國立交通大學

多媒體工程研究所

碩士論文

於模擬賽車遊戲學習超車與阻擋行為之研究

The Study of Learning Overtaking and Blocking Behaviors in a Simulated Car Racing Games

研 究 生：黃瀚賢

指導教授：王才沛　教授

中 華 民 國 一 〇 二 年 八 月

於模擬賽車遊戲學習超車與阻擋行為之研究

The Study of Learning Overtaking and Blocking Behaviors in a Simulated Car Racing Games

研 究 生：黃瀚賢　　Student：Han-hsien Huang

指導教授：王才沛　　Advisor：Tsai-pei Wang

國 立 交 通 大 學

多 媒 體 工 程 研 究 所

碩 士 論 文

A Thesis
Submitted to Institute of Multimedia Engineering
College of Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in
Computer Science
Aug 2013
Hsinchu, Taiwan, Republic of China

中華民國一○二年八月

於模擬賽車遊戲學習超車與阻擋行為之研究

學生：黃瀚賢　　指導教授：王才沛

國立交通大學多媒體工程所

摘要

這篇論文的內容是在研究 TORCS 平台上，以學習演算法訓練出來的賽車 AI，在不同阻擋者以及不同軌道條件下的表現。論文一開始會先從為何我們想要研究更具侵略性的 AI 駕駛行為開始講起，接著介紹 TORCS 這個開放程式碼的賽車遊戲平台架構以及在各個研討會或期刊上與 TORCS 相關的研究。因為 TORCS 所內建的賽車 AI 其駕駛行為皆不具侵略性，在超車時如果剩餘的軌道寬度沒有達到一定的安全寬度，則會取消超車行為，且內建 AI 並不存在阻擋行為，若後車速度較快，前車會將路徑讓給後車。所以我們利用學習演算法中的 Q-learning algorithm 來發展更具侵略性的超車與阻擋行為，並統計在不同情形下賽車的數據來評比表現。

The Study of Learning Overtaking and Blocking Behaviors in a Simulated Car Racing Games

Student：Han-Hsien Huang　　Advisor：Tsai-pei Wang

Institute of Multimedia Engineering

College of Computer Science

National Chiao Tung University

Abstract

In this thesis, we research the learning of overtaking and blocking behaviors of AI (non-human) drivers on TORCS, a simulated car racing platform. At first, we introduce why we focus on the more aggressive driving strategy. Secondly, we introduce the TORCS racing game architecture and some researches in TORCS or AI-related conference. Because of built-in AI didn't have aggressive driving skill, when performing overtaking, if remain track width has not reach to a safely width, the built-in AI will cancel the overtaking behavior. Additionally, the built-in AI didn't have blocking behaviors. If the opponent's speed was higher than theirs, it will let the opponent pass. In order to improve built-in AI's driving strategy, we use Q-learning algorithm for learning behaviors and analysis their overtaking and blocking performance in different conditions.

誌謝

兩年的研究所學業及此篇論文能順利完成，首先要先向我的指導教授，王才沛老師道謝。沒有老師在想法上的引導與實驗中按部就班的階段性目標的訂定，是無法完成這篇論文的。謝謝老師！並且，要謝謝我家人的全力支援，總是給我家人的溫暖，讓我可以不需擔心許許多多的雜事，專心致志的完成學業。在完成這篇論文之後，家人們求學時代的結晶總算是可以一起放在家中的書櫃上了。也謝謝實驗室夥伴，耿德、育任及向德，不論何時何事皆可以輕鬆地討論與交換意見，在感到困惑時也能互相打氣。謝謝學弟們，兆祥、台盛、經國、柏淵在幫助我們幾個學長收集駕駛資料或是標記 ground truth 時的貢獻。另外要提到的是分散在各個實驗室或是其他學校的同儕，在感到疲倦的時候總是因為你們一些可能較沒營養的話給逗得哈哈大笑，讓我總是能夠保持著一個愉快的心來寫程式、做實驗。雖然大家畢業的時間可能不太一樣，也不見得都在同一所學校，也或許有些人已經出社會工作了，但希望以後同學們還是可以常常聚會、聊天。也希望我的朋友們都能夠完成學業！最後，雖然完成了這篇論文，但我還是要再次感謝所有人，不論是有提到或是沒提到的人，因為沒有碰過這麼多形形色色的人是不會有今天的我。在這個崎嶇的路上，如果沒有其他人的幫助，我想自己一個人是很難撐得下去的。謝謝大家！

目錄

摘要.....i

Abstract.....ii

誌謝.....iii

目錄.....iv

圖例.....vi

表格.....vii

第一章　簡介.....1

1.1　研究目的.....1

1.2　論文架構.....3

第二章 TORCS 介紹與文獻縱覽.....4

2.1 TORCS 介紹與基本架構.....4

2.2 相關文獻縱覽.....6

第三章　實驗方法.....11

3.1 實驗中選用之 TORCS AI 介紹.....11

3.2 學習演算法：Q-learning.....12

3.3 使用的測試賽道.....14

3.4 超車行為學習.....16

3.4.1 基礎超車行為學習.....16

3.4.2 前車具阻擋行為時的超車行為學習與轉移實驗.....17

3.4.3 不同曲率半徑賽道上的超車行為學習與轉移實驗.....17

3.4.4 TORCS 內建賽道的超車行為實驗.....18

3.5 阻擋行為學習.....19

3.5.1 基礎阻擋行為學習.....19

3.5.2 阻擋行為學習之轉移實驗.....20

3.5.3 學習阻擋與超車行為後在不同曲率半徑賽道與 TORCS 內建賽道上之學習與實驗 21
第四章　實驗結果 22
4.1 格式說明 22
4.2 超車學習的成功率與時間統計 23
4.3 不同彎道曲率半徑時的超車行為表現與學習結果合併實驗 24
4.4 TORCS 內建賽道上的超車實驗 26
4.5 以內建 AI 為對手學習阻擋行為時超車成功率與時間 28
4.5.1 以 40 秒為判斷成功與否時的表現 29
4.5.2 對不同 AI 對手的阻擋學習結果的轉移實驗 32
4.6 不同彎道曲率半徑軌道與 TORCS 內建賽道上的阻擋行為實驗 34
第五章 結論與未來展望 37
5.1 結論 37
5.2 未來展望 38
參考文獻 39

圖例

圖 2-1　TORCS 實際遊戲畫面 …… 4

圖 3-1　測試軌道 …… 14

圖 3-2　改變彎道弧度的測試賽道（彎道弧度 40、90、150） …… 15

圖 3-3　TORCS 內建賽道 …… 15

圖 3-4　超車行為學習平均回饋 …… 16

圖 3-5　阻擋行為學習平均回饋 …… 20

圖 4-1　以 Fully reactive blocker 為學習對象的超車轉移實驗時間分布 …… 24

圖 4-2　學習阻擋行為之超車時間分佈圖 …… 30

圖 4-3　60 秒學習阻擋行為之超車時間分佈圖 …… 31

圖 4-4　以 lliaw 為學習對象的阻擋行為轉移實驗超車時間分布 …… 33

表格

表 2-1　環境變數 5

表 2-2　AI 端指令變數 6

表 3-1　輸入變數範圍與區間分布 13

表 3-2　修改後輸入參數名稱與範圍 13

表 4-1　統計格式中使用到的參數名稱及說明 22

表 4-2　前車有阻擋行為下的超車轉移實驗 23

表 4-3-1　TORCS 內建 berniw 與對手為 Fully reactive 之統計 25

表 4-3-2　TORCS 內建 berniw 與對手為 Slowly reactive 之統計 25

表 4-4　學習超車行為後對手為 Fully reactive 之統計 25

表 4-5　學習超車行為後對手為 Slowly reactive 之統計 26

表 4-6　TORCS 內建 Berniw 的各項參數統計結果 27

表 4-7　CG track 3 上的各項參數統計結果 27

表 4-8　Ruudskogen 上的各項參數統計結果 27

表 4-9　組合多個學習結果的各項參數統計結果 28

表 4-10 學習時間 40 秒下的超車成功率 29

表 4-11 學習時間 60 秒下的超車成功率 30

表 4-12 阻擋行為轉移實驗中 AI 超車成功率 32

表 4-13 在不同曲率半徑上以 lliaw 為學習對象的阻擋行為實驗 34

表 4-14 在不同彎道弧度上以 Q-learning AI 為學習對象的阻擋行為實驗 34

表 4-15 CG track 3 上面對 lliaw 的的阻擋行為統計結果 35

表 4-16 Ruudskogen 上面對 lliaw 的的阻擋行為統計結果 35

表 4-17 CG track 3 上面對 Q-learning overtaker 的阻擋行為統計結果 35

表 4-18 Ruudskogen 上面對 Q-learning overtaker 的阻擋行為統計結果 36

第一章 簡介

1.1 研究目的

在這個電子娛樂越來越發達的時代裡，不論是掌上型娛樂器或是大型機台直到電腦，都有大量的電子遊戲產品。而近年來的智慧型手機與平板電腦的崛起，也讓觸控操作成為了目前開發遊戲時多數開發者會考慮的操作模式之一。而在這些遊戲底下，又可以細分成許多不同的遊戲類型，包括角色扮演類、競技類等等。每種遊戲都有一定的支持者，想玩哪些遊戲端看玩家本身的喜好而定。

而在這篇論文中，成為主軸的遊戲類型即是競技類遊戲。這類型的遊戲在開始時就會讓玩家擁有一個屬於自己的角色，可能還可以跟不同玩家組隊，並在一個由遊戲設計者事先訂定好的規則底下，以自己的操縱技巧或戰術應用與對手競爭並想辦法取得勝利。這類型遊戲中，有許多是以體育運動項目為主，甚至還有體感操控方式可以幫助玩家更容易融入遊戲。而在競技遊戲中，最重要的一個要素就是對手。相信沒有人在比賽的時候是在一個沒有對手的環境下孤單地進行一項賽事的，若沒有對手，就不構成比賽的要件。對手的來源，在現今的遊戲中有很多都是經由網路連線將世界各地的玩家當作對手，可以同時與許多玩家一同競爭，並依據自己的實力在玩家排行榜上佔有一席之地也是一件很令人開心的事。

雖然可以跟各地的玩家一同進行遊戲是一件很開心的事情，但若是一開始信心不足，想先多做練習，卻又覺得只有自己一個人練習很無趣時，最好的幫手就是以遊戲中常常會有的電腦 AI 當作對手。這些由電腦 AI 控制的角色，我們統稱為 NPC（Non-player character），因為是由電腦控制，所以在行為上會偏向死板，所以若是玩家可以仔細觀察 NPC 的動作，並想出應對方式，就容易在對 NPC 的比賽中取得勝利。

電腦 AI 的強弱程度和遊戲的類型有關，如果一個遊戲規則較多且有一個最佳演算法的情況下，電腦方是具有相當優勢的。但若遊戲中規則較不嚴謹，則人

類玩家可能在不違反遊戲規則的情況下做出一個較複雜的動作，而這些動作如果當初設計電腦 AI 的程式設計者沒有想到，這就會成為 AI 的死角。為了能夠解決這種問題，所衍生出來的研究就叫做「Human imitation」，通常會使用到演化計算與學習演算法等等的學問，目的就是為了能讓電腦 AI 持續的學習到人類玩家或是比自己更強的電腦 AI 的習性與對應動作。

這篇論文的研究目的是想要研究如何改進賽車遊戲中的超車與阻擋的技巧。在這裡我們使用的遊戲平台是 TORCS（The Open Racing Car Simulator）。這個遊戲平台是一款開放原始碼的擬真賽車遊戲，遊戲當中很多賽車的參數都是參考真實世界中的情況來設定，連賽道也有很多是實際存在的比賽軌道。而 TORCS 也被很多研討會拿來當作指定的競賽平台，而這裡的競賽基本上是以 AI 與 AI 之間的比賽為主。在 TORCS 出現之前，很多相關研究的設定，不論是賽車參數或是賽道，都是由作者自己定義，而沒有一個統一的標準。所以研究成果之間不容易比較或實作。因為 TORCS 的出現，讓在這個領域中有研究熱忱的人擁有了一個可以互相比較的平台。而在 TORCS 底下廣大的研究範圍中，我們選擇了超車與阻擋行為作為這篇論文的研究主題，目的就是為了能夠讓賽車 AI 能夠有更像人類的駕駛行為。過去許多比賽中的 AI 即使有超車的行為，卻不會像現實比賽中，車手為了保持自己有利的地位而去阻擋後方來車的路徑。一般電腦 AI 都是以跑單圈賽道的時間表現作為評鑑的標準，但在現實的比賽中，若能保持自己有利的地位不受別人威脅，時常會出現一些更具侵略性的策略。所以我們選擇了研究如何產生更具侵略性的超車與阻擋行為作為主題。

