國立成功大學

電機工程學系

畢業專題

RRT*全域路徑規劃與 CBF-CLF 控制器

實現移動機器人自主導航

學生：陳致融

指導教授：鄭銘揚

日期 112 年 9 月

摘要

自主移動機器人技術是近年來機器人發展的重要課題之一，若希望機器人能夠根據周遭環境變化採取對應措施，做到自主移動能力，那麼路徑規劃方法會是移動機器人技術中重要的一個環節。在移動機器人中，要如何在有障礙物的環境中，規劃出一條合適的無障礙物的路徑，讓移動機器人能從起始點移動到達目標點且確保路徑是最短的，是一個很值得討論的議題。本專題主要提出地圖建構方法與兩種路徑規劃的方法，分別是 RRT* 與 CBF-CLF 控制器去實現路徑規劃。

第一章會介紹使用的移動機器人 Turtlebot3 burger，零件的組成與設備規格，接著闡述會使用到的移動機器人運動模型。

第二章首先將討論如何基於 RBPF 理論去實現 SLAM 並使用 Binary Occupancy maps 的方式來描述障礙物地圖。接著介紹 RRT 與 RRT*演算法的差異，再來使用 RRT*演算法實現路徑規劃，提供一種在本專題裡全域路徑規劃上避開障礙物的方法，驅動機器人部分則使用純追蹤演算法。

第三章將提供基於 CBF-CLF (Control Barrier Function - Control Lyapunov Function) 方法結合二次規劃(Quadratic Programming)設計出的控制器(CBF-CLF-QP)，運用 CBF-CLF-QP 在本專題上作為一種局域路徑規劃的方法。

第四章首先將介紹本專題的實驗環境，利用 LiDAR 將收回的環境資料在 Matlab 端構建一組 Occupancy grid maps，得到地圖後分別執行 RRT*搭配純追蹤演算法，以及利用 CBF-CLF-QP 控制器，這兩種方法的路徑規劃模擬，最後搭配上機實驗與這兩者的結果討論。第五章則是為本專題做一個結論與未來展望。

目錄

表目錄

圖目錄

第一章 移動機器人概述

1.1 硬體設備規格

各硬體的系統架構主體為車子(Turtlebot3 burger)，組合機件有底盤、馬達、輪子，車子上搭載的是 raspberry pi3 b+與 LiDAR 傳感器，需要的作業系統為 ubuntu20.04，以下圖 1.1、表 1 是我們本次專題所使用的移動式機器人的外觀與規格。

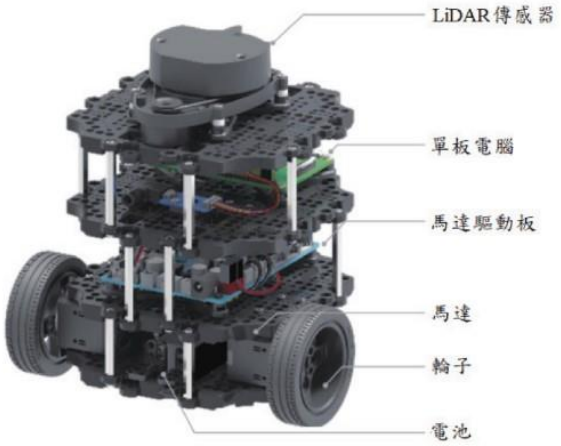


圖 1.1 機器人外觀

表 1 移動機器人規格

最大直行速度	0.22 (m/s)
最大旋轉速度	2.84 (rad/s)
最大載重	15 (kg)
尺寸(長*寬*高)	138 * 178 * 192 (mm)
機體重量	1 (kg)
爬坡高度	10 (mm)以下
單板電腦	Raspberry Pi3 Model B
雷射測距感測器	LDS-01
續航時間	2.5 (hr)
電源消耗	3.3 (V) / 800 (mA)
電池	11.1 (V) 1800 (mAh)

1.2 機器人運動模型

本專題使用杜賓斯車(Dubins car)模型。特別注意的是，這個模型原本是四輪車子的運動模型，如圖 1.2 所示。

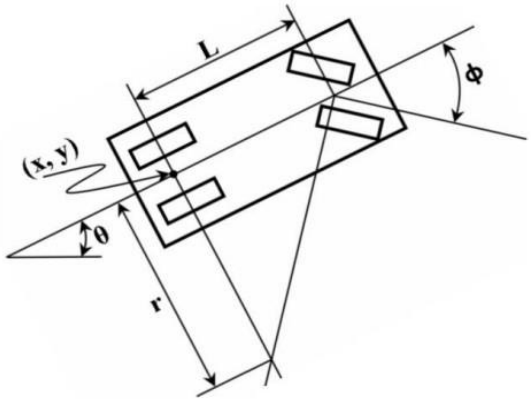


圖 1.2 杜賓斯車模型

定義角速度為 ω，輪胎的半徑 r、速度 v、後輪夾角 θ 、前後輪夾角 Ø 與中心點 (x,y)，假設驅動車子且條件是在純滾動與無滑動的情況下，杜賓斯車的動力學方程式如式(1.1)表示。

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \frac{v}{L}\tan\emptyset \end{bmatrix} \quad (1.1)$$

由於本專題移動式機器人的運動特性是利用兩輪速差來驅動，所以需要把距離 L 忽略不計，如圖 1.3 所示。

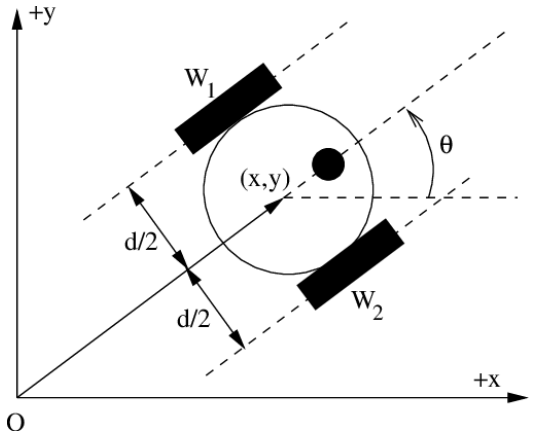


圖 1.3 移動機器人

假設驅動車子且條件是在純滾動與無滑動的情況下，x 和 y 方向的速度取決於車子的速度和角度，角度取決於角速度，用矩陣定義它們的關係，如式(1.2) :

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \end{bmatrix} \quad (1.2)$$

第二章 基於 RRT* 之移動機器人路徑規劃與自主導航方法

本章節首先將討論如何基於 RBPF 理論去實現 SLAM 並使用 Binary Occupancy maps 的方式來描述障礙物地圖。接著介紹 RRT 與 RRT*演算法的差異，再來使用 RRT*演算法實現路徑規劃，提供一種在本專題裡全域路徑規劃上避開障礙物的方法，驅動機器人部分則使用純追蹤演算法。

2.1 建構地圖

2.1.1 Binary Occupancy Maps

建構障礙物地圖的方式採用 Occupancy grid maps，是使用 Gmapping-SLAM 演算法所建構的地圖格式去建構地圖，它是幾何建圖 (Geometrical Mapping) 的一種，它的觀念是將整個環境切割成很多的網格(grid)，且每一個網格負責紀錄環境中某一個區域的狀態，如圖 2.1、2.2 所示。

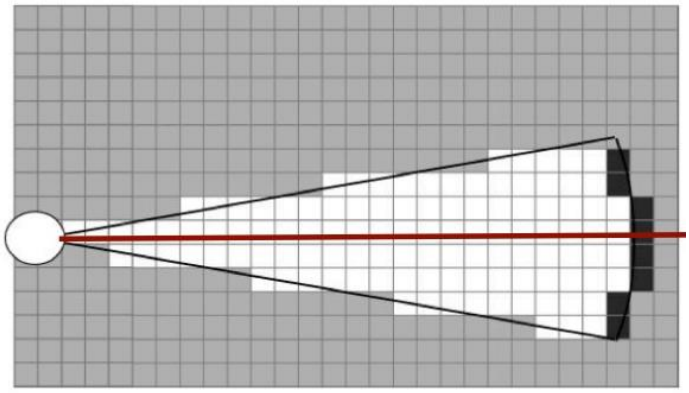


圖 2.1 網格切割圖

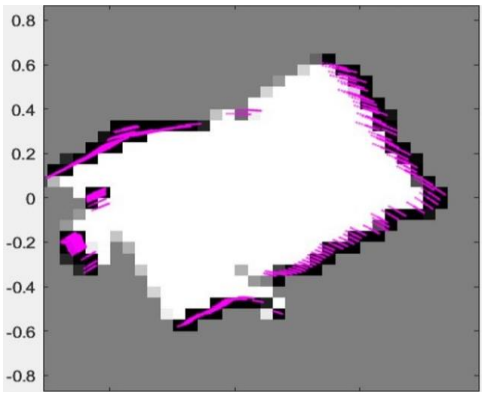


圖 2.2 定點掃描

圖 2.1 黑線畫出的扇形區域代表 LiDAR 掃到的範圍，而白色區域代表無障礙空間，黑色代表障礙空間，灰色區域代表未定空間。圖 2.2 是上機執行且定點掃描的示意圖，紫色區域是 LiDAR 掃到障礙物座標，經轉換後變為一組 Occupancy grid maps。

2.1.2 建立障礙物地圖

本專題在建立障礙物地圖上是基於 RBPF 理論(Rao-Blackwellized Particle Filter)實現 Gmapping-SLAM 並用 Occupancy Grid Maps 建構地圖。首先，Occupancy Grid Maps 的模型是將整個場景分成許多的小網格 m_i，並且用 m 表示整張地圖，每個小網格都有兩個值 0 或 1，0 代表白色，1 代表黑色。計算地圖時我們定義以下式(2.1) :

$$p\left(m\middle|z_{1:t},x_{1:t}\right) \tag{2.1}$$

假設初始狀態 x_0，此式代表到目前為止所有的狀態 $x_{1:t}$ 以及觀測資料 $z_{1:t}$ 給定的情況之下來算出整個地圖的機率。

因為地圖有很多的網格，假設地圖的小網格有 i 組，其可能就會有 2^i 組，所以為了之後的計算需要先假設這些小網格彼此獨立，因此藉由式(2.1)與式(2.2)可以合併推得式(2.3)。

$$p(m)=\prod_i p(m_i) \tag{2.2}$$

$$p\left(m\middle|z_{1:t},x_{1:t}\right)=\prod_i p\left(m_i\middle|z_{1:t},x_{1:t}\right) \tag{2.3}$$

接著把式(2.1)透過貝式定理得式(2.4)，然後再用馬可夫鏈得式(2.5)，其中 $P(z_t|m_i,x_t)$可以再用貝式定理得式(2.6)，代入式(2.5)去推得式(2.7)，從式(2.7)可以發現因為馬可夫鏈的關係少了一筆資料 x_t 不會影響結果，所以可以把 $p(m_i|x_t)$ 的條件拿掉，再度化簡後得式(2.8)。

$$p\left(m_i\middle|z_{1:t},x_{1:t}\right)=\frac{p\left(z_t\middle|m_i,z_{1:(t-1)},x_{1:t}\right)p\left(m_i\middle|z_{1:(t-1)},x_{1:t}\right)}{p\left(z_t\middle|z_{1:(t-1)},x_{1:t}\right)} \tag{2.4}$$

$$p\left(m_i\middle|z_{1:t},x_{1:t}\right)=\frac{p\left(z_t\middle|m_i,\ x_t\right)p\left(m_i\middle|z_{1:t-1},x_{1:t-1}\right)}{p\left(z_t\middle|z_{1:t-1},x_{1:t}\right)} \tag{2.5}$$

$$p\left(z_t\middle|m_i,x_t\right)=\frac{p\left(m_i\middle|z_t,x_t\right)p\left(z_t\middle|x_t\right)}{p\left(m_i\middle|x_t\right)} \tag{2.6}$$

$$p\left(m_i \middle| z_{1:t}, x_{1:t}\right) = \frac{p\left(m_i \middle| z_t, x_t\right) p\left(z_t \middle| x_t\right) p\left(m_i \middle| z_{1:t-1}, x_{1:t-1}\right)}{p\left(m_i \middle| x_t\right) p\left(z_t \middle| z_{1:t-1}, x_{1:t}\right)} \quad (2.7)$$

$$p\left(m_i \middle| z_{1:t}, x_{1:t}\right) = \frac{p\left(m_i \middle| z_t, x_t\right) p\left(z_t \middle| x_t\right) p\left(m_i \middle| z_{1:t-1}, x_{1:t-1}\right)}{p\left(m_i\right) p\left(z_t \middle| z_{1:t-1}, x_{1:t}\right)} \quad (2.8)$$

由於想要透過前一個時間點的地圖去推得目前時間點的地圖，也就是 t-1 時間要推向 t 時間的地圖，這裡定義一個計算機率的發生比（Log Odds Ratio）如式(2.9)：

$$l(x) = log \frac{p(x)}{1 - p(x)} \quad (2.9)$$

把式(2.8)代入式(2.9)的 p(x)，可以得到式(2.10)，形成一個遞迴式後就是我們想要的結果了，也就是從 t-1 時間可以推導出 t 時間的地圖機率。

$$l\left(m_i \middle| z_{1:t}, x_{1:t}\right) = l\left(m_i \middle| z_t, x_t\right) + l\left(m_i \middle| z_{t-1}, x_{t-1}\right) - l\left(m_i\right) \quad (2.10)$$

回到圖 2.1 且觀察紅線，可以看到黑色方格代表有障礙物存在。按照 Occupancy Grid Maps 的機率模型轉換，得到圖 2.3，縱軸是機率，橫軸是 LiDAR 前方距離。圖 2.3 可以看到 z 為測量距離，從區間 0 到 $z + d_1$ 之間訂為白色區域；區間 $z + d_1$ 到 $z + d_2$ 之間訂為黑色區域，之後的區間沒有收到資訊就定為灰色區域。

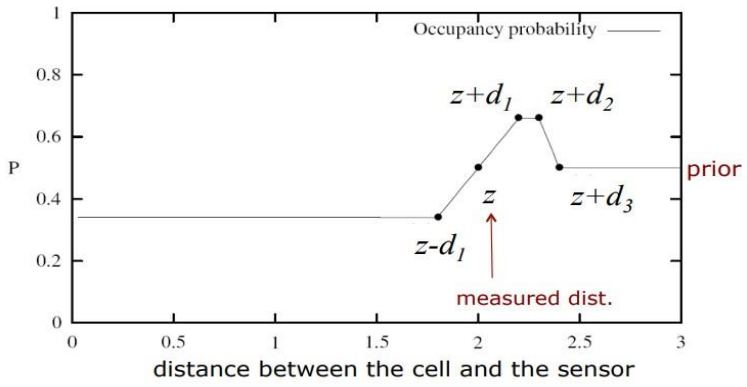


圖 2.3 結合機率分配圖

2.2 基於 RRT*路徑規劃

2.2.1 RRT 演算法

快速隨機搜索樹演算法(Rapidly-exploring Random Tree)簡稱 RRT 演算法，是一種隨機生成的資料結構，適合解決高維空間和復雜的路徑規劃問題。想法是以隨機產生點的方式通過一步步向目標點搜索前進，能有效避障，避免路徑陷入局部極小值，收斂速度快，RRT 示意圖如圖 2.4，RRT 的擴展如圖 2.5 所示。

