第一章 簡介

1.1 研究目的

在所有種類的遊戲中，賽車遊戲已有非常古老的歷史，古老的賽車 AI (Artificial Intelligence) 被應用在遙控機械車上面，目的在於單純地跑完指定的賽道到達終點，隨著資訊不斷地發達，賽車遊戲漸漸地出現在電腦上，AI 在賽車遊戲中的應用扮演了不可或缺的角色。一個良好設計的 AI，NPC(非玩家角色)，也就是電腦操控的賽車手，會讓玩家彷彿在遊戲中跟真人對戰一樣地刺激，使玩家獲得對戰的樂趣。

時至今日，隨著賽車技術的發展越來越成熟，人們開始思考能否在最短時間完成彎道，進而對彎道作出研究以發明出更好的”攻略”方法，從最短路徑、最小弧度過彎到外內外策略搭配彎心點的跑法，以及過彎前減速和加速出彎等等，都是能夠有效縮短過彎時間的方法。

早期的汽車控制器 AI，在面臨使用者設計的一連串彎道時，用”試誤”的方法找出更佳的路徑，這樣的方法往往不夠實際而且浪費太多時間，之後人們為了要讓控制器 AI 能夠變得更”聰明”，採用了在每個賽道上設立檢查點(check point)的方法，讓 AI 能夠循著設計好的路線跑出使用者想要的軌跡，然而這種方法並不理想，有些研究者則嘗試讓 AI 能夠獲得彎道資訊，以利 AI 能夠根據資訊以數學方式計算出相對應的過彎軌跡和入彎、出彎速度，使用此種方式所設計出來的 AI 在穩定度和時間的表現上有一定的水準，常見於現今的賽車遊戲中，

廣為一般設計者所接受。但對資訊的應用仍屬於較低階的方式，且變化性較低。

藉由多項的低階資訊可以組合成較高階的資訊，例如:彎道的曲率半徑與彎道類型，這兩項可以囊括大部分的彎道種類，AI 控制器根據上述資訊再結合自身與彎道距離，可以思考出較進階的過彎策略，決定入彎速度、入彎點、出彎速度與出彎點，以達到軌跡的高變化性與靈活度。

1.2 論文架構

本篇的第二章討論各種 TORCS 相關文獻的研究目的和方法以及結論，第三章為 AI 開發環境 TORCS 的介紹和 AI 引擎的基底架構，第四章詳述研究的方法，第五章展示研究的成果，第六章總結本篇結論以及未來展望。

第二章 文獻縱覽

2.1 玩家模擬(Player Imitation)

AI 應用已有相當的歷史，過去廣泛的被應用在策略遊戲，如圍棋、西洋棋、雷神之槌（即時 3D 射擊遊戲），近幾年來才慢慢地擴展至賽車遊戲方面。在[4]中展示了 TORCS(The Open Racing Car Simulator)在 2010 舉辦的官方比賽中的參賽 AI，其方向不是為了跑出最快的成績而是讓 AI 能夠良好的模擬出某玩家的行為模式，為了達成此目的，設計部分採用了類神經網路來推測出各種可能的軌跡和目標速度，AI 必須做出相對應的動作(檔次的升降、油門的控制、轉向的控制)來遵循以上的部分，類神經網路透過某位玩家的駕駛資料當作輸入來讓 AI 能夠在玩家未曾跑過的賽道上面也能重現該玩家的駕駛行為，結果顯示 AI 在模擬全程時約超過 20%的比例在模擬玩家時失誤(該區段的速度比玩家速度慢)，會造成此結果的原因是 AI 沒有辦法百分之百正確地模仿出玩家行為，中間只要出現一點點小失誤就可能導致無法預期的結果(碰撞、打滑、卡點、駛出軌道外)，AI 在模仿失誤時不會立即作出回復原有失誤的動作，而是透過類神經網路再規劃出一條新的預測軌跡，而此條新軌跡通常不是最好的，已經和原本的有所差距，如此一來當失誤一再發生時，就會導致 AI 出現不穩定的結果，這就是 AI 無法百分之百重現玩家駕駛行為的瓶頸。此篇提到可能降低失誤的方法是在訓練時採用數種與目標賽道相似程度高的賽道以避免因為賽道間的差異而造成 AI 發生失誤的機率。