而研究超車行為的論文雖然不多，但在本篇論文中要拿來比較的是“Learning to Overtake in TORCS Using Simple Reinforcement Learning,” 這篇由 D. Loiacono, A. Prete, P. L. Lanzi 及 L. Cardamone 等人所完成的論文。我們這裡的研究與其不同的地方是，在這篇參考論文中拿來作為被超車者的車輛皆是以一定速度開在直線跑道上且不具阻擋行為，研究主軸也只以研究後車的超車行為為主。

而我們的研究中，除了先以與參考文獻類似的方式進行超車行為的學習外，

也將學習演算法使用在駕駛 AI 的阻擋行為學習上。更進一步改變賽道的環境，包括不同的彎道曲率半徑及賽道長度，並利用這些具有變化的賽道讓駕駛 AI 去學習超車與阻擋行為，而不再只是單純學習在直線上的超車行為。最後利用 TORCS 內建的賽道進行學習與測試，希望可以看到學習過後的超車與阻擋行為是如何在實際賽道上有所發揮。

1.2 論文架構

本篇的第二章節首先介紹 TORCS 的環境架構，接著敘述與 Human imitation 及超車行為學習的相關文獻，第三章開始則是本篇論文進行實驗時所使用的各種方法與實驗時的環境條件介紹，第四章為統計結果的展示，第五章是總結本篇的結論以及未來展望。

第二章 TORCS 介紹與文獻縱覽

2.1 TORCS 介紹與基本架構

圖 2-1：TORCS 實際遊戲畫面

TORCS 這個模擬平台算是一個發展了一段時間的賽車模擬平台[1]，圖 2-1 為實際進行遊戲時的 screen shot。內部很多參數與賽車的行為模式都是仿照真實世界的行為所設計的，所以有很多與人工智慧相關的研討會採用 TORCS 這個平台作為一個可以讓發行論文者互相比較的標準，如 CIG（IEEE Conference on Computational Intelligence and Games）、CEC（IEEE Congress on Evolutionary Computation）等等的大型國際研討會中都有以 TORCS 為人工智慧行為模擬基礎的論文出現。在程式架構部分不管是競賽版或是一般版，基本上都是分成兩個部

分：駕駛 AI 的部分可以視為是競賽版中的 client 端，而遊戲執行引擎部分則為 server 端。遊戲進行中，遊戲引擎每 0.02 秒（遊戲中的一個 game tic）會將賽道與賽車的各個狀態與資料傳給駕駛 AI 並等待 0.01 秒，而 AI 會依據目前狀態決定加速、剎車與轉向等指令並回傳給 server 端。若駕駛 AI 沒有在 0.01 秒內將指令回傳，則會執行上次回傳的動作。表 2-1 和表 2-2 為遊戲中會使用到的變數：

表 2-1：環境變數

變數名稱	數值範圍(單位)	變數說明
Angle	[-π,+π] (rad)	車子前進方向與軌道方向的夾角。
curLapTime	[0, +∞) (sec)	在目前這一圈所花費的時間。
damage	[0, +∞) (point)	車子受到的損傷值。
distFromStart	[0, +∞) (m)	車子與起跑線的距離。
distRaced	[0,+∞) (m)	遊戲開始後所跑的總距離。
Focus	[0, 200] (m)	駕駛 AI 可操作的距離偵測器，共有 5 個，每個偵測器的角度皆間隔 1 度。用來測量車子在特定方向上與邊界的距離。
Fuel	[0, +∞) (1)	剩餘的汽油量。
Gear	{-1, 0, 1, …, 7}	排檔值，-1 為倒退檔，0 為空檔。
lastLapTime	[0, +∞) (s)	跑上一圈賽道時所花掉的時間。
opponents	[0, 200] (m)	共 36 個對手偵測器，分布在車子周圍間隔 π/18 的角度，偵測距離為 200 公尺。偵測器會傳回偵測範圍之中對手的距離。
racePos	{1, 2, …, N}	目前比賽中的排名，N 為車輛總數。
Rpm	[2000,7000] (rpm)	引擎轉速。
speedX	(-∞, +∞) (km/h)	賽車車頭方向上的速度。
speedY	(-∞, +∞) (km/h)	賽車車頭垂直方向上的速度。
speedZ	(-∞, +∞) (km/h)	賽車在 Z（高度）軸上的速度。
Track	[0, 200] (m)	包含 19 個距離偵測器，從車子前方的正右方(-π/2)到正左方(π/2)的範圍之內。
trackPos	(-∞, +∞)	表示車子和賽道中心線的距離，已經用賽道的寬度做正規化，-1 表示賽道右側邊界，+1 表示賽道左側邊界，大於+1 或是小於-1 的值代表車子超出邊界。
wheelSpinVeil	[0, +∞] (rad/s)	車輪轉動速度。
Z	[-∞, +∞] (m)	車和賽道表面的距離。

表 2-2：AI 端指令變數

變數名稱	數值範圍	變數說明
Accel	[0,1]	油門控制，0 為不踩油門，1 為全速。
Brake	[0,1]	剎車控制，0 為不踩剎車，1 為全煞。
Clutch	[0,1]	離合器控制，0 為無動作，1 為最大。
Gear	{-1, 0, 1, …, 7}	排檔控制。
Steering	[-1,1]	轉向控制，-1 為最大右轉，1 為最大左轉。
Focus	[-90, 90]	指定 focus 偵測器方向。

2.2 相關文獻縱覽

在遊戲的應用上，有時候如果只用固定的程式碼來建構 AI 或是判斷玩家行為，則可能效果不好且速度較慢。這時候機器學習就是一個重要的關鍵，利用這些學習演算法的自我學習能力，對開發者有很大的幫助。而在[2]中提到如 Xbox 的 Kinect，起初研發團隊想使用純圖學的方式，但後來發現這樣的處理速度太慢，於是導入機器學習的方法根據人體由某個動作移至接下來的動作的機率模型數據輔助辨識，又讓程式從上百萬張人體圖片中學習、辨識姿勢然後提出以抓身體驅幹及四肢為主要辨識目標，最後再以 3D camera 得到身體驅幹深度資訊幫助辨識。

而在 TORCS 中，實作一個具有競爭能力的駕駛 AI 的方法有很多種。在[3]中所用的是 Fuzzy logic，也就是模糊邏輯。模糊邏輯中具有程度的概念，而不像傳統邏輯中針對某一二元邏輯其輸出只有 0 及 1 兩種可能的值，模糊邏輯可以針對目前情況作出程度上的判斷。[3]先手動對各個輸入及輸出參數設計其成員函數，再利用演化計算去針對這些成員函數做演化。經過演化後，由其成員函數的演化可以看出駕駛 AI 對本身賽車位置變得更加敏感。[4]提到目前要做出一個類似人類駕駛員的駕駛 AI 有三個較主流的方式：監督式學習（Supervised Learning）、

類神經演化計算（Neuroevolution）以及規則演化計算（Rule-based evolution）。而[4]正是使用第一種方法來完成這個駕駛 AI。在這個部分共有兩種偵測賽道環境的方法：Range-finder sensor 及 Look-ahead sensor。第一種是 TORCS 比賽中使用的距離探測器，會回傳車子前方各角度距離賽道邊界的距離。第二種是取出車子前方固定長度的賽道，分割成多個等長的 segment 後計算每個 segment 的彎曲程度，0 為直線，而若是大於 0 的話代表右轉、小於 0 則代表左轉。經由上述兩種 sensor 取得了賽道的參數後，使用 K-nearest neighbor 及類神經網路來決定現在要採取什麼樣的行動。

而在我們的論文中，先經由學習演算法在訓練過程中找出在當下狀態裡的最佳行為並記錄下來，在結束學習過程後再找出過去對最佳行為也類似[4]中提到的方式。[5]結合了線性回歸分析與人工類神經網路來實做一個賽車 AI，而實驗結果顯示出這個作法可以贏過大多數現有的 AI。在[5]之前，同一批作者完成了一個以安全駕駛行為為主的駕駛 AI，以記憶錯誤發生地點與錯誤時的行為模式還有在賽道上調整速度的三段區域為主題。一開始，先找出數個學習用的軌道並紀錄在這些軌道上的 sensor-input，再對這些資料去取出有用的特徵，最後進入線性回歸分析利用當下環境參數來決定目標速度與類神經網路的學習階段。轉向控制部份，與大多數論文所使用的一樣以車頭前方的 track sensor 來當作判斷依據。而失誤處理部份則是紀錄 AI 在賽道的哪個地方發生錯誤，在錯誤點前後一段距離的區域中直接減速以避免衝出賽道或碰撞。

而[6]是採用 Spiking Neural Network（SNN）來發展一個駕駛 AI。SNN 是第三代的類神經網路，其開發理念是希望可以讓這個類神經網路更加接近生物大腦的運作方式，通常是用在分類問題或是機器人視覺分析上。在[6]中是由 Izhikevich 模組的神經元所組成的前饋網路來控制賽車 AI 的行為，這個網路中包含了八個輸入神經元與四個輸出神經元，且輸入與輸出神經元之間是直接對應，故架構中不存在隱藏層。類神經網路的輸入環境參數是賽車方向與軌道方向的夾角、賽車與軌道中線距離、賽車速度以及五個量測軌道邊界與賽車距離的 range-finder

sensor。而輸出部分，只簡單分為左右轉向與加速及剎車共四個。而在這篇論文中除了一般的健康度測試外，另外設了一個限制：當一個控制 AI 在計算過程中受到的損傷超過 1000，則馬上捨棄這個 AI 並開始另一個 AI 的計算。

賽車遊戲有很多種，所以針對是否可以將駕駛行為在不同遊戲間進行轉移以加速在另外一個遊戲中的學習效果也變成一個值得研究的課題。[7]即是有關轉移學習的一篇論文，是在 TORCS 與另一個賽車遊戲 VDrift 間進行實驗，兩個遊戲的差異是 TORCS 每 0.02 秒、VDrift 是每 0.01 秒更新一次，且 VDrift 在每個賽道上平均表現約比 TORCS 高出 32%。而在這篇論文的實驗中是以類神經網路學習的結果為轉移對象。首先在兩個遊戲間用兩邊現有的駕駛 AI 及兩款遊戲中的四個賽道上比較駕駛時間。比較完兩邊的時間表現後，接著是以 TORCS 為學習平台，測試在此平台上以類神經網路學習的結果是否可以不需調整即可應用在其他遊戲上。若使用在 TORCS 上的學習結果並進行學習，則其類神經網路的健康度在四個測試賽道上較原始駕駛 AI 成長了 14%～225%。而若不使用 TORCS 上的學習效果而直接開始學習，需要 37 個世代才能找到效果最好的駕駛 AI。若有使用，則只需要 4 個世代就能找到具有競爭力的 AI，所以經過實驗，轉移實驗的確是可以加速學習成果。

除了以 sensor 作為輸入來計算出當下車子的行動外，還有另外一種研究方向是事前便將關於賽道的資訊都蒐集完成後，利用這些資訊來計算出賽車如何在這個賽道上開得越快越好與近乎最短距離的路徑。[8]就是這個研究方向的其中一篇論文。在[8]中以基因演算法來找出最具競爭力的賽車路線，並將這個路線以一組 B'ezier curves 表示。在對控制點進行編碼時，當賽道某段的曲率半徑越大，則該區域的控制點就會越多。接下來的計算階段中，用了兩種計算方式。第一種是將計算出來的軌跡直接載入到 TORCS 程式中去模擬實際跑的時間與速度等等。而第二種是用估計的方式，計算這次軌跡跑出來的健康函數並以這個值來比較不同軌跡間的效果。最後，在模擬實驗的部份所呈現出來的結果表示，如果對控制點的編碼可以更加密集，效果應該能獲得提昇。而在估計實驗結果中，[8]所用

