy_rule_setting_creation()
        self.semantic_rule_setting_creation()
...
        self.m = PlotCanvas(self.data)
...
    def file_run_creation(self, datalist):
    #用來建造檔案選擇區塊介面
...
    def operation_type_creation(self):
    #用來建造規則規則相關運算設定介面
...
    def semantic_rule_setting_creation(self):
    #用來建造模糊變數參數設定區塊介面
...
    def fuzzy_rule_setting_creation(self):
    #用來建造規則後鍵部選擇區塊介面
...
    def file_changed(self):
    #若選擇檔案改變，會在右邊區域顯示新選擇地圖
        self.m.plot_map(self.file_choose.currentText())
    def run(self):
    #當start 按下後開啟一個thread來計算run.py中的模糊系統相關計算
...
```

5. run.py是用來處理模糊系統計算、碰撞處理的程式；為了避免程式lag，所以會把整個車子移動的計算存成一個詳細紀錄資料矩陣，當計算完成後才一次回傳整個記錄檔以供畫圖所需。

```
    @pyqtSlot()
    def run(self):
        """Run this function"""
         # 矩陣用來記錄中儲存了每次移動時 車子中心 x, y 座標，前方、左方45度、右方45度牆的距離與其座標，
                        車子方向盤角度，車子與水平夾角。
        trace_10d = []
...
        # 主要迴圈，計算車子移動相關資訊、模糊系統，跳出迴圈條件為車子撞牆
        while(not car.intersection(map_line)):

            # 判定是否抵達終點，若是則讓記錄檔不一樣長作為判斷依據
            if(end_area.contains(sp.Point(car_center))):
                trace_10d[1].append(0)
                break

            #以記錄檔長度來判斷是否初次執行，若第一次執行迴圈，設定初始值
            if (len(trace_10d[0]) == 0):

                # 設定檢查車子撞牆的半徑範圍
                r = ((max_px - min_px)**2 + (max_py - min_py)**2)**(0.5)
...
            else:
                #套用公式來更新新的車子中心座標、車子與水平夾角度
                car_center = (car_center[0]+M.cos(M.radians(fai+output))+
M.sin(M.radians(fai))*M.sin(M.radians(output)),
                              car_center[1] + M.sin(M.radians(fai + output)) -
M.sin(M.radians(output))*M.cos(M.radians(fai)))
...
                fai = fai - M.degrees(M.asin(2*M.sin(M.radians(output))/6))
            # dir, l, r line是前、左、右方的直線
            dir_line = [
                [x, y], [x + r * M.cos(M.radians(fai)), y + r * M.sin(M.radians(fai))]]
            l_line = [[x, y], [
                x + r * M.cos(M.radians(fai + 45)), y + r * M.sin(M.radians(fai + 45))]]
            r_line = [[x, y], [
                x + r * M.cos(M.radians(fai - 45)), y + r * M.sin(M.radians(fai - 45))]]

            #用上面的直線來判斷各方向的交點跟距離，以下以前方為例
            temp = sp.LineString(dir_line).intersection(map_line)
            temp = self.distance(temp, car_center)
...
            # 以前方距離模糊變數為例，根據設定參數來定義並計算與前方距離模糊變數的各個歸屬度，其餘模糊變數省略
            front_small = g_decreasing_funct(dir_dist, fuzzy_variable[0], fuzzy_variable[1])
            front_medium =gfun(dir_dist, fuzzy_variable[2], fuzzy_variable[3])
            front_large = gufun(dir_dist, fuzzy_variable[4], fuzzy_variable[5])
...