[5]中提到了在駕駛者即將過彎前，其可視範圍是有限的，只知道彎道前半段的資訊無法有效地辨識彎道的急緩程度，進而無法準確且安全的過彎，也點出了若只以 TORCS 內建的距離偵測器(track sensor)當作 AI 轉向依據時在辨識彎道時的盲點。所以為了能夠實現此想法，此篇論文展示了一個以 TORCS 內建的距離偵測器為主建立的賽道模型以提供 AI 學習，此種賽道辨識系統先讓 AI 一邊在暖身圈以低速安全的方式駕駛確保能夠順利抵達終點，一邊以偵測器所測得的資料透過簡單的分類器將沿途偵測到的賽道形狀分類成六種人類可輕易辨識彎曲程度等級的彎道類型，如此以直接建立完整的賽道模型，模擬人類在駕駛時對賽道的記憶，此模型提供的資訊有利於 AI 思考有如專家級的策略，並在剩下的圈數作出"規劃"，以達成更好的成績，此種 AI 的學習機制是由專家資訊和一個簡單的想法:知道整段賽道的情況比知道片段彎道來的更有利。模型建立完成之後，AI 會執行軌跡最佳化，進而控制油門、剎車、轉向時機點。最後將原有的 AI 去做參數的微調，達到模擬效果。作法是計算彎道緩急程度(ρ)定義 6 種賽道彎曲程度等級和初始軌跡、速度紀錄建立相應速度函數(corresponding speed matching function)，該函數可以視為一維線性函數來做最佳化和調整，動態的將賽道劃分等級區段之後，以類神經網路投入兩種不同的駕駛者(AI 和 human driver)下去訓練，分別找出最佳值，規劃出最佳軌跡和目標速度，以此比較兩者之間的差異。此種方法雖然能夠讓 AI 隨著駕駛時間越長，將跑法趨於更完美，但是因為受限於只能在暖身圈時收集資料，在比賽前半段時處於摸索階段，所以在表現上面剛開始會處於弱勢，結果顯示有些時候速度函數並沒有被訓練得很完美，有時候會低於官方網站提供的各彎道參考目標速度。

在[6]中作者提供了監督式的玩家模擬學習機制，在分類上是屬於直接模擬

的方式,直接模擬的方法是由玩家設定模擬的環境和限制條件以及目標效率評估標準,在限制的輸入和輸出下可以找出符合目標條件的結果。一般而言,採用直接模擬方式極度受限於玩家資料的完整度,往往需要大量且多樣的資料才可能有較高的模擬程度,本篇的目的在於藉由有限的玩家資料和賽道資訊讓AI能夠高效率的模擬學習玩家行為,藉由導入車子與賽道前方的高階資訊(軌跡和速度,前進方向、彎道彎曲程度)來取代低階資訊(車子與賽道的距離)的想法來改善直接模擬的限制,進而模擬出人類的高階行為,在本篇提供的方法下,模擬效率高達百分之八十五以上,讓直接模擬的方式有更高的實用性。在實際的作法上,作者使用了兩種不同的高低階資訊(1) 前瞻偵測器(lookahead sensor)和(2) 距離偵測器(track sensor)以及兩種演算方法:(1) K個最鄰近點(K-Nearest Neighbour) 和 (2) 類神經網路(Neural Networks evolved using NEAT),來作為AI在學習行為上的策略上的比較,為了能夠在原本只有最基本駕駛策略的AI車隨著駕駛時間的增加而演化出更多更好的玩家行為,制定有限數量的賽道,記錄玩家在這些賽道中的資料,之後讓AI在這些賽道之外的新跑道上憑藉玩家資料和演算方法演化出人類駕駛行為,其中包含目標速度、轉彎時機、油門和剎車以及入彎點等等,模擬出人類在不同環境下的行為差異,以達到模擬效果,在演化的過程中,也加入了預測錯誤的修正機制,讓AI可以即時的得知下一步預測目標與真正的目標之間的差異度,讓AI能夠即時作出修正的動作,以達到系統的穩定度,能夠更準確地模擬複雜的人類駕駛行為,藉由四種的搭配組合找出效率最佳的方法,作者在結論指出影響效率的主要原因有兩個:一是在玩家特徵的模擬程度上越高越有利,一是若AI嘗試只靠低階資訊作出高階駕駛行為則會嚴重降低效果。

