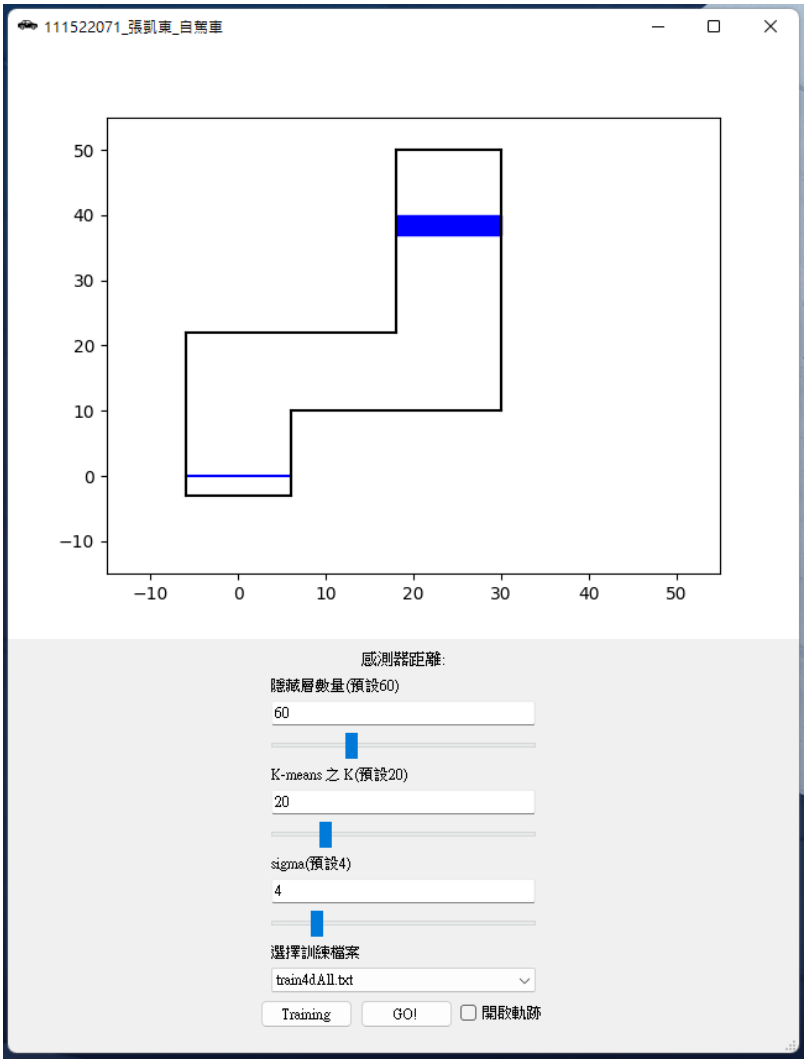
一、　程式介面說明 & 操作說明

介面說明:

主介面從上至下由一張圖片、感測器距離顯示方塊、四個可選擇參數(隱藏層、K-means 之 K、sigma、訓練檔案)、兩個按鈕(訓練、GO)以及一個核取方塊(開啟軌跡)組成。四個參數的預設值是跑過 4d 以及 6d 後都成功才選出來的數字，直接使用預設值下去訓練出的 model 沒意外可以成功走到終點。



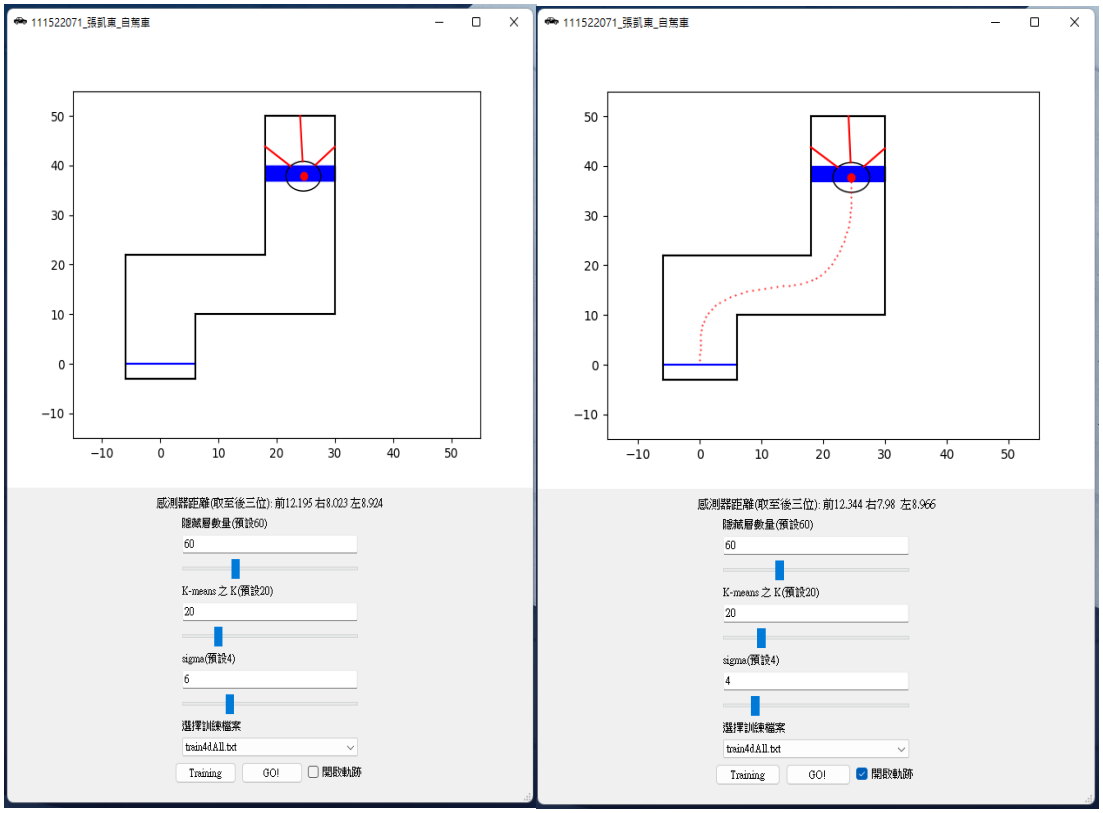
圖一、主介面

操作說明:

程式開始運行後會先依照預設值訓練一個預設 model(隱藏層 60、K=20、sigma=4、file="train4dAll.txt")，所以開啟時會停頓 1~2 秒(預設 model 訓練完後才會顯示視窗)，主視窗顯示後可以直接點擊"GO"按鈕讓自駕車使用預設 model 做移動。

想訓練新的 model 就把四個參數調整完後按下"Training"，調整的方式可以直接

輸入數字或是拖曳 silder 更改數值。Training 期間兩個按鈕會 disable，直至訓練完成後才會再次 enable，訓練完成後就可以按下”GO”用剛剛訓練完的 model 讓自駕車移動。(訓練注意: 參數請用整數；隱藏層數必須>=K ；K 越大訓練時間會越久，K 設置上限 100 時，訓練時間會落在 15 秒左右，隱藏層數量跟 sigma 的影響則不大)



圖二、設有”開啟軌跡”的核取方塊，勾選時會將車子軌跡一併畫出，取消勾選時則無軌跡。

二、 程式碼說明

main.py:

運行視窗應用程式，讀入 MyGUI.ui 並顯示主介面，import Playground、MyRBFN 等 class 做使用，可以訓練 model 也可以運行 model 並將結果視覺化於主介面。

MyRBFN.py:

模型主要架構，包含讀入訓練檔案、基底函數計算、虛擬反置矩陣運算、模型訓練、模型預測等等。

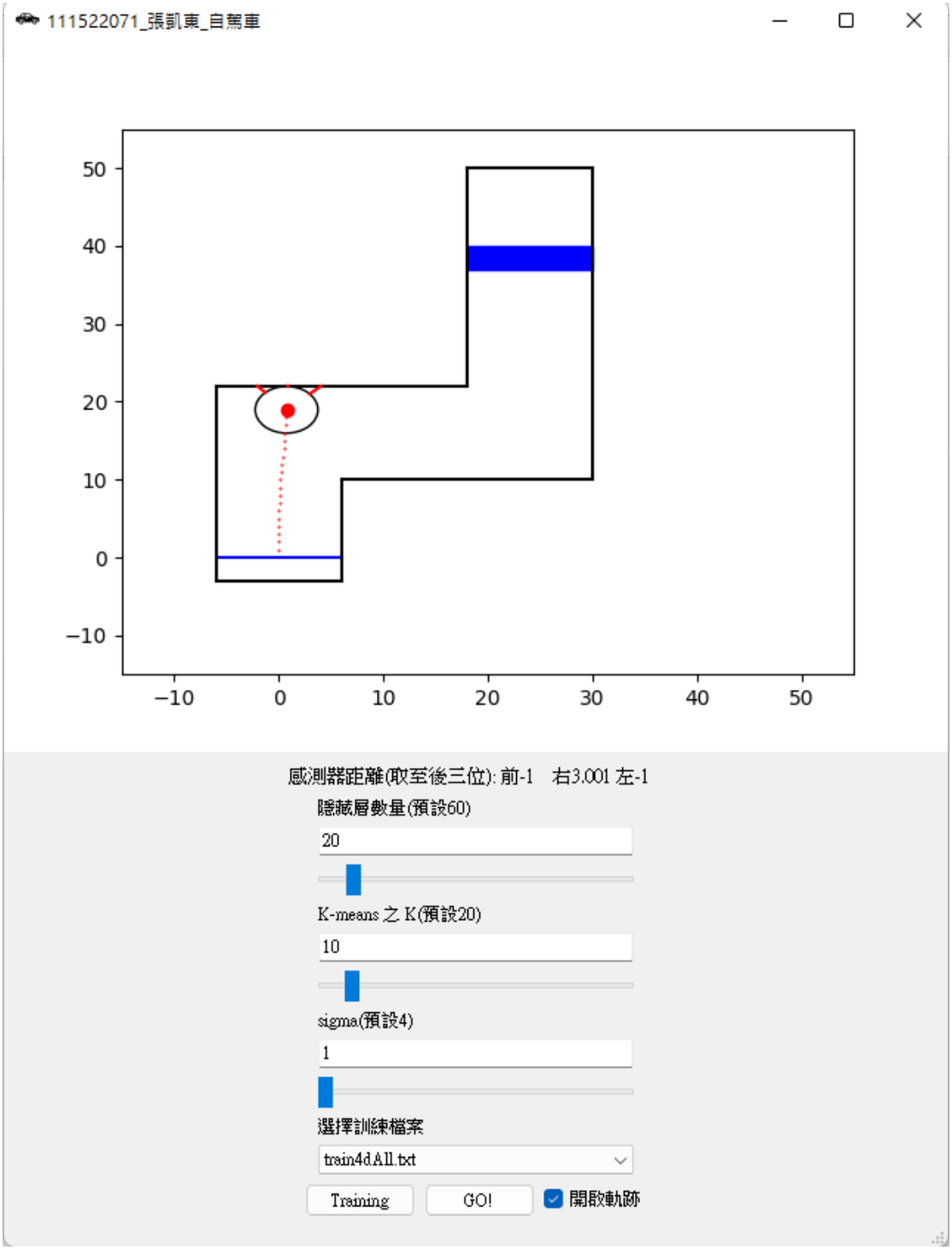
MyKMeans.py:

簡單寫的一個 K-Means 演算法，根據給定的 K 值選出 K 個簇中心，如果第 t+1 次分類完的簇中心等同於第 t 次則會馬上結束分群，或是達到最大分群次數 50 次也會結束分群。每次會選出一個資料點與當前的簇中心做歐式距離計算，並分配到最適合的簇。

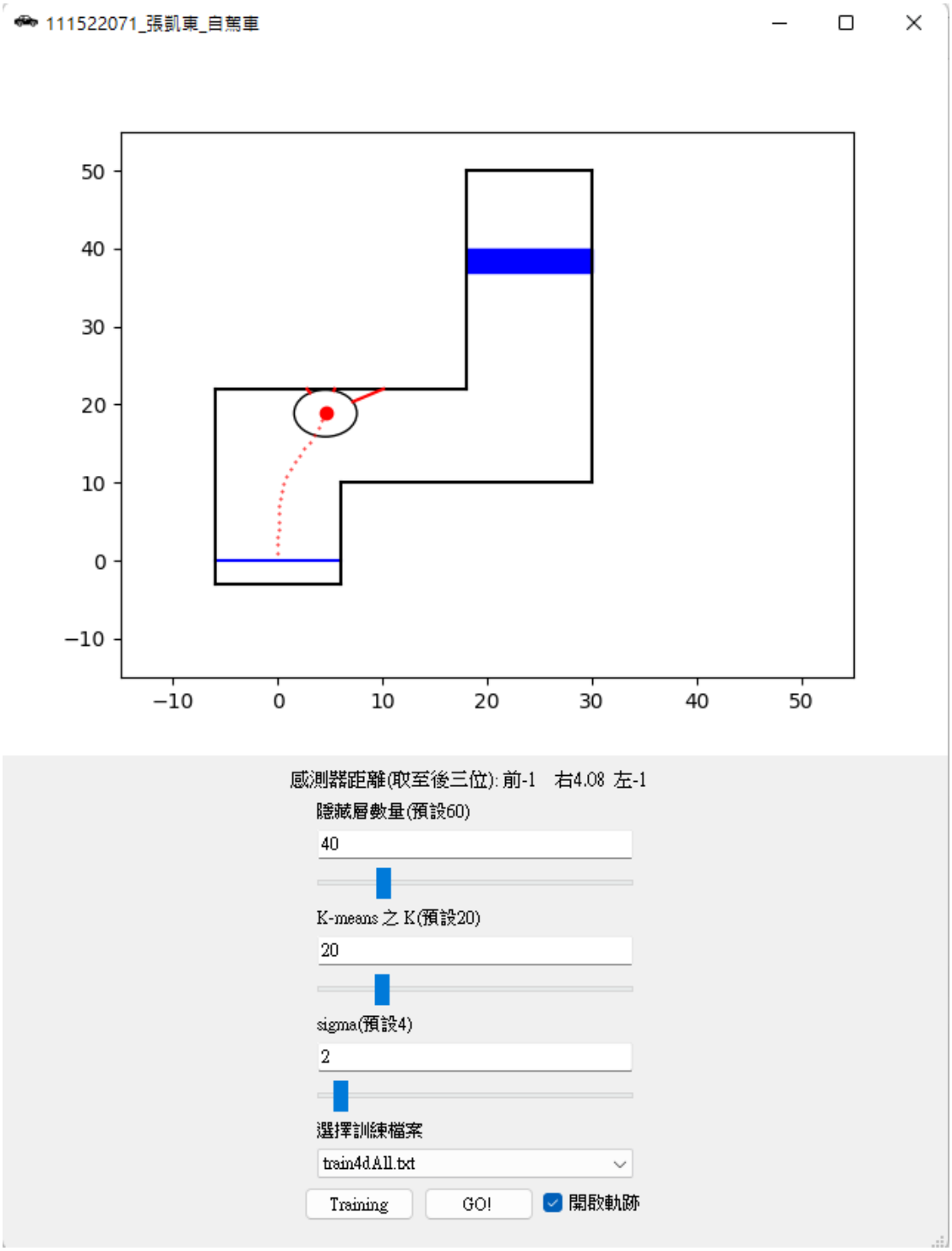
simple_playground_new.py & simple_geometry.py

