私立東海大學資訊工程研究所

碩士論文

指導教授：石志雄　博士

作球策略對撞球賽局之影響探討

(Effects of cue ball positioning strategies on billiard game)

研究生：林政儒

中華民國 100 年 7 月

摘要

撞球是物理中碰撞、旋轉與人類戰略技巧行為結合; 具備力、智慧與美感的一種運動，就像許多運動一樣受到物理定律主宰。優秀的玩家在力道控制、物理碰撞、摩差力影響甚至物體的移動上有著卓越的感知。撞球台遊戲是一種選擇使得我們能夠在模擬中嘗試各種技巧，為了在模擬的遊戲世界中能夠更貼近現實，除了坊間撞球遊戲軟體著重於讓遊戲畫面更為真實細膩之外，遊戲軟體中 AI 的人工智能是必扮演著不可或缺的角色。因此本研究基於可幫助玩家擊球練習之 AI 雛型加入控制擊球力道藉此控制母球以利於下一球進攻之策略性人工智慧。且著重於將人類真實之撞球行為，包含考慮力道、瞄準、球型分佈等加入模擬，比較加入不同策略後對球局之影響。

最後與固定力道 AI 的不同力度比較，顯示出有考慮出桿力道之 AI 能夠擁有更低的進球桿數(每次出桿記為一桿)且合併擁有較低的犯規次數(母球洗袋犯規)，如此便同時具備不同固定力道 AI 的優點。另外考慮 AI 瞄準失誤的情況，當兩 AI 具備相同範圍的瞄準失誤且失誤範圍在兩度內，我們也能得到與上敘相同的結果與趨勢，說明此系統具備相當程度之一般性。

Abstract

Billiard is a sport which combined physics collision, rotate, and strategic skills of human behavior with strength, wisdom and beauty. It is based on physics laws just like the other sports. Good players have an excellent sense on the force controlled, influence of friction and the movement of balls. It is a good choice to allow me to apply many of the techniques on the demo game. Besides focused on the game screen, AI is plays an important role in order to make game more likely the reality. We add the force controlled system to control the mother ball to benefit the next hit of target ball based on the AI prototype which can help players to hit balls. The simulation focus on the human behavior including strength, aim, ball distribution and etc. of pool.

Finally, compare with the constant force AI on different force level. It shows that force controlled AI cando well than the constant force AI in times of hit and have the lower times of illegal hit at the same time.In other world, the force controlled AI has the advantages that constant force AI had in different force level. We got the similar results when we add the miss angle of two in the two AI type. It explain that our system is generalized.

目錄

圖次

表次

第一章前言

1.1 研究背景與目的

本研究模擬根據花式撞球 14-1 規則。14-1 是一種使用白色母球及 15 顆號碼球的花式撞球玩法[1]。每次擊球前必須先指定子球號碼及袋口，成功該子球撞入該袋口，才能得分並繼續擊球，否則即換由對方擊球。開局前，15 顆子球緊密排在球台一側的三角框線裡，最前端的子球中心剛好落在腳點上。開球時若未得分，必須至少讓兩顆子球碰到台邊，否則就是犯規而扣分。

撞球比賽一直是受大眾歡迎的遊戲，它以某些形式存在或以各種不同的規則娛樂大眾已經有數百年的歷史。此遊戲是由一擊球桿，一顆母球，及數顆不同顏色之子球所構成。比賽在一有六個球袋之布質球桌上進行。玩家藉由推送擊球桿，打擊一個母球，當此開端母球沿擊球桿推送方向前進，導致母球和另一個子球（目標球）碰撞後，希望目標子球得以進入一個挑選的球袋之內。若玩家無法順利將目標子球擊入目標球袋，則需交出擊球權。最後擊入越多子球之玩家獲勝。

最近，人工智慧研究已經成功運用在許多方面。比賽遊戲已經在機器人工程領域中尤其接受廣泛的研究，其中亦包括撞球遊戲。而支援機器人進行遊戲之 AI 亦有需多優異的理論可分析出最佳擊球策略，但皆缺乏一數位平台來模擬其實際運作的效果，以利其與真正撞球台環境的相互整合並提供一經濟上可行的可靠系統。我們開發一套可與使用者對戰的 AI 為目標，以此為前提架構出基本的遊戲平台

與 AI 雛型。此一 AI 雛型系統可幫忙使用者擬定撞球策略並預先為接著的幾次撞擊確保最大的得勝可能。此 AI 雛型系統會判斷包括經過計算球的滾動入袋的偏離、母球滾動和桌子的摩擦力，然後已發展的模糊邏輯運算法則將應用這些數值專為特定位置的母球計算直接撞擊其他每個子球的難度和角度。接著可以為可撞擊的目標球及其對應的底袋產生一優先順序，然後，那順序最前面的目標球的撞擊點連同開始運轉的初速度都能同時用專家系統來決定。這些將顯示在互動圖解式的使用者平台上，引導使用者移動他們的撞球桿，使用者僅需移動球桿，配合電腦指示的方向對準，以母球中心所延伸而出的中心線來擊球。

另外在球桌對子球的第一次擊球，在母球與子球的碰撞以後，改變母球位置決定它連續擊球成功。我們能從母球上畫一條理想的醒目的線來擊中子球進入根據最大容忍角度選擇標準的球袋。更加進一步的擴大理想的線演算法包括母球改變位置的理想的速度演算法。改變母球位置是要提早要求的複雜的行動。一個新的提議繼續地申請最大容忍角度查尋兩次。在先碰撞的擊球和第二在碰撞軌道。另外對最大容忍角度標準，也提出一個新的可看見的子球計數標準協助母球改變位置。這個標準根據零容忍角度區域的分析被開發了。

控制改變位置的指令在一個圖形接口首先分析母球的位置然後被顯示系統利用母球，子球地點和球桿自動追蹤的視覺系統。用戶能根據瞄準方向和在電腦顯示器上顯示在撞球桌的擊球的速度的視覺指南來調整球桿。視覺指南包括顯示被分析的速度和實際速度的一條提示桿並在第一次碰撞以後適當地改變最佳的

下次擊球母球位置。

用戶必須擊出母球多次，以便分析實際球桿的擊球速度。現實世界和虛擬世界撞球為一條精確提示桿和球桿速度演算法為一個最小平方差實驗結果的最大容忍角度重新定位策略在使用我們的訓練設施後，具有不同的技能級別的使用者都執行出相同的使用者在沒有經過指導的結果。這不但證明我們的訓練設備和重新定位的演算法的可靠性，也證明在指導使用者獲得最佳性能的重新定位演算法的有效性。

1.2 研究目的

在撞球比賽中，若單次進球成功率高達八成甚至九成，一個球員在擊球時是否有作球往往成為影響勝負的關鍵，有作球的球員通常可以連進好幾球，沒有作球的球員可能常常會錯失下一球的進球，此時並非瞄準上面的失誤而是沒有考慮下一球擊球，往往使得母球跑到一個沒辦法將子球打進袋的位置上，可能是進攻路線被阻擋或是母球洗袋犯規，所以我們用模擬的方式研究作球策略是否會能夠讓一盤球局中擁有較低的出桿次數或連續進球數。

第二章文獻探討

2.1 文獻

Jeff Lander 於 1999 為了能夠了解碰撞、摩差力等物理在撞球上的交互作用關係所製作 3D 撞球碰撞雛形，以方便在模擬中實踐其想法及技巧，已完成場景、球與球桿各種物件建構，基礎數學程式以及物理碰撞、摩差力等演算法，理論實做完整度相當高但缺乏娛樂性。以此為基礎，但是外觀及功能上皆缺乏完整性，所以我們增加子球數及球袋，建構出較為完整之互動式遊戲平台，並且加入可幫助擊球訓練的 AI 系統雛型，以驗證此一平台之可用性與實用性。