另外有一些相關的例子，[7]中提到AI在辨識過彎之後的直線應該更加準確且爭取時間提早加速，以提升整體效能，另外如何安全有效地超越敵車且避免非必要的碰撞也是提升效能的關鍵，作者採用著名類神經網路演算法(NeuroEvolution of Augmenting Topologies)，”演進”過彎和超車的技巧兩部分，並且改良了原有的距離偵測器(track sensor)，取其最前方的三個角度自訂出新的偵測器(frontal sensor)，目的在於正確判斷車子目前處於過彎前還是過彎後的狀態，以利爭取時間加速，輸入三個參數(偵測器距離、前進方向、速度)，輸出參數控制過彎後油門和轉向的控制，讓AI作出彎後立即加速的駕駛行為。另一部分，將敵車偵測器(opponent sensor)依角度分成三個等級，來區分敵車距離和相對位置(前方、並列、後方)，以上述當作類神經網路演算法的輸入，尋找最佳的超車行為，增進效能。研究結果顯示，只單一增進過彎技巧或超車技巧對效能並沒有顯著提升，但兩者都使用的效果卻有明顯提升，此篇著重於重點加強式的方法，以演算法增進多項方面的駕駛技巧對整體效能有顯著改善。

[8]提到在多車競賽的模式下，在某些情況下刻意阻擋敵車路線，可以讓自己贏得更有利的條件，阻擋敵車的策略，對本身的最佳化沒有幫助，唯一的目的是可以在某些特定模式比賽下贏得最後勝利。本篇提供了進階的阻擋敵車策略，在單純環境下，透過使用模糊邏輯（Fuzzy Logic）的方法建立玩家模型，制定三種不同積極度的阻擋策略，從遊戲引擎本身提供的環境參數、敵方車軌跡當作輸入經由模糊系統產生預測目標速度與目標方向，進而再由這些參數當作輸入進而一步再產生油門、煞車、轉向、檔次等直接控制車子的加速和行進方向，設定

了時間限制和敵我車順位以及距離來作為阻敵策略成功率的評估標準，結論是在實驗設定的單一環境下(一段直線賽道)採用積極度越高的阻擋策略可以表現出顯著的結果，但是考量到賽車環境的複雜度遠高於實驗設定的環境以及敵車在不同環境下會作出實驗預期外的行為，在真正的實用度上面有待加強，另外阻擋策略也偏向簡單，僅僅隨著敵車軌跡的參考度和提升反應時間無法作出大方向的思考策略，只對單一目標的思考在多車同時競爭的環境下顯得不足。值得一提的是，這種以敵車軌跡為參考和自身位置為考量而思考出阻擋敵車路線的擬人類高階思考模式，在玩家模擬上有很大的研究空間。

[9]中的建模方法主要訴求是在模擬程度和效能之間做取捨，希望藉由直接模擬和間接模擬方法的混合使用能夠找出效能與玩家相似度之最佳平衡點。在使用玩家紀錄來建立操控模型(直接模擬)和微調共通模型(間接模擬)之間設定目標達成率(Fitness Measures)的方式，如:在固定時間內跑出最遠距離，過某些彎道時的轉向操縱值不可過大，某些賽道的目標速度值等等，在直接模擬和間接模擬建立的模型中，設定二到三個目標物件(最短完成時間、速度、轉向)，藉由MOEA(Evolutionary multi-objective optimization algorithm)的方式，以目標物件達成率，找出在Pareto平面的非支配解(Pareto front ofNon-dominated solutions)，這些解即代表了兩種模擬方法之間的權重值，找出最佳平衡點，最後在以玩家收集資料範圍外的新賽道，測試其結果的模擬度和強韌度，結果顯示是應新賽道的強韌度方面，用此種方法可以很容易達成，但在模擬相似度方面，若玩家的駕駛行為偏向保守則相似度會比偏向積極的玩家來的高，其原因是因為模仿較積極的玩家容易出現失誤，而失誤會造成原有的強韌度下降，所以此篇的

演算法是偏向於在確保足夠強韌度的情況下達到最高模擬的模擬程度，作者也提到了其他還未嘗試的類似演算法，有可能突破目前方法的模擬程度，未來可能會與本篇方法作比較。

2.2 軌跡最佳化(Trajectory Optimization)