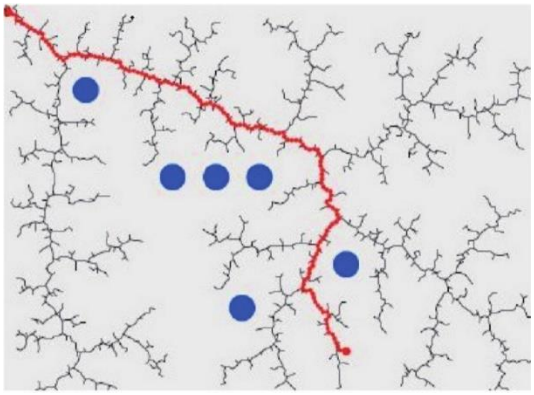


圖 2.4 RRT 示意圖

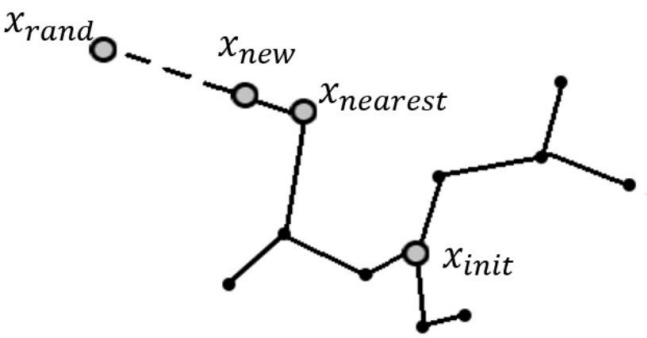


圖 2.5 RRT 擴展結構

RRT 的步驟大概如下：

1. 定義初始位置 x_{init}、終點、採樣點數、點與點間距離 d
2. 空間中分佈樣本點 x_{rand}
3. 在已知樹的點集合中找出最近的節點 $x_{nearest}$
4. 確定 $x_{nearest}$ 之後向 x_{rand} 的直線方向延伸一段距離，產生 x_{new}
5. 判斷 $x_{nearest}$ 與 x_{new} 之間，如果有障礙物就放棄生長，回步驟 2 尋找新的 x_{rand}
6. 將 x_{new} 加入樹中，若還沒到終點就回到步驟 2

2.2.2 RRT*演算法

RRT 採用隨機採樣的方式做路徑規劃，運算速度較快，但最後獲得的往往是可行路徑而非最佳路徑，因此需要優化。RRT*演算法是一個改良的版本，RRT*是在 RRT 的基礎上增加了重寫和隨機重連兩步。主要區別是增加了對新節點 x_{new} 的重新計算過程，目的是把 x_{new} 加入到樹後，為它重新尋找一個新的父節點，使它代價(cost)降低。

RRT*相對 RRT 最大的改進就是會有重寫這個過程，如此一來可以確保代價降低，以漸進式的方式尋找最佳路徑。RRT*缺點也很明顯，因為引入了重寫，表示計算量大使生成路徑效率低，甚至在複雜的環境或狹窄的縫隙不好找到路徑。本專題會採用 RRT*去做運算，因為障礙物地圖相比較單純，用 RRT 找到的路徑往往比較具有優勢，所以採用 RRT*做路徑規劃。

2.3 Pure Pursuit 演算法

前一小節提到，RRT*演算法能做好路徑規劃的工作，此時需要驅動機器人移動在規劃好的路徑上。純追蹤演算法(Pure Pursuit)是一種路徑追蹤的辦法，Pure Pursuit 是依照機器人當前的姿態與速度去計算出下一次運動的角速度與線速度，使得機器人能夠按照規畫好的理想路徑上行走，如圖 2.6。

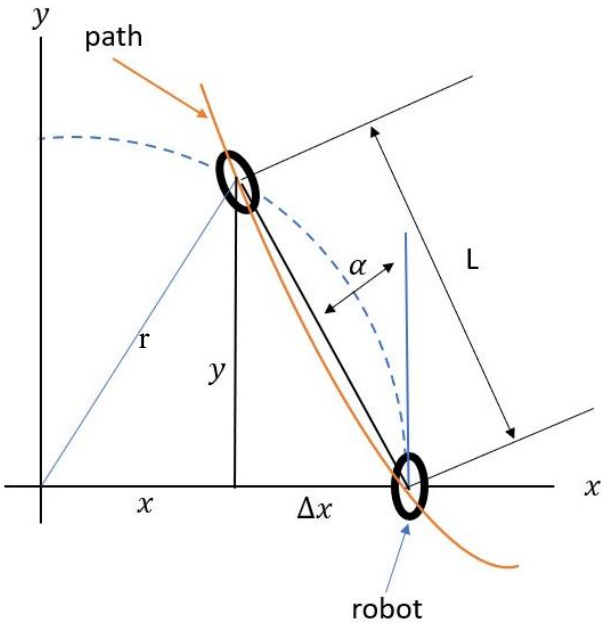


圖 2.6 Pure Pursuit motion

純追蹤演算法的概念是要計算目前位置移動到下一個位置的圓弧曲率 λ ，圓弧的半徑 r 為圓弧曲率 λ 的倒數。定義此運動模型的 Δx 是起始點與目標點在 X 軸上的位移，L 是起始點與目標點距離。由圖 2.6 上可以得到式(2.11)、(2.12)、(2.13)。

$$\Delta x^2 + y^2 = L^2 \qquad (2.11)$$

$$x + \Delta x = r \qquad (2.12)$$

$$x^2 + y^2 = r^2 \qquad (2.13)$$

由式(2.11)、(2.12)、(2.13)可以化簡得到 λ ，如式(2.14)。

$$\lambda = \frac{2\Delta x}{L^2} \qquad (2.14)$$

圓弧曲率決定了機器人該以多大的轉彎半徑行走，曲率值會隨著機器人接近或遠離路徑而變化。另外純追蹤演算法還有一個重要的參數是移動距離 L，定義 α 為起始點與目標點的轉向角，根據正弦定理可得式(2.15)且整理得式(2.16)。

$$\frac{L}{\sin 2\alpha} = \frac{r}{\sin(\frac{\pi}{2} - \alpha)} \qquad (2.15)$$

$$L = 2r\sin\alpha \qquad (2.16)$$

在純追蹤演算法裡面，如果調整前視距離(Lookahead Distance)會使移動距離 L 不同，移動距離是線速度與角速度的組合，如果前視距離調整的較小，機器人將會更快的去反應出路徑的變化並加以控制調整，如圖 2.7，機器人因為前視距離變小的關係需要更反覆的變動線速度與角速度，產生多組組合，這樣會造成機器人容易追著路徑走導致有震盪擺動的情況。為了改善情形，可以選擇加大前視距離，如圖 2.8，如此一來路徑就會更平滑，但曲率變大也容易影響準確度，偏離理想路徑。總結來說，純追蹤演算法若想得到好的效果，就要調整好的前視距離。

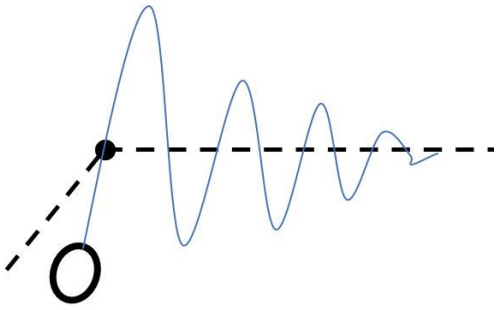


圖 2.7 前視距離小

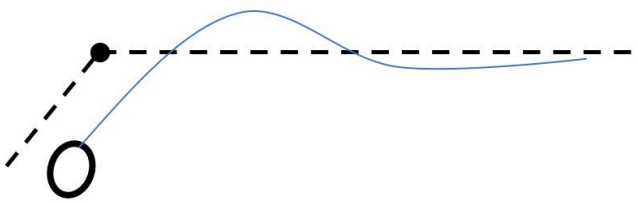


圖 2.8 前視距離大

第三章 基於 CBF-CLF 控制器之移動機器人自主導航方法

上章提出使用 RRT*作為全域路徑規劃的一種方法。本章節將提供基於 CBF-CLF (Control Barrier Function - Control Lyapunov Function)方法結合二次規劃(Quadratic Programming)設計出的控制器(CBF-CLF-QP)，運用 CBF-CLF-QP 在本專題上作為另一種局域路徑規劃的方法。

3.1 Control Lyapunov Function

3.1.1 Lyapunov 穩定性

CLF 是一種在非線性系統控制中常使用的方法，首先，CLF 是根據 Lyapunov 穩定性分析且延伸出來的，可以利用此特性設計控制器確保非線性系統的穩定性與收斂性，此小節要討論 Lyapunov 穩定性的性質。以下定義一個非線性系統如式(3.1)

$$\dot{x} = f(x) \tag{3.1}$$

其中 $x \in \mathbb{R}^n$ 為系統狀態，f 是局部利普希茨連續。再來考慮如圖 3.1 與圖 3.2。

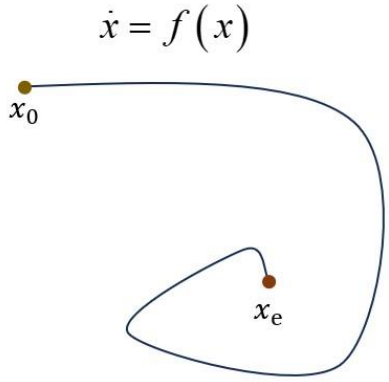


圖 3.1 系統狀態軌跡

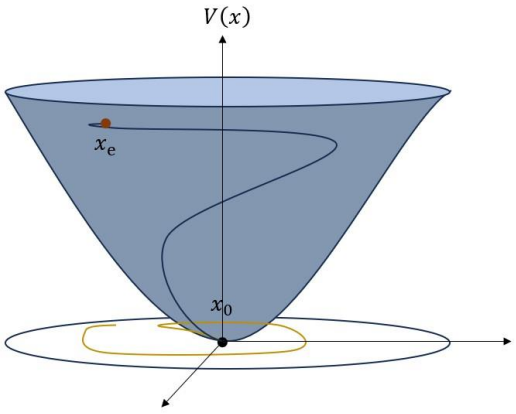


圖 3.2 平滑有界函數$V(x)$

假設起始狀態 x_0 與平衡點 x_e，圖 3.2 的深藍色線為狀態 x_0 到 x_e，然而黃線是將系統狀態投影到平面上。

有了起始狀態 x_0 與平衡狀態 x_e 後，定義一個有界的平滑函數 $V(x):\mathbb{R}^n \to \mathbb{R}$，使得以下式(3.2)、式(3.3)條件成立：

$$V(x_e)=0, V(x)>0, x \neq x_e \tag{3.2}$$

$$\dot{V}(x)=\frac{\partial V}{\partial x} f(x)<0, x \neq x_e \tag{3.3}$$

3.1.2 非線性仿射系統

在 3.1.1 小節提出一個非線性系統式(3.1)，此系統沒有輸入(input)，因此在這個小節引入一個非線性仿射系統(Control Affine System)，如式(3.4)。

$$\dot{x}=f(x)+g(x)u \tag{3.4}$$

其中 $x \in \mathbb{R}^n$ 為系統狀態，$u \in U \subset \mathbb{R}^m$ 為系統輸入，$f(x) \in \mathbb{R}^n$ 和 $g(x) \in \mathbb{R}^{n \times m}$，$f$ 和 g 皆是局部利普希茨連續。U 可以表達為一個凸多胞形如式(3.5)，其中 $A_0 \in \mathbb{R}^{p \times m}$ 且 $b_0 \in \mathbb{R}^p$。

$$U=\left\{u \in \mathbb{R}^m \mid A_0 u \leq b_0\right\} \tag{3.5}$$

3.1.3 CLF 與 ES-CLF

定義 3.1 : CLF (Control Lyapunov Function)

回到圖 3.2，假設一個有界的平滑函數 $V(x):\mathbb{R}^n \to \mathbb{R}$ 它連續可微，考慮一個非線性仿射系統 $\dot{x}=f(x)+g(x)u$，如果有一個常數 $c>0$ 存在且滿足下列三個條件：

1. $\Omega_c=\{x \in \mathbb{R}^n \mid V(x)<c\}$

2. $V(x)>0$ for all $s \in \mathbb{R}^n \setminus \{x_e\}, V(x_e)=0$

3. $\inf_{u \in U} \dot{V}(x,u)<0$ for all $x \in \Omega_c \setminus \{x_e\}$

上面第三點可以看到，$V(x,u)$ 做微分並且希望它的值可以小於 0，而 Ω_c 為吸引域 (Region of Attraction)，代表任意狀態 x 只要在 Ω_c 裡頭，隨著時間會漸進穩定到 x_e。而特性如下:

$$\forall x_0 \in \Omega_c,\ \exists t \in [0,\infty) \quad s.t.\ \ x(0) = x_0 \ \text{且}\ \lim_{t\to\infty} x(t) = x_e$$

滿足上述條件的 $V(x)$ 就是 CLF，為了後面的書寫方便，令 $L_f V(x) = \frac{\partial V}{\partial x} f(x)$ 和 $L_g V(x) = \frac{\partial V}{\partial x} g(x)$，而系統微分的簡化如式(3.6)。

$$\inf_{u\in\mathbb{R}^m} \dot{V}(x,u) = L_f V(x) + L_g V(x)u < 0 \qquad (3.6)$$

定義 3.2 : ES-CLF (Exponentially Stabilizing Control Lyapunov Function)

由定義 3.1，可見 CLF 它擁有漸進穩定性，但是對於系統來說，需要知道到底多快才能收斂平衡狀態，如圖 3.3，這時引入一個參數 $\lambda > 0$，代表衰減速度，假設一個有界的平滑函數 $V(x):\mathbb{R}^n \to \mathbb{R}$ 它連續可微且為正定矩陣，滿足如式(3.7)的話則稱為 ES-CLF，它可以確保系統的指數穩定性。

$$\inf_{u\in U} L_f V(x) + L_g V(x)u + \lambda V(x) \le 0,\ \text{for}\ \exists u \in U, \lambda > 0 \qquad (3.7)$$