2.2 探討解決之問題描述

Jeff Lander 於 1999 所製作 3D 撞球碰撞雛形，已完成場景及碰撞、摩差力等演算法，但其桌面缺少可將球擊入的袋口，以及能夠模擬球入袋的功能。

1. 除了基本遊戲平台不完整外，原本的系統尚欠缺可幫助擊球之 AI，來輔助使用者決定出桿角度，提高擊球入袋的成功率[21]。

2. 在上敘 AI能夠正確的將出桿角度計算出來之後，希望能夠考慮擊球力道，如此一來便能夠在進攻下一顆目標球時得到越小的擊球角度，一般來說擊球角度越小以及母球與目標球之距離愈接近時能夠提高進球率，比較加入了此擊球策略

後對球局產生的影響，我們期待能夠降低失誤且提高平均連桿數。

2.3 問題 1 之解決構想與方法

將原場景讀出，此場景之球桌並無球袋，在繪圖軟體中修改桌面，使其具有與正規花式撞球球桌的六個球袋，再回到原程式中與原場景結合。修改相關的程式碼，使球移動至球袋所在的座標時，球可停止移動並將球消除，以達到在視覺上球滾入袋中的效果。再加入擊球路徑演算法，使 AI 系統對各個子球與球袋做計算判斷出擊球成功率最高的子球與球袋之組合，優先打擊，再以此計算出正確的擊球角度，並換算成正確的出桿角度，最後在互動平台上指示出來供使用者參考，以輔助玩家提高擊球成功率，或者由 AI 自行打擊。

一、外觀部分

1. 將程式內場景以 .obj 檔案格式讀取。

2. 用 Rhinoceros 4.0 Evaluation 轉檔成 .3ds 檔。

3. 使用 Google SketchUp7 讀取 .3ds 檔，修改場景，再輸出成 .obj 檔。

4. Rhinoceros 4.0 Evaluation 校正座標。

5. 讀回原程式中，修改成與變動的場景相關之程式碼。

二、路徑演算法

1. 求出六個球袋中最易打進之球袋。

2. 母球撞擊子球之最佳角度。

3. 移動球桿至最佳擊球位置。

三、分析

1.外觀部分

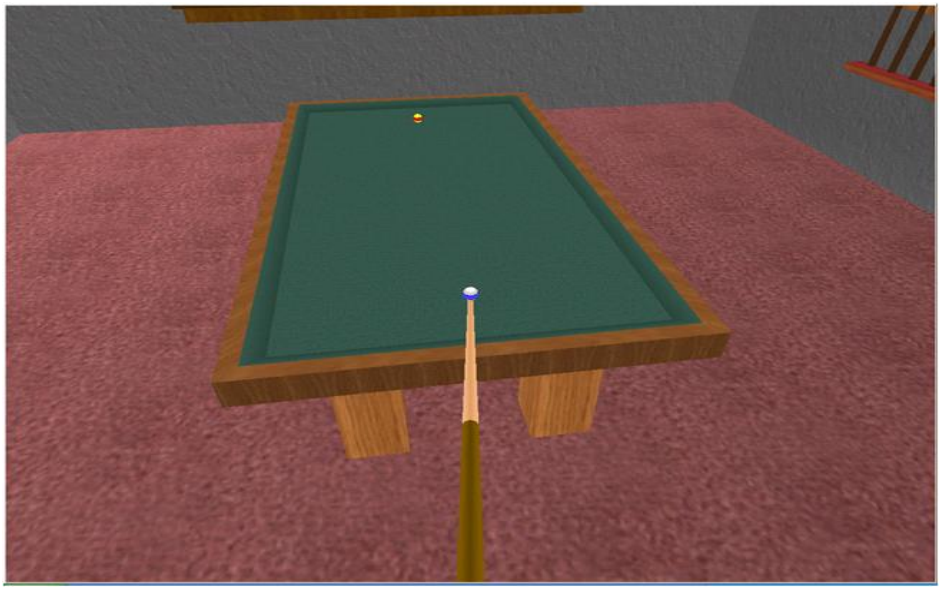

圖 1 原始桌面圖

1.1 新增 methods 使原場景(圖 1)以 table.obj 檔案格式從程式中讀取出來。

1.2 將 table.obj 用 Rhinoceros 4.0 Evaluation 匯入，再匯出成 .3ds 檔。

1.3 使用 Google SketchUp7 讀取 .3ds 檔(圖 2)，去除背景，將原球桌增加六個球袋，加厚顆星邊(圖 3)，再輸出成 .obj 檔(需選擇以全三角形構成)。