[11]提供了低階的最佳化方法，作者認為尋找最佳軌跡是任何賽車遊戲AI的基本，大部分是透過人類設計出最短時間內跑出最長的距離的路線，由於TORCS中的整個賽道是由很多的一定距離的區段賽道組合而成，所以可以在區段賽道中的起點計算出最佳的路線到終點，再把數個最佳區段路線組合而成一整段賽道的路線，而AI就可以沿著這些計算出的最佳路線已達到最佳化目的，本篇則提供了一個自動尋找最佳路徑的方法，藉由基因演算法在兩個互相衝突的方法中找出最佳的權衡:最短路徑(Shortest Path)和最小曲度路徑(Minimum Curvature Path)，找出最佳的組合，以達到最佳化。實作方法是分別先分別計算以最短路徑和最小曲度路徑演算法計算出整段軌跡在區段中的起點和終點，給定一連串不同的權重值，再利用基因演算法以平均單圈時間最短者為評估標準在兩種路徑中尋找出最佳權重，最後再與只用單一方法的平均最佳時間相比較。結果顯示在十一個不同的賽道中有八個賽道中的最佳軌跡所需時間少於只用單一方法的時間，整體表現突出，在低階最佳化中有顯著效果。

2.3 基底架構(Baseline Architecture)

在AI駕駛引擎基本架構的實作方面，[10]將各種駕駛可能的基本思維與行為以六種行為模型構成完整的系統，包含檔次升降、目標速度、油門剎車控制、轉

向控制和偵測對手車有利於超車或者避免碰撞以及車輛跑出賽道外或者卡點的回復機制，其中的目標速度模組是以模糊系統作為基底，以距離偵測器(track sensor)當作輸入，經過系統輸出目標速度值，控制油門或剎車，文中提及的各項模組皆是屬於較低階的機制，適合AI設計者快速建立一個基本駕駛引擎。

第三章 TORCS 與 AI 基底架構

3.1 TORCS(The Open Racing Car Simulator)

TORCS [12]是一款高技術水平且公開原始程式碼的競賽車輛模擬器，程式碼是在 LINUX 環境下由 C/C++寫成，目前最新的版本是 1.3.1，程式碼通過 GPL 認可，TORCS 提供複雜的物理引擎，全 3D 可視環境，數種多樣性的賽道，多樣性的車輛模組，以及多變的遊戲競賽模式(例如:快速競賽模式、耐力賽模式、錦標賽模式和練習模式...等等)，現實賽車中的很多細節都可以被物裡引擎精確地模擬，例如:車子和車子之間的牽引、空氣動力原理、油耗等。

遊戲中的車輛都被各自設計的 AI 操控駕駛，AI 可以在各個控制步驟中(每個遊戲單位時間)擷取目前的競賽狀況，包含自身車輛和賽道之間的資訊，自身與敵方車輛間的資訊，都可以透過如表 3-1 所示的偵測器中取得，而各種控制動作的指令則由表 3-2 所示效應器(effector)接收直接轉成控制行為，以利接下來作出油門、剎車、轉向、離合器、升降檔...等等的控制，程式碼中提供了數個已開發的 AI 模組，新入門的玩家可以透過這些原有的模組當作基本架構建立屬於自己的 AI 模組。

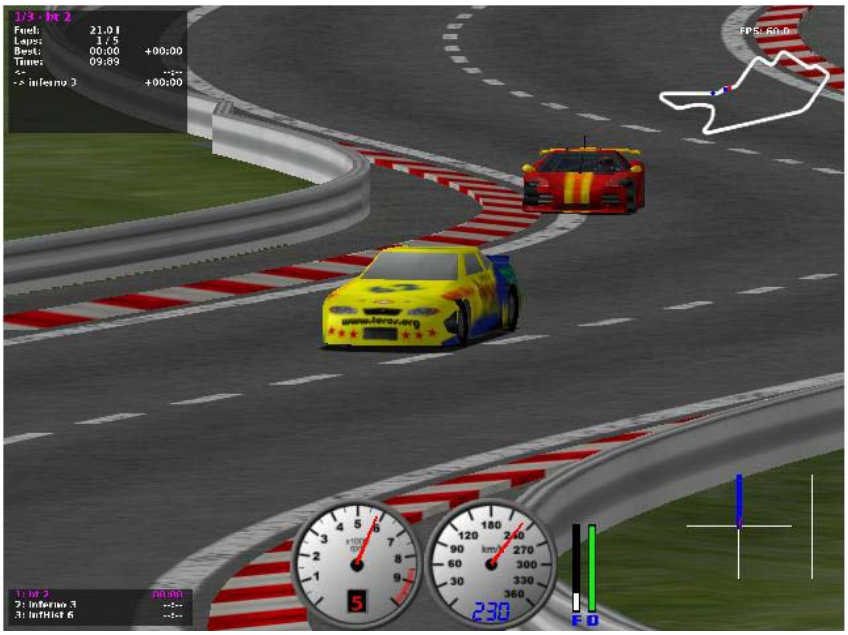


圖 3-1　TORCS 遊戲截圖

TORCS 提供了極佳的 AI 開發和測試環境，它具有以下優點:

- TORCS 比一般市面上商業賽車遊戲更具有研究價值，提供了進階的模模和客製化的環境適合計算智能的研究者當作效能評估的基準。
- 複雜的遊戲引擎可模擬各種物裡效果，對於賽車的擬真度比一般商業賽車遊戲來得更真實。
- 有熱誠的 AI 開發者可以隨時對原有的引擎作修改和更新，或者將開發好的 AI 與大家分享。

由於 TORCS 在開發上的簡易性，以軟體開發 AI 並且在遊戲中測試其效能，被視為是計算智能研究者泛用的開發平台。

表 3-1TORCS 偵測器列表

名稱	範圍（單位）	描述
angle	[-π, +π]（rad）	車子與賽道中心線的夾角。
curLapTime	[0,—]（Sec）	目前圈的完成累積時間。
damage	[0, —]（point）	車子因碰撞造成的損傷程度。
distFromStart	[0, —]（m）	車子與起跑線的距離，範圍是從零到賽道全長。
distRaced	[0, —]（m）	車子在起跑後的累積里程數。
fuel	[0, —]（l）	車子目前的燃料存量。
gear	{-1,0,1,2,…,6}	目前檔次: -1 為倒退檔，0 為空檔以及 1-6 檔。
lastLapTime	[0, —]（s）	完成最後一圈的累積時間。
track	[0,100]（m）	19 個 Sensors 從-180°到180°每間隔10°為單位設置 Sensor，偵測範圍從 0（m）到 100（m），偵測賽道邊界與自身的距離。
opponents	[0,100]（m）	36 個 Sensors 從-180°到180°每間隔10°為單位設置 Sensor，偵測範圍從 0（m）到 100（m），偵測敵方車與自身的距離。
racePos	1,2,…	車子起跑時的相對位置排序。
rpm	[2000,10000]	引擎每分鐘的轉動數。
speedX	—（km/h）	車子垂直方向的速度分量。
speedY	—（km/h）	車子水平方向的速度分量。
position	—	車子距離賽道中心線的距離，其值依照賽道寬

		度作 Normalization，-1 代表車子在賽道右邊界線上，1 代表車子在左邊界線上，值大於 1 或者小於 -1 代表車子在左(右)邊邊界外。
wheelVel	[0, —](rad/sec)	車子輪胎轉動速度。

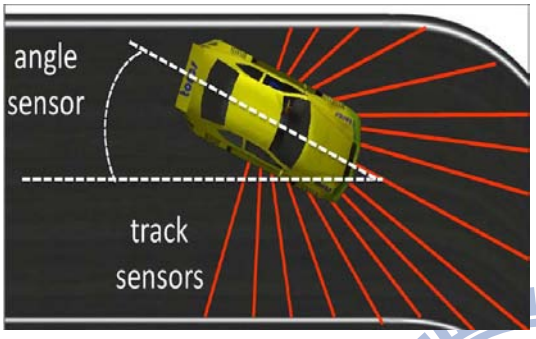


圖 3-2 表 3-1 中部分偵測器示意圖 1

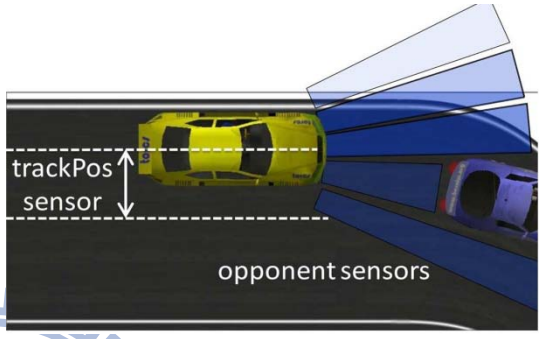


圖 3-3 表 3-1 中部分偵測器示意圖 2

表 3-2 TORCS 效應器列表

名稱	範圍（單位）	描述
accel	[0,1]	模擬車子的油門，0 代表放空，1 代表全滿。
brake	[0,1]	模擬車子的剎車，0 代表放空，1 代表全滿。
gear	-1, 0, 1, 2, …, 6	車子目前的檔次，-1 代表 R 檔，0 代表空檔。
steer	[-1,1]	車子的轉向程度，-1 代表方向盤打向最右邊，1 代表最左邊。