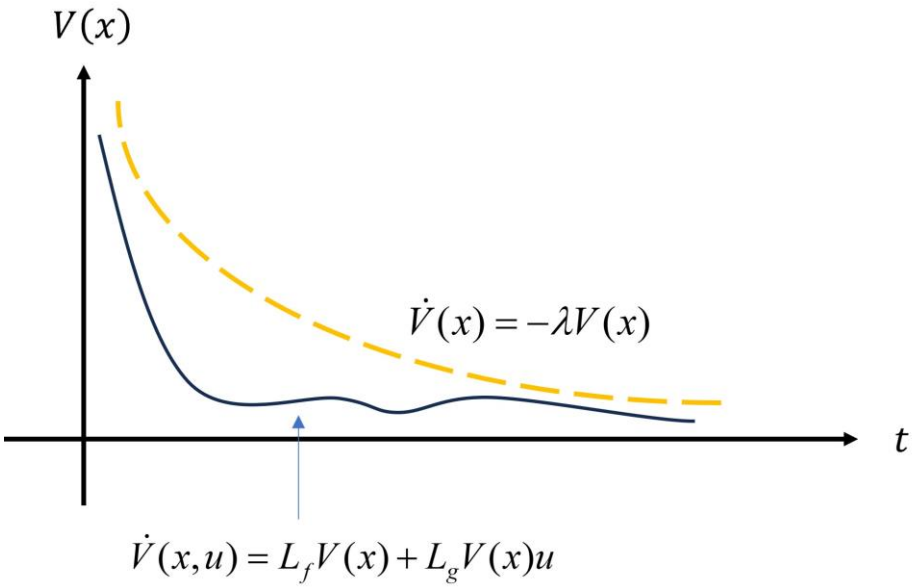


圖 3.3 ES-CLF 收斂性

3.2 Control Barrier Function

3.2.1 Nagumo Invariance Principle

CBF 是基於 Nagumo 不變性定理的集合不變性而延伸出的安全避障函數，而 Nagumo 不變性定理與 Lyapunov 穩定性非常類似。

首先考慮一個非線性系統 $\dot{x} = f(x)$ 如圖(3.4)，接著考慮集合 $\mathcal{C}$ 如圖 3.5。

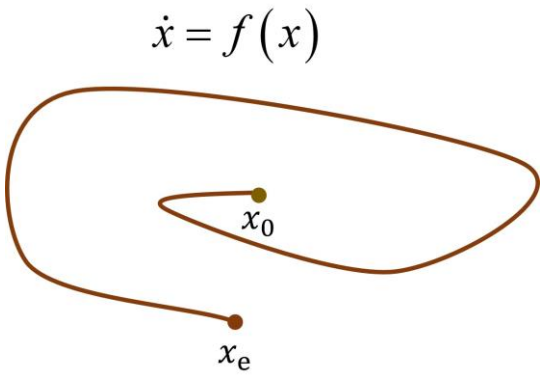


圖 3.4 系統狀態軌跡

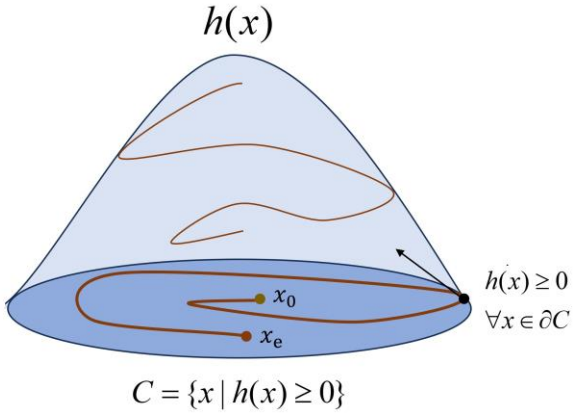


圖 3.5 平滑有界函數 $h(x)$

其中 $x \in \mathbb{R}^n$ 為系統狀態，f 是局部利普希茨連續的函數，$h: \mathbb{R}^n \to \mathbb{R}$ 是一個連續可微的函數。假設閉集合 $\mathcal{C}$ 的定義滿足以下式(3.8)、式(3.9)、式(3.10)：

$$\mathcal{C} = \{x \in \mathbb{R}^n \mid h(x) \geq 0\} \tag{3.8}$$

$$\partial\mathcal{C} = \{x \in \mathbb{R}^n \mid h(x) = 0\} \tag{3.9}$$

$$Int(\mathcal{C}) = \{x \in \mathbb{R}^n \mid h(x) > 0\} \tag{3.10}$$

$\partial\mathcal{C}$ 為邊界，$Int(\mathcal{C})$ 為內部空間，假設狀態 x 到了邊界上，我們希望狀態能夠回到閉集合 $\mathcal{C}$ 而不是跑出集合，因此，要考慮並討論邊界的問題。若對於 $h(x)$ 微分可以得式(3.11)，$\dot{x}$ 為運動方向，$\frac{\partial h(x)}{\partial x}$ 為法向量方向。

$$\dot{h(x)} = \frac{\partial h(x)}{\partial x}\dot{x} > 0 \qquad (3.11)$$

$$\frac{\partial h(x)}{\partial x} > 0 \qquad (3.12)$$

由式(3.12)可以得知，法向量方向與$h(x)$有關，當$h(x) > 0$，法向量會指向集合$\mathcal{C}$內部，反之則指向集合$\mathcal{C}$外部。式(3.11)就是法向量與運動方向$\dot{x}$的內積，如圖 3.6，從中可以發現法向量與運動方向必定夾銳角，確保當狀態x到邊界上時，運動方向會落入集合$\mathcal{C}$，因此若滿足式(3.11)與式(3.12)兩個條件，邊界上的問題就能解決。

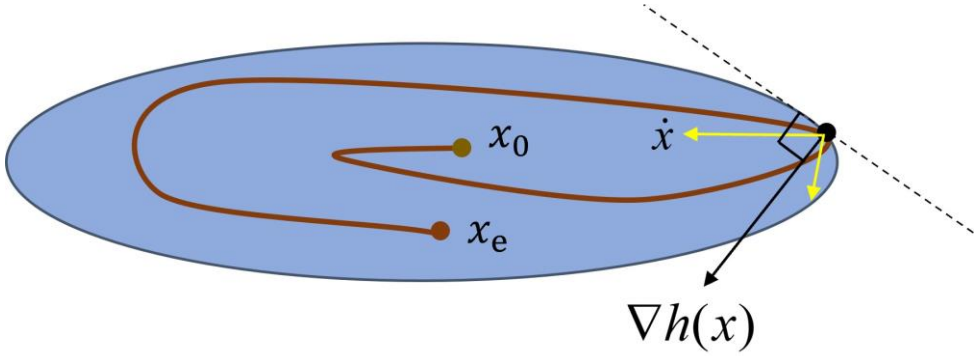


圖 3.6 邊界問題

結論則是 Nagumo 不變性定理希望閉集合有$\mathcal{C}$有集合不變性，在閉集合$\mathcal{C}$裡頭並且遇到邊界會反彈，利用這個想法結合控制的概念延伸出 CBF。

3.2.2 RCBF 與 ZCBF

CBF 可以分為 RCBF 與 ZCBF，RCBF 的其中一種形式是假設對數函數$B(x)$作為障礙函數(Barrier Function)如式(3.13)，而 ZCBF 是直接使用$h(x)$。相同於 3.1 節的 CLF，在 CBF 也要使用一個非線性仿射系統如式(3.4)。

$$B(x) = -\log\left(\frac{h(x)}{1+h(x)}\right) \qquad (3.13)$$

定義 3.3 : RCBF (Reciprocal Control Barrier Functions)

由式(3.8)、式(3.9)、式(3.10)和式(3.13)可以滿足式(3.14)

$$\inf_{x\in Int(C)} B(x)\geq 0,\ \lim_{x\to\partial C} B(x)=\infty \tag{3.14}$$

一般傳統的限制條件而言如式(3.15)，但這條式子可以給予一個放鬆係數η，η為一個大於 0 的值，使得式(3.15)可以放鬆限制條件，如式(3.16)。當遠離邊界$\partial\mathcal{C}$時，允許$\dot{B(x)}$成長，反之靠近邊界$\partial\mathcal{C}$時，$\dot{B(x)}$會趨近於 0。

$$\dot{B(x)}=-\frac{\dot{h(x)}}{h(x)+h(x)^2}\leq 0 \tag{3.15}$$

$$\dot{B(x)}\leq\frac{\eta}{B(x)} \tag{3.16}$$

有了非線性系統式(3.4)和集合$\mathcal{C}$，一個連續可微函數$B:Int(\mathcal{C})\to\mathbb{R}$，若存在一個$\kappa_\infty$類函數$\alpha$，那麼上述這些條件可以定義出 RCBF 且存在限制條件式(3.17)。

$$\inf_{u\in U}\left[L_fB(x)+L_gB(x)u-\alpha\left(B(x)\right)\right]\leq 0,\ \text{for } \exists u\in U \tag{3.17}$$

接著，令函數$\alpha\left(B(x)\right)=\frac{\gamma}{B(x)}$，$\gamma$為一個大於 0 的常數，定義如式(3.18)。

$$\inf_{u\in U}\left[L_fB(x)+L_gB(x)u-\frac{\gamma}{B(x)}\right]\leq 0,\ \text{for } \exists u\in U,\gamma>0 \tag{3.18}$$

定義 3.4 : ZCBF (Zeroing Control Barrier Functions)

首先，ZCBF 的障礙函數設置為$B(x)=h(x)$，$h(x):\mathbb{R}^n\to\mathbb{R}$，$B$為連續可微函數並且滿足式(3.19)、式(3.20)。同定義 3.3 的 RCBF，γ加入後得到限制條件式(3.21)。

$$\mathcal{C}=\{x\,|\,B(x)\geq 0\} \tag{3.19}$$

$$\nabla B(x)\neq 0,\ \text{for } x\in\partial\mathcal{C} \tag{3.20}$$

$$\sup_{u\in U}\left[L_fB(x)+L_gB(x)u+\gamma B(x)\right]\geq 0,\ \text{for } \exists u\in U,\gamma>0 \tag{3.21}$$

兩種方法中，因為 RCBF 由式(3.14)得知當靠近邊界時會趨近無窮大，使得運算上非常的不方便，所以本專題選擇用 ZCBF 如圖 3.7 來進行模擬與實作。

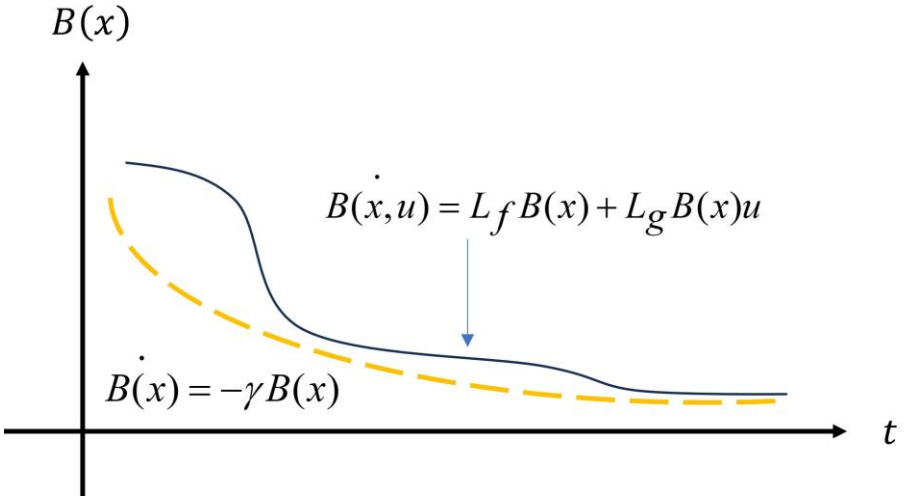


圖 3.7 ZCBF 示意圖

3.3 CBF-CLF-QP

CLF 保證了控制器的穩定性使移動機器人能到達終點，而 CBF 保障了安全問題，也就是避障。此節需將 CLF 與 CBF 結合，運用二次規劃(QP)的方式去設計控制器。由定義 3.2 可以得知，使用 ES-CLF 可以保障系統的指數穩定性，3.2.2 小節有提到使用 ZCBF 的原因，但 QP 求解時，滿足 ES-CLF 和 ZCBF 的限制條件可能會沒有交集，從而導致找不到可行解，而這時引入解決辦法，就是在 ES-CLF 的限制條件加入放鬆變量 δ 再做 QP 如式(3.22) ，QP 的限制條件如式(3.23)。

$$u(x) = \arg\min \left(u - u_{ref}\right)^T H \left(u - u_{ref}\right) + p\delta^2 \qquad (3.22)$$

$$\text{subject to :} \quad L_f V(x) + L_g V(x)u + \lambda V(x) \le \delta$$

$$L_f B(x) + L_g B(x)u + \gamma B(x) \ge 0 \qquad (3.23)$$

$$\text{input :} \quad u \in U$$

其中，H 是一個正定矩陣，u_{ref} 是期望輸入，常數 $\lambda,\gamma>0$，懲罰係數 $p>0$，δ 是放鬆變量。然而允許 $\delta>0$ 而放寬 ES-CLF 的條件，使 QP 的可行解域更大，這個觀念在本次專題的實作中非常重要，系統的穩定性固然重要，但我們更要求安全性的問題，也就是 CBF 的限制條件必須要優先考慮，適時的放鬆系統的穩定性在實作上效果會好上許多。常數 λ,γ 對於系統與 QP 求解的可行性也在這裡能夠得出結論，如表 2。

表 2 λ,γ 對系統與 QP 的關係

變化	λ	γ
增加	系統收斂速度增快	QP 求解可行性提高
減少	QP 求解可行性提高	系統安全性提高

第四章 模擬與實驗

本章節首先將介紹本專題的實驗環境，利用 LiDAR 將收回的環境資料在 Matlab 端構建一組 Occupancy grid maps，得到地圖後分別執行 RRT*搭配純追蹤演算法，以及利用 CBF-CLF-QP 控制器，這兩種方法的路徑規劃模擬，最後搭配上機實驗與這兩者的結果討論。

4.1 實驗場域與地圖建構

4.1.1 實驗場域

本專題實驗的場景是電機系 92883 實驗室(8F)裡的方形場域，如圖 4.1，場域裡的障礙物是由海綿所組成，障礙物的搭建必需要高於移動機器人的 LiDAR 高度，否則會偵測不到障礙物的資料。

圖 4.1 方形障礙場域 (105cm*148cm)

4.1.2 地圖建構

圖 4.2 為地圖建構的流程圖，首先連接 Matlab 與移動機器人，尤於本專題實作用的 Turtlebot3 有時初始位置在 x, y 其中一個方向的值有很大的偏差，解決辦法是透過關機重新連線來修正誤差值。

表 3 為本專題利用 lidarSLAM 函式感測周遭環境資訊，透過 scansAndPoses 函式與 buildMap 函式轉換成 Occupancy grid map，成為我們在實作 RRT*與 CBF-CLF-QP 的地圖。除此之外，在蒐集資訊的過程中，要確保每個點的資訊有按照順序存進來，避免建構地圖時點的順序位置不對而無法進行迭代運算。圖 4.3 為機器人繞行場域的資訊圖。