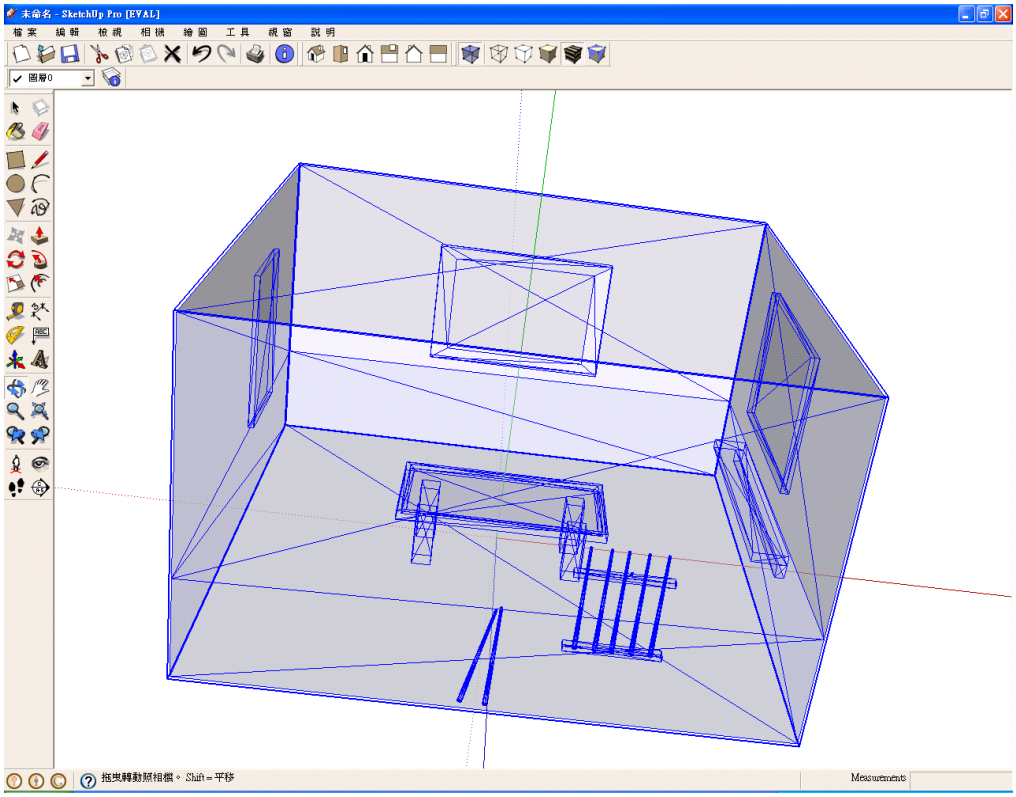

圖 2 在 Google SketchUp7 讀入原圖

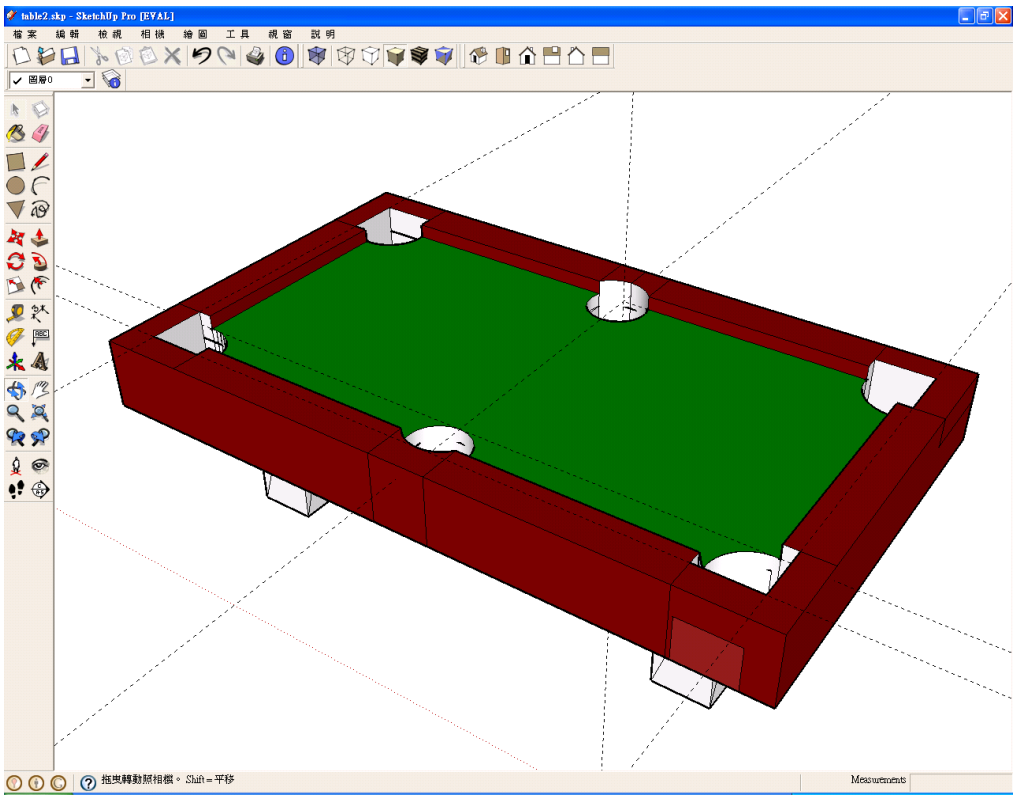

圖 3 在 Google SketchUp7 修改桌面

1.4 在 Rhinoceros 4.0 Evaluation 環境之下將新球桌與原場景之座標比對且校正。

(如圖 4)

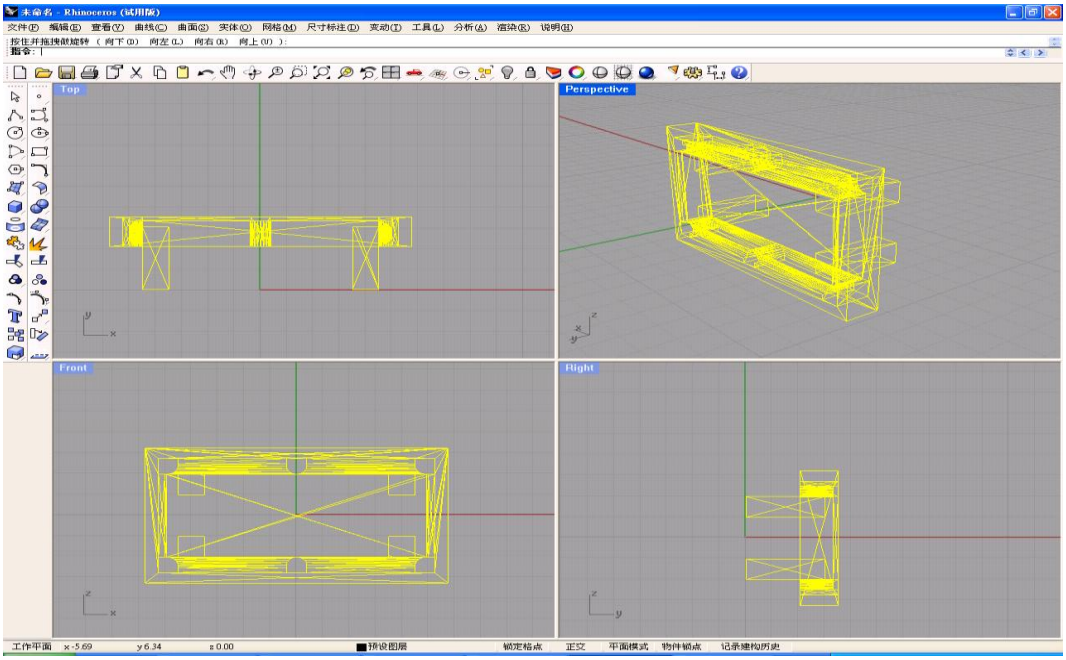

圖 4 在 Rhinoceros 4.0 Evaluation 校正座標

1.5 在原程式中新增數個可讀取 .obj 檔案格式之 methods。(讀入後如圖 5)

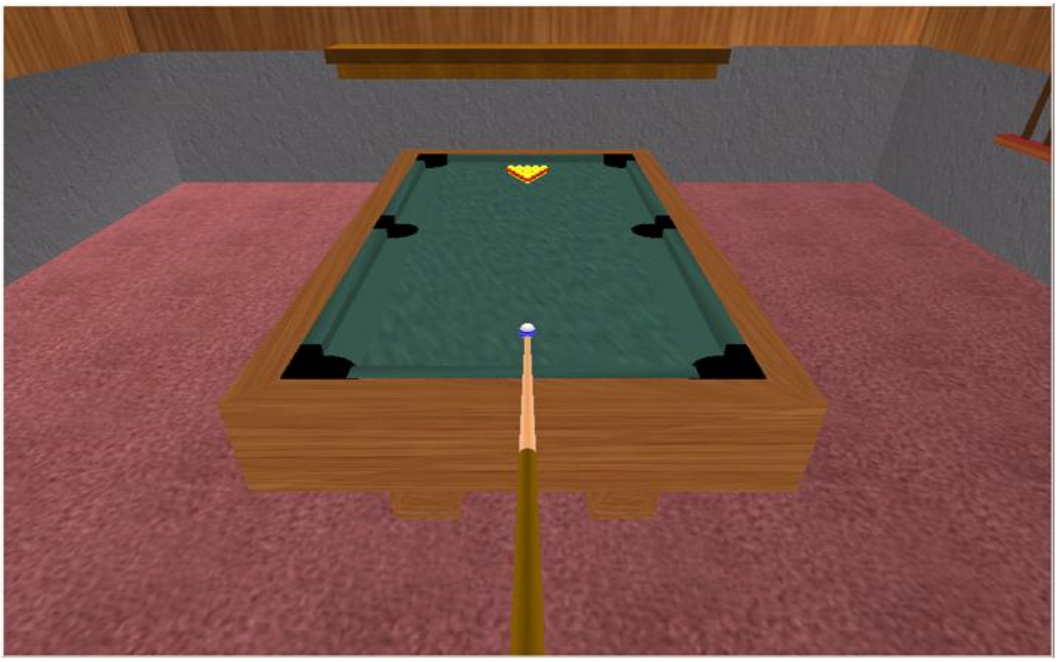

圖 5 修改後的新桌面

2.容許角度區域

如圖 6. **R**、**r** 分別表示袋口半徑與球半徑，L，l 分別表示子球到袋口距離與母球到子球的距離，a、b 表示最正確之擊球角度與容許之誤差角度，c、d 為 a、b 之不同觀點。當選定子球攻擊時我們計算容許值 d 並挑選所有子球中 d 值最大的攻擊。

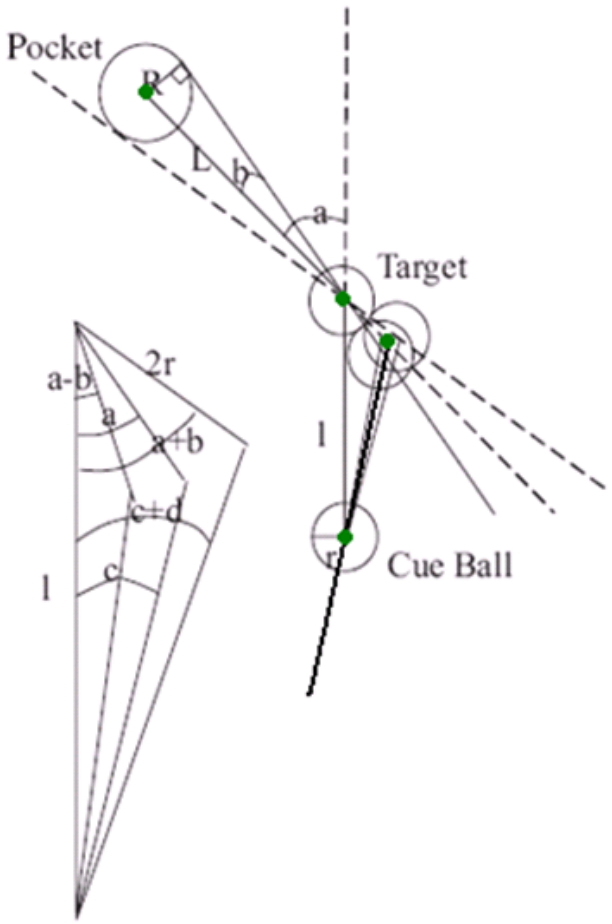

圖 6 擊球路徑演算法示意圖

2.1 求出六個球袋中最易打進之球袋，d值越大表示擊球入袋的機率越高。

$$b = a\sin(R/L),$$

$$d = a\sin\left\{\frac{2r\sin(a+b)}{[4r^2+l^2-4rl\cos(a+b)]^{1/2}}\right\} - c,$$