3.2 AI 引擎之基底架構

在設計進階的 AI 使其有更高階的技巧之前，AI 本身必須要有一些較低階的基本駕駛行為，如此一來才能讓更高等的模擬行為或駕駛技巧輕易地嵌入其中。AI 控制的基本架構由五個不同功能的模組組成，分別是:

1. 升降檔控制(Gear Control model)
2. 目標速度(Target Speed model)
3. 速度控制(Speed Control model)
4. 轉向控制(Steer Control model)
5. 回復控制(Recovery Control model)

升降檔控制

升降檔控制模組主要由車子馬達的轉速作為判斷依據執行維持、升檔或降檔，每段檔次都有其轉速最大值和最小值，超過則升高檔次，反之則降低檔次，轉速表如表 3-3 所示，檔次控制系統接收車子目前檔次和馬達轉速當作輸入，判斷是否升降檔，再將參數傳遞給效應器作出反應。

表 3-3　檔次轉速表

目前檔次	1	2	3	4	5	6
升檔	≧9000	≧9000	≧8000	≧8000	≧8000	
降檔		≦3000	≦3000	≦3000	≦3500	≦3500

目標速度

此部分設計參考[10]中之第三章 3.2 節所提到之 Target Speed Model，目標速度模組主要的目的是計算出車子在每個遊戲單位時間中，在賽道區段中車子該有的速度值，，須考慮兩種情況:賽道外和賽道內，在賽道外的情況，表示車子目前需要回復至賽道內，不採用此模組而由回復控制模組負責，若在賽道內，則以圖 3-4 中的三個距離偵測器值當作輸入: (S[9]代表圖 3-4 中標號 9 所側得之距離，S[8]代表標號 8 測得距離…以此類推)

1. 正前方偵測器距離:S[9]。
2. 正負 10 度角:S[8]，S[10]之中取最大值。(Maximum(S[8],S[10]))。
3. 正負 20 度角:S[7]，S[11]之中取最大值。(Maximum(S[7],S[11]))。

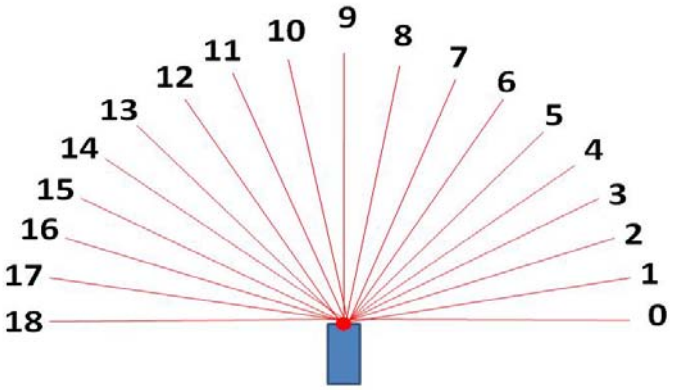


圖 3-4 距離偵測器示意圖

上述每個輸入變數將會被圖 3-6 中(a)、(b)和(c)中所示的成員函數(membership function):Low, Medium, High，代表不同距離所屬之等級，經由模糊系統轉成模糊化值。