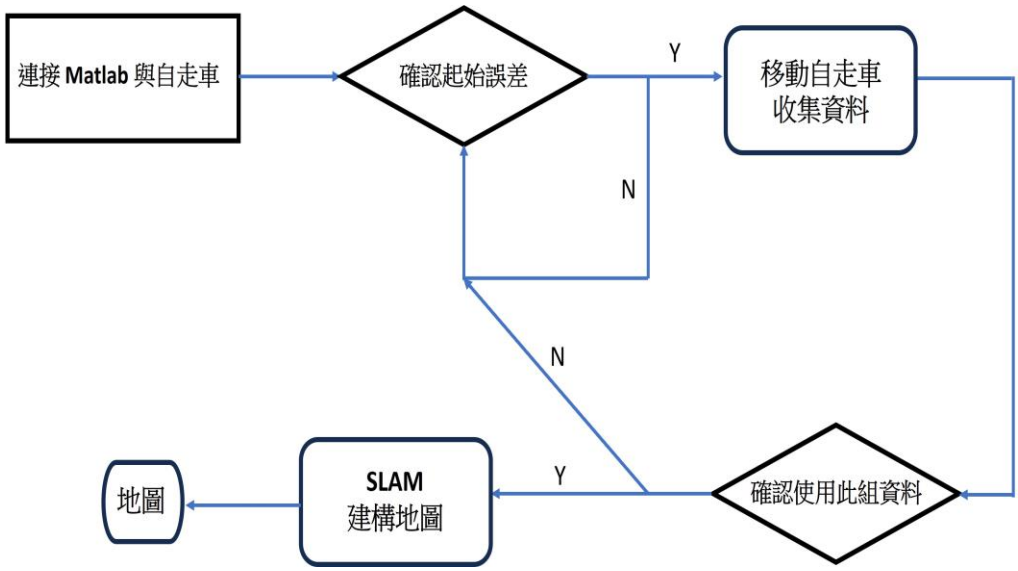


圖 4.2 SLAM 建圖流程

表 3 lidarSLAM 函式

lidarSLAM	simultaneous localization and mapping form lidar scan
addScan	Add scan to lidar SLAM map
buildMap	Build Occupancy Map from Lidar Scans and Poses
scansAndPoses	Extract scans and corresponding poses

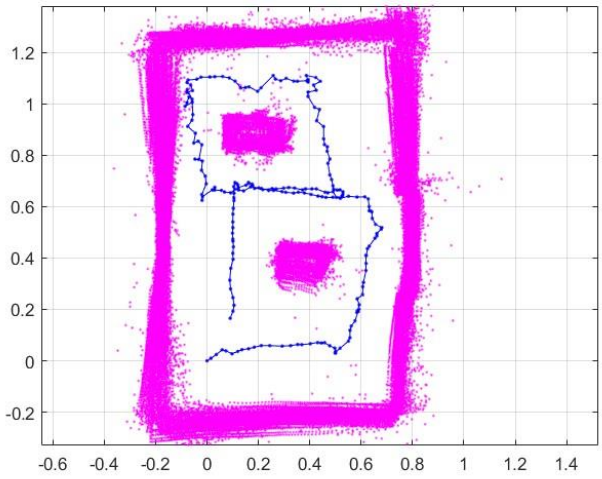


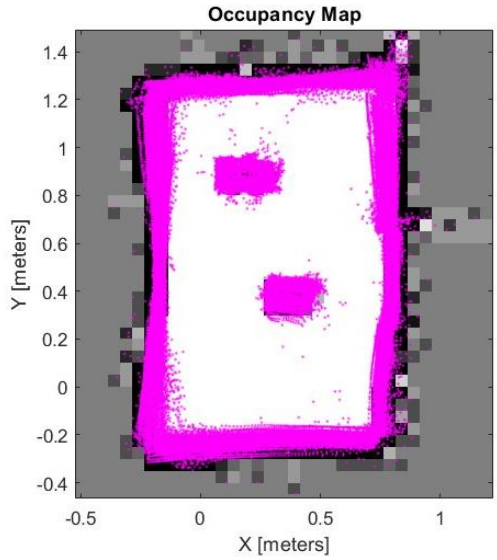
圖 4.3 移動機器人繞行資訊

建構地圖中，地圖的參數也是很重要的一個問題，表 4 是 lidarSLAM 函式裡頭要定義且對地圖影響較大的參數。

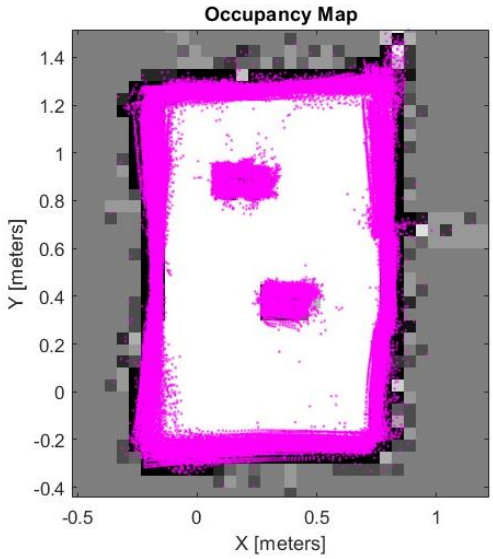
表 4 地圖重要參數

LoopClosureThreshold	Threshold for accepting loop closures
LoopClosureSearchRadius	search radius for loop closure detection
MapResolution	Resolution of occupancy grid map

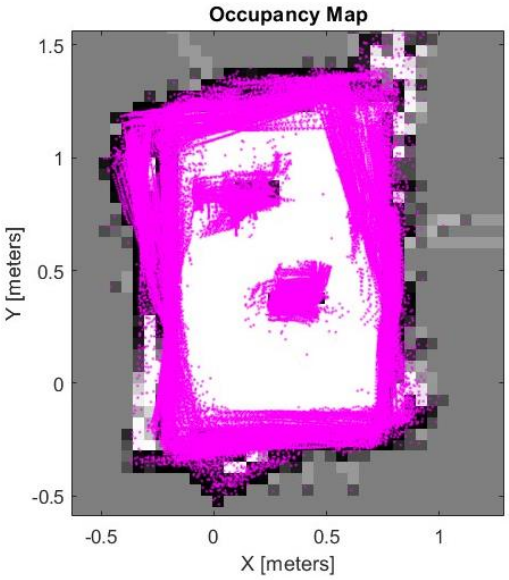
首先，LoopClosureThreshold 與 LoopClosureSearchRadius 的設定會影響到地圖運算的結果與時間。LoopClosureThreshold 為門檻值，若它設定的值越大，有助於排除環線閉合(Loop closure)檢查時錯誤的情況，但過大的門檻值有可能會有反效果，若太多的資料被排除，留下的數據太少就有可能建構不出地圖。LoopClosureSearchRadius 為迴路搜索半徑，本專題固定搜索半徑，一致設定為 3 公尺。參數設定方法為門檻值除以搜索半徑，圖 4.3 參數分別為 200/3、300/3、350/3、500/3，可以看到門檻值設定過高導致地圖無法順利的被建構出來，而運行的時間如表 5。



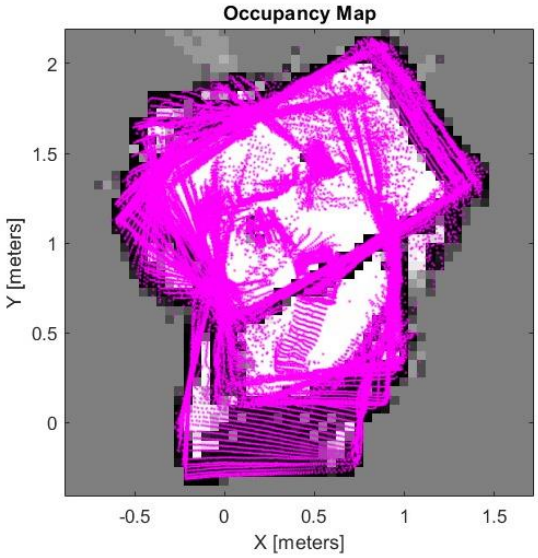
(a) 參數 = 200 / 3



(b) 參數 = 300 / 3



(c) 參數 = 350 / 3



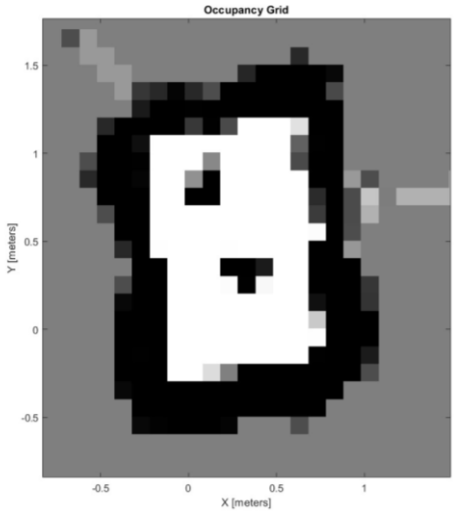
(d) 參數 = 500 / 3

圖 4.4 參數設定(門檻值除以搜索半徑)

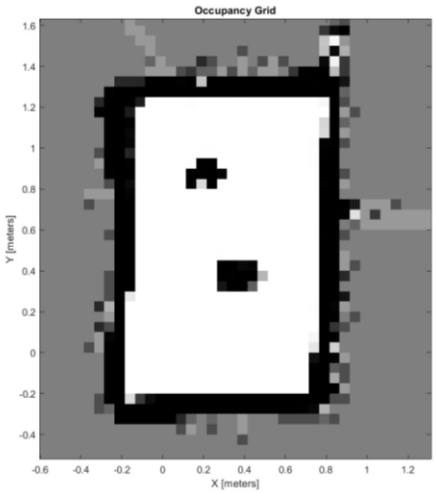
表 5 門檻值與地圖運算時間

參數 = 200 / 3	79.2 (s)	參數 = 300 / 3	91.8 (s)
參數 = 350 / 3	82.5 (s)	參數 = 500 / 3	74.6 (s)

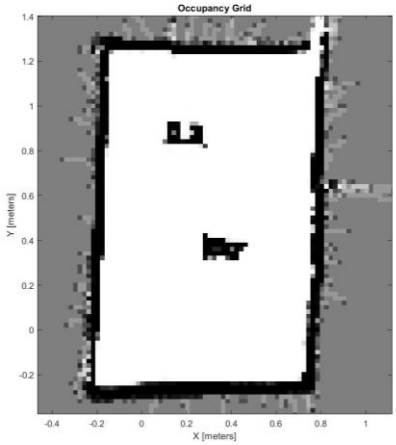
接著 MapResolution 為地圖的解析度，如果值為 20 代表一公尺裡有 20 個等分點，換言之地圖解析度會是 5 公分，調整解析度的參數與建圖比較如圖 4.4。然而解析度值越高，地圖運算的時間就越長，如表 6。



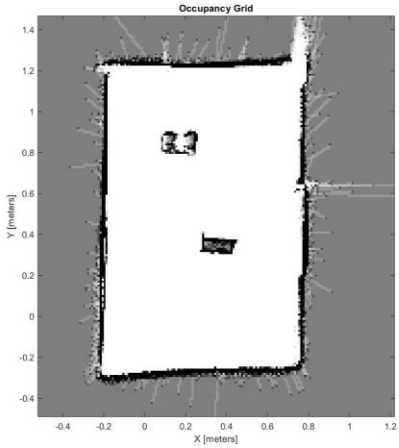
(a) MapResolution = 10



(b) MapResolution = 20



(c) MapResolution = 50



(d) MapResolution = 100

圖 4.5 不同的解析度參數

表 6 解析度與地圖運算時間

MapResolution = 10	79.2 (s)	MapResolution = 20	87.8 (s)
MapResolution = 50	123.4 (s)	MapResolution =100	515.6 (s)

4.2 RRT*全域路徑規劃與實驗

4.2.1 RRT*路徑規劃

圖 4.6 為 RRT*路徑規劃與驅動機器人到終點的流程圖，首先連接 Matlab 與機器人，表 7 為本專題利用 plannerRRTStar 函式建立搜索樹，代入 validatorOccupancyMap 函式當作限制條件後，找出可行路徑。為了避免太靠近障礙物的路徑產生，使用 isPathValid、clearance 函式檢查，直到建立的隨機搜索樹有可行的路徑產生為止。

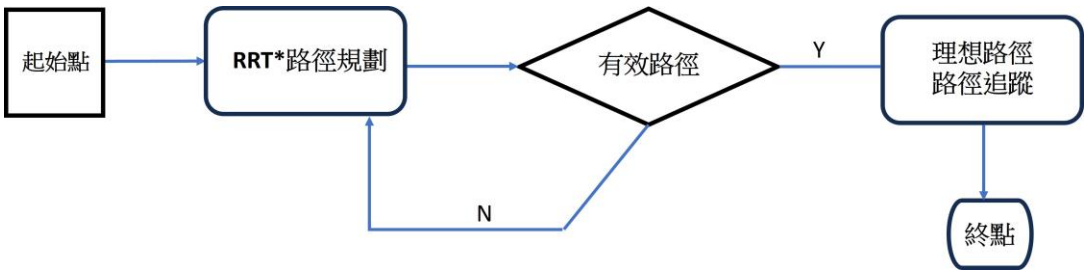


圖 4.6 RRT*流程圖

表 7 plannerRRTStar 函式

plannerRRTStar	RRT* planner
StateSpaceSE2	SE2 state space
validatorOccupancyMap	State validator based on 2D grid map
isPathValid	Determine if planned path is obstacle free
clearance	Minimum clearance of path

考慮移動機器人的長與寬 138 (mm)* 178(mm)，地圖上的邊界與障礙物需要做膨脹處理，而膨脹的大小約為 9 公分，圖 4.7 為膨脹後的地圖。在 RRT*路徑規劃中，參數 MaxIterations 與 MaxConnectionDistance 會影響到路徑長度、運算時間與是否為平滑路徑。首先，MaxIterations 值越大代表分支越多，路徑會比較平滑，但所需要的時間也會變多；反之，路徑會曲折而運算時間縮短。MaxConnectionDistance 為每個航點之間的距離，為了使純追蹤演算法能更好的追上路徑，路徑要盡量平滑，在本專題皆設定為 0.5 公尺，而模擬過程中幾乎只會調整 MaxIterations 的值，如圖 4.8。