2.2 母球撞擊子球之最佳角度。

$$c = a\sin\left[\frac{2r\sin a}{(4r^2+l^2-4rl\cos a)^{1/2}}\right].$$

2.3 移動球桿至最佳擊球位置。

$$\overrightarrow{b1b0'} = \overrightarrow{pb1} * \frac{2r}{L}$$

$$b0' = b1 + \overrightarrow{b1b0'}$$

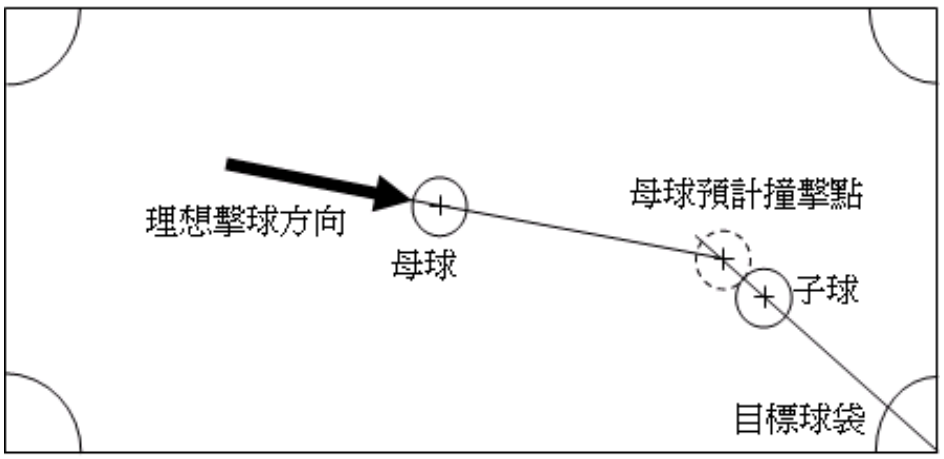

圖 7 最佳擊球角度示意圖

四、程式碼修改步驟

1.匯出：

在 RENDERWORLD.cpp 檔案中，新增 void GetObjFile(char *filename)，可將 g_Scene 中資料輸出成.obj 檔。

2.匯入：

在 RENDERWORLD.cpp 檔案中，新增一個 LoadObjFile() 用來讀取 .obj 檔中所有 vertex、tri、quard 的值。

新增 AddSceneFile()，將讀入的.obj 檔修改相關的 vertex 索引值後加進 g_Scene 中，與原場景做結合。

將新增球袋後的桌子放回原場景中：

felt_6hole_v3: 桌面上的桌布。

felt_6hole_v35: 延伸至桌邊的桌布。

felt_6hole_*_v3: 分成 a，b，c 3 個部份，可給予不同的顏色增加立體感。

修改 RENDERWORLD.cpp 中 RenderScene()部分，可將原本的桌子及桌布隱藏，且能保留其他場景。

3. 模擬球入袋情形

修改 GameSim.cpp 檔案中的 CheckForCollisions()、ResolveCollisions()、Simulate()，RENDERWORLD.cpp 檔案中 RenderCueAndBalls()。在 externs.h 的

enumtCollisionTypes 中新增 INPOCKET 狀態。可呈現出當球移動或者與牆產生碰撞時，若在 6 個洞的座標內，將球的速度 v=0、子球 y 軸座標設為 0（球掉到地上）、母球移回初始位置。

4. 增加球數成為完整遊戲

子球數增為 15 顆，並設定相關的初始位置：GameSim.cpp 中，BALL_COUNT 改為 16、MAX_CONTACTS 改為 20 ，t_Ballg_Ball[BALL_COUNT]、SetupBalls() 中設定 15 顆球的初始位置。

RENDERWORD.cpp 中，新增 BALL_COUNT 常數，從 RenderCueAndBalls() 中可直接控制子球的數目

5. 設置擊球瞄準輔助桿

新增一支球桿 g_CueStick2:

在 externs.h 中新增 extern t_CueStick g_CueStick2; GameSim.cpp 中新增 t_CueStick g_CueStick2， InitGame() 設定初始位置、方向 RENDERWORD.cpp 中 RenderCueAndBalls() 產出球杆使輔助球桿可隨著計算結果改變方向，指出最佳擊球角度。

model.h 中新增 CUEMODEL_2 為輔助球桿的 model， externs.h 中新增 extern t_CueStick g_CueStick2; GameSim.cpp 中新增 t_CueStick g_CueStick2; InitGame() 設定初始位置、方向 RENDERWORD.cpp 中

RenderCueAndBalls() 產出球桿， VIEW.cpp 中修改 LRESULT CALLBACK WndProcView() 的 case WM_LBUTTONDOWN:，使球桿隨母球移動。

6. 加入擊球路徑演算法

RENDERWORD.cpp 中新增 SetPocket()，設定 6 個球袋的座標及半徑；新增 GetOptAngle()，可判斷出最易擊球入袋的球袋及最佳擊球角度，與傳回輔助桿應旋轉的角度。