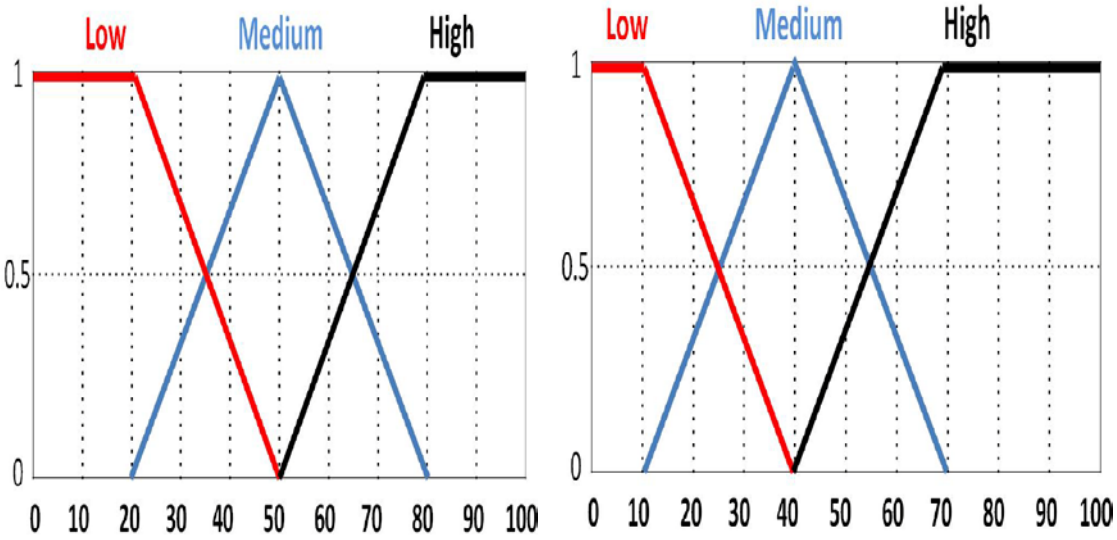
這套模糊系統就是此模組的核心，其模糊規則如圖 3-5 所示，藉由車子正前

方的距離偵測器來判斷車子目前處於何種狀態。

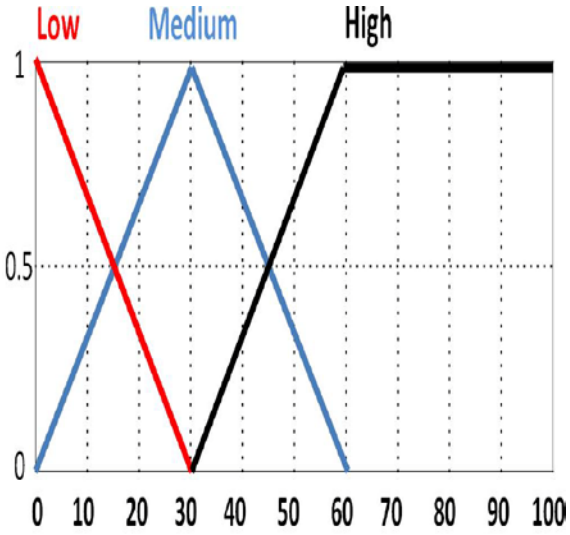
最後的輸出值:目標速度($Target_{\text{speed}}$)，如圖 3-7 所示的單一輸出，最大值 200 km/hr，最小值 50 km/hr。目標速度值將會被送至速度模組當作輸入，以控制油門和煞車，將目前速度升(降)至目標速度值。

1. If $front$ is $High$ then $Target_{speed}$ is TS_1
2. If $front$ is $Medium$ then $Target_{speed}$ is TS_2
3. If $front$ is Low and Max_{10} is $High$ then $Target_{speed}$ is TS_3
4. If $front$ is Low and Max_{10} is $Medium$ then $Target_{speed}$ is TS_4
5. If $front$ is Low and Max_{10} is Low and Max_{20} is $High$ then $Target_{speed}$ is TS_5
6. If $front$ is Low and Max_{10} is Low and Max_{20} is $Medium$ then $Target_{speed}$ is TS_6
7. If $front$ is Low and Max_{10} is Low and Max_{20} is Low then $Target_{speed}$ is TS_7

圖 3-5 目標速度系統中的模糊規則



(a) (b)



(c)

圖 3-6　模糊規則中的成員函數

(a)為 Front(b)為 Max10 (c)Max20

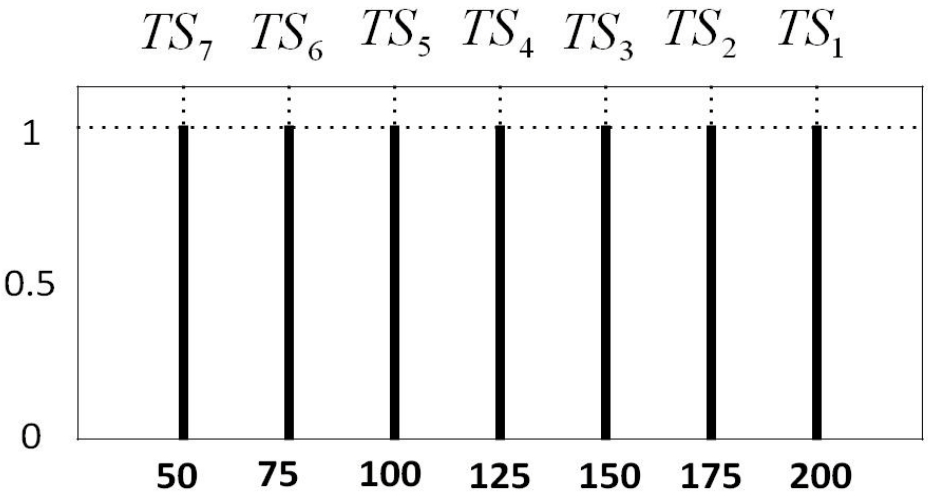


圖 3-7　目標速度各輸出值

速度控制

此部分設計參考[10]中之第三章 3.3 節所提到之 Speed Control Model，速

度控制模組主要目的是將目標速度值當作輸入，經過函式(1)計算出加速或減速，將值輸出至效應器(effector)的油門和煞車，以達到車子在該賽道區段的目標速度。Gas 值範圍落在[-1,1]之間，當 Gas < 0 時，|Gas|為煞車效應器(brake effector)的輸入值，反之當 Gas >= 0 時，Gas 值則為油門效應器(accel effector)的輸入值。