來計算健康函數的計算模組可能沒有辦法完全符合想解決的問題。

超車與阻擋行為對 AI 來說較不直觀，因為人類駕駛可能會想要保持自己優勢的地位，而寧可放棄本身的最佳路徑也想擋住後方來車的路徑，所以在這麼多相關研究中，即使有超車行為的研究，也鮮少有阻擋行為的論文出現。尤其 TORCS 本身內建的 AI 中甚至沒有所謂阻擋的行為，只會讓出自己的路線使後方速度快的車輛可以超越。與這篇論文的方向最為相關的論文為[9]與[10]這兩篇。其中[9]是以強化學習演算法（Q-Learning）來進行超車行為的模擬，在[9]中將超車行為分成了兩個部分，分別是軌跡與剎車延遲。軌跡部分主要是控制車子在軌道上的位置，而剎車延遲則是控制車輛準備進入彎道前剎車的時間。主要目標是延後剎車的時間，讓超車者可以利用與前車的時間差與速度完成超車。[9]在進行學習前，先將賽車參數統合成 4 個主要用到的變數：Dist y 為兩車在賽道軸上的距離，Dist x 則為兩車間水平距離，Pos 為超車者自己離軌道中心線的距離，ΔSpeed 則為兩車的速度差值。四個不同的輸入參數與區間共可以產生 4000 多個 state，而每個 state 中都會依據學習結果更新底下對應動作的 Q-value。前面提到在[9]中，有分成兩個部分的實驗，分別是軌跡與剎車延遲。在軌跡實驗中，可以將車子的行為視為在某一個狀態下往左移動一單位、往右移動一單位或是保持原本行進路線。利用改變超車者目前的軌跡，進而完成更具侵略性的超車行為。而剎車延遲是在快要進入彎道時，若能將剎車時間稍微延後，則就很有可能利用這一小段時間的速度差在進入彎道前順利超車。這部分用到的輸入參數則是上面表 2-3 所寫 Dist y 與 ΔSpeed 及新加入的 dist turn，代表要進入到下一個彎道前的距離。[9]的實驗中尚有考慮到不同空氣動力模式下的情形，不過在本篇論文中只考慮在標準空氣動力中的學習。而在學習演算法的回饋部分，[9]中的回饋系統設計如下：

$$r_t = \begin{cases} 1，超車成功 \\ -1，超車者撞毀 \\ 0，\quad 其他 \end{cases} \qquad (1)$$

在[9]的兩部分的實驗中的每個細項皆跑 10000 次的學習，而當學習次數約進行 5000 次之後，回饋系統所回傳的值的平均會趨於穩定，也就是學習的行為會趨於穩定。軌跡學習部分，Q-Learning 的結果以成功率來看，皆可以以 90%的成功機率壓倒 TORCS 內建的 Berniw Driver 的 70%。在車子的最高速度數值上也是約超出內建 AI 約 7～8 Km/h。剎車延遲部分，成功機率更是以 88.1%遠遠領先內建 AI 的 18.8%。

[10]是以模糊邏輯來對前車阻擋後車的行為作出行為模式的分類並以這些分類結果來判斷現在所要採取的最好行為為何並進行超車。模糊理論是一種可以將目前情況以程度來進行分辨的邏輯的一種。與只有 1 跟 0 兩種可能的經典邏輯不同，使用到模糊邏輯決定一個參數時需要使用到成員函數（Membership Function），成員函數底下可能針對各種不同情況而由多個梯形或三角形函數組成。在[8]中除了針對後車進行超車行為的判斷外也對前車，也就是阻擋者設計了三種不同的模式，分別是受限制、延遲反應與即時反應。不同之處在於受限制者的可使用軌道寬度是受到限制的，這個 AI 不能開到離軌道邊界太近的地方。而延遲反應者是接收到後車前一秒的狀態來進行阻擋動作。即時反應的 AI 則不受上面兩者的限制，可以隨時針對超車者目前的動作進行阻擋。不過[10]依然是以超車行為的學習為目標，阻擋行為是 hard coded。在結果部分，受限制者的阻擋結果除了[10]所提出的模糊理論超車系統外，對 TORCS 內建的其他 AI 來說，都無法完成超車。這個結果是可以預期的，因為內建的 AI 在超車行為上都不夠侵略，不會為了要進行超車而讓自己有碰到或衝出軌道邊界的可能。所以當阻擋者稍微貼近賽道某側的邊線時，內建的 AI 會取消想超車的行為。而在延遲反應部分，較聰明的 TORCS 內建 AI 則可以超車成功，其中以 Berniw、Lliaw 及 Tita 以超過 90%的超車成功機率為最高。也因為這個結果，讓本篇論文後面針對阻擋行為的學習時也主要以 Lliaw 及 Tita 做為學習對象。而我們最主要使用的到阻擋者策略也是採用[10]中所提到的 Slowly reactive 及 Fully reactive blocker。

第三章　實驗方法

在這一章裡面，我們會先以使用學習演算法的駕駛 AI 去跟一個不考慮後方車輛路徑的駕駛 AI 進行學習以得到一個最基本的超車行為模式。接著再陸續將前方阻擋車輛導入[10]中所提到的兩種阻擋行為模式，並在各種不同的測試軌道及 TORCS 中內建的賽道上進行學習與實驗。最後再利用 TORCS 內建的 AI 與有經過學習演算法改進過的 AI 來幫助進行阻擋行為的學習。

3.1 實驗中選用之 TORCS AI 介紹

在將學習演算法導入負責超車的 AI 之前，我們需要先產生出一個只會最簡單的行為模式的賽車 AI 作為前方被超車的對象。而在這個部分是選用了 TORCS 中內建的 BT driver。這個賽車 AI 的策略是以開在軌道中間為主，而我們根據[9]將此 AI 的最高速度控制在 140 km/h～150 km/h 之間。若在較複雜的賽道上，內建 AI 可能會因為本身駕駛行為的關係造成名次間的變動，但若是在簡單的賽道上，則 AI 的速度會接近一致，使得賽車間的排名不會有所變化，所以我們需要限制阻擋者的最高速度。最高速度會因直線的長度與曲率半徑的大小而有些許變動。另外，我們將此 AI 調整為不會針對對手車輛做出反應，而是單純只開在自己設定的路徑上。若不作此調整，則當後方車輛速度高於自己時，TORCS 中內建的 AI 都會傾向將路徑讓給後方速度較快的車輛通過，如此一來便完全無法針對超車行為進行學習。

而負責超車的 AI 部分，我們為了能與[9]作比較，所以選用了與[9]相同的駕駛 AI：berniw，而超車者的速度我們並不作特別的調整，保持其原有的速度，最高速度約在 290 km/h 附近。當後方車輛在接近前方車輛時，TORCS 內建的 AI 會自動將速度降下來並跟在前車後方等待超車機會，等到 AI 判斷有足夠寬敞的空間可以進行超車時才會提昇速度。

一開始的超車行為我們也模仿[9]製作出了具有長直線且彎道長度短的賽道供學習使用，而學習時只在測試賽道上的直線段中進行，這個部分只要調整兩車在彎道時的速度相同即可。因為測試軌道難度不高，且除了兩側彎道部份外皆為直線。故當偵測到當下軌道區塊具有曲率半徑時，將兩車的加速指令與速度上限統一便可簡單做到。在阻擋行為的學習中，如同超車實驗，考慮到使用學習演算法時基礎駕駛 AI 的一致性，我們負責阻擋的駕駛 AI 的也是選擇 berniw，而超車者的部份我們根據[10]，使用了 lliaw、tita 以及 bt 以利比較。

3.2 學習演算法：Q-learning

在這篇論文中的學習演算法使用的是 Q-Learning 演算法。Q-Learning 演算法是一種增強式學習演算法，在學習過程中不需借助外在所給予的指令即可以自行找出在當下情況中最適合的行為。也因為我們並不一定可以知道在某個狀況下應該要採取何種行為才是最佳解，所以選擇了這個有能力自己找出解答的演算法來使用。

這個演算法用一個函數 Q(s,a)來代表在狀態 s 時進行動作 a 的質量（Quality）。當 AI 每次選擇一個動作時，回傳的回饋值與到達的新狀態會與上一個狀態與前一次動作相關。這個演算法的核心概念是迭代更新對應狀態與動作的值。式（2）為演算法公式：

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t(s_t, a_t) \times \left[R_{t+1} + \gamma \times \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t)\right] \quad (2)$$

等式左邊為更新後的 Q-table value，等式右邊第一項為更新前的 Q-table value，第二項為學習率（Learning Rate），此處將其設為 0.5。大括弧中的第一項為採取某個行動時所得到的報酬（Reward），而 γ 為 Discount Factor，主要用處是調整

前後 Reward 的重要性，此處將其設為 0.8。而第三項是下一個狀態時，Q-table 中的最大值。大括弧中所有項目和即為此次學習的學習值（Learning Value）。

導入 Q-learning 演算法時，[9]中將變數處理成表 3-1 的形式，Q-learning 中狀態即是以兩張表中 4 個變數的各個區間組合而成，每個變數的區間個數分別為 6、10、8 及 9 個，經組合後共有 4320 個狀態。而我們希望可以讓狀態數分切的更細微，於是將變數修改成表 3-2。由下面的兩張表可以看出我們將數值範圍縮小，但並不改變區間數量。

表 3-1：輸入變數範圍與區間分布

變數名稱	數值範圍	區間
Dist y	[0,200]	[0 10), [10 20), [20 30), [30 50), [50 100), [100 200)
Dist x	[-25,25]	[-25 -15), [-15 -5), [-5 -3), [-3 -1), [-1 0), [0 1), [1 3), [3 5), [5 15), [15 25)
Pos	[-10,10]	[-10 -5), [-5 -2), [-2 -1), [-1 0), [0 1), [1 2), [2 5), [5 10)
ΔSpeed	[-300,300]	[-300 0), [0 30), [30 60), [60 90), [90 120), [120 150), [150 200), [200 250), [250 300)

表 3-2：修改後輸入參數名稱與範圍

變數名稱	數值範圍	區間
Dist y	[0,30]	[0 5), [5 10), [10 15), [15 20), [20 25), [25 30)
Dist x	[-15,15]	[-15 -10), [-10 -5), [-5 -3), [-3 -1), [-1 0), [0 1), [1 3), [3 5), [5 10), [10 15)
Pos	[-8,8]	[-8 -3), [-3 -2), [-2 -1), [-1 0), [0 1), [1 2), [2 3), [3 8)
ΔSpeed	[0,250]	[0 10), [10 20), [20 30), [30 50), [50 70), [70 100), [100 150), [150 200), [200 250)

兩個表格當中 Dist y 為兩車間垂直距離，Dist x 為兩車間水平距離、Pos 為本身離軌道中心線的距離，最後 ΔSpeed 為兩車間速度差。而當作基礎駕駛 AI 的部

分則是按照[9]選用 Berniw 來當作修改的基礎。在每個狀態底下共有三個不同的可能行為，分別為往左移動一單位距離、往右移動一單位距離與維持原本方向。這三種動作都有自己的 Q-Value，而在學習初期是以隨機方式決定當下採取的行動並依這次行動的結果來更新 Q-value table 中的值，當 Q-Value 達到一定門檻時才會採用該狀態底下擁有最大 Q 值的行為。回傳 Reward 的設定也參考[9]。而在兩側超出賽道部分會有 3 公尺草地與牆壁。所以兩車水平距離及學習者本身位置會稍大於賽道寬度

3.3 使用的測試賽道

首先在開始實驗之前我們需要一個具有足夠長度直線的軌道，在 TORCS 中雖然有很多軌道且當中不乏具長直線者，但這些軌道通常較複雜且總距離過長。所以我們利用 TORCS 安裝版中所附的軌道編輯器自製了一個軌道，如圖 3-1。此測試軌道總長度為 4408 公尺，且上下直線部份各為 2000 公尺，軌道寬度為 10 公尺。

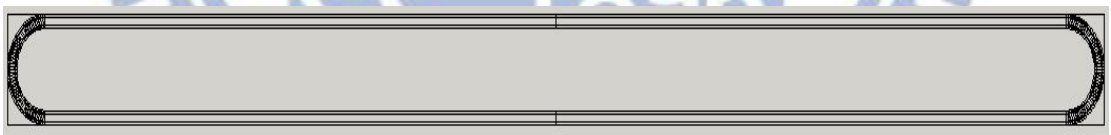


圖 3-1：測試軌道

在後面的實驗中我們也另外設計了與圖 3-1 類似的賽道，但直線長度部份縮短為 500 公尺，並改變其彎道曲率半徑，希望可以得到在彎道上超車的學習機會，如圖 3-2。圖中的賽道寬度皆為 10 公尺，而我們在後續的實驗中尚有使用到 15 公尺寬度的賽道。