7.創新性

建構出一虛擬平台可驗證最佳進球路徑，設計出一瞄準用之輔助桿指引出最佳路徑，玩家將球桿拖曳至與輔助桿重疊，即可提高進球機率。

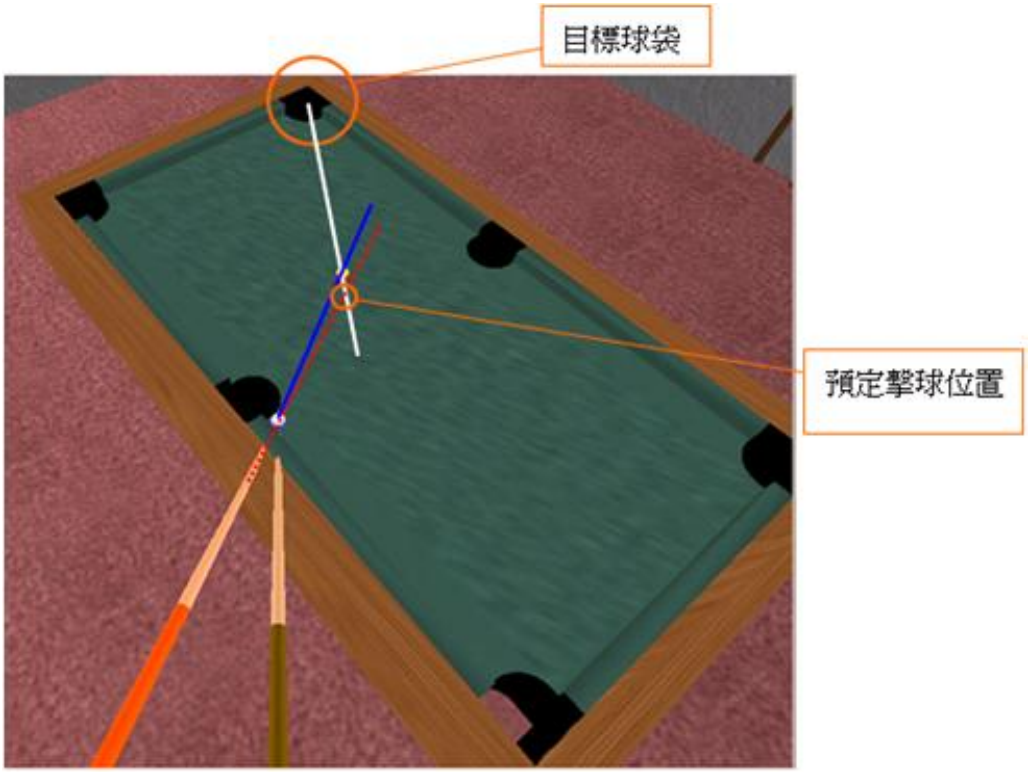

圖 8 輔助桿指引示意圖

第三章　研究方法

3.1　簡易研究流程

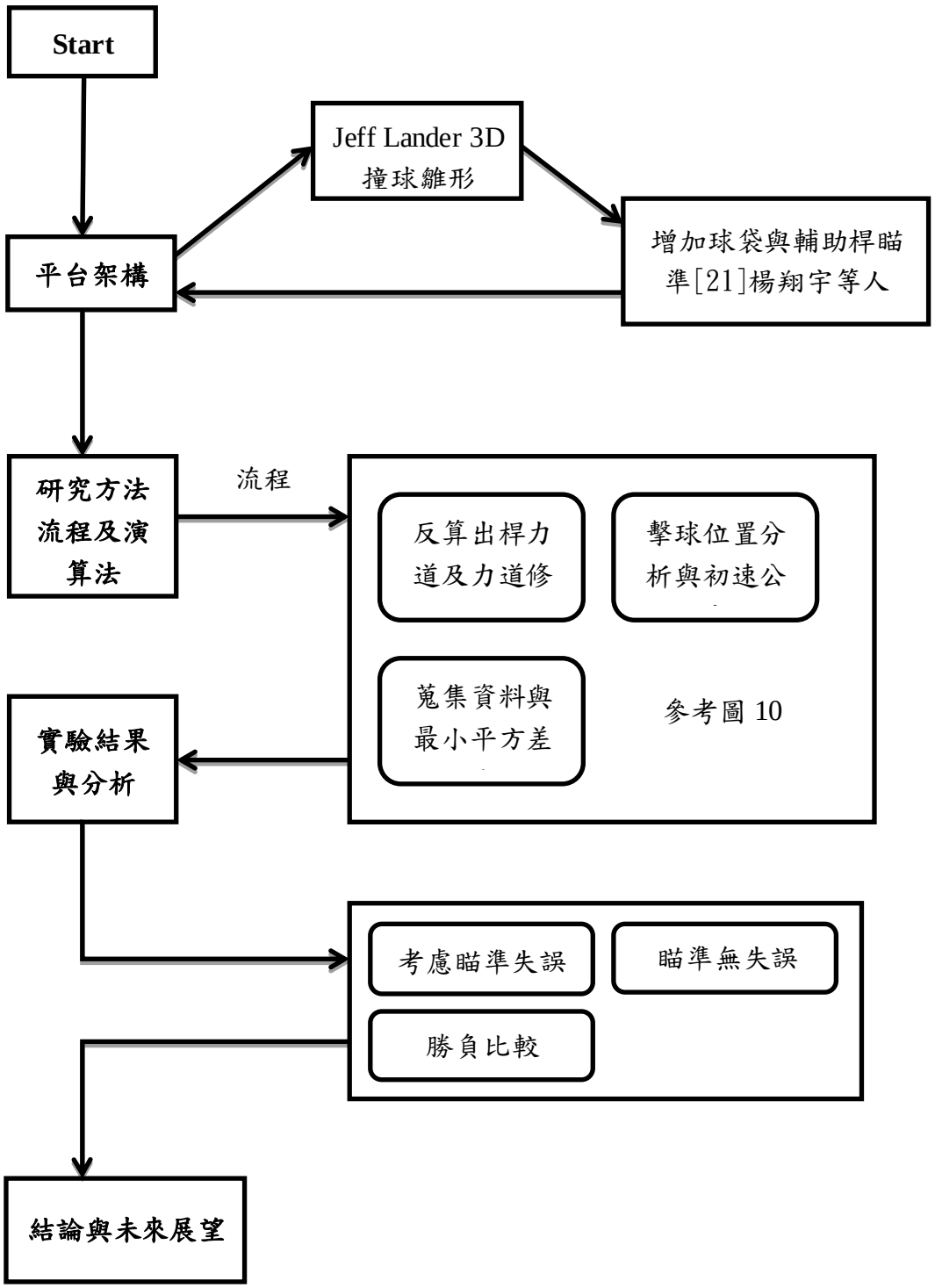

圖 9 簡易研究流程

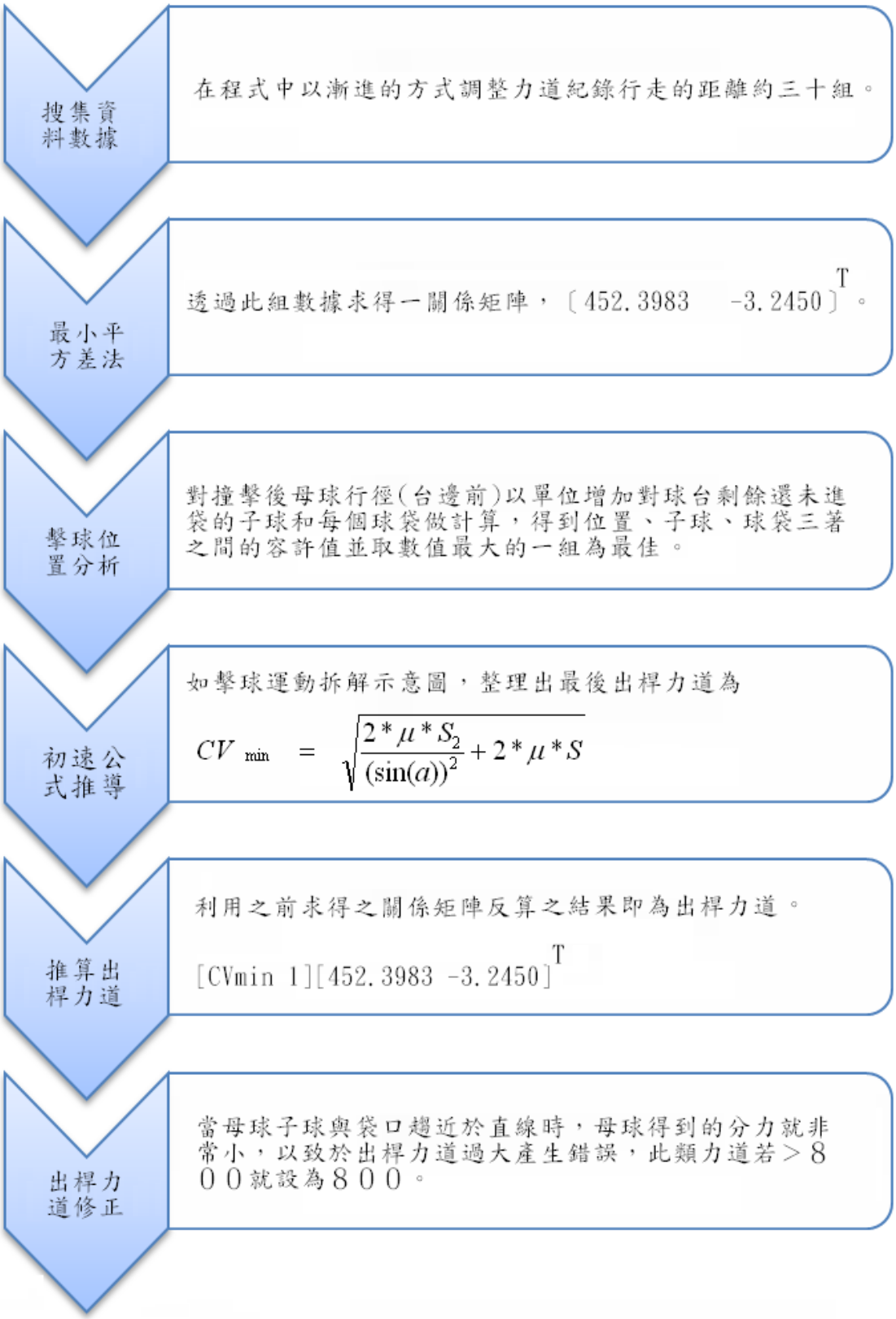
搜集資料數據
在程式中以漸進的方式調整力道紀錄行走的距離約三十組。
最小平方差法
透過此組數據求得一關係矩陣，〔452.3983 -3.2450〕T。
擊球位置分析
對撞擊後母球行徑(台邊前)以單位增加對球台剩餘還未進袋的子球和每個球袋做計算，得到位置、子球、球袋三著之間的容許值並取數值最大的一組為最佳。
初速公式推導
如擊球運動拆解示意圖，整理出最後出桿力道為
$CV_{min} = \sqrt{\frac{2*\mu*S_2}{(\sin(a))^2} + 2*\mu*S}$
推算出桿力道
利用之前求得之關係矩陣反算之結果即為出桿力道。
[CVmin 1][452.3983 -3.2450]T
出桿力道修正
當母球子球與袋口趨近於直線時，母球得到的分力就非常小，以致於出桿力道過大產生錯誤，此類力道若＞800就設為800。