            #用離散重心法來去模糊化
```

```
            output_l = []
            for i in range(-400, 410):
                output_l.append(max(min(front_small, lr_s, rule[0][i+400]),
                                    min(front_small, lr_m, rule[1][i+400]),
                                    min(front_small, lr_l, rule[2][i+400]),
                                    min(front_medium, lr_s, rule[3][i+400]),
                                    min(front_medium, lr_m, rule[4][i+400]),
                                    min(front_medium, lr_l, rule[5][i+400]),
                                    min(front_large, lr_s, rule[6][i+400]),
                                    min(front_large, lr_m, rule[7][i+400]),
                                    min(front_large, lr_l, rule[8][i+400])))
...
            for i in range(-400, 410):
                up += i/10*output_l[i+400]
                base += output_l[i+400]
            if base != 0:
                output = up/base
...
        #回傳記錄檔
        self.signals.result.emit(trace_10d)
```

6. 最後來介紹plot.py，他是負責處理和右側畫圖有關的部分，所以將初始地圖更新、劃出移動軌跡、、、等都是在這部分完成。

```
class PlotCanvas(FigureCanvas):
#初始右側區域的Figurecanvas(matplotlib中畫圖的一種類別)，並設定其相關設定、參數。
    def __init__(self, dataset={}):
        #設定大小、座標軸等設定
        fig = Figure(figsize=(8, 8), dpi=100)
        self.ax = fig.add_subplot(111)
        self.ax.axis('equal')
...
        imap = list(self.dataset.keys())[0]
        #設定完成後，把讀入的第一個檔案傳給plot_map, 來顯示初始地圖在已設定好的FigureCanvas上
        self.plot_map(imap)

    def plot_map(self, imap):
    #用來更新整個地圖的function
        self.ax.clear()

        # 用shapely來創造出 Polygon 類別(包括面積)的終點區域
...
        end_area = shapely.geometry.Polygon(ea)

        # 用shapely來創造出 LineString 類別(只有線)的地圖邊界區域
...
        map_line = shapely.geometry.LineString(verts)

        # 用shapely的圓來作為車子
        inicar = shapely.geometry.Point(self.dataset["{}".format(
            imap)].start[0], self.dataset["{}".format(imap)].start[1]).buffer(3)
...
```

```
        # 初始地圖物件都創好後，把它們上不同顏色，並加到畫布上來顯示
        self.ax.plot(*np.array(map_line).T, color='k',linewidth=3,
solid_capstyle='round')
        self.ax.add_patch(descartes.PolygonPatch(
            end_area, fc='red', alpha=0.7))
        self.car = self.ax.add_patch(
            descartes.PolygonPatch(inicar, fc='royalblue', alpha=0.5))
        self.draw()
    def plot_car(self, list9d):
    #用來產生車子移動動畫、軌跡，並輸出train4D.txt、train6D.txt的function
...
        # 因為在run.py中產生的紀錄檔會把矩陣長度是否相同作為抵達終點訊號，所以每當這裡判斷長度
不相同時，就可以視作車子可以抵達終點，要輸出train4D.txt、train6D.txt。(此處以4D為例)
        if (len(list9d[0]) != len(list9d[1])):
            outpath = join(os.path.realpath(os.path.join(
                os.getcwd(), os.path.dirname(__file__))), "train4D.txt")
            with open(outpath, "w") as fp:
                for i in range(0, int(len(list9d[0]))):
                    s = ''
                    for j in range(2, 6):
                        if j == 5:
                            s = s + str('{:.7f}'.format(list9d[j][i]))
                        else:
                            s = s + str('{:.7f}'.format(list9d[j][i])) + ' '
                    fp.write(s+'\n')
...

        #根據run.py中產生的紀錄檔來產生動畫、軌跡
        for i in range(0, len(list9d[0])):
            #這個條件用來判斷處理到達終點車子要變綠色
            if (len(list9d[0]) != len(list9d[1])) and (i+1 == len(list9d[0])):
...
            #這個條件用來判斷處理撞到牆，車子要變紅色
            elif (len(list9d[0]) == len(list9d[1])) and (i+1 == len(list9d[0])):
...
            #其他代表車子還沒到，在地圖上刷新車子、距離線、軌跡
            else:
                newcar = shapely.geometry.Point(*self.car_center).buffer(3)
            self.dir = self.ax.arrow(
                *self.car_center, *self.dir_point, head_width=1, head_length=1,
fc='gold', ec='k')
            front_line = self.ax.plot([front_point[0], list9d[6][i][0]], [
                                      front_point[1], list9d[6][i][1]], linestyle='--',
color='navy')
            right_line = self.ax.plot([right_point[0], list9d[7][i][0]], [
                                      right_point[1], list9d[7][i][1]], linestyle='--',
color='navy')
            left_line = self.ax.plot([left_point[0], list9d[8][i][0]], [
                                     left_point[1], list9d[8][i][1]], linestyle='--',
color='navy')
```

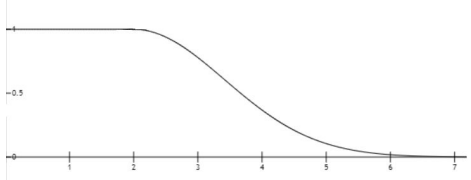
```
...
            self.trace = shapely.geometry.Point(*self.car_center).buffer(0.1)
            self.draw()