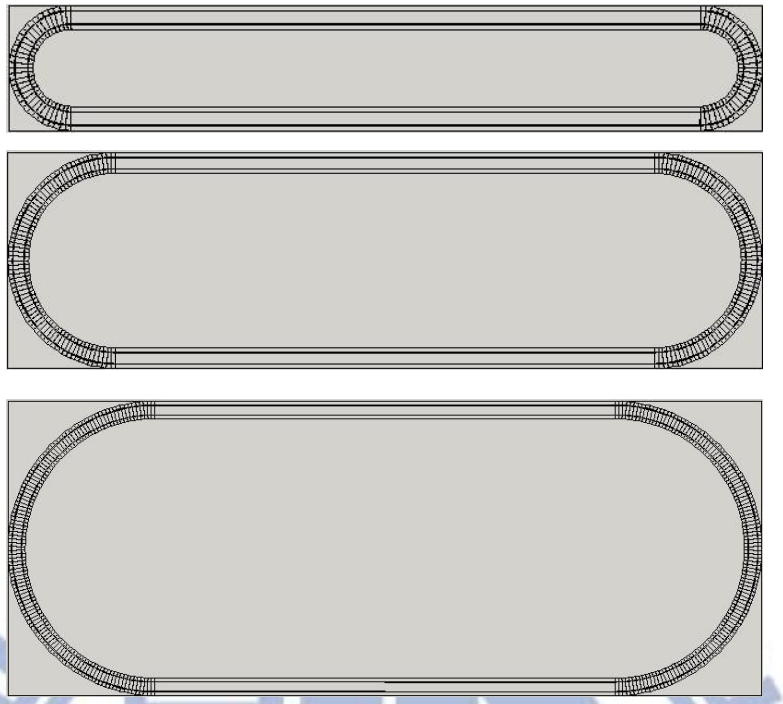
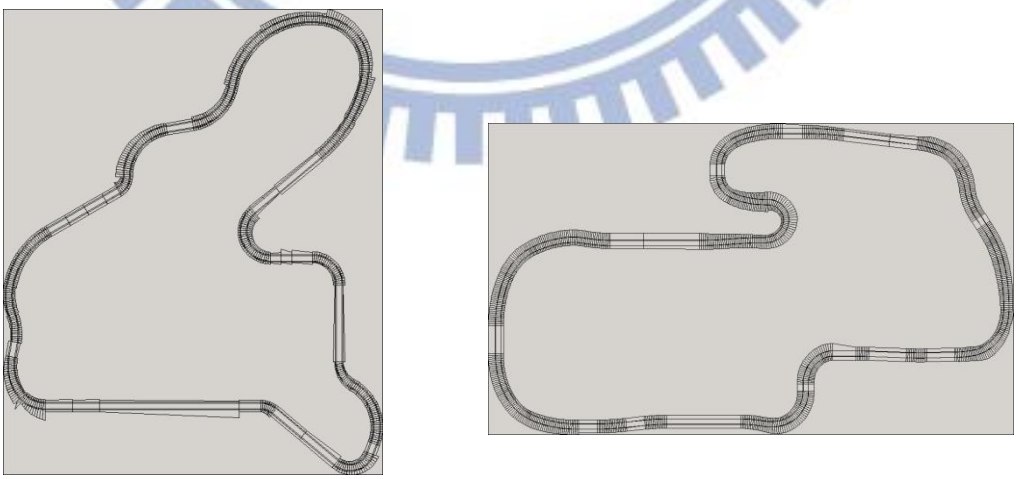


圖 3-2：改變彎道弧度的測試賽道（曲率半徑 40m、90m、150m）

最後的實驗中我們用到了 TORCS 中內建的兩個賽道，我們選擇 TORCS 中兩個道路寬度在 10 公尺附近、難度中等的賽道來測試之前學習結果是否可以應用到實際賽道。圖 3-3 為所選出的兩個測試賽道。圖中(a)為 CG track 3，道路寬度 10 公尺、總長度為 2843.1 公尺，(b)為 Ruudskogen，寬度為 11 公尺、總長度為 3274.2 公尺。



(a) (b)

圖 3-3　TORCS 內建賽道(a) CG track 3，(b) Ruudskogen

3.4 超車行為學習

3.4.1 基礎超車行為學習

我們的第一個實驗使用只開在自己預設路徑上的駕駛 AI 為被超車者，及導入 Q-learning 演算法的駕駛 AI 為超車者，將這兩個駕駛 AI 放到測試軌道上（圖 3-1）進行學習。在每次跑 35 圈中，約可以進行 15 次的學習。在跑過共 400 次之後就約有 6000 次的學習機會。比對[9]中，當學習次數約達到 5000 次之後，Q-Learning 演算法中平均回饋的值會趨於穩定。圖 3-4 為學習時所得到的平均回饋圖，縱軸為平均回饋值，橫軸為學習次數。學習過程中，每 10 個 game tic(0.2 秒)選擇一次新的動作。

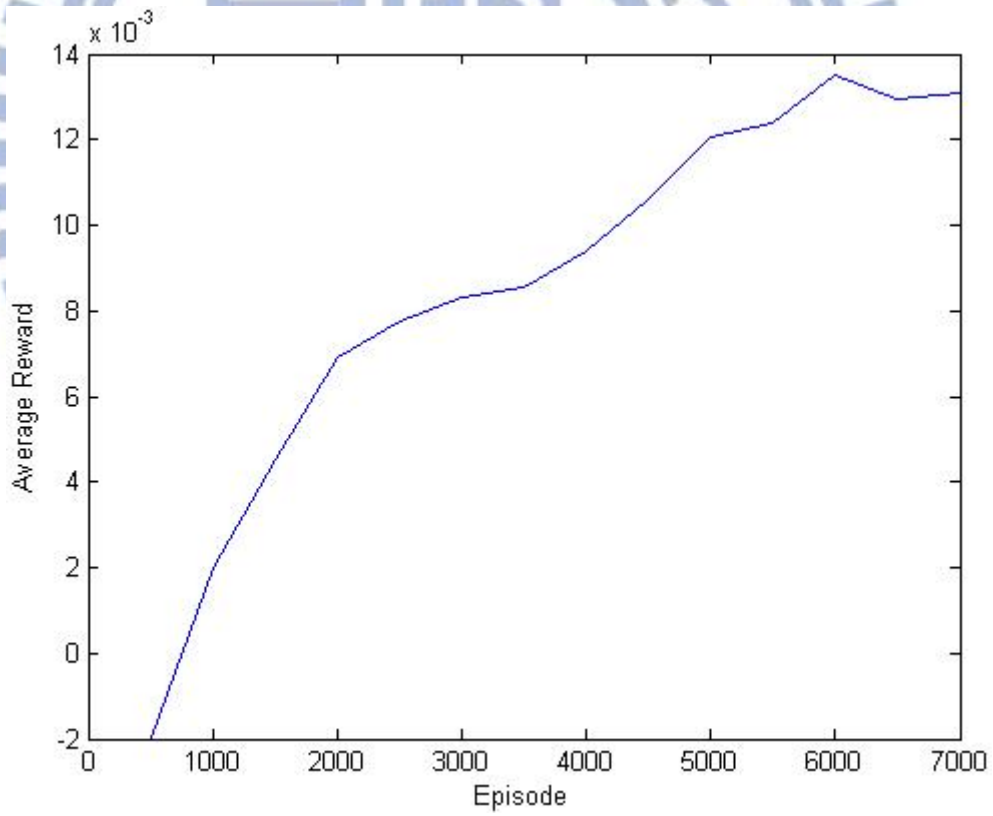


圖 3-4：超車行為學習平均回饋

在這裡的實驗中，我們將超車的學習限制在賽道直線上，彎道中會將超車者的速度限制與被超車者相同。經過觀察，當後方的超車車輛開始減速後約 40 秒

時，若尚未超車成功，則可能已經在測試軌道中進入入彎或出彎階段。所以我將40 秒設為一個界限，若超過 40 秒沒有超車成功，則此次超車行為視為失敗並回傳 Reward 為-1。後續實驗中也繼續使用此設定。

3.4.2 前車具阻擋行為時的超車行為學習與轉移實驗

一開始我們所得到的超車行為學習結果，其學習的對象為一個只開在軌道中線上且不會考慮後方來車狀態的駕駛 AI。而現在我參考[10]將此簡單的 AI 再進行一些修改，將其改成以下兩種版本：Slowly Reactive 及 Fully Reactive，前者是對後方車輛的位置在一秒後進行反應並往後方車輛位置偏移來阻擋，後者則為即時反應，不需等待一秒的反應時間便立即往後方車輛位置進行偏移。加上最原始的對手，也就是只開在軌道中線上的 AI，便有了三個不同的學習對象。

在這個階段進行轉移實驗的目的是為了要觀察超車者在面對具有不同阻擋行為的阻擋者時，其超車能力的變化。先對不同阻擋行為的阻擋者分別進行學習，得到學習結果後再跟其他阻擋者進行超車行為的測試，如：與 Fully reactive blocker 學習後，利用這個學習結果與 Slowly reactive blocker 及不具阻擋行為的 AI 進行測試，並觀察其超車成功率與時間分布的變化。

3.4.3 不同曲率半徑賽道上的超車行為學習與轉移實驗

在這個小節中，我們使用的是圖 3-2 的賽道。主要目的是改變彎道部分的曲率半徑，希望可以看看在不同曲率半徑的彎道中超車行為的改變。另外也準備了不同賽道寬度的版本，每個賽道皆有寬度為 10 公尺及 15 公尺兩種版本可以使用。並且在這個部分的訓練過程中，取消了在彎道上兩輛車速度必須相同的設定。因為是在不同彎道曲率半徑上的軌道中進行實驗，所以想看看在不同彎道曲率半徑時，超車行為會有怎樣的變化。若不取消此設定，則因為此處軌道直線長度較短，可能造成賽車出彎後無法重新將速度加速到最高導致超車次數大幅下降。而取消

兩車在彎道上速度必須相同的設定還有另外一個考量：希望可以達成每次在軌道不同的位置上開始進行超車行為。當超車者一圈一圈超過前車時，下次再碰到要超車的情形時，有極大的可能是從與上次超車行為不同的起點開始。

接下來在上面三個軌道上與兩種不同阻擋策略（Slowly reactive、Fully reactive）的駕駛 AI 進行學習，學習完的結果都會存檔以利後續實驗的進行。在接下來的實驗中，為了得知學習結果是否可在具不同彎道曲率半徑的軌道上進行學習行為的轉移，所以會利用不同學習對象與軌道的配對產生出多組結果。而在轉移部份，我們目前只針對相同軌道寬度的學習結果進行轉移實驗。

轉移實驗的下一步則是嘗試合併學習結果，例如以彎道曲率半徑為 40m 及 90m 軌道上所訓練出來的結果，實驗是否可以使用在彎道曲率半徑為 65m 的軌道上。這部分又另外做了兩個賽道，分別是曲率半徑為 65m 及 120m 的賽道以利實驗使用。曲率半徑 120m 的賽道是提供給在曲率半徑 90m 及 150m 兩個賽道中所學習出來的結果使用。我們統計在 1500 秒內超車者的各項參數來評比學習成果。

初步觀察的結果顯示，若軌道寬度為 15 公尺，則有超車學習者在一定時間內超車次數只會略大於無學習行為時的駕駛 AI。但當軌道寬度為 10 公尺時，則無學習超車行為之 AI 其超車成功次數皆在 0～2 次之間，而學習後平均皆可達到 10 次，且即使超車次數較多，超車者所受到的損傷值也不見得會比較高。從以上可以簡單的看出當軌道寬度較大時，超車行為學習成效較差。所以後續的阻擋實驗及 TORCS 內建賽道上的實驗中我們都以賽道寬度為 10 公尺的軌道為主。

3.4.4 TORCS 內建賽道的超車行為實驗

超車行為實驗的最後，我們在 TORCS 內建的賽道（圖 3-3）中進行實驗，此處我們只利用前面實驗中學習的結果來進行測試，不會再次學習。在這個階段因為這兩個賽道長度都較原本用來學習的賽道長，所以這個部份我們將原本 1500