圖 10 研究方法流程圖

3.2 最小平方差法

X: 原資料矩陣，L:轉置矩陣，Y: 對應資料矩陣，有以下的關係

$$X \bullet L = Y$$

$$\left(X^{t} \bullet X\right) \bullet L = X^{t} \bullet Y$$

$$\left(X^{t} \bullet X\right)^{-1} \bullet \left(X^{t} \bullet X\right) \bullet L = \left(X^{t} \bullet X\right)^{-1} \left(X^{t} \bullet Y\right)$$

$$L = \left(X^{t} \bullet X\right)^{-1} \bullet \left(X^{t} \bullet Y\right)$$

首先必須要知道出桿力道與母球行走距離之關係，藉由測試各力度所得到的母球行走距離的多組數據計算後反推出一關係式，此關係式可換算此系統下力度與母球行走的距離，用來套入到控制力道演算法之數學關係式中就能夠得到更佳的擊球力道達成提高進球率之效果。在求取未知數據的部分我們利用最小平方差法，是一種數學優化技術。它通過最小化誤差的平方和尋找數據的最佳函數匹配，利用最小平方法可以簡便地求得未知的數據，並使得這些求得的數據與實際數據之間誤差的平方和為最小。以下便是我們利用此方法得到力道、行走距離之間的關係，透過此組數據求得一關係矩陣，在利用此矩陣與球距求得我們需要的出桿力道，此方法會有些許誤差但誤差非常微小可由下面的表 1.及圖 10.得知。

反算所得力量	力道 Y	初速度平方
268.5715	270	**0.361002**
258.9662	260	**0.335939**
249.3327	250	**0.311708**
239.6672	240	**0.288308**
229.5477	230	**0.264787**
220.0662	220	**0.243657**
210.3527	210	**0.222921**
200.5834	200	**0.202996**
190.8297	190	**0.184033**
180.9950	180	**0.165854**
170.5048	170	**0.147505**
160.9787	160	**0.131774**
151.1183	150	**0.116425**
141.1899	140	**0.101930**
127.2160	130	**0.083161**
121.1070	120	**0.075555**
110.2628	110	**0.062952**
100.6915	100	**0.052783**
90.6219	90	**0.043051**
80.5587	80	**0.034315**
70.3014	70	**0.026429**
60.3101	60	**0.019736**
48.4612	50	**0.013063**
39.1817	40	**0.008795**
29.2018	30	**0.005144**
19.2297	20	**0.002468**
10.1524	**10**	**0.000877**

表 1 最小平方差數據資料

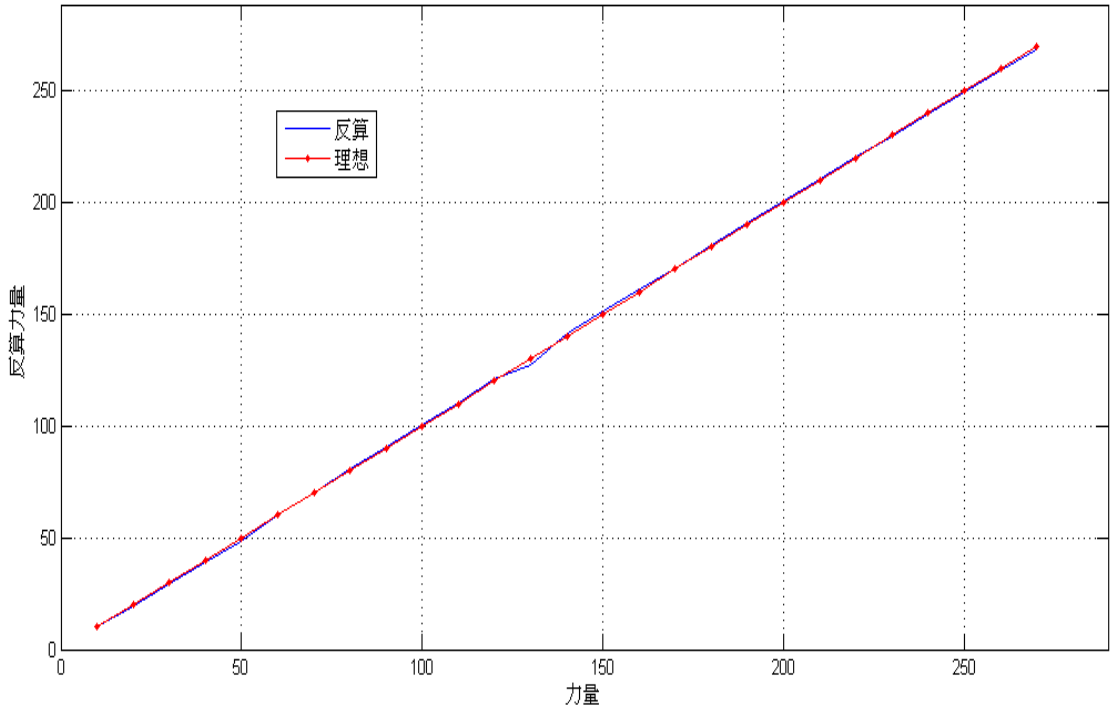

圖 11 最小平方差結果對照圖

此一組數據所得到的關係矩陣為 L = [452.3983 -3.2450]^T，平均誤差值 0.5503%。往後實驗力道部分便是由此矩陣計算而得。

3.3.1 演算法假設

1.演算法計算路線不考慮與其他子球碰撞。

2.下面 S_2 方向蒐集之母球行進路線上的點僅止於與台邊碰撞之前。

3.碰撞反彈均假設為理想(碰撞分離角度為 90 度)。

4.物理及數學計算由內部 library 負責。

5.此系統各個變數單位與現實無關為系統下之相對數值。

3.3.2 力道計算演算法

此章節將說明我們使用何種方法來完成力道計算演算法，其中包括物理碰撞力的拆解以及數學函數計算和向量分析。由圖，白色為母球(mother ball)，綠色為子球或稱目標球(target ball)，和球桌相同顏色的球為假想球（母球擊中此位置方能使子球進入 S_1方向的球袋）。S_2為擊球後母球行進方向（理論定義與 S_1垂直），我們目的就是在此行進方向上找出一個位置使得下一球變得更容易進攻。判斷的準則採用容許值，因為容許值若越大代表允許得瞄準偏差越大，我們以這為一個標準，瞄準偏差越大能夠吸收人為瞄準失誤範圍越廣，若是失誤夠小仍能夠進球，所以我們視為越容易打擊。

Step 1:

此方法優先考慮要把子球打進袋口，換句話說母球勢必先在 S_2方向上行走一段距離後停止，此延伸方向至球台邊（S_2方向）會分成一百等分，每一單位為基礎對球檯上剩餘還未進袋的子球和每個球袋做計算，得到位置、子球、球袋三著之間的容許值並取數值最大的一組為最佳，如此就能保證下一球是球檯上最容易進攻的母球位置、子球與代口之組合。

Step 2:

此位置必定在 S_2 方向上，如最佳位置示意圖之Ｐ點，利用假想球與此位置之距離帶入計算公式，便可以計算出擊球力道使母球停在此位置上，利於下一球

之打擊，此部份將在下節細部說明。

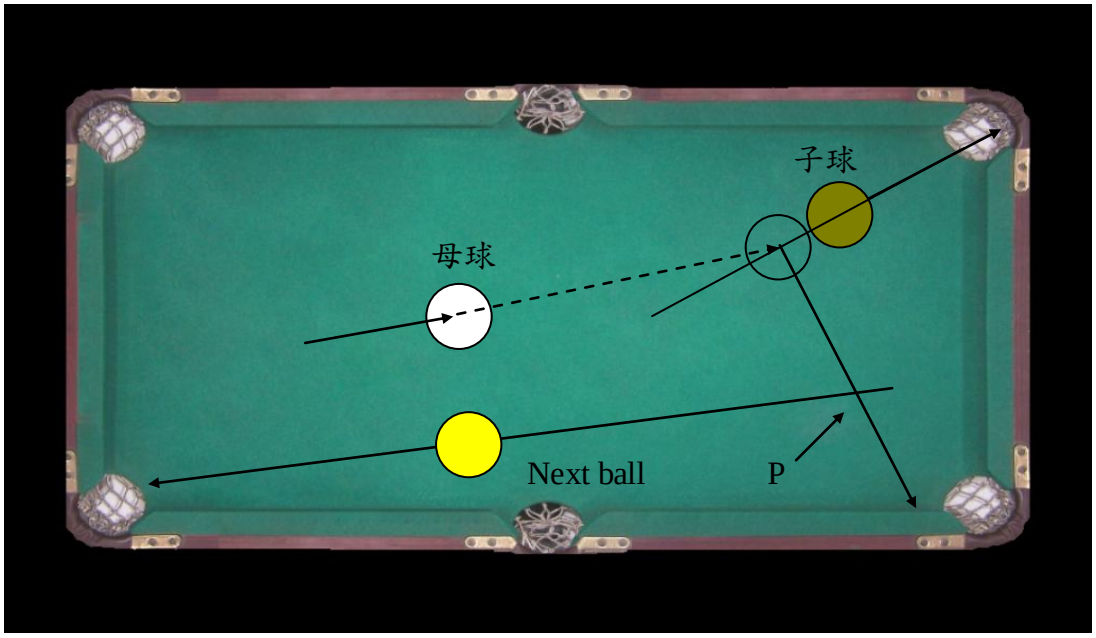

圖 12 擊球位置分析示意圖

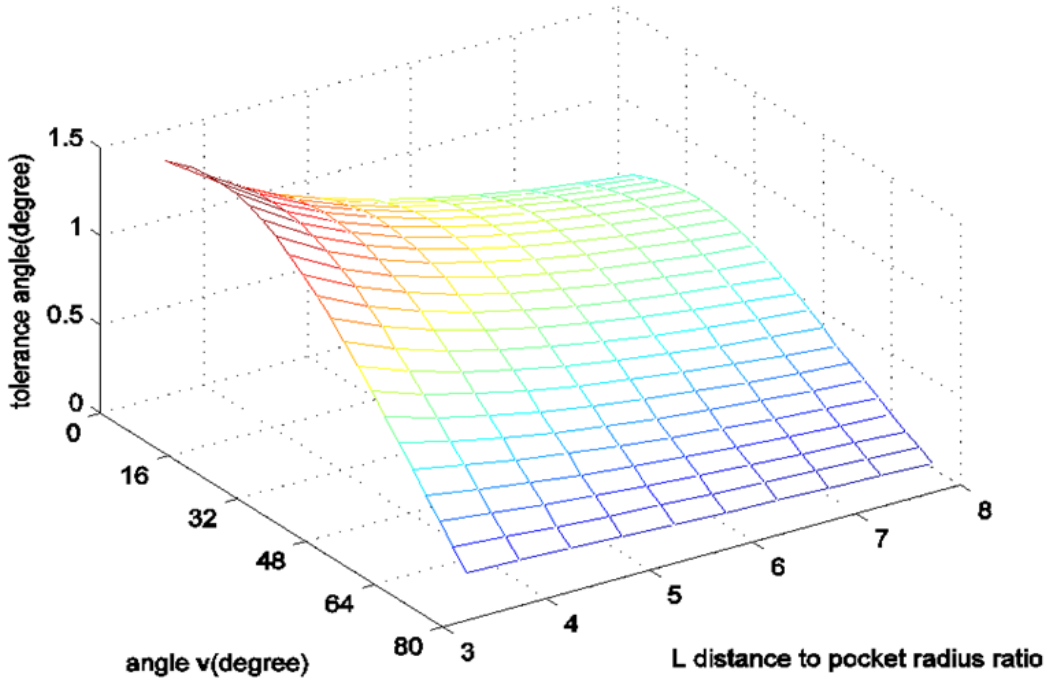

圖 13 角度變化與距離遠近之容許值變化圖

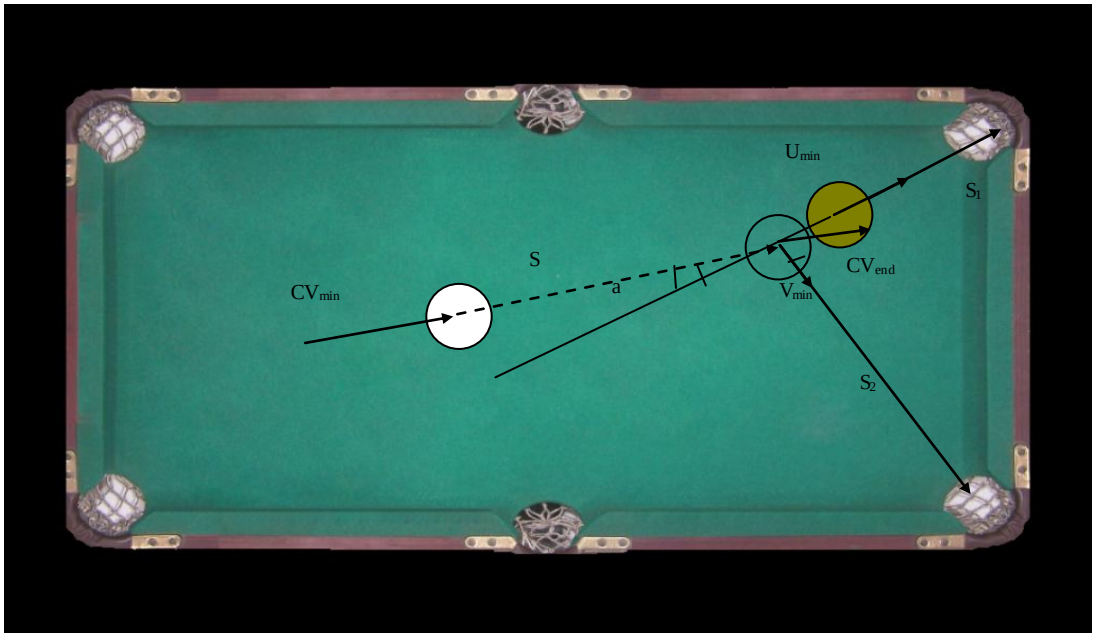

圖 14 擊球位置分析圖

3.4 演算法之數學說明與計算流程

3.2 說明了力道計算演算法之方法與大體精髓。此節將更利用數學公式細部說明其意義與計算步驟。公式如以下所示，公式(1)中，v 為終端速度 Umin 為初速，μ 和 S 分別為球桌之摩差係數和子球行走距離。公式(2)與(1)相同，參考上圖可推得公式(3)。在 S_2 方向上之分力為母球擊球後之行走方向，若母球沒有洗袋犯規其終端速度為零，由公式(1)帶入末速度為零、初速度 Vmin 和行走距離 S_2 可得到公式(4)，整理可得公式(5)，再將(3)帶入(5)整理後便可得到(6)。Umin 可表示成 CVmin 乘以 cos(a)再帶入(3)整理就可以得到公式(9)。利用(1)可得到公式(7)，(4)、(7)and(9)可以得到公式(10)。