經過微調後的助教給的 code。

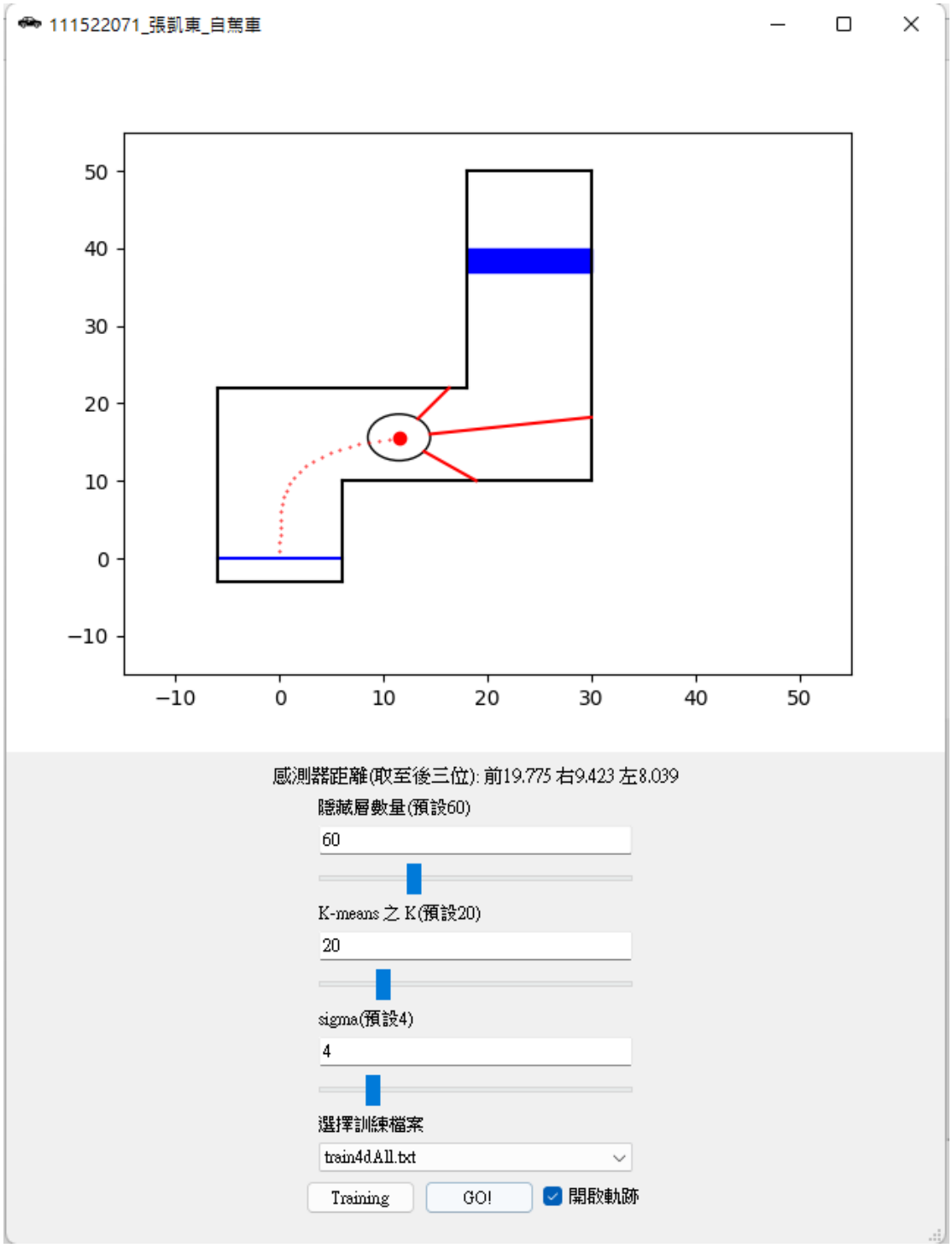
三、　　實驗結果(包含移動軌跡截圖)



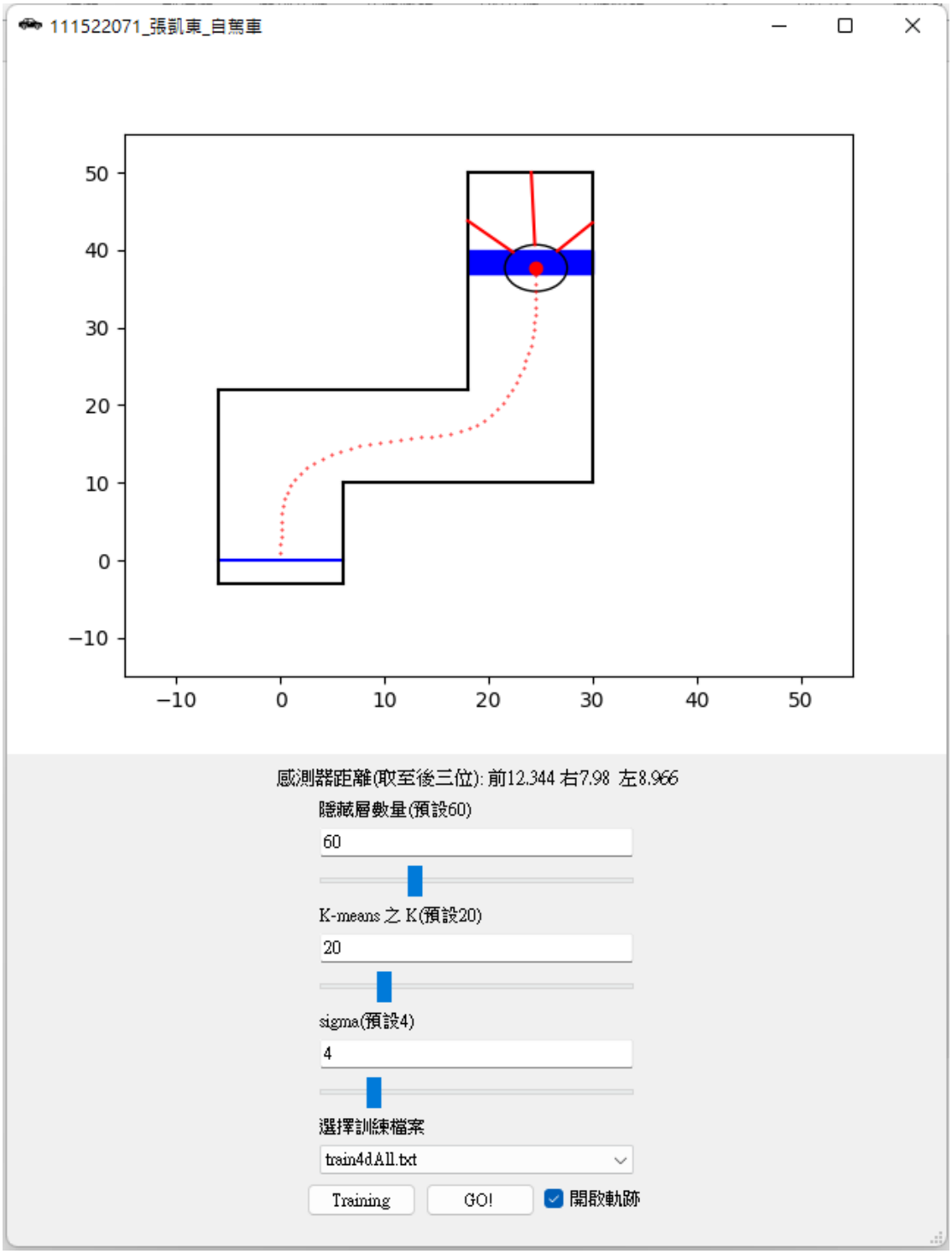
最一開始使用 20 層以及 K=10、sigma=1 去做訓練，發現車子只能微微右轉但免不了撞牆。