秒的時間設定改為 2400 秒。我們使用在前面的幾次不同實驗中所獲得的學習檔案來進行這裡的實驗，包括在較窄（10 公尺）的軌道上依不同彎道曲率半徑（40m、65m、90m、120m、150m）所得到學習結果，其中原本曲率半徑為 65m 及 120m 的兩個檔案是由另外 3 個檔案做內插得出。

這個實驗當中我們嘗試以兩種不同的方法來使用之前的學習結果。第一種方式是將五種曲率半徑的結果各進行一次為時 2400 秒的實驗。另外一個實驗中則同時使用為原始訓練結果的三個檔案（曲率半徑 40m、90m、150m），在要進行超車行為時，選擇使用與目前所在的軌道位置曲率半徑最相近的學習檔案來決定要採取的動作。

3.5 阻擋行為學習

3.5.1 基礎阻擋行為學習

在 TORCS 中當後方車輛的速度大於前方車輛的時候，前方車輛會將自己的路徑讓出來，讓後方車輛可以順利超過前方車輛。內建 AI 沒有阻擋後方車輛的概念，所以在進行學習之前，我們需要先讓阻擋者知道自己是否已經被後方車輛所超越。若在某個學習階段，當採取某一動作時，可以讓自己在一定時間中沒有被後方車輛超越，則此次採取此行動的結果為成功。採取動作的間隔在這邊我們設為 15 個 game tic，若在這 15 個 game tic 中沒有被對手超越，則此次動作回傳為成功。

在進行這部份實驗時，除了一樣使用 berniw 作為阻擋者修改基礎，且此處的最高速度也限制在 140～150 km/h 間，也沿用表格 3-2 中的輸入參數範圍來將車子間的狀態區分成不同的 state 並對應三種動作，分別是：往左移動一單位、往右移動一單位與維持現在行進方向。這個實驗中主要先以 TORCS 中內建的駕駛 AI 作為超車者，依據不同的駕駛行為共挑出了 lliaw、tita、bt 三者來學習。其

中 lliaw 及 tita 這兩個駕駛 AI 其行為模式與 berniw 較相近，都是與前車有一小段距離時，會將其速度降低並跟在前車後方尋找超車機會。較不同的地方是對於複雜賽道時，兩者的路徑規畫會與 berniw 不同，但在此處因為賽道都較簡單，所以不易看出其差異。而 bt 就如同前面也介紹過，是以開在賽道中線上為主的 AI。在碰到要超車的情形時，會提前改變自己與賽道中線的偏移量。而測試賽道則是先使用圖 3-1 中的長直線賽道。

圖 3-5 為阻擋行為學習時的平均回饋值。圖 3-5 中可以看出其平均回饋值都較學習超車行為時低，這是由於阻擋行為時較容易發生撞毀的情況。而多次實驗中，約經過 3000 次學習階段後，回傳的回饋值會趨於穩定。

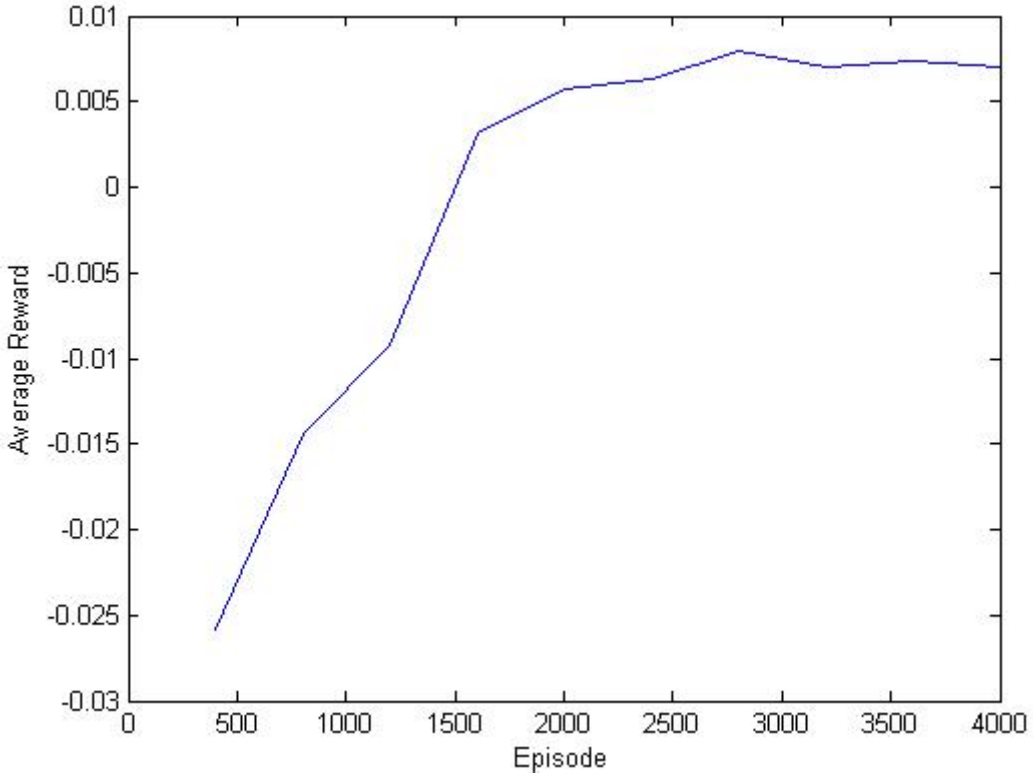


圖 3-5：阻擋行為學習平均回饋

3.5.2 阻擋行為學習之轉移實驗

我們進行阻擋行為的學習時，如同 3.5.1 節中所提到的，共使用了三個不同

的駕駛 AI 作為學習對象，所以在進行轉移實驗時，直接組合這三個不同的學習結果與對手 AI 來進行轉移實驗並統計其超車成功機率與時間分佈即可。而在這個部份因為都是使用學習時間限制為 40 秒所得到的學習結果，經由初步的觀察後發現，被超車時間容易集中在 40 秒後的區間中。推測可能是 40 秒過後容易進入彎道，導致後車具有更大的超車空間且學習時間不足以讓阻擋者學到這段時間裡所應採取的行為。所以我們另外針對這個現象，將學習時間拉長至 60 秒，並觀察其超車時間的分佈。

3.5.3 學習阻擋與超車行為後在不同曲率半徑賽道與 TORCS 內建賽道上之學習與實驗

在這節中，以重複 3.4.3 與 3.4.4 的實驗步驟為主，但在這個部分將學習演算法及多個擁有不同彎道曲率半徑的軌道導入阻擋行為的學習中。最後再放到 TORCS 內建的賽道上進行測試，測試條件等等都與前面兩節中的實驗相同。阻擋實驗中，作為對手的超車者，我們選定了在 3.5.1 節中學習後可以使阻擋者有較好表現且行為模式與 berniw（作為 Q-learning 基底的駕駛 AI）較相似的 lliaw 以及經過 Q-learning 學習後的超車 AI 作為學習對象。並且，為了比較學習後的阻擋成效，我們選擇了 Fully reactive blocker 作為最後結果的比較對象。

第四章　實驗結果

4.1 格式說明

首先，我們在這邊先說明後面會出現的結果格式的意義。在這邊一共會有兩種不同的結果表示方式，首先在以 TORCS 內建的 AI 為學習對象來發展阻擋行為時，是用統計 200 次阻擋行為的時間所得到的直方圖以及對應我們所設定的時間限制下的成功率，當一次超車或阻擋行為超過設定的 40 秒時則此次行為將被視為失敗。另外一種則是為了比較多個學習結果彼此做轉移實驗時所用到的格式，這個部分不是限制行為次數與每次行為的時間上限，而是設定 1500 秒的時間限制，在這 1500 秒內讓賽車 AI 在軌道上任意的跑，時間內約可讓賽車在軌道上跑約 45～55 圈的圈數，視軌道單圈長度與賽車的速度而定。而我們採計的參數如表 4-1。而這些參數的數值都取自超車者。而內建賽道上的實驗則因為賽道長度較長，所以會將 1500 秒改成 2400 秒。

表 4-1：統計格式中使用到的參數名稱及說明

參數名稱	單位	說明
dist_raced	公尺	時間內經過的總距離
dmg	一單位	時間內賽車所受到的損傷
Max_spd	Km/h	出現過的最高速度
min_spd	Km/h	出現過的最低速度
lap_diff	圈	時間內成功超車的次數

4.2 超車學習的成功率與時間統計

在 3.4.1 中，當對手不具阻擋行為且最高速度在 140～150 km/h 時，利用 Q-learning 演算法所訓練出來的駕駛 AI 其超車成功率平均達 91%，而平均超車時間為 22.18 秒。而在這節中要展示的結果是在有學習超車行為下，針對前車三種不同模式，即無阻擋行為、Slowly reactive 與 Fully reactive 三種行為模式在圖 3-1 的測試軌道上做轉移實驗所對應的超車成功率與時間分布。結果如表 4-2 與圖 4-1。表 4-2 中縱向的阻擋策略名稱代表的是 training 時的學習對象，而橫向則為進行測試時所面對的對手。

表 4-2：前車有阻擋行為下的超車轉移實驗

Training/Testing	None blocking	Slowly reactive	Fully reactive
None blocking	91%	55%	51%
Slowly reactive	81.5%	83.5%	78.5%
Fully reactive	90%	82.5%	80%

表 4-2 中對角線部分為原始學習結果，在面對 Slowly reactive 及 Fully reactive blocker 時，超車成功率與前車無阻擋行為的情況相比有些下降。尤其比較面對 Slowly reactive 及無阻擋行為的結果時，下降幅度較大，顯示當前車具阻擋行為時對於超車學習有明顯的影響。

由表 4-2 我們可以看出以不具阻擋行為的前車為對手所學習出來的 AI，在轉移實驗中面對具有阻擋行為的另外兩者時，其成功率大幅度下降。這個結果可能是因為在進行學習時因為學習對象不具阻擋行為，而在轉移實驗中面對有阻擋行為的對手時，超車的 AI 沒有對應的行為導致成功率的下降。而分別以 Slowly reactive 及 Fully reactive blocker 為學習對象並進行轉移實驗時，其成功率並未下降太多，以 Fully reactive blocker 為學習對象並面對另外兩者進行測試時，甚至有表現較好的情形。這是因為在學習時，因對手行為較複雜，所以學習成效較好。

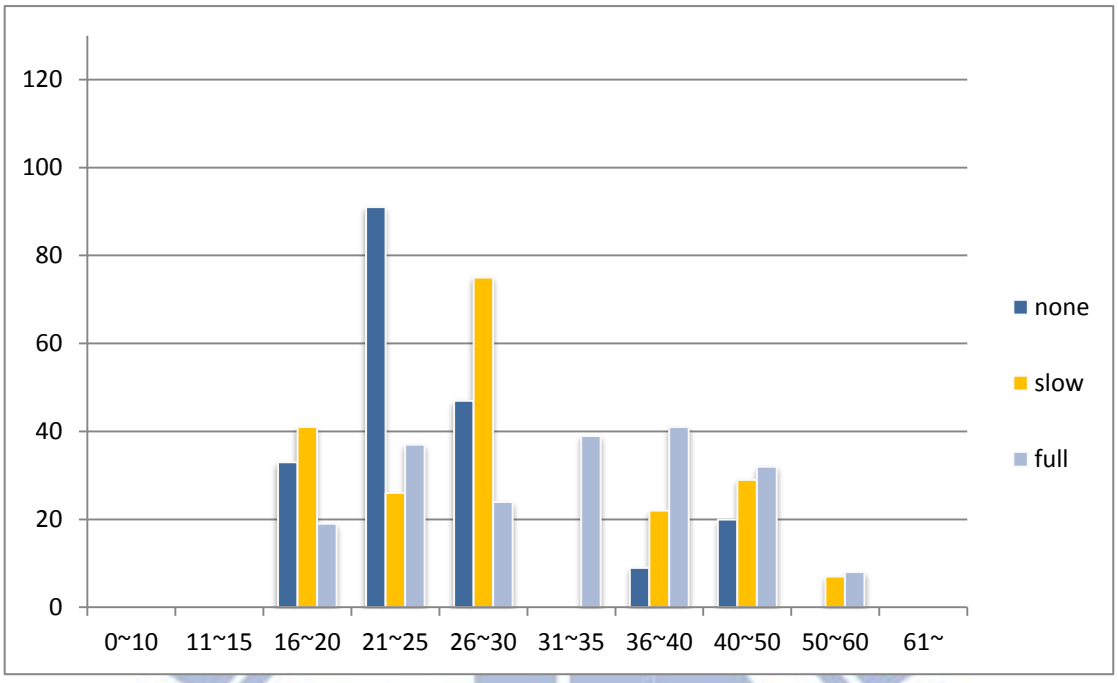


圖 4-1：以 Fully reactive blocker 為學習對象的超車轉移實驗時間分布，每個時間區間中左邊是以 Fully reactive 為學習對象，並使用無阻擋策略 AI 測試結果，中間是以 Fully reactive 為學習對象，並使用 Slowly reactive AI 測試結果，右邊是以 Fully reactive 為學習對象，並使用 Fully reactive AI 測試結果