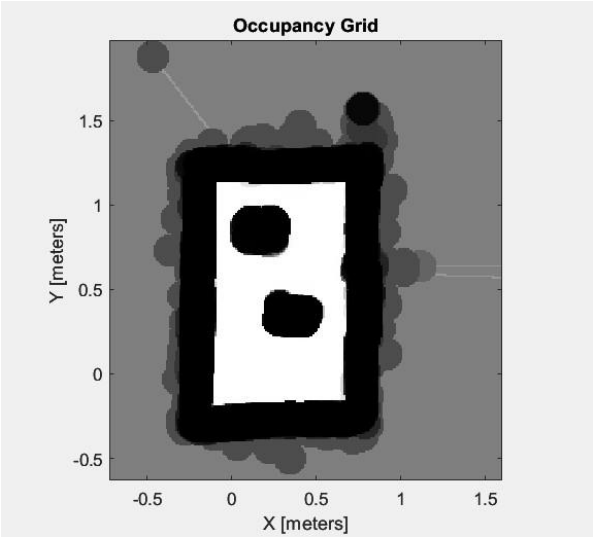
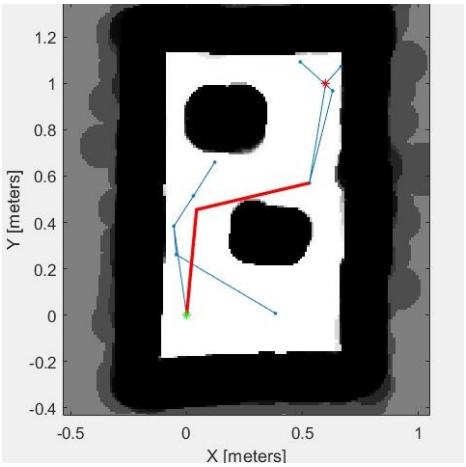
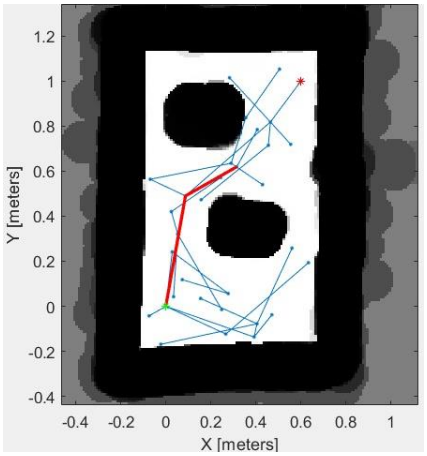


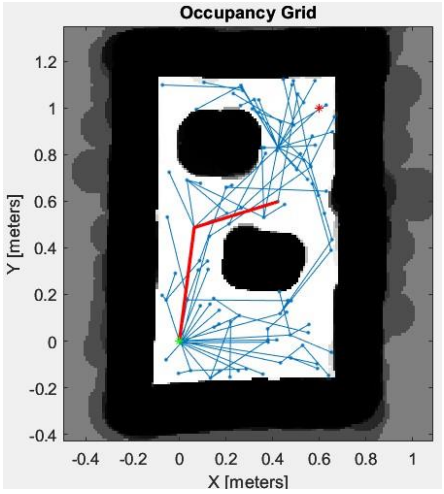
圖 4.7 膨脹環境地圖



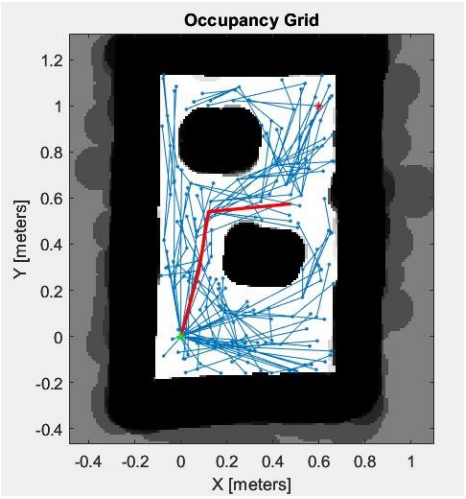
(a) MaxIterations = 200



(b) MaxIterations = 500



(c) MaxIterations = 1000



(d) MaxIterations = 2000

圖 4.8 不同 MaxIterations

4.2.2 RRT*實驗討論

上機實驗執行 RRT*路徑規劃的軌跡且驅動機器人的方法採用 2.3 節的純追蹤演算法來進行路徑追蹤，經由上一小節調整不同的 MaxIterations，執行的秒數如表 8。在實際應用上，挑選演算法秒數較短且路徑平滑為最好的選擇，綜合表 8 與圖 4.8 的結果，最後選用 MaxIterations = 500 實作本次實驗並且得到最佳結果，如圖 4.9。

圖 4.10 的綠線為機器人的軌跡追蹤，實際移動過程如圖 4.11，從第一列由左向右開始為 RRT*實驗影片的截圖，共 12 張，執行時間 18 秒。其中，第一張到第三張照片是自走車在原地旋轉到下一個航點的方向，剩餘照片則為行駛到終點。

表 8 不同的 MaxIterations，執行時間

MaxIterations = 200	8.72 (s)	MaxIterations = 500	9.28 (s)
MaxIterations = 1000	13.7 (s)	MaxIterations = 2000	62.6 (s)

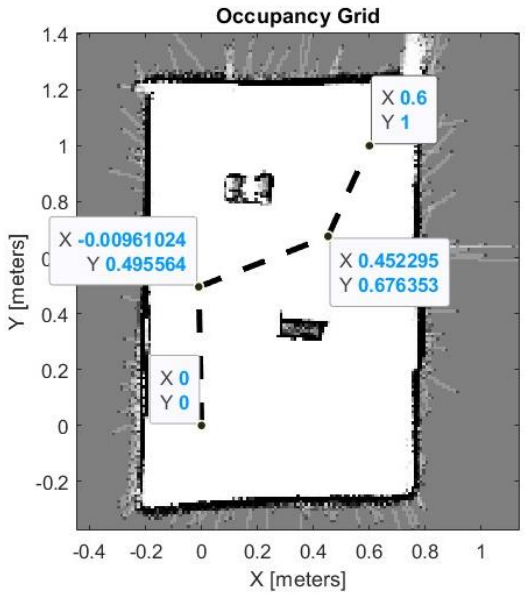


圖 4.9 RRT*路徑規劃航點

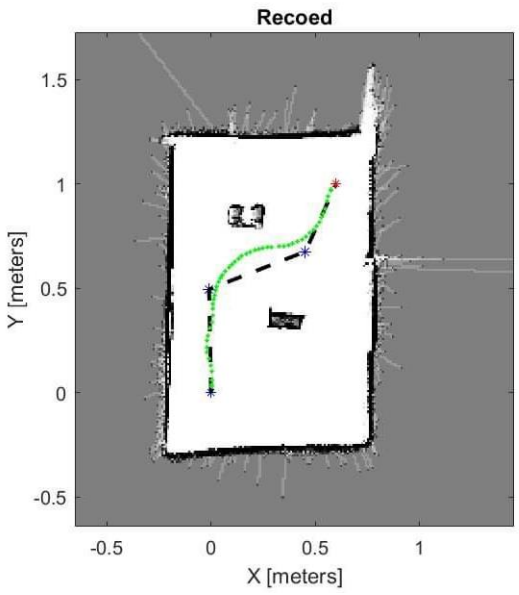


圖 4.10 軌跡追蹤

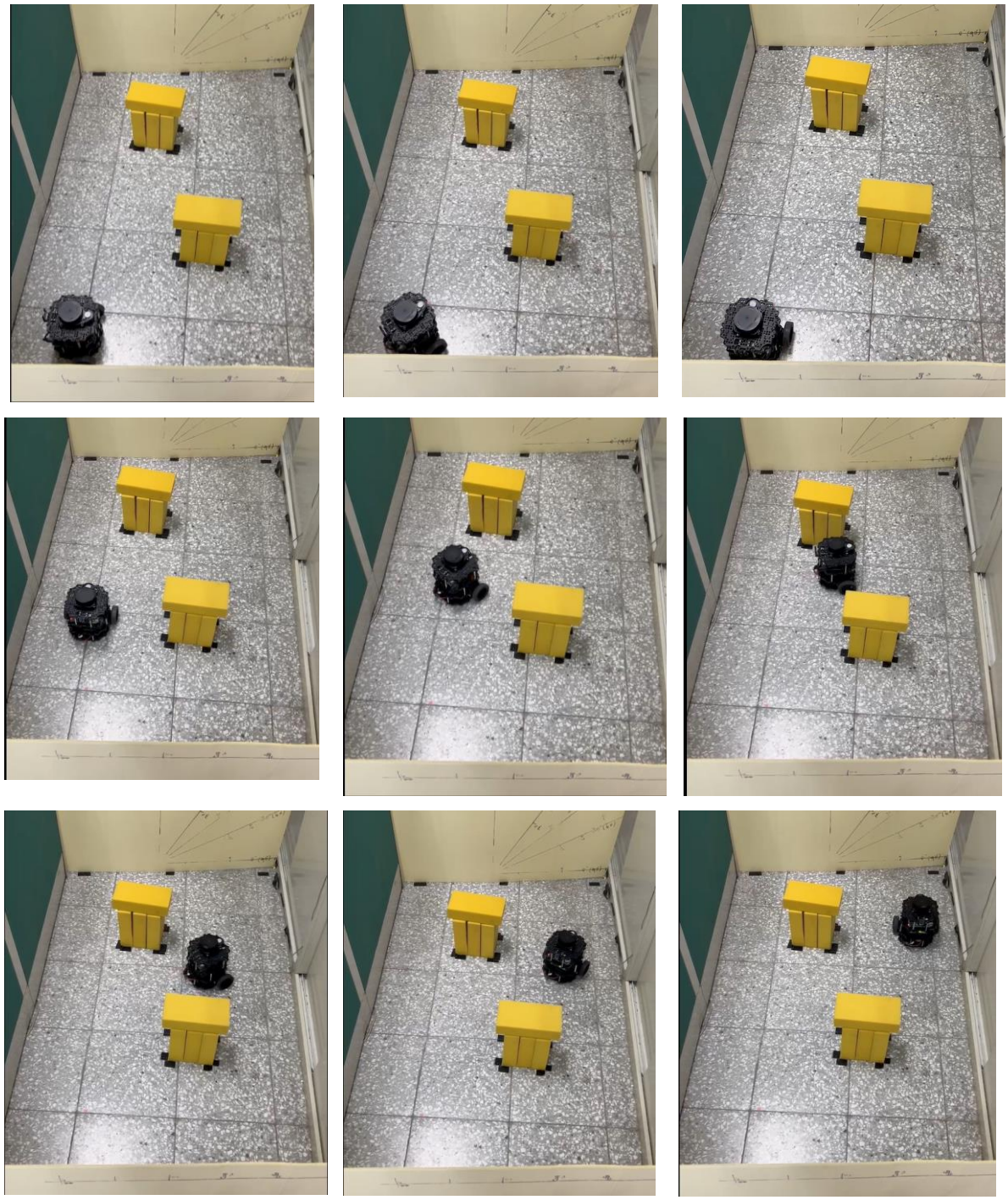


圖 4.11 RRT*上機過程

參考影片：https://youtube.com/shorts/o15MSMuil8s?feature=share

4.3 CBF-CLF-QP 控制器實驗

4.3.1 控制器設計與設立限制條件

圖 4.12 為 CBF-CLF-QP 控制器與驅動機器人到終點的流程圖，首先連接 Matlab 與機器人，表 9 為 CBF-CLF-QP 在模擬的過程中重要的參數。

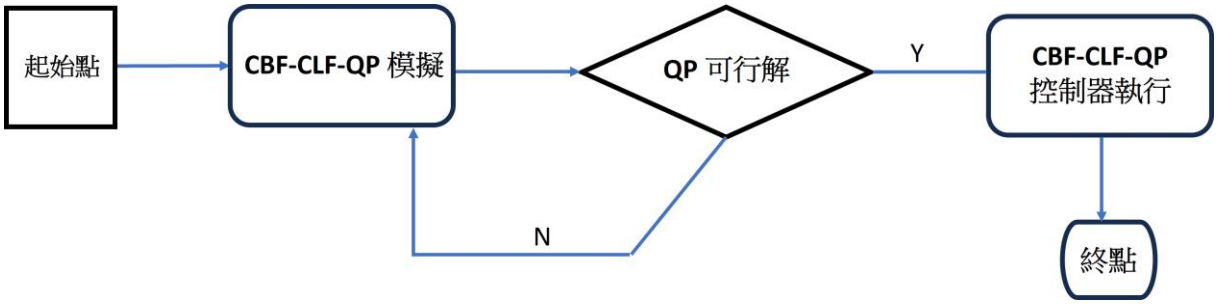


圖 4.12 CBF-CLF-QP 流程圖

表 9 CBF-CLF-QP 重要參數

Param.clf.rate	λ	CLF 衰減速度
Param.clf.rate	γ	CBF 衰減速度
Param.weight.input	H	QP 解的正定矩陣(調整用)
Param.weight.slack	p	QP 解的懲罰因子

設計 CBF-CLF-QP 控制器與模擬的步驟流程如下：

1. 建構一個非線性系統如 3.1.2 小節。

2. 移動機器人的動力學方程式如 1.2 節，放入系統。

3. 定義 CLF，$V(x)=\inf_{u\in U} L_f V(x)+L_g V(x)u+\lambda V(x)\le 0$ 與參數 λ 。

4. 定義 CBF，$B(x)=\sup_{u\in U}\left[L_f B(x)+L_g B(x)u+\gamma B(x)\right]\ge 0$ 與參數 γ 。

5. 將 $V(x)$ 和 $B(x)$ 結合，設定 H 與 p ，使用 QP 求可行解。

6. QP 若找不到解，則重新回到步驟 3、4、5 設定參數。

CLF 的λ可以決定系統的收斂速度，CBF 部分，把距離設置成 CBF，將障礙物當作 CBF 的限制條件，如果地圖中有 n 個障礙物就設立 n 條限制條件。本專題在環境地圖建構中確定了障礙物有兩個，CBF 就設定兩個限制條件。設定的方法為取出兩個障礙物的中心座標位置，分別為$(0.15, 0.85)$、$(0.35, 0.35)$，以半徑為 10 公分進行建模，如圖 4.13。除此之外，也需要對移動機器人的半徑 9 公分納入考慮。