            #畫完、顯示完後馬上移除，下次才能直接畫新的
            self.dir.remove()
            self.car.remove()
...

            #設定刷新頻率
            loop = QEventLoop()
            QTimer.singleShot(20, loop.quit)
            loop.exec_()
```

C. 模糊規則、函數的設計理由及分析

1. 模糊變數的設計參考老師上課提供的意見，使用語意式模糊規則，並避免參數過多太複雜，因此前件部只取兩個模糊變數分別為，前方距離、左45度距離減右45度距離。後件部的模糊變數則是設定為唯一輸出，方向盤角度。至於措辭集所有模糊變數都是large、medium、small三個的集合。因此我的模糊規則的形式就是: If 前方距離 is {large, medium, small} and 左45度距離減右45度距離 is {large, medium, small}, then 方向盤角度 is {large, medium, small}，而為了全面地考量全部情形，所以規則數就是前件部兩個變數的措辭集之排列組合，總共9種，如下表所示。另外需要注意的是因為左減右距離跟方向盤角度並非都是大於0，會有正負，所以我將這兩個模糊變數的samll定義為偏負，medium代表趨近0，large代表偏正。

前方距離	small	small	small	medium	medium	medium	large	large	large
左45度距離減右45度距離	small	medium	large	small	medium	large	small	medium	large
方向盤角度(使用者設定)	{small, medium, large}	{small, medium, large}	{small, medium, large}	{small, medium, large}	{small, medium, large}	{small, medium, large}	{small, medium, large}	{small, medium, large}	{small, medium, large}