4.3 不同彎道曲率半徑時的超車行為表現與學習結果合併實驗

這個小節中，我們會把在不同彎道曲率半徑上學習與測試的結果展示出來，包含不同彎道曲率半徑間學習結果互相轉移的轉移實驗與兩組學習檔案如何合併並提供給另一個賽道使用。這一步稱作合併實驗，共有使用曲率半徑 40m、90m 與曲率半徑 90m、150m 使用內插法產出的曲率半徑 65m 及 120m 兩組檔案。

首先，表 4-3-1 是沒有實作超車行為時，在 10 公尺賽道上對 Fully reactive blocker 的表現，而 4-3-2 則是對 Slowly reactive blocker 的表現。表 4-4 是在有實作學習演算法的情形下對手的阻擋策略為 Fully reactive 且軌道寬度為 10 公尺的結果，表 4-5 則是對 Slowly reactive 時的表現。

雖然另外有賽道寬度為 15 公尺的實驗結果，但當寬度為 15 公尺時，因為賽道本身已經足夠寬，導致表現與未使用學習演算法時差異不大。唯一需要另外提

的一點是，當軌道寬度為 15 公尺且彎道曲率半徑越大，如曲率半徑為 120m 或 150m 時，使用學習演算法的 AI 表現甚至略低於原始 AI。所以我們在這邊以 10 公尺寬度的結果為主。而在表格內容中，因為測試中得到的每一筆結果並不會完全相同，所以我們對每個測試組合都是進行 5 次的統計，而最後呈現出來的每項數值我們都會以平均值加減標準差的格式來表達。

表 4-3-1：TORCS 內建 berniw 與對手為 Fully reactive 之統計

	Rad 40m	Rad 65m	Rad 90m	Rad 120m	Rad 150m
dist_raced	60244	66602	75654	79788	81321
dmg	1071.0	0.0	0.0	0.0	0.0
Max_spd	230.8	240.9	251.2	263.4	280.3
min_spd	32.3	116.1	149.6	161.3	168.8
lap_diff	0.0	0.0	1.0	1.0	1.0

表 4-3-2：TORCS 內建 berniw 與對手為 Slowly reactive 之統計

	Rad 40m	Rad 65m	Rad 90m	Rad 120m	Rad 150m
dist_raced	61251	69707	74939	78476	80518
dmg	0.0	0.0	0.0	0.0	0.0
Max_spd	229.7	241.0	251.3	263.2	280.5
min_spd	41.2	120.1	133.3	158.3	47.0
lap_diff	1.0	1.0	1.0	1.0	1.0

表 4-4：學習超車行為後對手為 Fully reactive 之統計

	Rad 40m	Rad 65m	Rad 90m	Rad 120m	Rad 150m
dist_raced	67853 ± 1561	77168 ± 1191	86690 ± 389	97860 ± 2123	108607 ± 573
dmg	407.6 ± 359.6	0.0 ± 0.0	62.4 ± 51.5	0.0 ± 0.0	123.4 ± 246.8
Max_spd	231.0 ± 0.6	242.8 ± 0.2	251.6 ± 0.2	265.7 ± 0.3	282.8 ± 0.9
min_spd	41.2 ± 7.0	67.7 ± 17.3	60.6 ± 41.9	151.9 ± 2.5	38.8 ± 2.8
lap_diff	10.0 ± 1.4	8.8 ± 1.2	8.4 ± 0.5	11.6 ± 1.0	15.6 ± 0.5

表 4-5：學習超車行為後對手為 Slowly reactive 之統計

	Rad 40m	Rad 65m	Rad 90m	Rad 120m	Rad 150m
dist_raced	68667 ± 677	77660 ± 461	88140 ± 109	90068 ± 1969	105307 ± 834
dmg	117.4 ± 141.2	0.0 ± 0.0	18.0 ± 36.0	0.0 ± 0.0	0.0 ± 0.0
Max_spd	231.6 ± 0.4	242.9 ± 0.0	251.7 ± 0.1	265.3 ± 0.2	282.9 ± 0.0
min_spd	37.2 ± 0.2	120.9 ± 0.2	133.1 ± 7.6	50.3 ± 17.3	110.9 ± 24.7
lap_diff	11.0 ± 0.9	9.0 ± 0.0	9.0 ± 0.0	11.8 ± 1.2	17.2 ± 0.7

由表 4-4 及表 4-5 與表 4-3-1 及表 4-3-2 中的結果相比可以看出使用了學習演算法後，超車行為的表現在超車圈數以及行進距離上皆有大幅成長，雖然在沒有學習演算法的情形下損傷值大部分皆為 0，但在有使用學習演算法的結果中，損傷值的成長也不至於太高。對比超車圈數與行進距離，我認為這是一個合理的 trade off。在不同的彎道曲率半徑上，都可以有一定程度的表現。但 Fully reactive 與 Slowly reactive 兩種阻擋策略在簡單的測試軌道上差異較不明顯，目前可以看出在以 Slowly reactive 為對手時，所受到的損傷以及超車圈數這兩個參數上的表現略優於 Fully reactive。在接下來的實驗結果中我們使用的超車對象主要為 Fully reactive blocker。而在合併實驗的部份，也就是在彎道曲率半徑為 65m 以及 120m 的賽道上，也可以從上面的結果中看出對於一個沒有實際學習過的賽道也有一定的效果。根據這個結果，我們在稍後 TORCS 內建賽道上的實驗中，也是直接利用在這邊進行學習過後的 AI 來進行。

4.4 TORCS 內建賽道上的超車實驗

這節中，我們使用了在 3.3 節提到的兩個 TORCS 內建賽道來進行超車實驗的測試，對手是 Fully reactive blocker。我們首先使用與表 4-4 中使用的相同的五個 AI 進行超車實驗。表 4-6 是 TORCS 內建的 Berniw 與 Fully reactive blocker 進行實驗的統計結果。表 4-7 是有使用學習演算法進行超車行為學習並在 CG track

3 上進行測試得到的結果，而表 4-8 則是在 Ruudskogen 上進行測試的結果。表 4-7 與 4-8 中橫軸代表所使用的學習結果是從哪一個訓練軌道所學習而來的。

表 4-6：TORCS 內建 Berniw 的各項參數統計結果

	dist_raced	dmg	Max_spd	min_spd	lap_diff
CG track3	103499	166	227.3	30.3	3
Ruudskogen	116255	1412	244.2	28.6	6

表 4-7：CG track 3 上的各項參數統計結果

	Rad 40m	Rad 65m	Rad 90m	Rad 120m	Rad 150m
dist_raced	109779 ± 1422	110107 ± 882	109605 ± 873	110962 ± 464	111062 ± 386
dmg	757.4 ± 721.2	673.8 ± 425.5	1177.4 ± 1035.9	1021.4 ± 657.0	873.0 ± 407.3
Max_spd	226.2 ± 0.8	226.9 ± 0.4	226.4 ± 0.6	226.4 ± 0.5	226.4 ± 0.6
min_spd	34.9 ± 10.8	35.2 ± 8.5	34.4 ± 9.5	36.2 ± 17.1	55.1 ± 5.5
lap_diff	4.8 ± 0.4	4.8 ± 0.4	4.8 ± 0.4	5.0 ± 0.0	5.0 ± 0.0

表 4-8：Ruudskogen 上的各項參數統計結果

	Rad 40m	Rad 65m	Rad 90m	Rad 120m	Rad 150m
dist_raced	120783 ± 1669	121419 ± 700	121683 ± 1142	119343 ± 5518	121952 ± 262
dmg	418.0 ± 348.8	595.8 ± 373.5	533.8 ± 347.7	366.2 ± 291.0	396.2 ± 360.0
Max_spd	245.1 ± 0.5	244.4 ± 1.2	244.8 ± 0.4	245.1 ± 0.4	244.8 ± 1.0
min_spd	23.0 ± 11.6	29.1 ± 1.8	28.4 ± 2.1	24.4 ± 6.9	31.4 ± 4.6
lap_diff	6.6 ± 0.5	6.8 ± 0.4	6.8 ± 0.4	6.8 ± 0.4	7.0 ± 0.0

雖然在挑選 TORCS 中的內建賽道時我們已經盡可能挑選彎道較少且寬度較接近測試軌道的兩個賽道，但因為內建賽道的彎道依然還是較複雜（可能有連續彎道的情形），所以從表 4-7 及表 4-8 中可以看出在超車圈數的表現上都只比沒有學習超車行為的原始 AI 高上 2 圈左右，但在 CG track 3 上受到的損傷值較高，表

示在超車時可能因為賽道寬度為較窄的 10 公尺且彎道複雜度較高造成兩車間較常發生碰撞。

接著在表 4-9 則是則同時使用原始訓練結果的三個檔案(曲率半徑 40m、90m、150m)，在要進行超車行為時，選擇與目前所在的軌道位置曲率半徑最接近的學習檔案來決定要採取的動作。

表 4-9：組合多個學習結果的各項參數統計結果

	dist_raced	dmg	Max_spd	min_spd	lap_diff
CG track3	110690 ± 403	790.0 ± 156.6	226.4 ± 0.4	33.8 ± 7.2	4.8 ± 0.4
Ruudskogen	123412 ± 568	691.6 ± 797.1	244.4 ± 1.1	29.8 ± 1.2	7.0 ± 0.0

表 4-9 的結果中，在組合了多個學習檔案後，在 CG track 3 上的表現並沒有太大差異，但在超車者所受到的損傷值中有些許下降。而在 Ruudskogen 這個賽道上的結果，時間內超車者所跑的距離比只使用單一檔案時為高，但所受到的損傷則比只使用單一檔案時高。在 Ruudskogen 的原始結果中只有一次統計的傷害值特別高，其餘的傷害值皆低於 600。從兩個實驗結果中可以看出，使用多個學習檔案的實驗對於超車表現的改進是有效果的。

4.5 以內建 AI 為對手學習阻擋行為時超車成功率與時間

在這部分，我們的學習者是前方負責阻擋的車輛，這個阻擋者我們依然是使用 Berniw 作為一個基礎的駕駛 AI，而負責超車的 AI 則使用內建的 Tita、Lliaw 及 BT 為對象來幫助學習。而這三個內建 AI 與 Fully reactive blocker 做了一次簡單的實驗結果後發現，在 200 次的統計時間內，除了 BT 有在每次行為的限制時間內成功超車一次外，皆因超過規定時間而被判定為失敗。

4.5.1 以 40 秒為判斷成功與否時的表現

在第三章中有提到過阻擋行為時的表現因學習時間長度不同對超車時間分布也會有所變化。表 4-10 為以 40 秒為學習時間限制下經 200 次學習階段所統計出的成功率與成功次數。在超車成功率中百分比旁邊的括號中代表 200 次學習階段中成功超車的次數。平均超車時間的部分只採計在40秒內有成功超車的次數。超車時間分布如圖 4-2。圖 4-2 分別是以 lliaw、tita 及 bt 為對手時的超車時間分佈。圖中可以看到被超車時間大部分集中在 41~50 秒的區間中，這部分推測是在使用圖 3-1 這個具有較長直線的測試賽道時，兩車在這個時間帶中正準備入彎或出彎，且因為學習時間限制為 40 秒，所以沒有學習到這部份該有的行為。與 Fully reactive blocker 比較後發現，在面對 TORCS 內建 AI 這些行為模式較固定的對手時，Q-learning blocker 可能會對內建 AI 中較常出現的模式過分學習（overfitting），如 lliaw 與 tita，這兩個駕駛 AI 會在接近前車時將速度降低，而 Q-learning blocker 只需與超車者較靠近的一側稍微偏移即可在一段時間內不被超車者超過。也因此若下次超車的情況與前次的行為有些不同便很有可能被超越，所以在這裡 Q-learning blocker 的表現會較 Fully reactive blocker 差。同樣的，bt 在超車時也以向賽道內側偏移為主，若前車與內側距離過近（接近彎道時）才會向外側偏移進行超車。