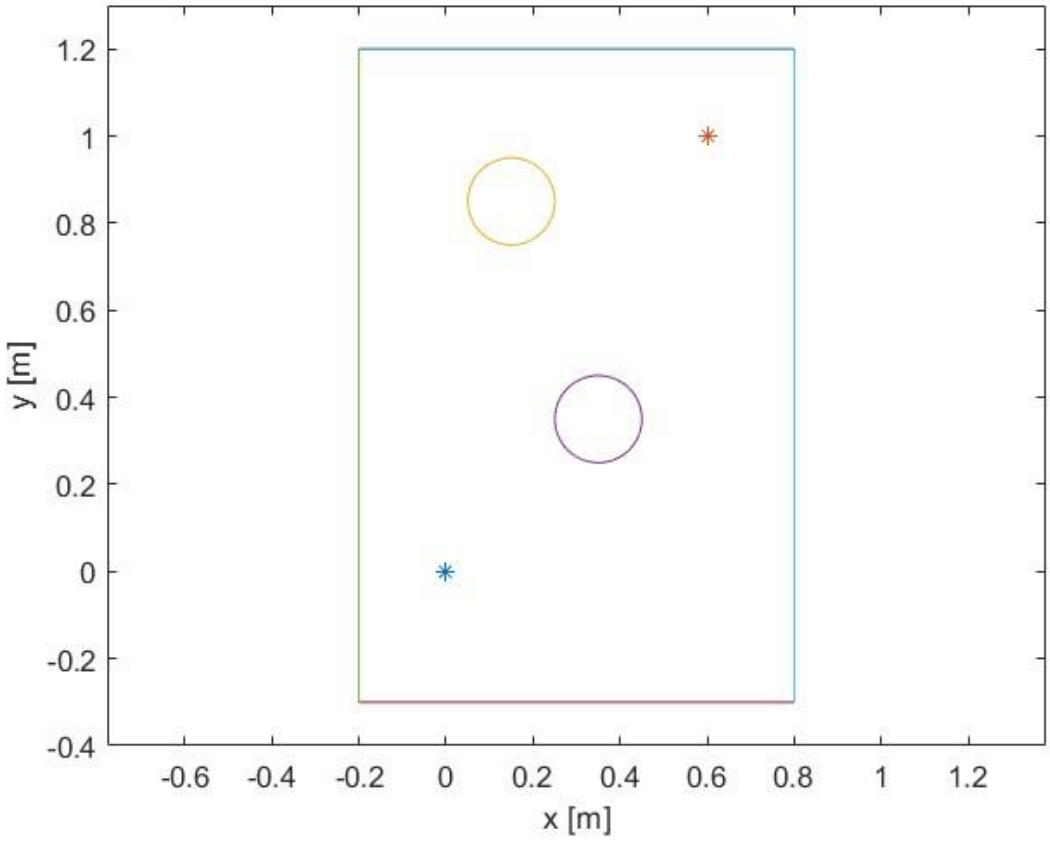


圖 4.13 CBF-CLF-QP 使用地圖

4.3.2 CBF-CLF-QP 模擬

如上小節的表 9 提到的 CBF-CLF-QP 重要參數，這些參數只要稍做更動都有可能使結果產生變化，這小節要來討論關於調整參數會帶來的不同模擬結果。首先，起始點的位置$(0,0)$，目標點的位置$(0.6,1)$，移動機器人的初始姿態$(0,0,0)$，第一種測試為不同速度下，調整不同的參數所帶來的結果；第二種為初始姿態的不同角度下，如初始姿態$(0,0,\pi/2)$，調整不同的參數所帶來的結果。

第一種測試：不同速度下，調整不同的參數

(1) $v=0.1(m/s)$，$x=(0,0,0)$，針對 CLF 的 λ，γ 值固定，如圖 4.14、4.15。

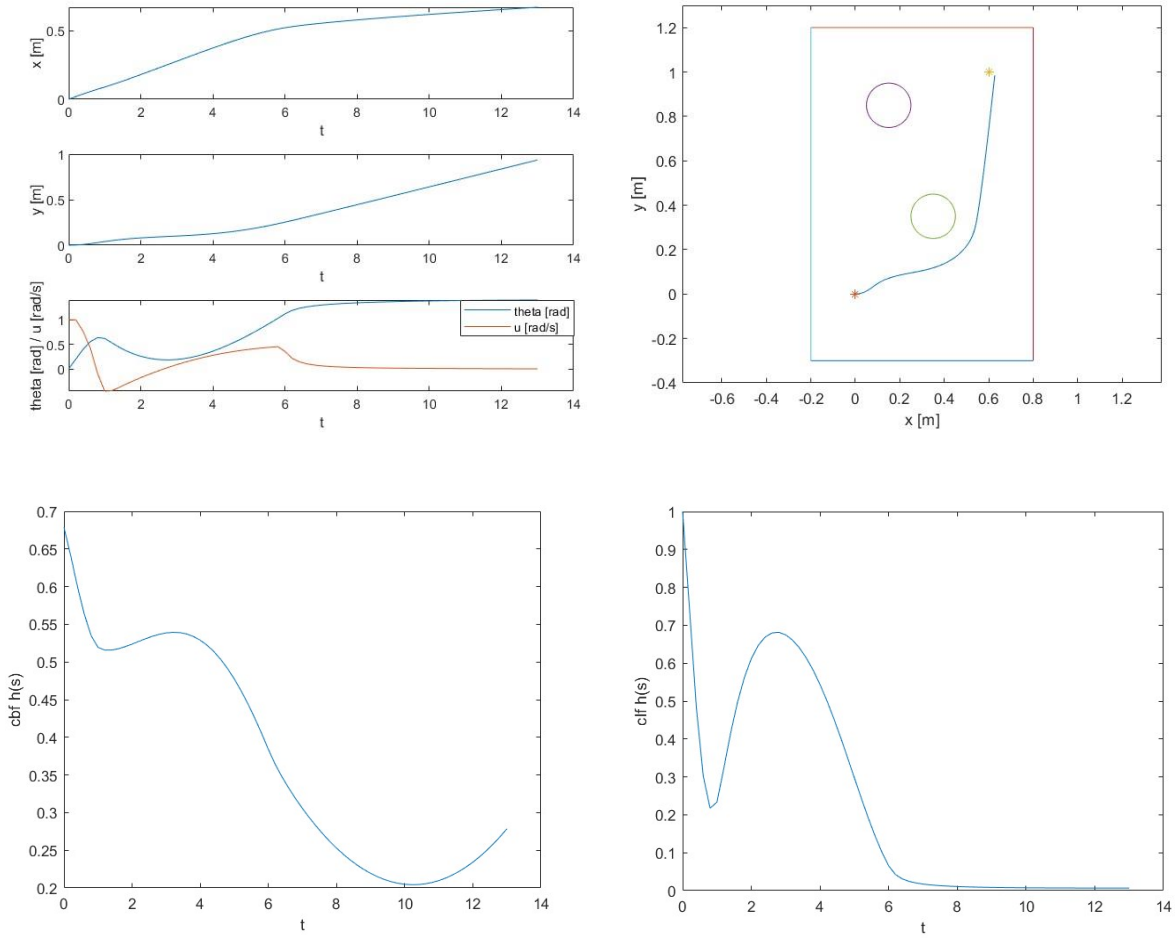
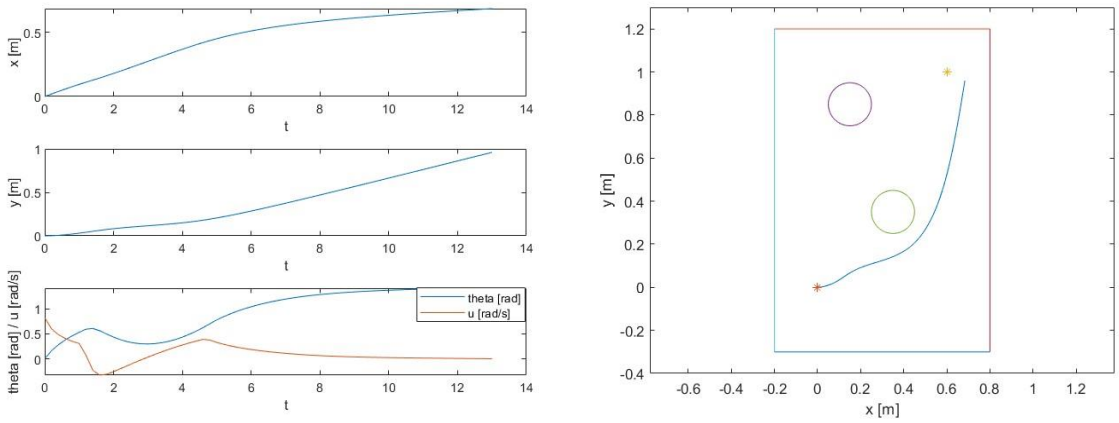


圖 4.14 λ=5, $v=0.1(m/s)$, $\gamma=1$



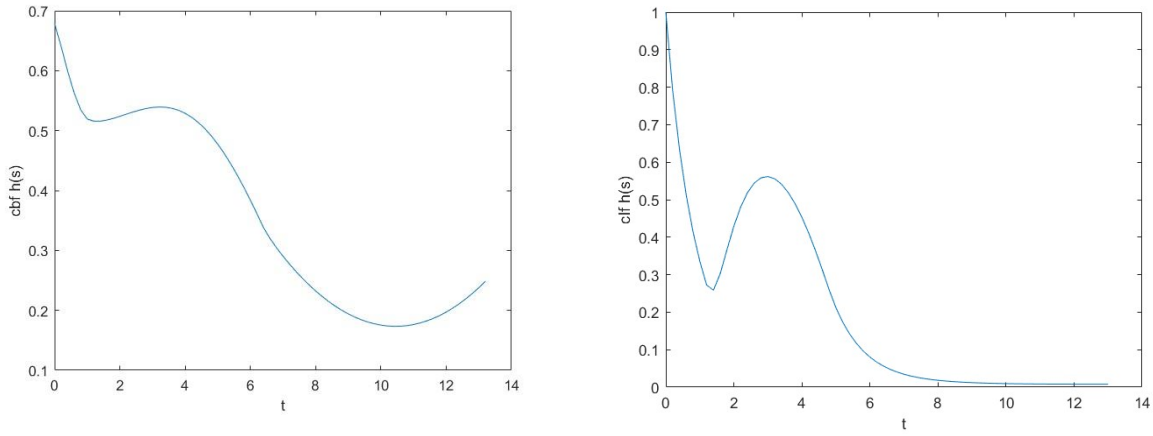


圖 4.15 λ=1, $v=0.1(m/s)$, $\gamma=1$

(2) $v=0.1(m/s)$，$x=(0,0,0)$，針對 CBF 的γ，λ 值固定，如圖 4.16、圖 4.17。

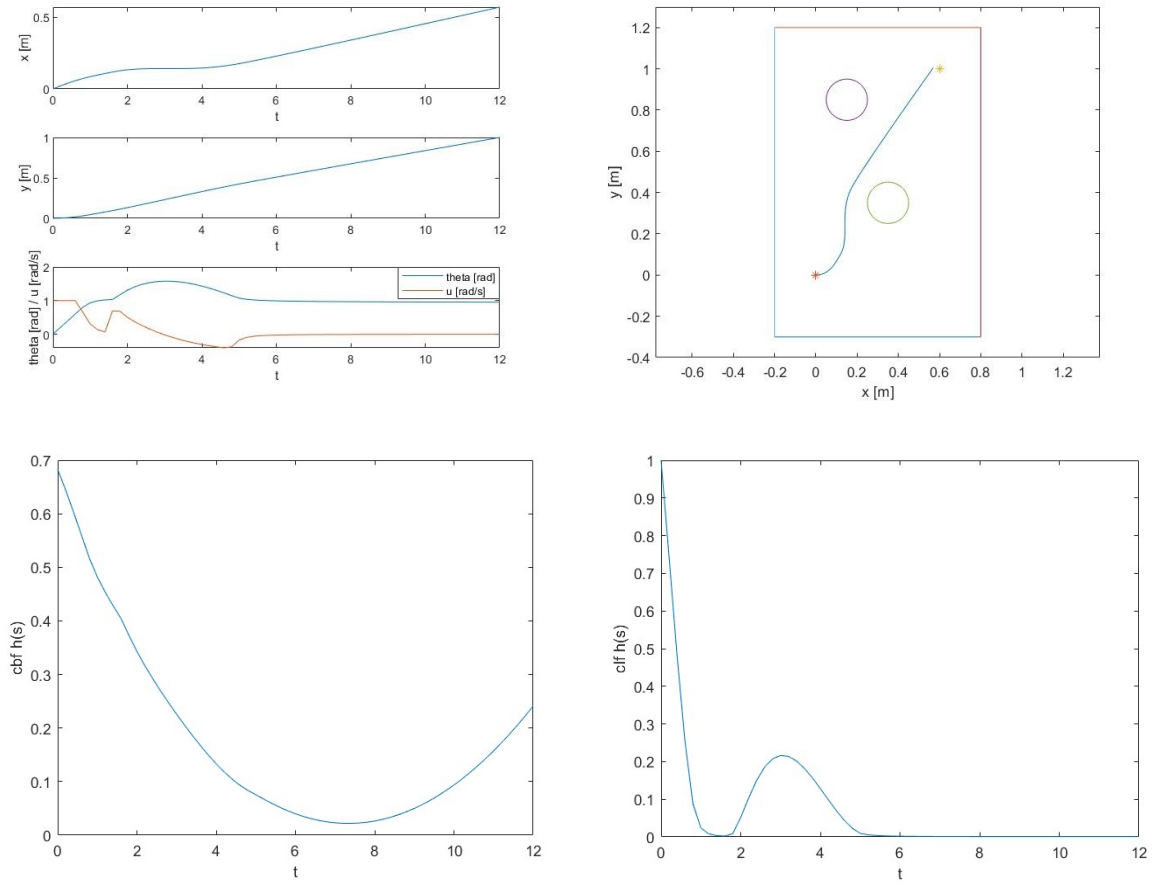


圖 4.16 λ=5, $v=0.1(m/s)$, $\gamma=2$

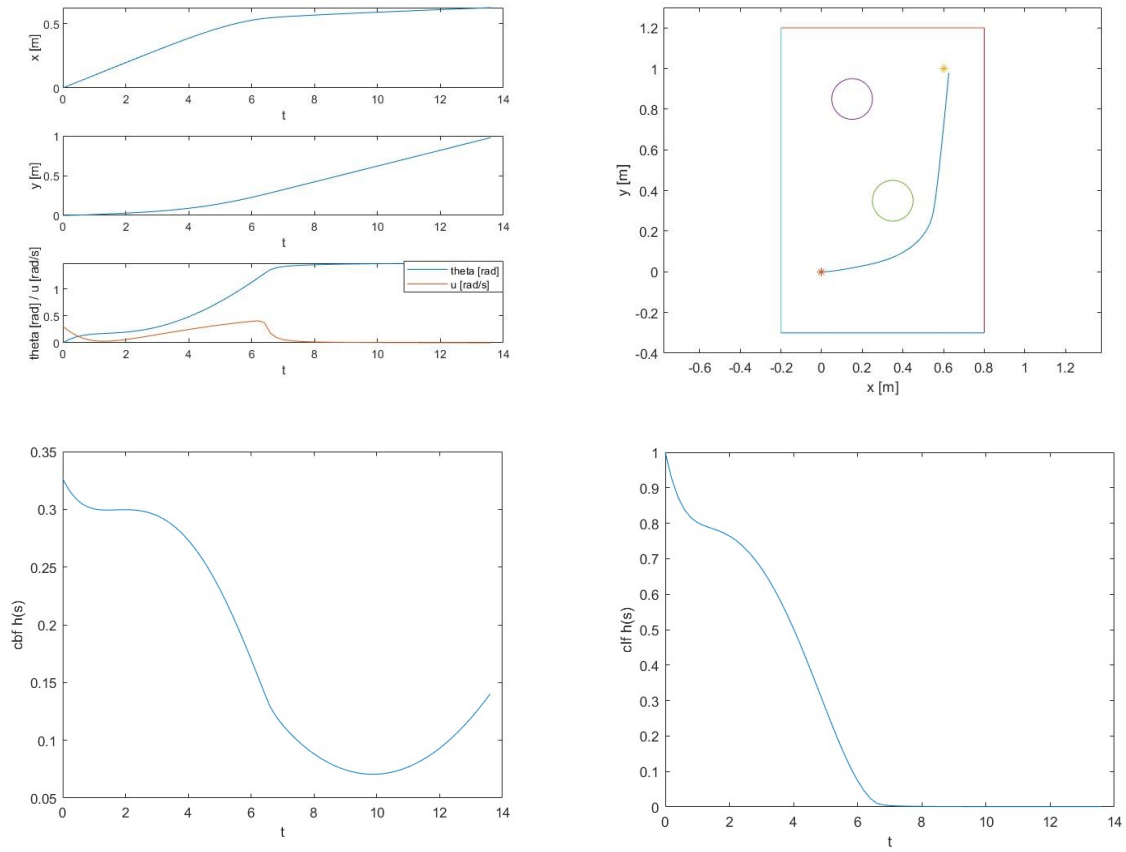


圖 4.17 λ=5, $v = 0.1(m/s)$, $\gamma = 0.5$

圖 4.14、4.15 針對 CLF 的λ，修改不同的λ會使系統的收斂速度與最後終點位置有差別，λ調越大，CLF 收斂的越快，QP 可行解限制提高，距離目標點越接近；λ調越小，CLF 收斂的越慢，QP 可行解限制提低，但距離目標點也容易有更大的誤差。這兩張圖可以得知λ調越大是越好的，距離目標點越近是我們的主要目標。