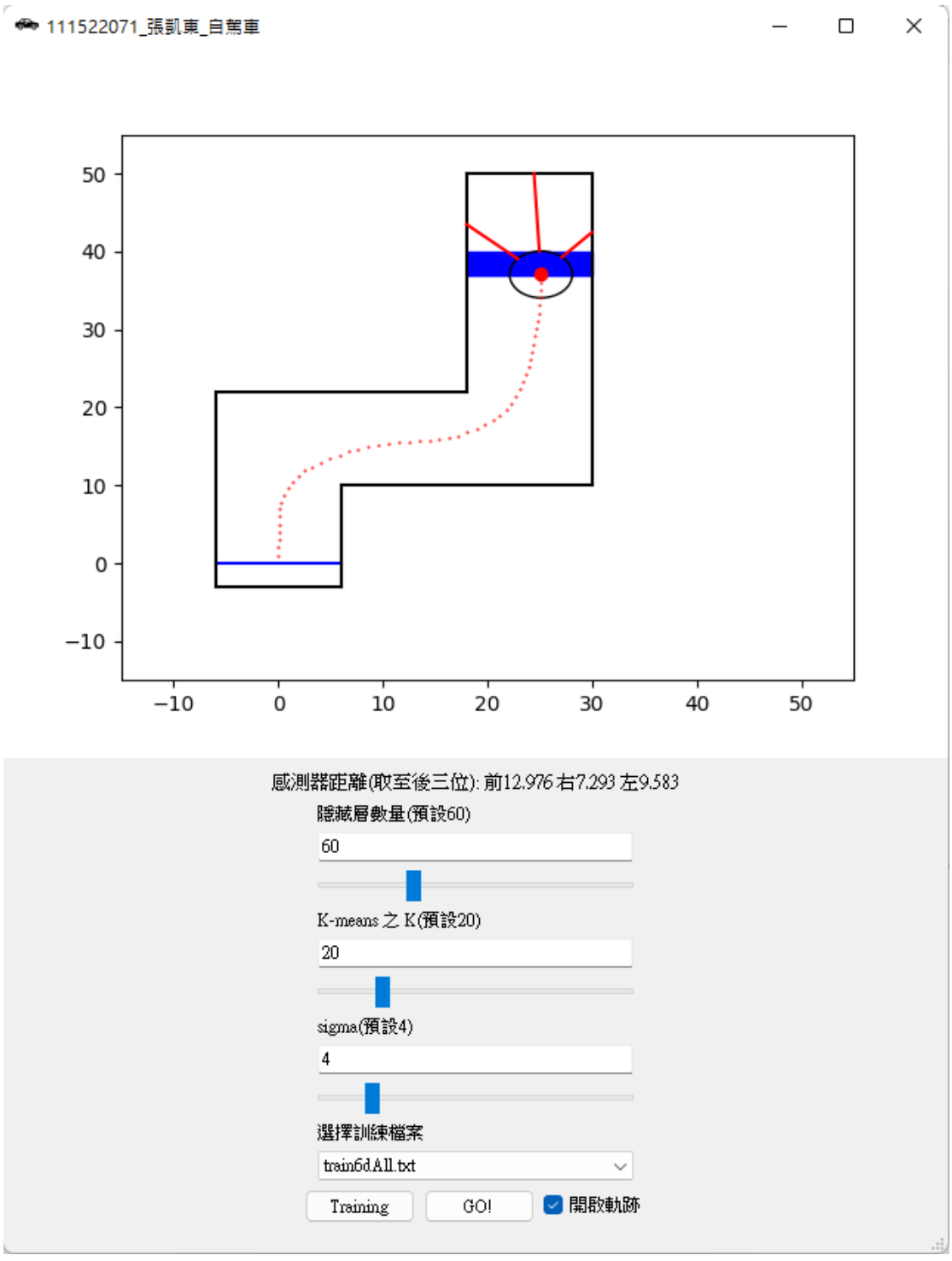
再來調高 K 以及 sigma 讓 model 可以模擬出更接近原始數據的預測，車子右轉的弧度更大了。



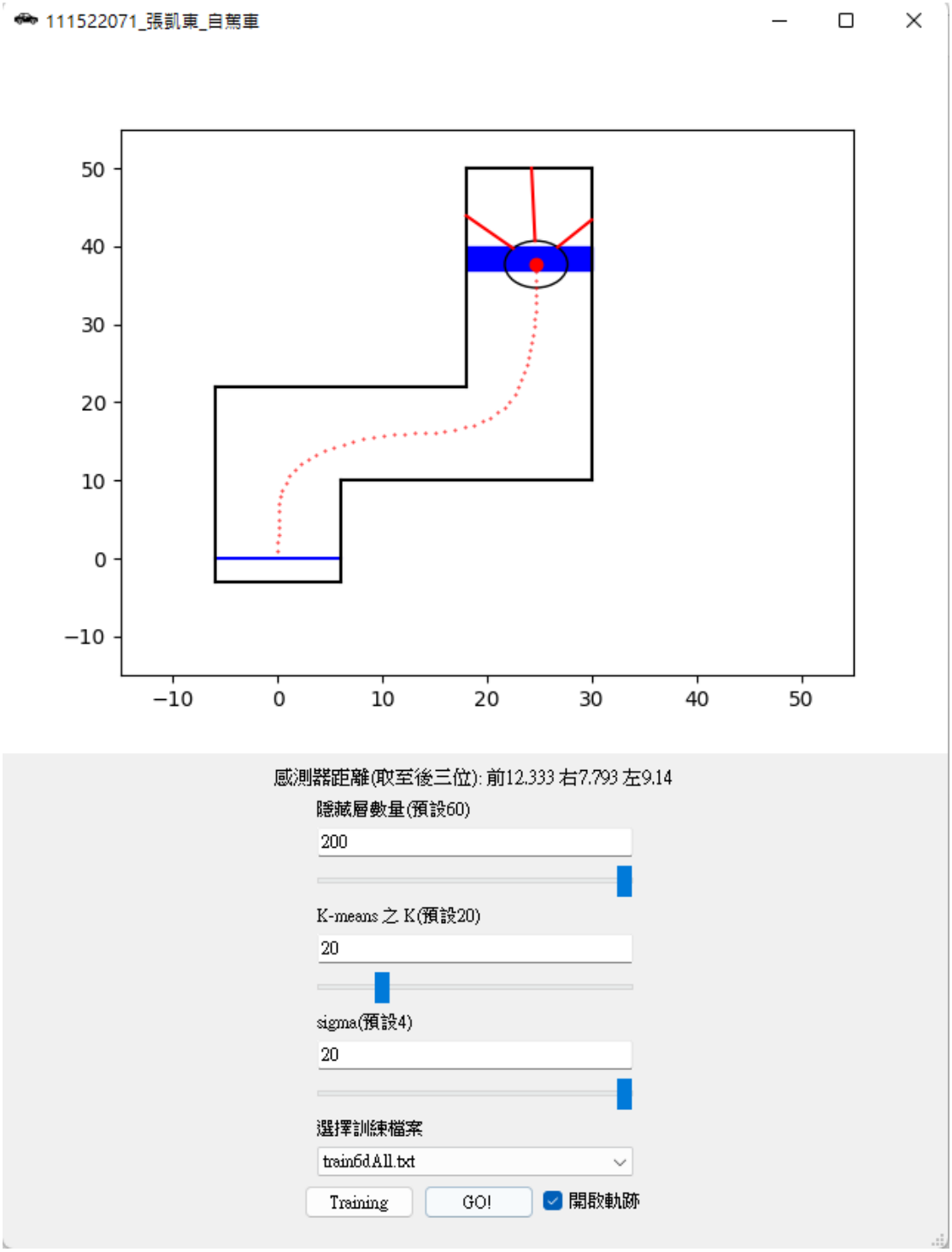
最後則是發現在隱藏層 60 以及 K=20、sigma=4 的情況下成功彎過第一個彎，並也成功彎過第二個彎。


111522071_張凱東_自駕車
50
40
30
20
10
0
−10
−10
0
10
20
30
40
50
感測器距離(取至後三位): 前12.344 右7.98 左8.966
隱藏層數量(預設60)
60
K-means 之 K(預設20)
20
sigma(預設4)
4
選擇訓練檔案
train4dAll.txt
Training
GO!
開啟軌跡