表 4-10：學習時間 40 秒下的超車成功率

對手駕駛 AI	超車成功率
Lliaw	28% （56/200）
Tita	28.5% （57/200）
Bt	30.5% （61/200）

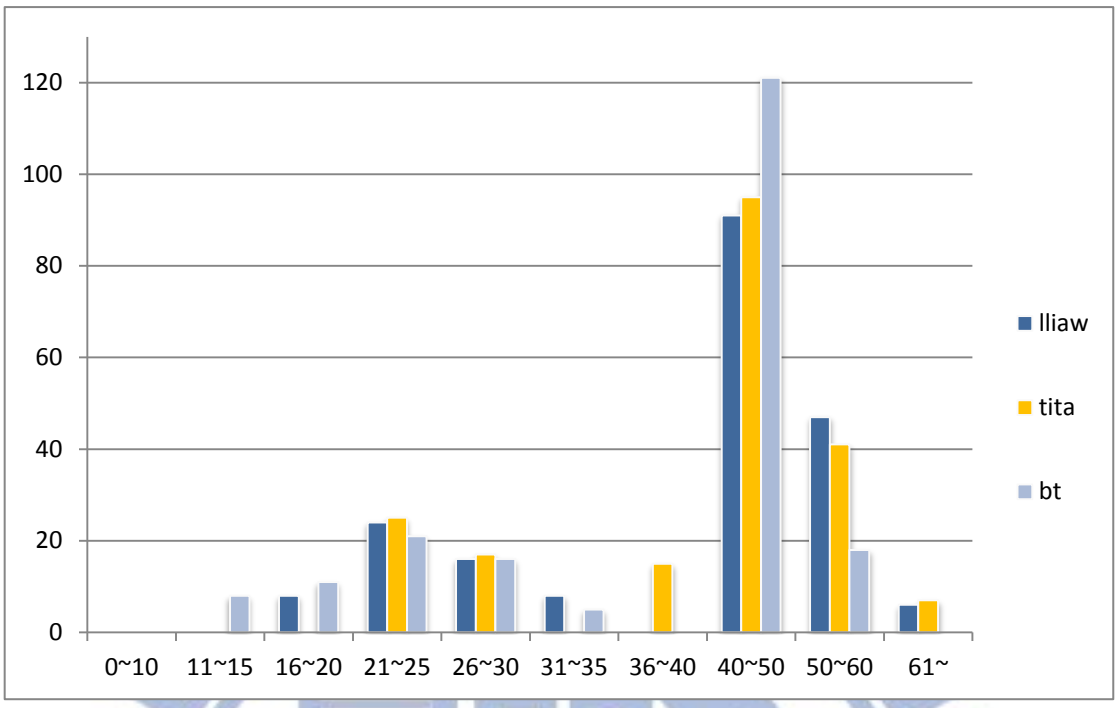


圖 4-2:學習阻擋行為之超車時間分佈圖:每個時間區間中左邊是 lliaw 測試結果，中間是 tita 測試結果而右邊則是 bt 測試結果。

因為在學習時間被限制為 40 秒的情況下，我們發現在超車時間的分布裡，出現了超車時間都擠在 40 秒後的區間之中，推測可能是因為時間限制的關係導致沒有學到針對 40 秒以後的行為。於是我們將學習時間拉長至 60 秒再次觀察其時間分布。表 4-11 是以 60 秒為學習界限時經 200 次學習階段所統計出的成功率與成功次數。在超車成功率中百分比旁邊的括號中代表 200 次學習階段中成功超車的次數。時間分布如圖 4-3。

表 4-11：學習時間 60 秒下的超車成功率

對手駕駛 AI	超車成功率
Lliaw	26.5% （53/200）
Tita	28% （56/200）
Bt	31% （62/200）

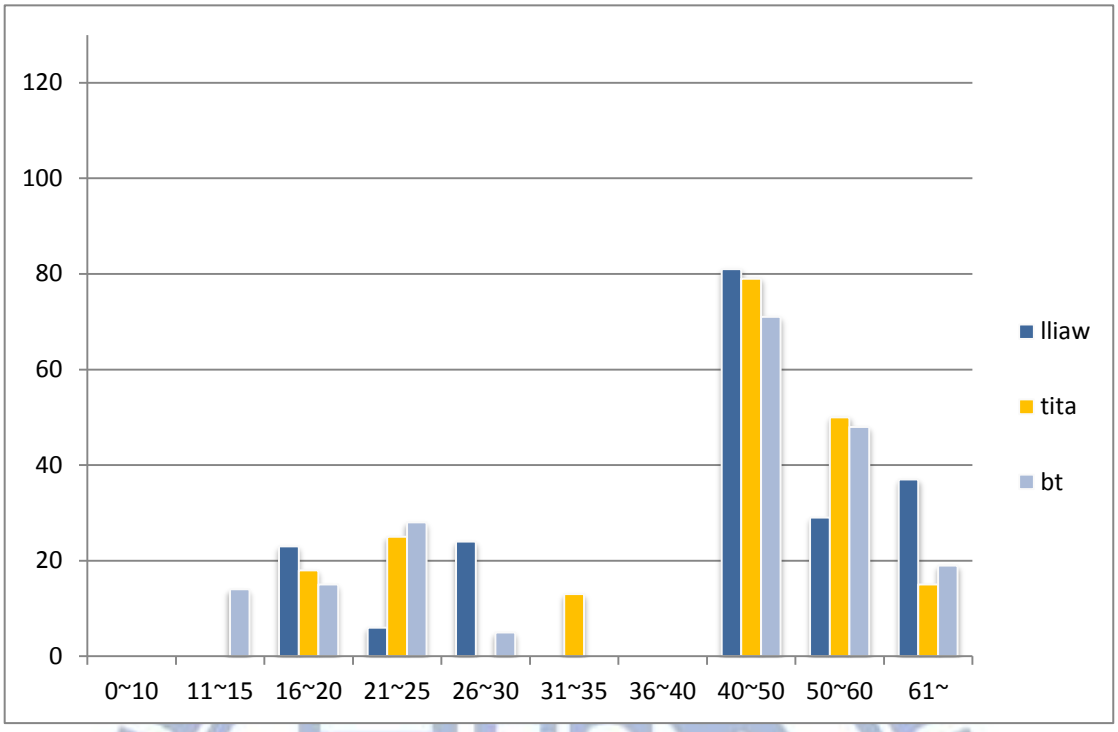


圖 4-3：60 秒學習阻擋行為之超車時間分佈圖：每個時間區間中左邊是 lliaw 測試結果，中間是 tita 測試結果而右邊則是 bt 測試結果。

與表 4-10 及圖 4-2 比較後可以發現當我們將學習時間拉長至 60 秒後，在 lliaw 與 tita 為對手時的超車成功率上的確有些許下降且三者的時間分佈皆有往後分散的趨勢，表示阻擋者對於彎道上的學習也有一定效果存在。

當使用圖 3-1 的長直線測試賽道，且後車為了方便比較，使用與 Fully reactive blocker 進行學習後的 Q-learning overtaker 時，在對 Q-learning blocker 進行三次不同學習，此處 Q-learning overtaker 不再更新其 Q-table 內動作對應的值，只對阻擋行為進行學習。經測試，三次學習超車成功率分別為 37%、39.5%及 36.5%，平均成功率可達到約 37.7%。對比表 4-2 Q-learning blocker 與 Fully reactive blocker 的測試結果中可以知道在長直線賽道上，Q-learning blocker 的表現優於 Fully reactive blocker。實際觀察兩者之駕駛行為後發現，Fully reactive blocker 是對後車的位置去調整自己與賽道中心線的距離，所以當後車行進路線貼近賽道邊緣時，Fully reactive blocker 也會讓自己的行進路線貼近邊緣，造成另外一側有非常大寬度的賽道，當後車切換路徑的速度夠快時（如 Q-learning overtaker），則較容易被

超越。相對的，Q-learning blocker 在直線上很少會讓自己貼近賽道邊緣，在後方有來車時，都與賽道邊緣維持一段距離。如此行為模式讓超車者較難超車，因為 AI 不會開到超過賽道邊緣的草地上，使得某一側的賽道空間不足以進行超車，而要切換至另一側時，依然有很大機會被阻擋者用這種方法阻擋下來。但在預備入彎與出彎時，因為 berniw 在過彎時會讓賽車由外側開進內側，再由內側開回外側，使得在彎道的出入口附近會有較容易被超車的情形發生。而前面提到的面對內建 AI 時可能有 overfitting 的問題在這邊較不嚴重，因為 Q-learning overtaker 切換路徑的速度較 AI 快，所以學習的時候會出現的情況也較多。被超車成功率的上升主要因為 Q-learning overtaker 的超車行為較強。

4.5.2 對不同 AI 對手的阻擋學習結果的轉移實驗

表 4-12 是針對 3 個不同的對手 AI 所學習出來的阻擋行為作轉移實驗得到的超車成功機率與超車時間分布圖。在表 4-12 中，縱向的 AI 名稱代表的是 training 時使用的超車 AI，而橫向則為進行測試時所面對的超車 AI。

表 4-12：阻擋行為轉移實驗中 AI 超車成功率

Training/Testing	Lliaw	Tita	Bt
Lliaw	28%	35%	40%
Tita	36%	28.5%	42.5%
Bt	33.5%	33%	30.5%

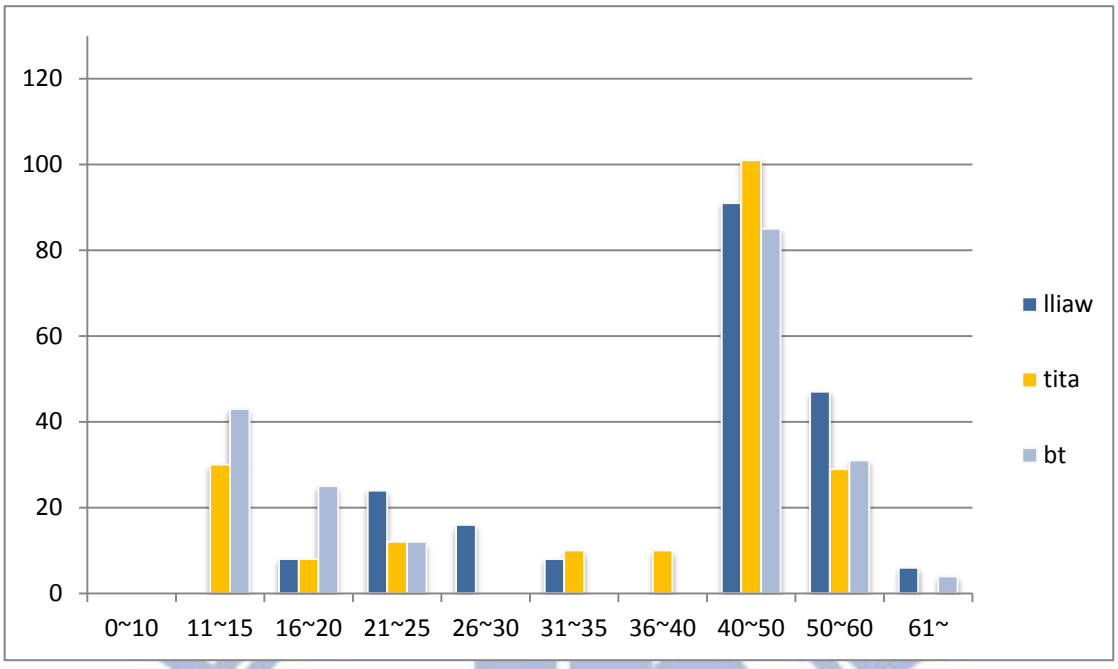


圖 4-4：以 lliaw 為學習對象的阻擋行為轉移實驗超車時間分布：圖中每個時間區間左邊是以 lliaw 為學習對象，並使用 lliaw 測試結果，中間是以 lliaw 為學習對象，並使用 tita 測試結果，右邊則是以 lliaw 為學習對象，並使用 bt 測試結果。