2. 而變數的歸屬函數部分，我選擇用高斯函數，雖然計算可能比較大，但因為有標準差可以控制曲線的坡度，所以我認為比較好。此外我在程式中有稍微修改高斯函數，預設只要是small的模糊變數，其歸屬函數是單調遞減的高斯函數，也就是當x小於平均(mean)時，歸數度都強制設為1，假設我設定mean是5，當x小於5時，歸屬度都是1，這樣其實應該更符合small的認定，因此這樣比一般的高斯函數更能符合事實。如下圖所示。同理可推medium的模糊變數，其歸屬函數是應該是一般高斯函數，large的模糊變數，其歸屬函數是單調遞增的高斯函數。



3. 模糊系統相關的運算方式因為只求方便好做，所以模糊交集選擇最小值，模糊聯集選擇最大值，模糊蘊含選Mamdani implication，這樣過程就可以如課本上化減，去模糊化方式則沒有難易之分，效果好壞要試才知道，所以我就先選擇用離散重心法實作。

D. 實驗結果與分析

1. 參數太多不可能全部用盲測找出一組可行解，所以如課本所說模糊規則的建立要靠人類專家提供，所以我先依據常理選擇後件部的措辭，但人類總有疏漏不足的地方，因此若有問題我打算最後在用嘗試的方式調整。**(注意: 方向盤的small為偏向左轉，medium偏直走，large偏向右轉)**

前件部	後件部
If 前方距離 small and 左減右 small, then...	前方快撞牆了，左邊比右邊小，所以方向盤角度應該要向右轉，才能盡可能避免撞牆，所以先設為large
If 前方距離 small and 左減右 medium, then...	前方快撞牆了，左邊右邊距離差不多，所以方向盤角度向右轉或左都可以，先設為large
If 前方距離 small and 左減右 large, then...	前方快撞牆了，左邊比右邊大，所以方向盤角度應該要向左轉，才能盡可能避免撞牆，所以先設為samll
If 前方距離 mediuml and 左減右 small, then...	前方距離一般，左邊比右邊小，所以方向盤角度應該要向右轉，才能盡可能避免撞牆，所以先設為large
If 前方距離 mediuml and 左減右 medium, then...	前方距離一般，左邊右邊差不多，所以方向盤角度應該先不用轉，所以先設為medium