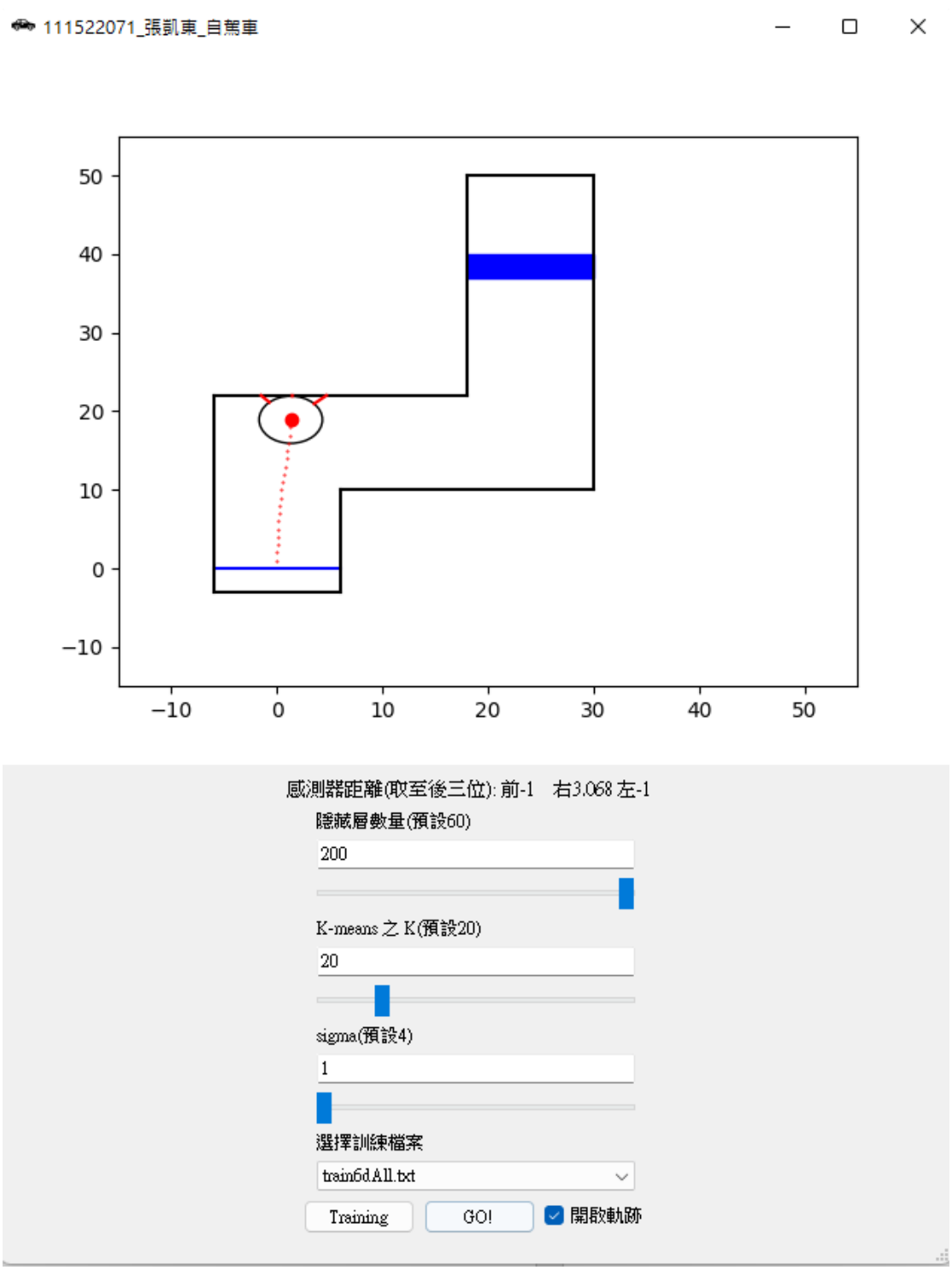
成功到達終點。



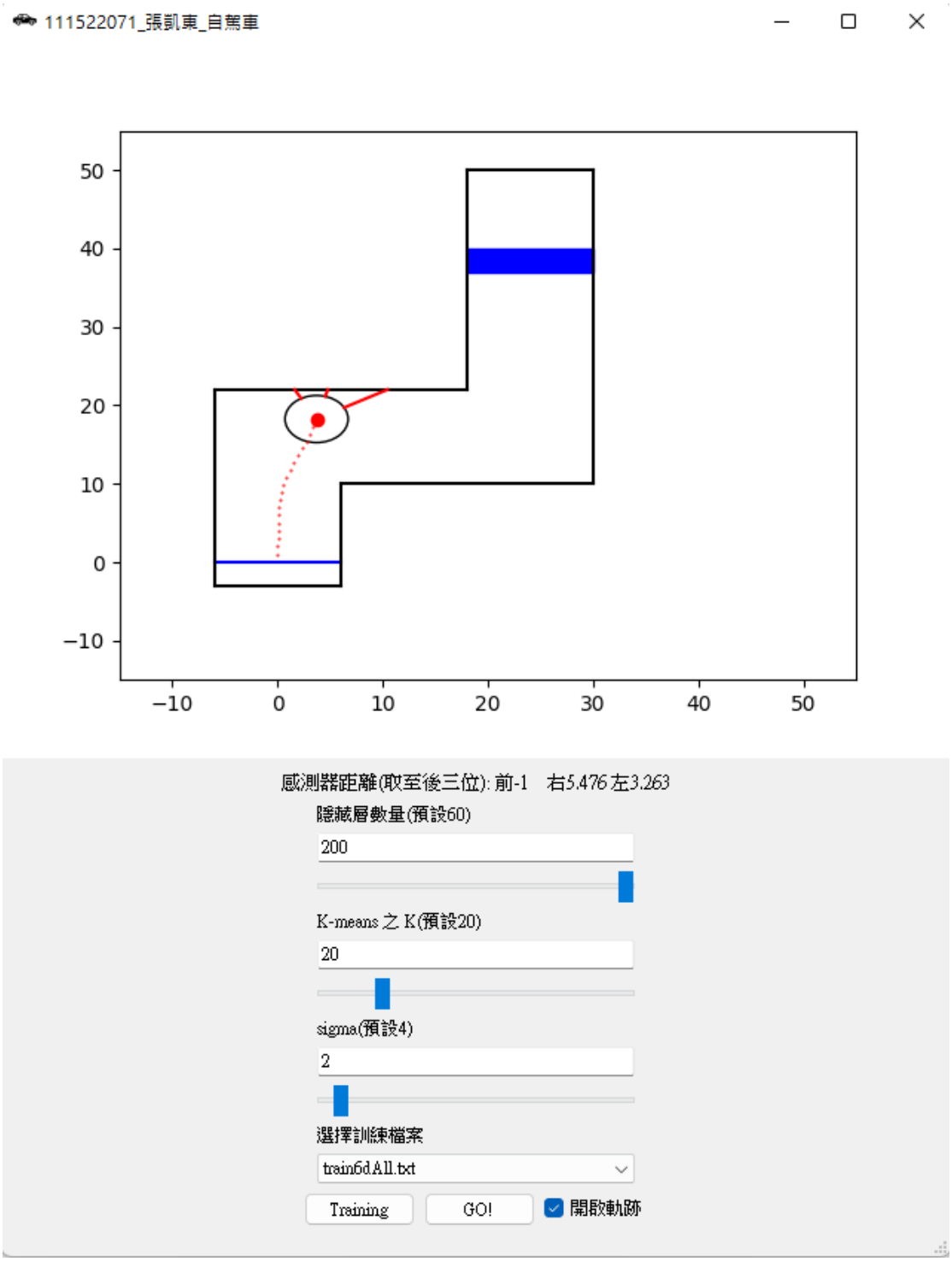
直接使用 60/20/4 的配置去訓練”train6dAll.txt”也發現可以直接到達終點。