$$Gas(speed - Target_{speed}) = -1 + \frac{2}{1+e^{speed-Target_{speed}}} \quad (1)$$

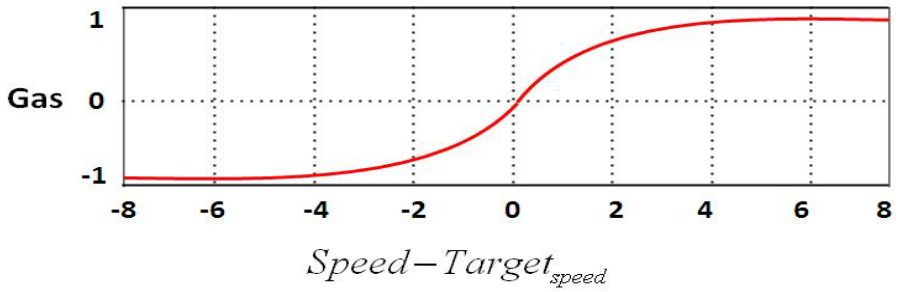


圖 3-8 油門程度與函式輸出值之關係圖

轉向控制

轉向控制模組負責將計算出的轉向值傳遞給轉向效應器(steer effector)以控制車子的轉彎方向。

車子的轉向需考慮三種情況: (1)車子在賽道內 (2)車子在賽道外 (3)車子處於倒退檔(gear=-1)，(2)情況下讓車子先回到賽道內為首要目標，所以屬於回復控制的範圍，(3)情況下，表示車子目前可能是卡點或是車子經碰撞後車頭處於逆方向，回復機制判斷後將檔次設定成倒退檔，藉由公式(2)可計算出在倒退檔狀態下回到賽道內轉向效應器(steer effector)所需的轉向值，angle 屬於表 3-1 之一的偵測器，*steerlock* 代表車輪轉向之最大值(約 45 度，為弧度值)。

$$steer = \frac{-angle}{steerlock} \quad (2)$$

在(1)的情況下，採用公式(3)計算轉向值(S[10]代表圖 3-4 中標號 9 所側得之距離，S[8]代表標號 8 測得距離…以此類推)，採用分佈在正前方左右 40 度內的 8 個偵測器，隨著偵測器偏離正前方的角度增加，其權重值(0.5, 1.0, 1.5, 3.0)越高，隨著彎道越急轉彎幅度則越大。

整體而言，公式的目的是讓車子維持在賽道的正中央，若車子即將進入彎道時，某半邊的偵測器距離會增加，另外半邊會減少，如此可計算出正確的偏差值並轉向。

$$steer = \frac{(0.5*S[10]+1.0*S[11]+1.5*S[12]+3.0*S[13])-(0.5*S[8]+1.0*S[7]+1.5*S[6]+3.0*S[5])}{S[10]+S[11]+S[12]+S[13]+S[8]+S[7]+S[6]+S[5]} \quad (3)$$

回復控制

回復控制模組負責判斷車子目前是否在賽道內，有無卡點情況，並且控制多個效應器(檔次、轉向、油門)讓車子回到賽道內正常駕駛。回復機制有兩個步驟，第一步先判斷車子是否卡點，如果是則藉由倒退檔讓車子回到賽道內，第二步再判斷車子是否在賽道外穩定駕駛狀態，如果車子因為在賽道外打滑則先停止轉向，等車子回到穩定狀態後再將車子駛回賽道內，虛擬程式碼如下圖 3-9, 圖 3-10 所示。

```
If(car is stuck)
        steer = direction to track
        gear = reverse
        accelerator = full
        brake = empty
else
        car is not stuck
end
```

圖 3-9　回復控制虛擬碼 1

```
if(car is offtrack and gear is not on reverse)
        if(car is sliping)
            steer = no control
        else
            steer = direction of the ce nter of ne xt track s egment
        end
 else
      Co mmon Drive
end
```

圖3-10　回復控制虛擬碼2