If 前方距離 mediuml and 左減右 large, then...	前方距離一般，左邊比右邊大，所以方向盤角度應該向左轉，所以設為small
If 前方距離 largel and 左減右 small, then...	雖然前方空間還有，但左邊比右邊小，所以方向盤角度應該要向右轉，為避免繳正不及，所以先設為large
If 前方距離 largel and 左減右 medium, then...	前方空間還很大，左右距離也差不多，所以方向盤角度應該不太需要轉，所以先設mediuml
If 前方距離 largel and 左減右 large, then...	雖然前方空間還很大，但右距比左小，為避免繳正不及，所以方向盤角度應該要先向左轉，所以先設為small

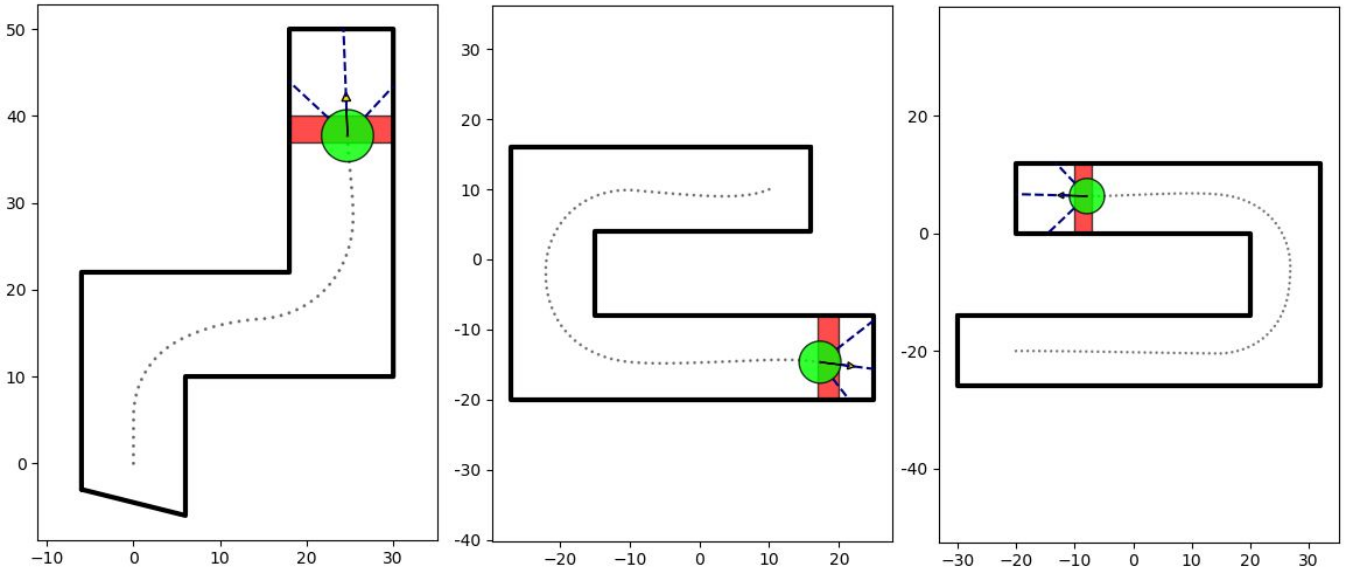
2. 再來是參數的部分，一開始設定時，依樣可以先根據經驗設定，最後若在遇到錯誤時再不斷的調整來解決。最後測試參數、設定結果如下圖所示，已經設定為預設值，在程式開啟時就會設定好，大部分圖可以過，但唯獨下面提到的大轉彎不行。

	mean of small	SD of small	mean of medi...	SD of medium	mean of large	SD of large
Front dist.	3.00	10.000	12.00	5.000	20.00	5.000
L-R dist.	-8.00	5.000	0.00	5.000	6.00	3.000
Result	-10.00	20.000	0.00	21.000	13.00	18.000

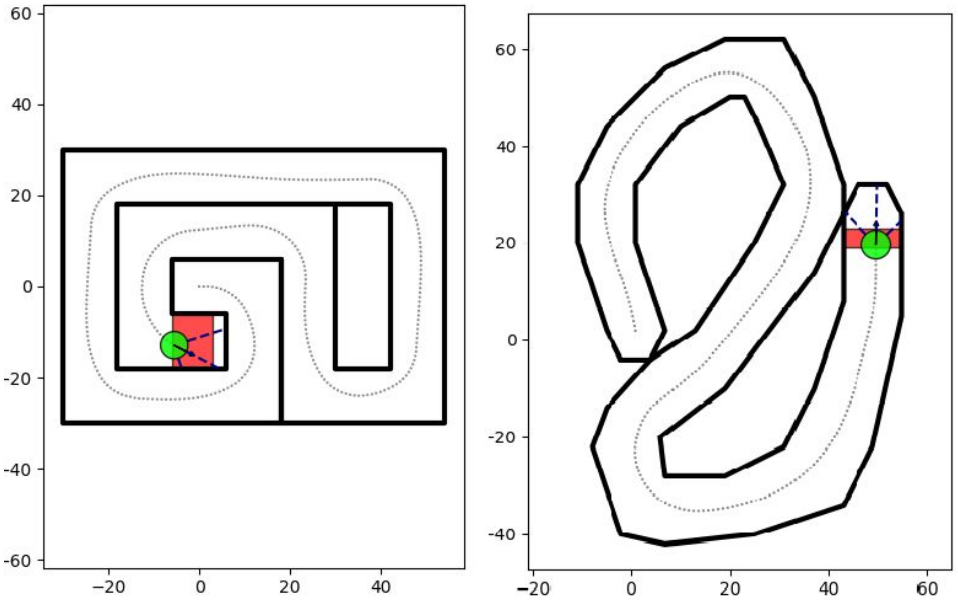
Front dist.	small	small	small	medium	medium	medium	large	large	large
L-R dist.	small	medium	large	small	medium	large	small	medium	large
Result	large	large	small	large	medium	small	large	medium	small