表 4-12 及圖 4-4 中我們還是以 40 秒版本的學習結果來進行轉移實驗。從表 4-12 可以看到以某個駕駛 AI 為學習對象去面對另外兩個駕駛 AI 進行阻擋行為時，被超車的成功率雖皆有上升，尤其是在面對 bt 時上升較多，對比前車（為了方便比較，此處的前車為不對超車者做出反應的 berniw）無阻擋行為時，lliaw 的超車成功率為 100%、tita 為 97%以及 bt 的 100%，可以發現學習的阻擋行為經過轉移測試後依然有一定的效果。以 bt 為測試對象時，因為 bt 這個駕駛 AI 會在距離前車較遠處便開始進行路徑的偏移，與另外兩者在駕駛行為上有較明顯的不同，所以我們推測因為學習對象與測試對象其駕駛模式不同使得超車成功率上升。而圖 4-4 是以 lliaw 的轉移實驗為範例將時間分佈列出，圖中可以看到雖然對手不同時，超車時間分佈皆不同，但在 40 秒後的區間中都有類似的趨勢，也就是因為通常到了 40 秒時直線區段通常已經結束，所以在使用 40 秒學習時間的版本時，大部分的被超車時間會因為沒有學習到在彎道上應採取的行動而集中在這個區間。而以其他兩個駕駛 AI 作為學習對象的轉移實驗也有類似的情形發生。

4.6 不同彎道曲率半徑軌道與 TORCS 內建賽道上的阻擋行為實驗

最後在這裡我們用 3.5.3 中提到的實驗方法來對阻擋行為進行實驗。而用來學習的超車者分別是 TORCS 中內建的 lliaw 以及有經過 Q-learning 演算法改進的 berniw。表 4-13 及表 4-14 分別是以 lliaw 及學習過後的 AI 在曲率半徑為 40m、90m 及 150m 三種測試軌道上學習的結果。

表 4-13：在不同曲率半徑上以 lliaw 為學習對象的阻擋行為實驗

	Rad 40m	Rad 90m	Rad 150m
dist_raced	65046 ± 422	83388 ± 1037	98868 ± 2693
dmg	1448.8 ± 936.4	522.6 ± 577.9	650.6 ± 570.9
Max_spd	231.0 ± 0.3	249.9 ± 0.3	283.1 ± 0.3
min_spd	43.7 ± 21.9	80.3 ± 19.6	44.1 ± 4.3
lap_diff	7.2 ± 0.4	5.2 ± 0.4	10.8 ± 0.74

表 4-14：在不同曲率半徑上以 Q-learning AI 為學習對象的阻擋行為實驗

	Rad 40m	Rad 90m	Rad 150m
dist_raced	66774 ± 1837	87071 ± 2184	103414 ± 2506
dmg	904.4 ± 1128.5	1029.6 ± 870.8	1696.8 ± 1242.2
Max_spd	229.9 ± 0.6	251.3 ± 0.0	279.6 ± 0.7
min_spd	46.9 ± 10.4	67.6 ± 8.3	32.4 ± 4.0
lap_diff	9.4 ± 0.8	7.6 ± 0.8	14.0 ± 0.0

將表 4-13 與表 4-3-1 比較後發現以 TORCS 內建 AI 為學習對象時，在超車圈數上的表現 Q-learning blocker 落後 Fully reactive blocker，但表 4-14 與表 4-4 相比後可以看出以 Q-learning overtaker 為學習對象時，在超車圈數上僅是略優於 Fully reactive blocker，且以損傷值來說都偏高。初步的推測是賽道直線較短且未限制兩車在彎道上的速度需一致所導致，使得前車在彎道上要進行阻擋時的反應

不夠快，容易出現超車者已經處在一個可以超車的位置，而阻擋者卻依然想要阻止超車者進而發生碰撞造成雙方的損傷也依然被超越過去。理想上，若阻擋者反應足夠快，則應該盡可能讓超車者開在阻擋者後方。接下來的 TORCS 內建賽道測試結果如下：

表 4-15：CG track 3 上面對 lliaw 的阻擋行為統計結果

	Rad 40m	Rad 90m	Rad 150m
dist_raced	106007 ± 1101	106200 ± 535	106564 ± 557
dmg	1237.0 ± 1001.1	416.8 ± 360.2	1289.2 ± 1241.2
Max_spd	226.2 ± 0.2	226.4 ± 0.4	226.1 ± 0.2
min_spd	51.1 ± 4.1	55.6 ± 4.7	61.6 ± 7.2
lap_diff	4.8 ± 0.4	4.4 ± 0.4	4.8 ± 0.4

表 4-16：Ruudskogen 上面對 lliaw 的阻擋行為統計結果

	Rad 40m	Rad 90m	Rad 150m
dist_raced	119980 ± 525	120577 ± 198	119804 ± 844
dmg	254.0 ± 132.8	0.0 ± 0.0	506.4 ± 291.9
Max_spd	245.6 ± 0.7	245.5 ± 0.6	244.9 ± 0.5
min_spd	59.9 ± 3.9	60.6 ± 3.3	56.8 ± 8.4
lap_diff	6.0 ± 0.0	6.0 ± 0.0	6.0 ± 0.0

首先討論的是 lliaw 的測試結果，在表 4-15 與表 4-16 中，由於 CG track 3 的軌道寬度較窄（10 公尺）且彎道較多，使得學習過後的 AI 因為本身駕駛行為的關係，容易貼近賽道邊緣，對超車者來說可用來超車的空間較大，與表 4-6 相比，在超車圈數上落後且受到的損傷較高。而在 Ruudskogen 的表現則與表 4-6 中的結果相差不大。

表 4-17：CG track 3 上面對 Q-learning overtaker 的阻擋行為統計結果

	Rad 40m	Rad 90m	Rad 150m
dist_raced	109033 ± 1946	110149 ± 669	110142 ± 842
dmg	1924.4 ± 406.6	2098.6 ± 832.6	1627.0 ± 665.9
Max_spd	225.1 ± 0.6	225.3 ± 0.5	225.0 ± 0.4
min_spd	34.9 ± 6.6	47.3 ± 15.1	41.6 ± 23.6
lap_diff	5.0 ± 0.0	5.0 ± 0.0	5.2 ± 0.4

表 4-18：Ruudskogen 上面對 Q-learning overtaker 的阻擋行為統計結果

	Rad 40m	Rad 90m	Rad 150m
dist_raced	125017 ± 402	125199 ± 226	123286 ± 2490
dmg	235.4 ± 412.8	29.4 ± 2.8	840 ± 383.4
Max_spd	244.5 ± 0.4	244.0 ± 1.3	244.1 ± 1.5
min_spd	46.9 ± 12.4	54.8 ± 12.4	41.5 ± 19.1
lap_diff	7.0 ± 0.0	7.0 ± 0.0	7.0 ± 0.0

表 4-17 與表 4-7 兩邊的結果相比，在行進距離與超車圈數上並沒有太大的差別，但損傷值來說是 Q-learning blocker 這邊較高，從這裡也可以得到與前面表 4-13 及 4-14 中類似的結論，也就是反應速度不夠快，導致損傷值的提高。而比較表 4-18 及表 4-8 的結果也有類似的情況出現。

第五章 結論與未來展望

5.1 結論

在這篇論文中，我們首先使用 Q-learning 演算法面對一個行為簡單的對手來學習基礎的超車行為。而在有了基礎的超車能力之後我們又利用相同的演算法去面對具有阻擋能力的對手並從中學習對應的行為，從超車行為的轉移實驗中我們知道了對不同對手所學習到的超車行為是有一定程度可以共通使用的，且當對手的行為模式越複雜，則所學習到的對應行為也會越多，對於轉移實驗來說甚至會提升超車者的學習成效。在寬度為 10 公尺的賽道上與未進行學習演算法改進的原始 AI 相比，超車表現上有大幅的成長。而在不同彎道曲率半徑的情況下，也依然可以從中學習到不錯的超車行為。最後在 TORCS 內建的賽道上，我們也可以看出即使在較複雜的賽道上進行超車行為實驗，其表現依然較內建的 AI 好。

接著我們進行阻擋行為的學習時也利用了 TORCS 中內建的三種不同的駕駛 AI 作為對手，在這邊的實驗中我們也證明了學習結果的可轉移性以及拉長學習時間對於超車時間分布的改變。在長直線測試賽道中，以 Q-learning 學習阻擋行為的 AI 在面對以 Q-learning 學習超車行為的 AI 時，阻擋行為的表現甚至比 Fully reactive blocker 好。而學習阻擋行為的 AI 在具有不同彎道曲率半徑的賽道以及 TORCS 中內建的真實賽道上證明了以學習演算法進行改進的駕駛 AI 其阻擋行為在非直線以及較複雜的賽道上依然具有一定程度的阻擋能力。唯一較不足的地方是，阻擋行為的學習在較複雜的軌道上表現不如 Fully reactive blocker 且受到的損傷值較高。對於這個現象，可能的原因如同第四章中所提到，可能因為阻擋者對於後車的反應不夠靈敏導致在超車圈數以及損傷值這兩個參數上的表現較差。

5.2 未來展望

最後，在結束這一連串的實驗後，依然有許多課題可以繼續做下去。例如運用其他學習演算法來比較各個學習演算法間表現是否會有差異且學習所花費的時間能否縮短，又或是利用類神經網路或模糊邏輯等等不同的方式來做超車與阻擋行為的研究。尤其要提到的是阻擋行為這部分，因為 TORCS 中的內建 AI 並沒有阻擋的概念，且在實作上也較困難，所以目前為止有關阻擋行為的研究並不多。多數的相關研究中，若有阻擋行為的描述，通常都是 hard coded，而不是針對這個行為去進行學習或優化，所以通常都只是將這些行為較固定的阻擋行為 AI 用來當作超車行為測試的對象。未來若可以將更多車輛間的資訊導入學習演算法中，相信對於阻擋行為的學習可以有更好的成效出現。在阻擋行為中對於後車的反應不夠靈敏這個問題上，我們或許可以提高阻擋者偵測後車的有效距離，讓阻擋者在較遠的地方就可以偵測到後車，也能有更多的時間針對後車做出反應。而賽車場上不只有超車與阻擋這兩種較特殊的駕駛行為，未來甚至可以對其他賽車行為模式進行研究，如：特殊過彎技巧等等。在這篇論文中都是以一個超車者與一個阻擋者作為實驗基礎，但在超過兩人競賽的環境下，超車與阻擋的行為又有可能發生變化，這也是未來一個可能的研究課題。

參考文獻

[1] "The open racing car simultaor website, " http://torcs.sourceforge.net/. [Online]. Available: http://torcs.sourceforge.net/

[2] "The Motion Tech Behind Kinect, " http://theinstitute.ieee.org/.

[3] D. Perez, G. Recio, Y. Saez and Pedro Isasi, "Evolving a Fuzzy Controller for a Car Racing Competition," in *IEEE Symposium on Computational Intelligence and Games*, pp. 263-270, 2009.

[4] L. Cardamone, D. Loiacono and P. L. Lanzi, "Learning Drivers for TORCS through Imitation Using Supervised Methods," in *IEEE Symposium on Computational Intelligence and Games*, pp. 148-155, 2009.

[5] K-J. Kim, J-H. Seo, J-G. Park and J-C. Na, "Generalization of TORCS car racing controllers with artificial neural networks and linear regression analysis," in *ELSEVIER Neurocomputing*, vol. 88, pp. 87-99, 2012.

[6] E. Yee and J. Teo, "Evolutionary Spiking Neural Networks as Racing Car Controllers," in *Hybrid Intelligent Systems (HIS)*, pp. 411-416, 2011.

[7] L. Cardamone, A. Caiazzo, D. Loiacono and P. L. Lanzi, "Transfer of Driving Behaviors Across Different Racing Games," in *IEEE Symposium on Computational Intelligence and Games*, pp. 227-234, 2011.

[8] M. Botta, V. Gautieri, D. Loiacono and P. L. Lanzi, "Evolving the Optimal Racing Line in a High-End Racing Game," in *IEEE Symposium on Computational Intelligence and Games*, pp. 108-115, 2012.

[9] D. Loiacono, A. Prete, P. L. Lanzi and L. Cardamone, "Learning to Overtake in TORCS Using Simple Reinforcement Learning," in *IEEE Symposium on Computational Intelligence and Games*, pp. 1-8, 2010.

[10] E. Onieva, L. Cardamone, D. Loiacono and P. L. Lanzi, "Overtaking Opponents with Blocking Strategies Using Fuzzy Logic," in *IEEE Symposium on Computational Intelligence and Games*, pp.123-130, 2010.