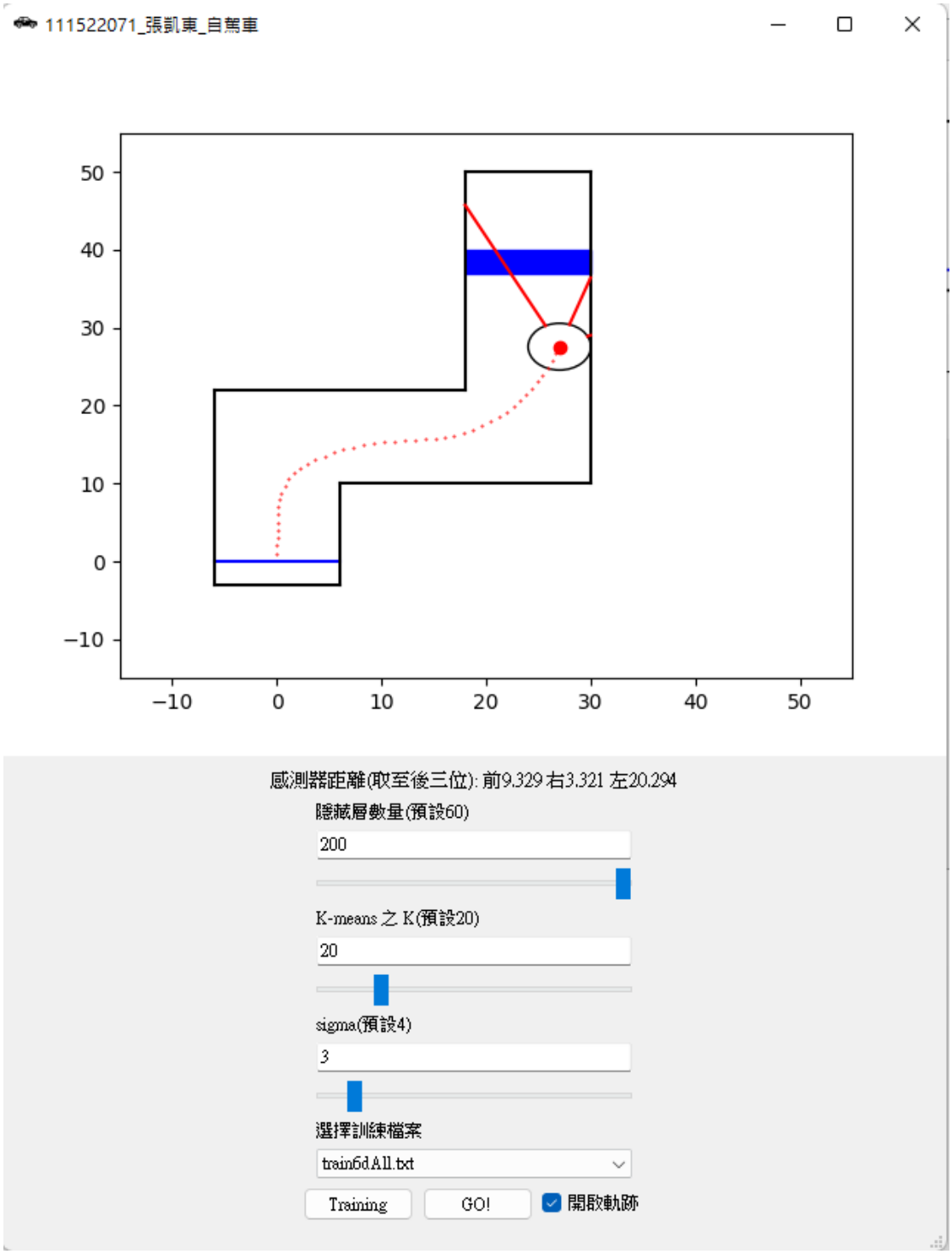
進一步將隱藏層及 sigma 調到 200/20 的配置，發現過彎有稍微更滑順的感覺。



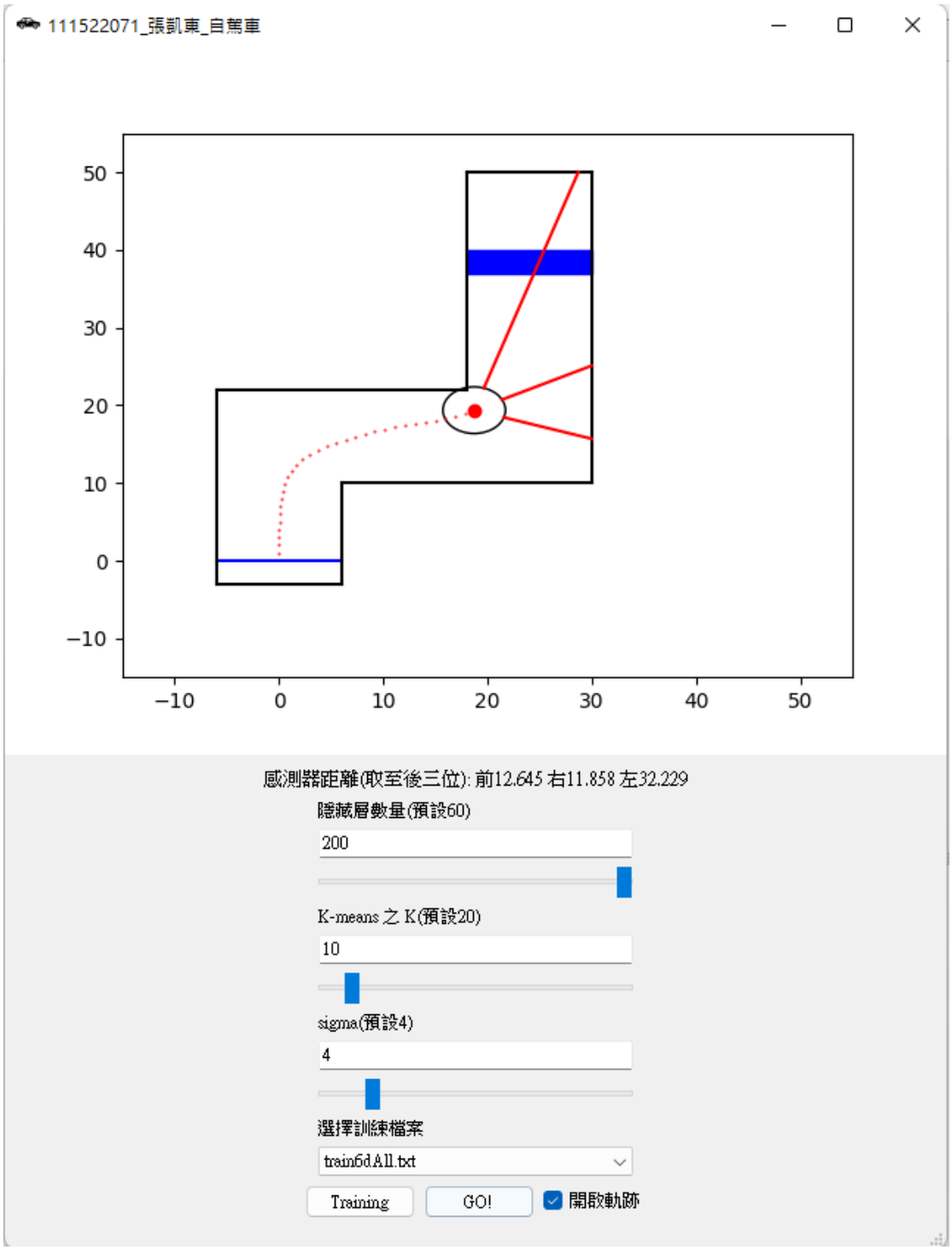
回頭將 sigma 調回 1 發現直接撞爛。


111522071_張凱東_自駕車
50
40
30
20
10
0
−10
−10
0
10
20
30
40
50
感測器距離(取至後三位): 前-1 右5.476 左3.263
隱藏層數量(預設60)
200
K-means 之 K(預設20)
20
sigma(預設4)
2
選擇訓練檔案
train6dAll.txt
Training
GO!
開啟軌跡