3. 實驗地圖的是由吳艭所提供，只自行額外加入一個地圖測試迴轉空間影響，以下為實驗截圖軌跡

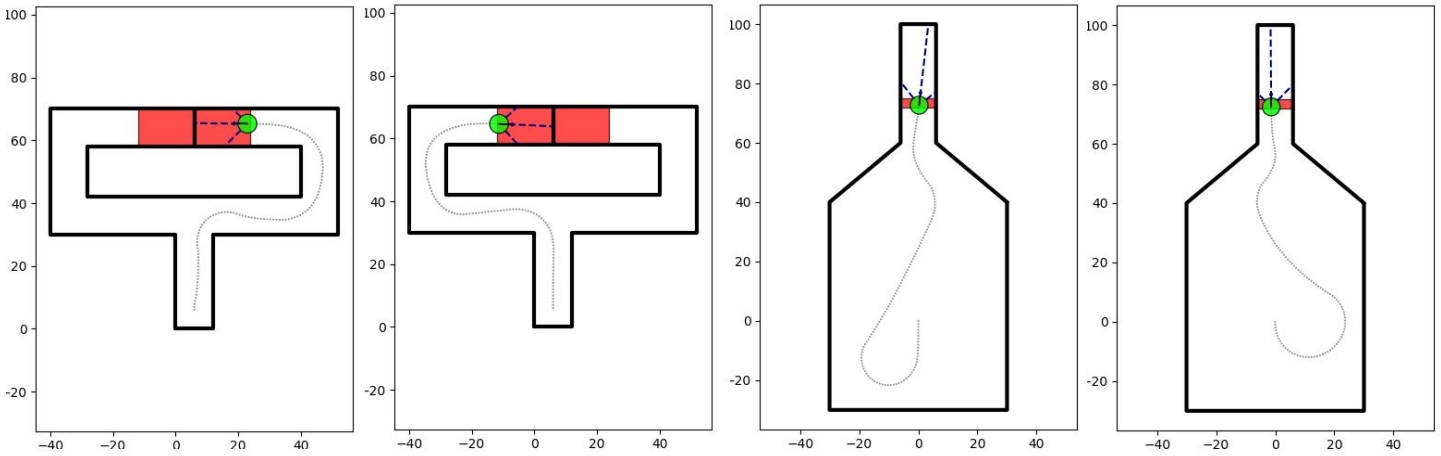
i. 基本轉彎



ii. 連續複雜轉彎



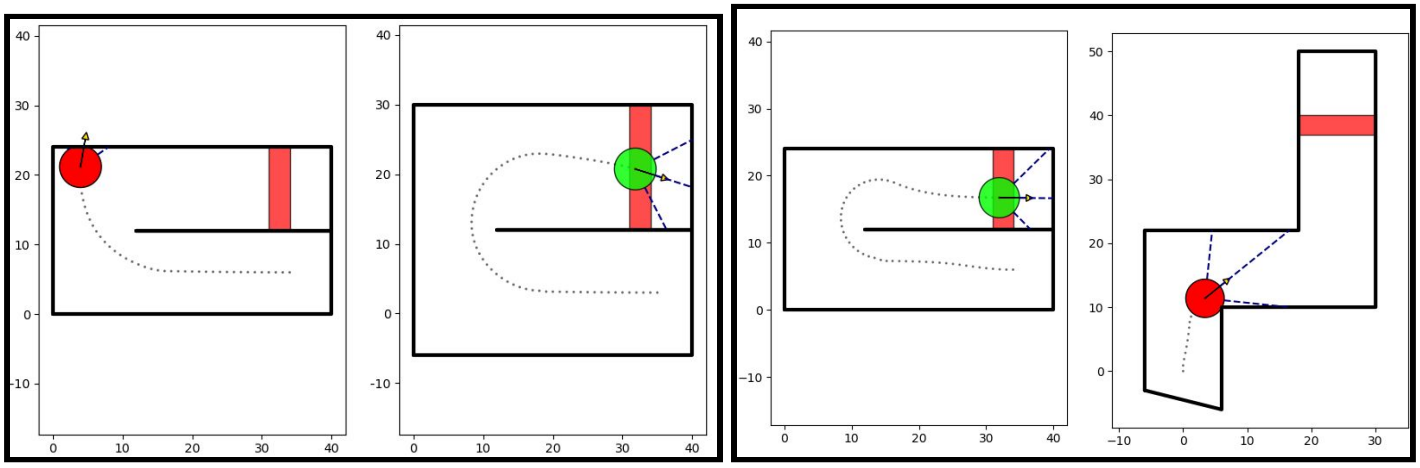
iii. 左右距離相同: 隨著越靠近牆，但左右都等距，這時候會往左轉，還是右轉就和模糊變數L-R 為medium時(代表左右等距)所對應的後件部的設定有關了，以下為兩種轉彎的範例。



原參數轉向為上圖2、3，下面為上圖1、4之設定

Front dist.	small	small	small	medium	medium	medium	large	large	large
L-R dist.	small	medium	large	small	medium	large	small	medium	large
Result	large	small	small	large	small	small	large	small	small

iv. 大轉彎: 若迴轉空間較小，除非特化參數來專門應對，不然無法通過；但特化參數其他正常地圖便會無法通過。



左圖為預設資料，只能通過迴轉空間較大的。右圖顯示特化參數可以通過小迴轉空間，但連基本測資過不了，下列為特化參數。

	mean of small	SD of small	mean of medi...	SD of medium	mean of large	SD of large
Front dist.	25.00	5.000	40.00	3.000	59.00	2.000
L-R dist.	-8.00	5.000	0.00	5.000	6.00	3.000
Result	-5.00	20.000	0.00	21.000	40.00	22.000

Front dist.	small	small	small	medium	medium	medium	large	large	large
L-R dist.	small	medium	large	small	medium	large	small	medium	large
Result	large	large	small	large	large	small	large	large	large