圖 4.16、4.17 針對 CBF 的γ，如果使用到 CBF 做為避障功能，其值會接近或趨近於零，修改不同的γ會使系統的安全性改變。γ調的越大，CBF 限制條件放鬆，安全性降低；γ調的越小，CBF 限制條件嚴謹，安全性提高。這兩張圖可以得知γ調越小是越好的，安全性問題是路徑規劃裡最重要的一環，能到的了終點但過程中發生碰撞就不是成功的路徑了。

(3) $v=0.15(m/s)$，$x=(0,0,0)$，針對 CLF 的γ，λ值固定，如圖 4.18、4.19。

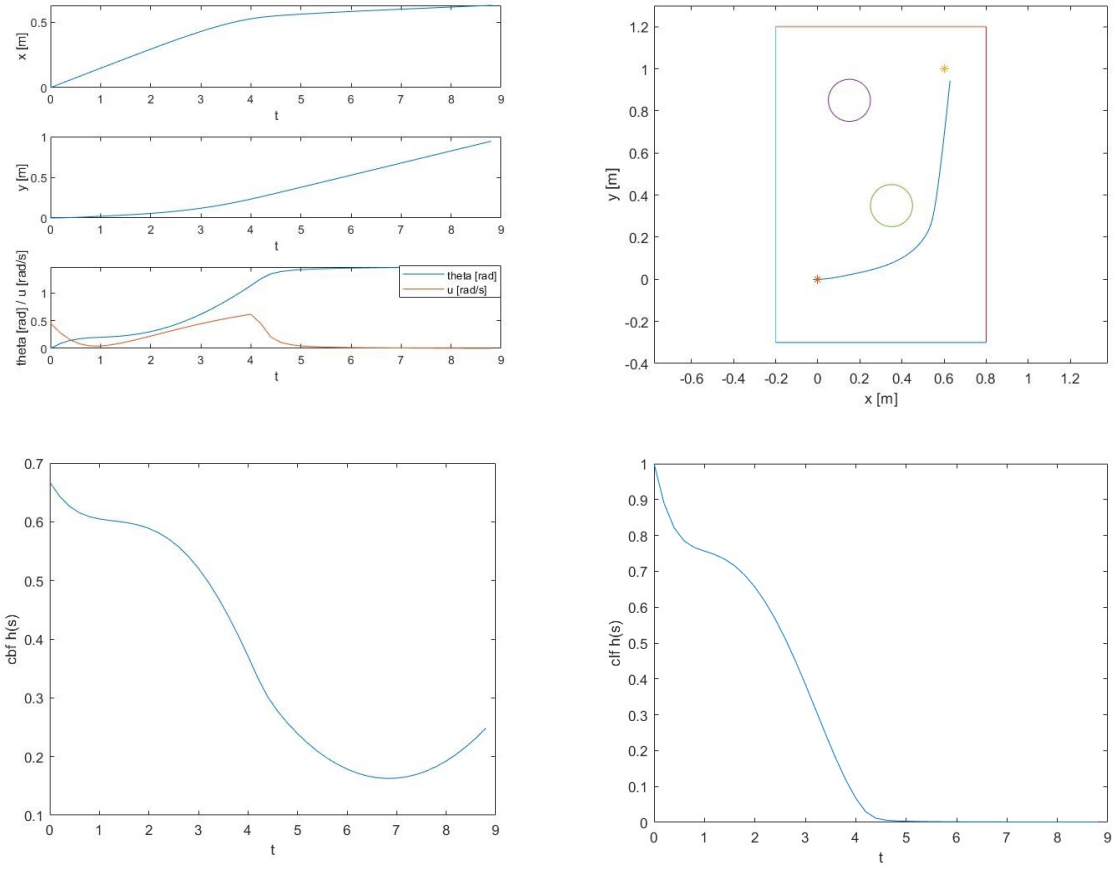
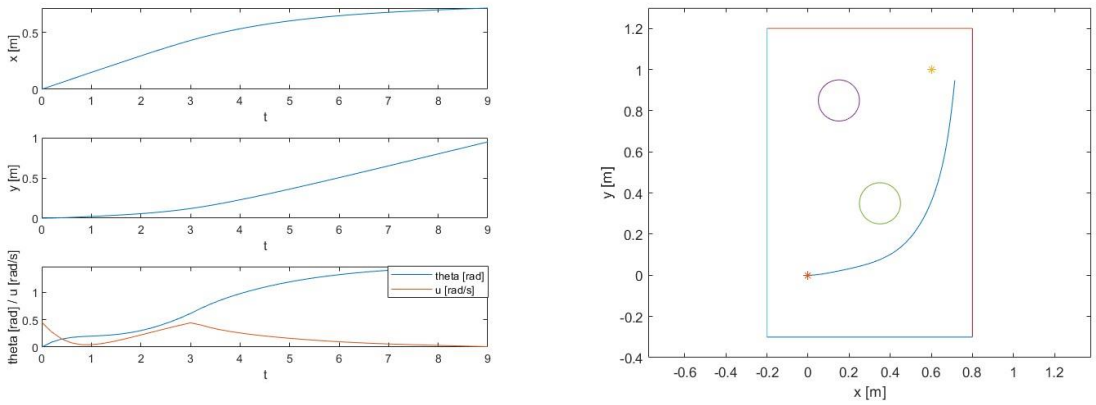


圖 4.18 λ=5, $v=0.15(m/s)$, $\gamma=1$



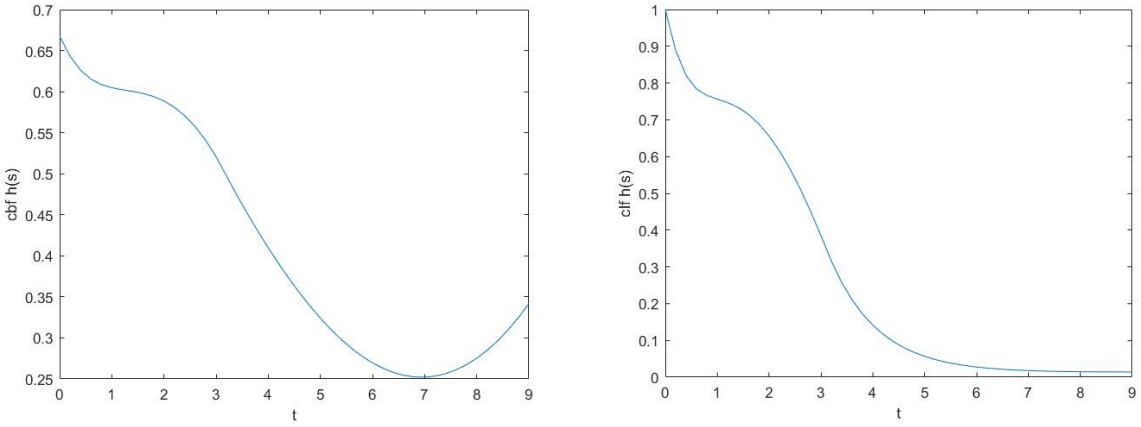


圖 4.19 λ=1, $v=0.15(m/s)$, γ=1

(4) $v=0.15(m/s)$，$x=(0,0,0)$，針對 CBF 的γ，λ值固定，如圖 4.20、4.21。

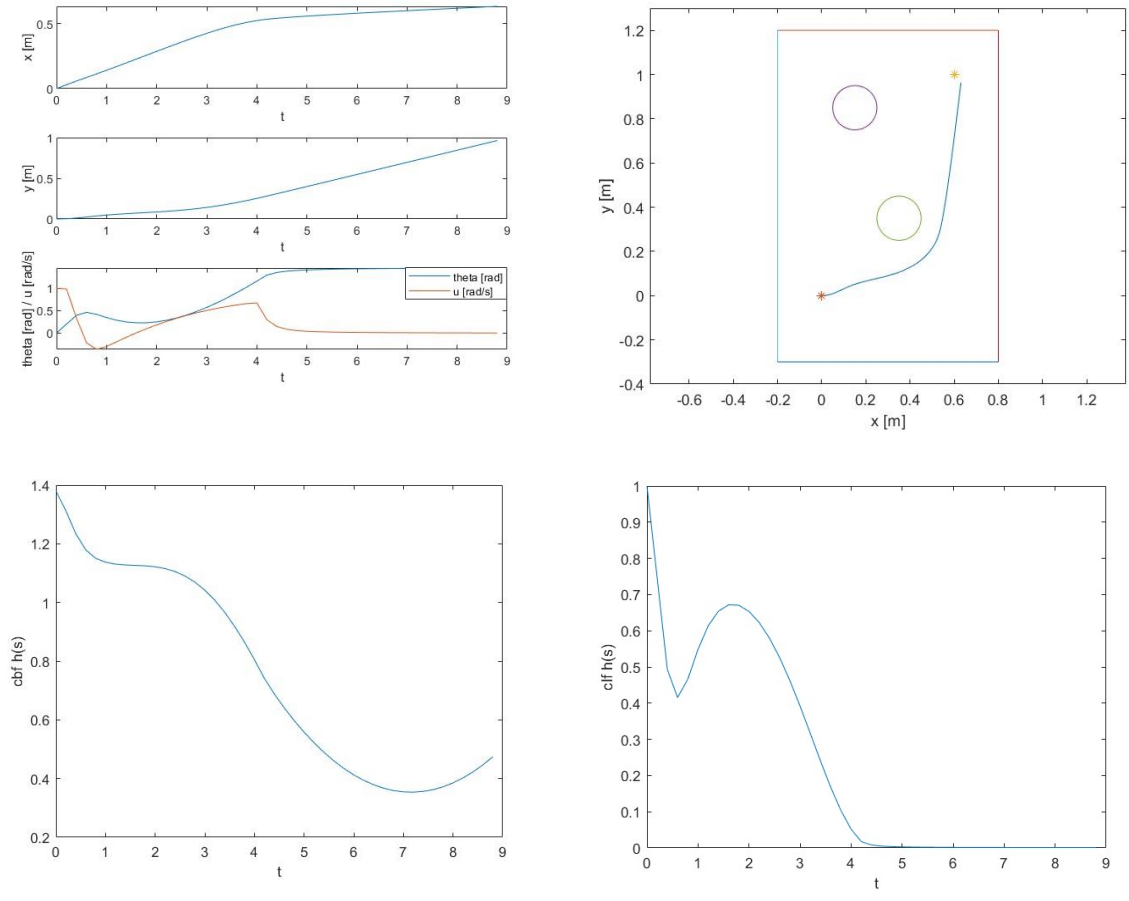


圖 4.20 λ=1, $v=0.15(m/s)$, $\gamma=2$

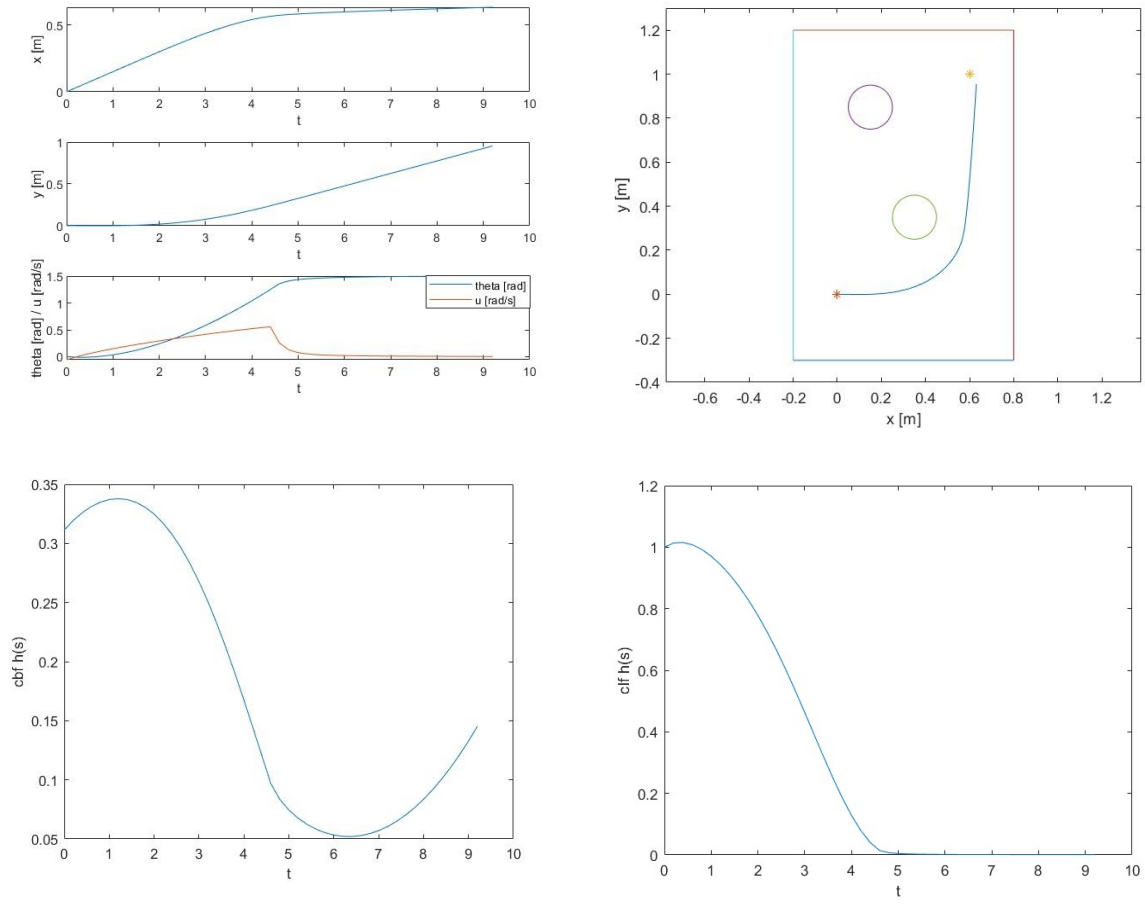


圖 4.21 λ=1, $v = 0.15(m/s)$, $\gamma = 0.5$

圖 4.18、4.19、4.20、4.21 針對λ,γ的結論與圖 4.14、4.15、4.16、4.17 一樣，這邊的重點為針對速度不同的測試，確定找的到 QP 的可行解後，就要開始著重於速度的部分。模擬出速度若是越快，是否能成功規劃出路徑，能夠使移動機器人加速移動到終點，提升效率。在(1)、(2)的測試中，速度設定為$v = 0.1(m/s)$，模擬的時間為 11.8-13.4 秒，而(3)、(4)測試中，時間約為 8.8-9.2 秒，大大節省了到終點的時間，接著考慮第二種測試。