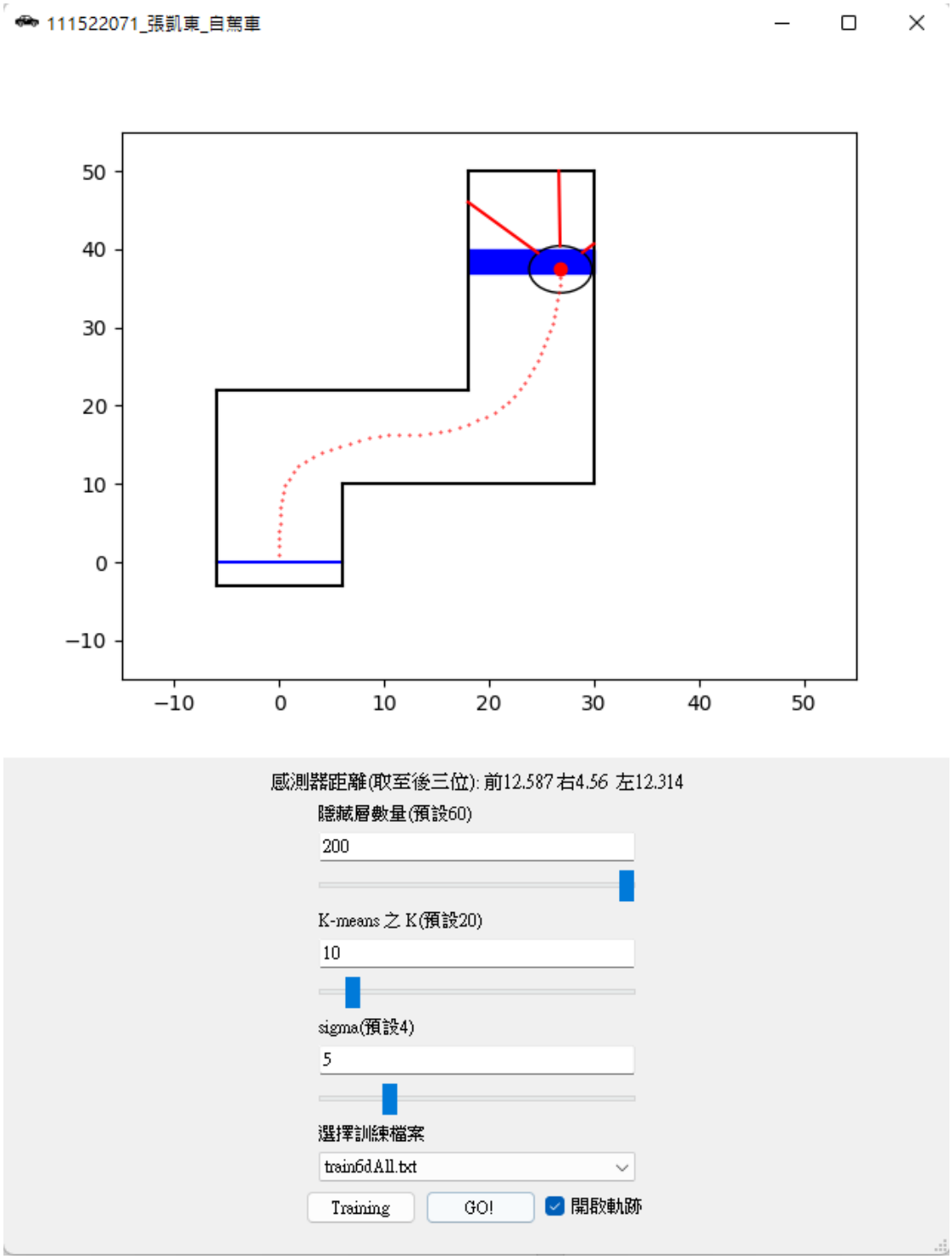
Sigma=2 也是撞爛。



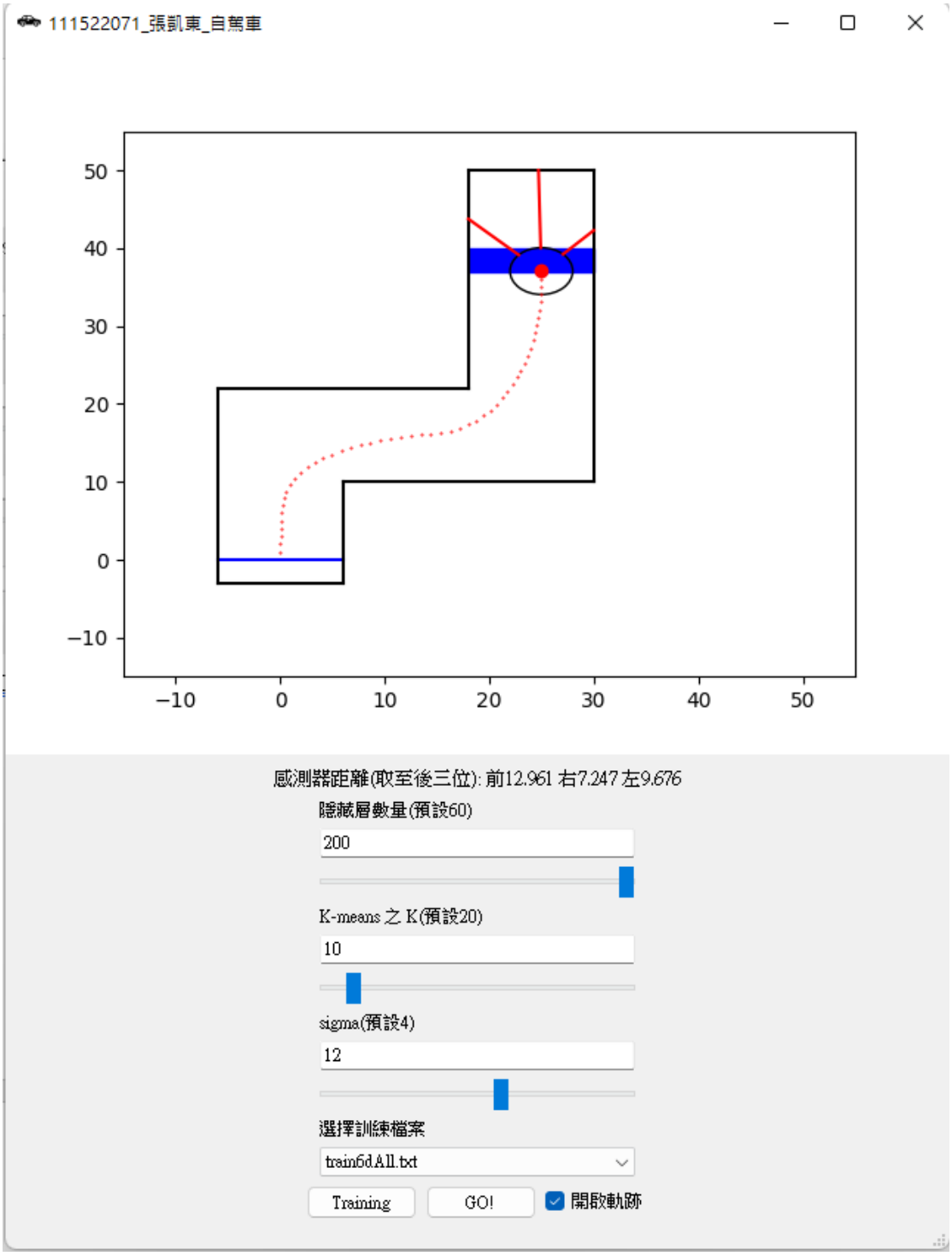
Sigma=3 雖然過了第一個彎但第二個彎直接外拋撞爛。



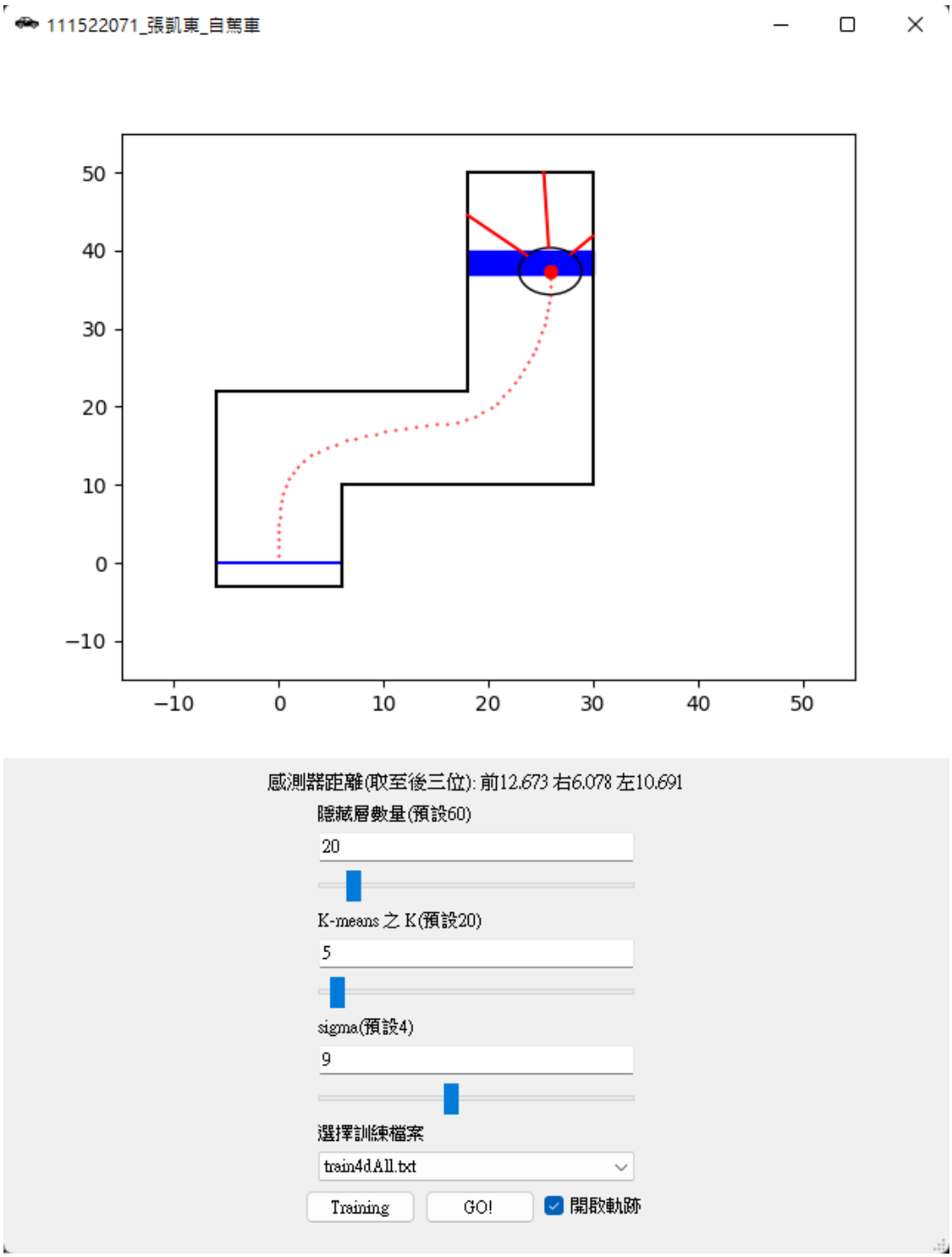
即使隱藏層=200，sigma=4，K 太小也會擦撞到牆壁。



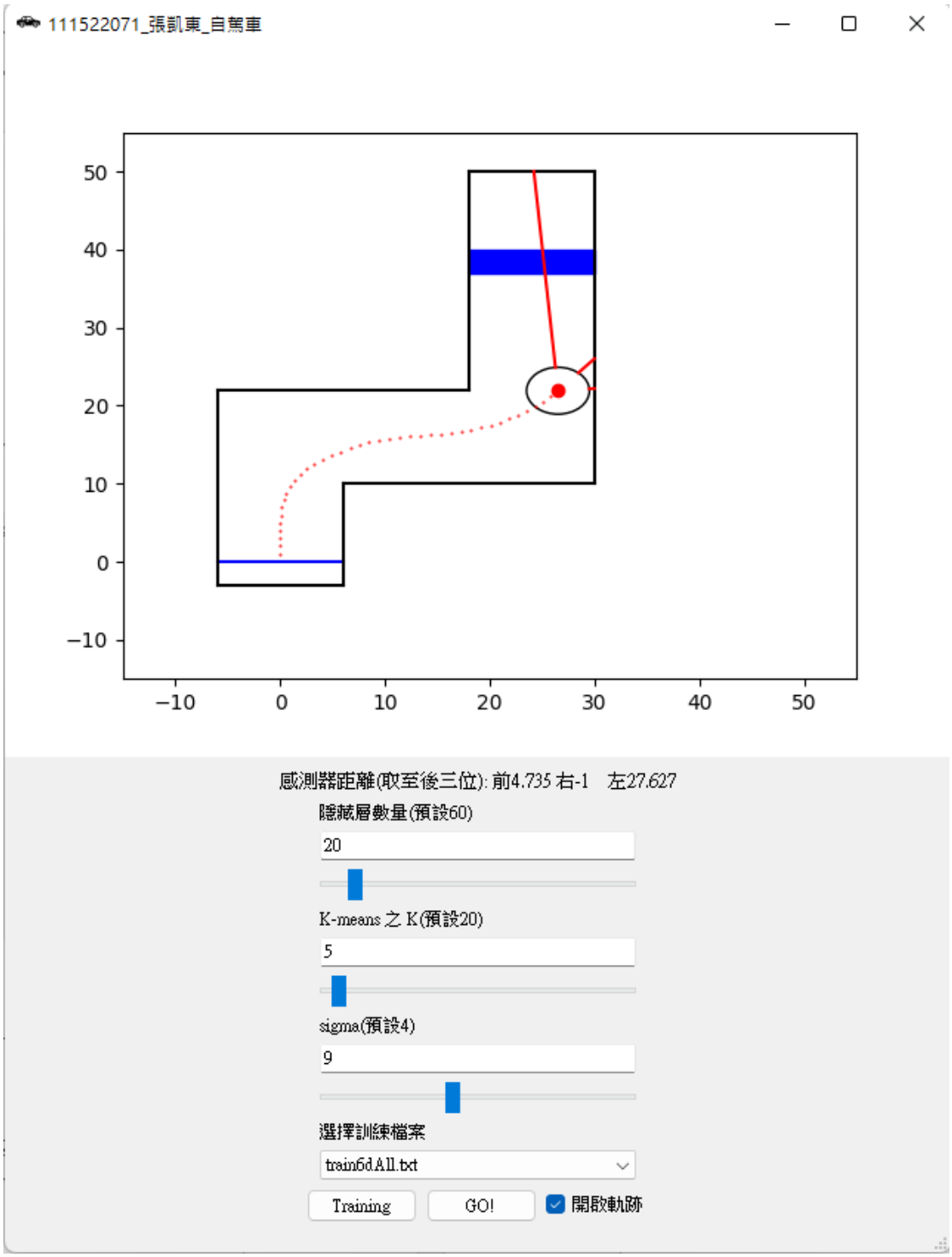
把 sigma 調大可以成功把外拋的車子龍頭再轉回來。



再次調大 sigma 發現，即使 K 不理想加大 sigma 依然有機會可以讓模型學出正確答案。



相同為比較艱難的設定下”train4dAll.txt”可以走到終點。



相同為比較艱難的設定下”train46All.txt”走不到終點。

四、 分析

這次的作業我用的是 RBFN 模型並使用 K-means 做群聚分析，基底函數的選擇如圖三，更新 model 時使用虛擬反置矩陣更新參數。

$$\varphi_j(\underline{x}) = \exp\left(-\frac{\left\|\underline{x} - \underline{m}_j\right\|^2}{2\sigma_j^2}\right)$$

圖三、選擇之基底函數

在一連串實驗後發現，相比於隱藏層的數量，RBFN 在選擇 K 跟 Sigma 時更為重要，不好的 K 無法找出 Ground Truth 的分布，太小的 sigma 無法讓模型預測出更靈活的答案，導致出現方線盤轉不過去的現象發生。

這其中我覺得 sigma 的重要程度又大於 K，因為 sigma 能直接反映出模型的靈活度，越靈活的模型越能找到最佳解，上面也有實驗發現相同的 K 不同的 sigma，比較大的 sigma 表現較好。

至於訓練檔案的選擇，我發現 4d 比較容易走到終點，6d 因為要考慮車子座標反而不容易走到終點，我覺得是因為維度從四維提升到六維的情況下，需要預測的 Ground Truth 更複雜，所以 6d 需要更大的 sigma 跟 K 才能得到比較好的表現，相同設定下 6d 表現則會比 4d 差。

五、 結語

以上是這次的作業說明，因為是使用 RBFN 實作再請助教幫我額外加分，謝謝。