第二種測試：初始姿態的不同角度

(5) $v=0.15(m/s)$，$x=(0,0,\pi/2)$，如圖 4.22。

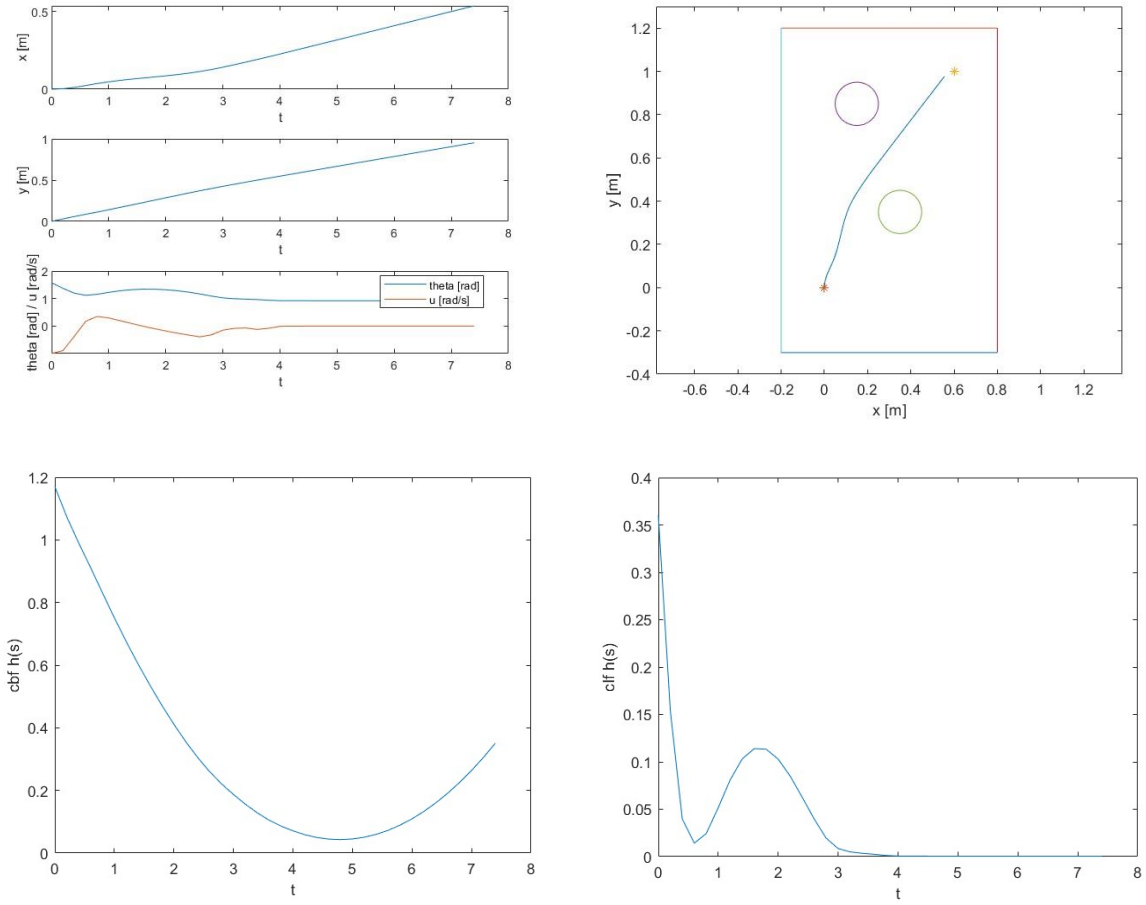


圖 4.22 λ=5, $v=0.15(m/s)$, $\gamma=0.5$

圖 4.22 針對初始姿態的不同角度做討論，可以發現起始角度改為$\pi/2$後，模擬上移動機器人也可以從兩障礙物中間通過，甚至移動時間縮短至 7.8 秒，最後距離目標點的誤差跟第一種測試比也是最小的。

從模擬中得到的結論就是，CLF 的λ值越大越好，收斂速度快會使移動機器人更能接近目標，CBF 的γ值越小越好，系統安全性可以提高，至於起始角度應該由最適合的方向做調整，若調整後的角度其前方有障礙物存在，為了避障速度就無法調大，模擬中移動機器人約在$\pi/4$處有障礙物存在，本專題模擬中角度設定$\pi/6\sim\pi/3$幾乎得不到可行解，所以起始位置角度也是一大因素考量。

4.3.3 CBF-CLF-QP 實驗討論

上機實驗執行 CBF-CLF-QP 控制器的軌跡進行路徑追蹤，經由上一小節調整不同的 CLF 與 CBF 參數，執行的秒數如表 10。根據模擬結果，本專題挑選 $x=(0,0,0)$，λ=5, $\gamma=0.5, v=0.15(m/s)$ 這組參數進行上機實驗，圖 4.23 為本次上機實驗最後的軌跡追蹤圖。

表 10 不同速度下，調整不同的參數，執行時間

速度、起始位置	CLF λ=5 CBF γ=1	CLF λ=1 CBF γ=1	CLF $\lambda=5$ CBF γ=2	CLF $\lambda=5$ CBF γ=0.5
$v=0.1(m/s)$，$x=(0,0,0)$	13.02 (s)	12.92 (s)	11.74 (s)	13.40 (s)
$v=0.15(m/s)$，$x=(0,0,0)$	8.94 (s)	8.72 (s)	8.80 (s)	9.24 (s)
$v=0.15(m/s)$，$x=(0,0,\pi/2)$	11.20 (s)	10.22 (s)	7.78 (s)	11.10 (s)

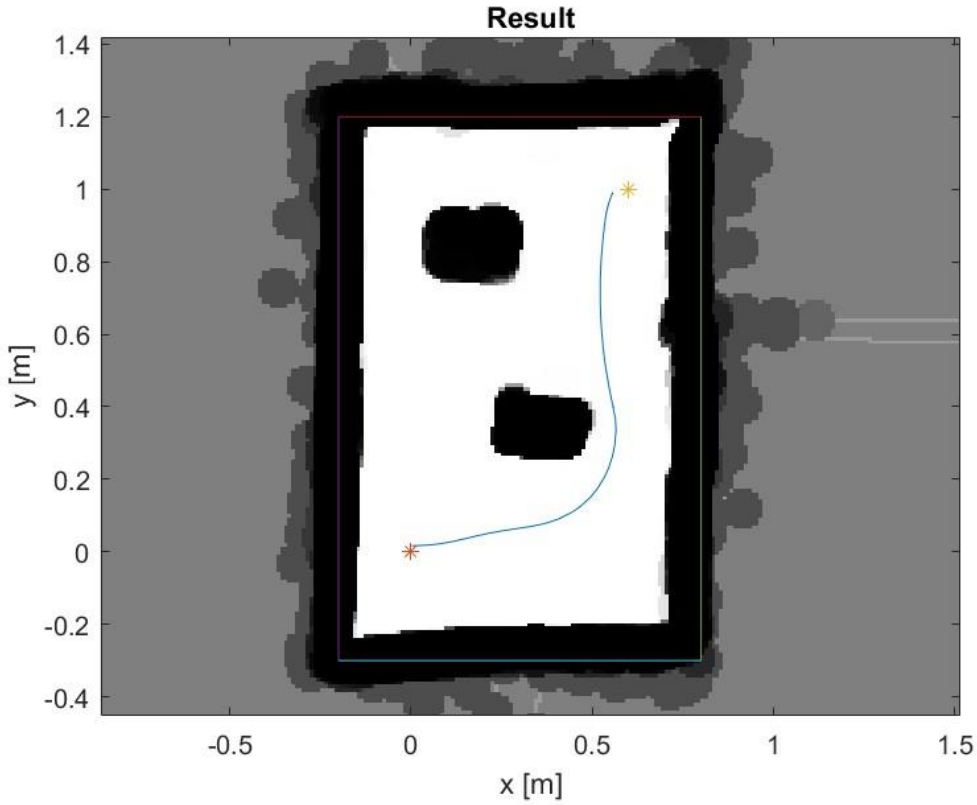


圖 4.23 實驗軌跡追蹤圖

實際移動過程如圖 4.24，從第一列由左向右開始為 CBF-CLF-QP 實驗影片的截圖，共 5 張，執行時間 9.4 秒。

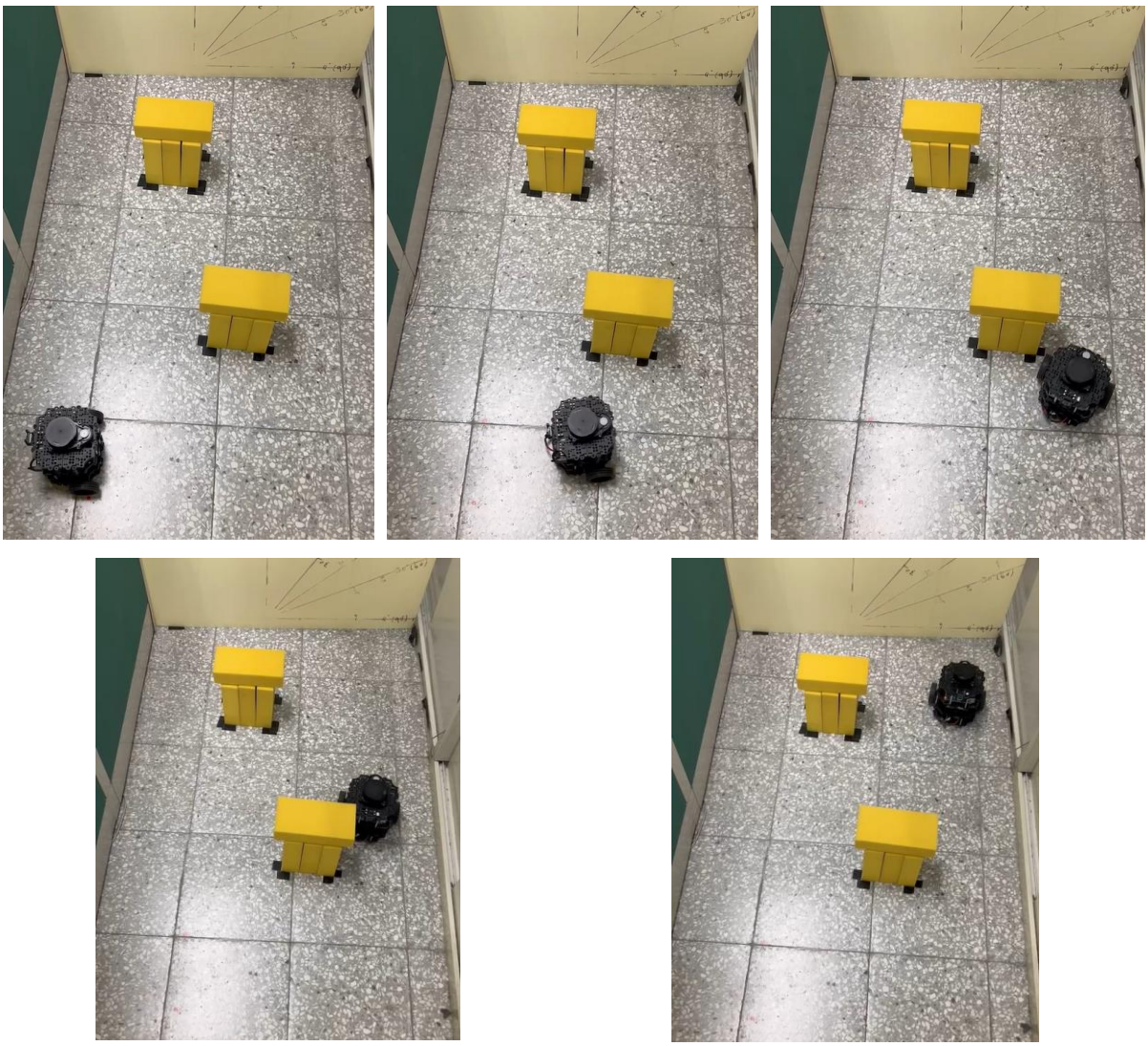


圖 4.24 CBF-CLF-QP 上機過程

參考影片：https://youtube.com/shorts/1l6QkgQpiS4?feature=share

第五章 結論與未來展望

本專題使用 ROS2 的系統架構來建立導航系統，並使用 MATLAB 結合在移動機器人上，採用 Occupancy grid maps 建構一張完整的地圖，使用 Gmapping-SLAM 演算法應用在實際場域中，移動機器人目前本專題包括了兩種路徑規劃方法，分別是第二章的 RRT*與第三章的 CBF-CLF-QP 控制器，實現避障能力。在章節 4.2、4.3 模擬與實驗的結果呈現來看，CBF-CLF-QP 控制器比 RRT*在本專題上更具有優勢，CBF-CLF-QP 控制器的優點在於對系統穩定性和安全性要求可以有一定的保障，模擬與上機的時間也比 RRT*來的快速。

RRT*的優點是適合解決高維空間和復雜的路徑規劃問題，CBF-CLF-QP 控制器能夠保障系統的穩定性與安全性，缺點是使用 QP 會在計算成本上有較大的負擔，然而 CBF 與 CLF 設下的限制條件有時會導致 QP 沒有可行解，目前正規劃嘗試使用線性二次調節器 LQR(Linear Quadratic Regulator)來避免求解 QP 的問題，結合 CBF 與 RRT*來設計一個新的控制器，透過全域結合局域的方式，希望能夠改善路徑規劃的效率。