$$\upsilon^2 = U^2_{\min} - 2*\mu*S \quad \text{(1)}$$

$$U_{\min} = \sqrt{2*\mu*S_1} \quad \text{(2)}$$

$$V_{\min} = U_{\min} * \tan(a) \quad \text{(3)}$$

$$0 = V^2_{\min} - 2*\mu*S_2 \quad \text{(4)}$$

$$S_2 = \frac{V^2_{\min}}{2*\mu} \quad \text{(5)}$$

$$S_2 = \frac{(U_{\min} * \tan(a))^2}{2*\mu} \quad \text{(6)}$$

$$CV^2_{end} = CV^2_{\min} - 2*\mu*S \quad \text{(7)}$$

$$CV^2_{\min} = CV^2_{end} + 2*\mu*S \quad \text{(8)}$$

$$CV_{end} = \frac{V_{\min}}{\sin(a)} \quad \text{(9)}$$

$$CV^2_{\min} = \frac{2*\mu*S_2}{(\sin(a))^2} + 2*\mu*S \quad \text{(10)}$$

$$CV_{\min} = \sqrt{\frac{2*\mu*S_2}{(\sin(a))^2} + 2*\mu*S} \quad \text{(11)}$$

利用公式(10)，可由演算法的前段在 S_2 方向上得到一個最有利於攻擊下一球的位置，此位置與假想球中心之距離即為 S_2 長度值，且 S、sin(a)和 μ 的數值均為已知或可事先求得，最後對 CV^2min 開根號方可得到出桿擊球之力道，此力道可使母球將子球及入袋口並且停留在 S_2 方向上最有利進攻下一球的位置上（角度越小），如此可有效提升進球率。

3.5 擊球運動及時間複雜度說明

我們利用理想中的物理碰撞在母球與一個選擇中的子球碰撞以後預言母球的可能停留的位置。目標是擊中那個被選擇的子球進入一個最近的球袋。因為最近的球袋比起其他球袋更容易瞄準而且進球率高這是非常自然的選擇。這裡我們關心母球擊球的初速(CVmin)是否為最佳的。要達到這樣目標，首先要考慮的是球杆必須要以預定的初速擊出母球使母球擊中一個被選中的子球使子球入袋。給出這樣初速，子球剛好能夠進入球袋，並且母球將移動到固定位置為止。為了使母球在下次擊球時停在最好擊球的位置，如最佳位置示意圖 P 點，預定的初速能大於或小於剛好可以將子球送入袋口的力道，但我們必須保證力道大於等於該力道，這初速取決於子球到球袋和子球到母球的距離。

愈得知 CVmin，如圖 13.，我們將真實數據帶入 3.3 之公式，最後由公式(11)可得到 CVmin，其中 S_2 長度需要特別計算，將 S_2 方向上的位置分割為三十等分每個位置對球桌上的每顆子球做計算，其計算內容為容許值大小如同我們在選擇要打哪顆子球一樣，將所有位置對所有子球計算取出容許值最大之位置及子球組合，作為下一次打擊的組合。此位置與假想球中心之長度訂為 S_2，就可以利用(11)計算出 CVmin，以此力道出桿打擊即可將子球打進球袋並且母球撞擊子球後利用分離後的力道移動至預定的位置上，以利於下一球打擊。

過程中擊球位置分析，母球反彈後的路線上至台邊距離 L，每隔球直徑的十分之一蒐集一次位置，最後這些位置都會與球袋、子球計算容許值。我們假設球

直徑為 R、球袋及子球個數分別為 m、n，蒐集的位置有 10L/R 個，設為 k，m 為常數，每找到一個最佳位置我們要做 m* n* k 次運算，所以我們計算的複雜度為 O(nk)。

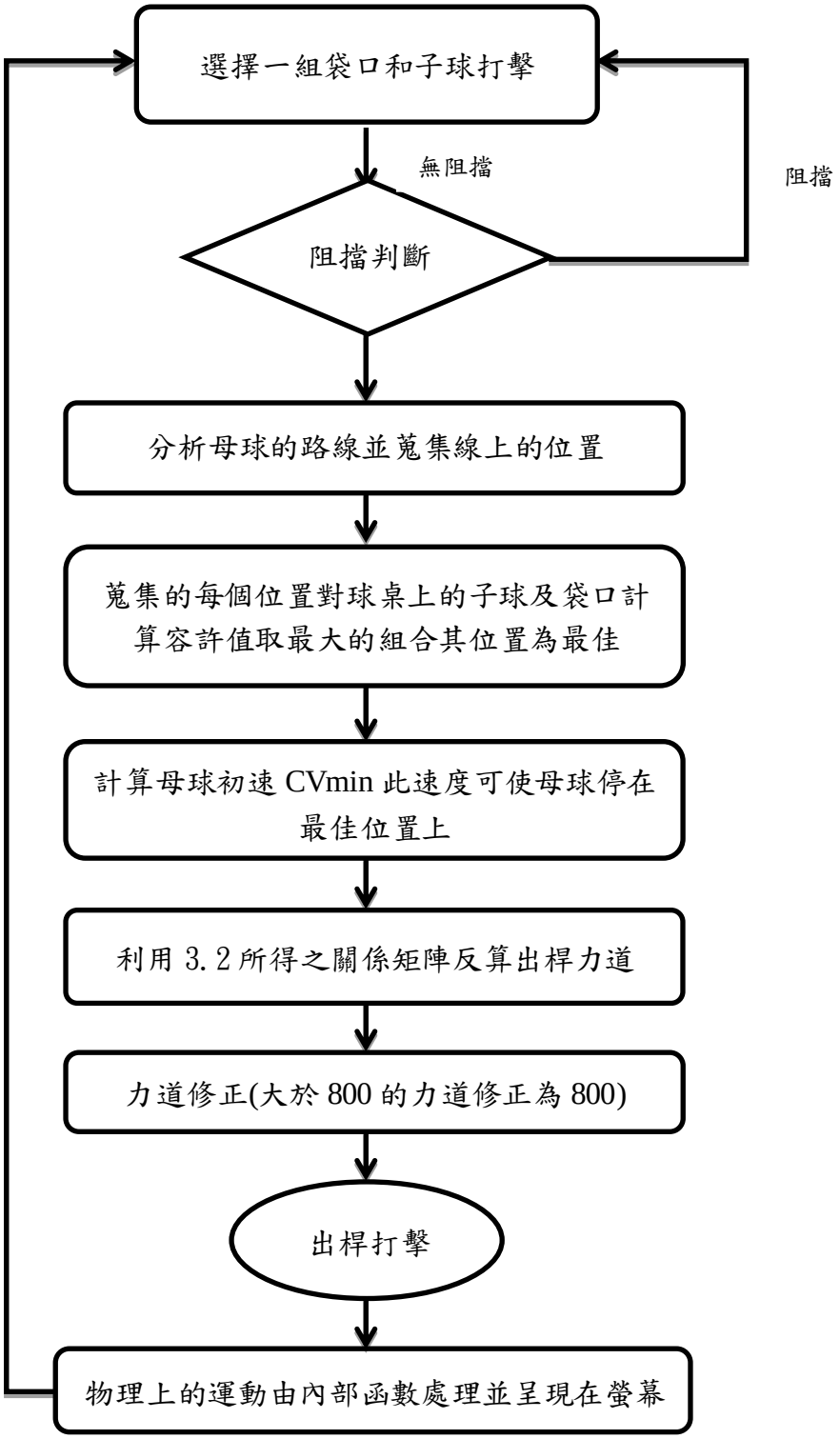

圖 15 演算法流程圖

3.6 阻擋情況分析

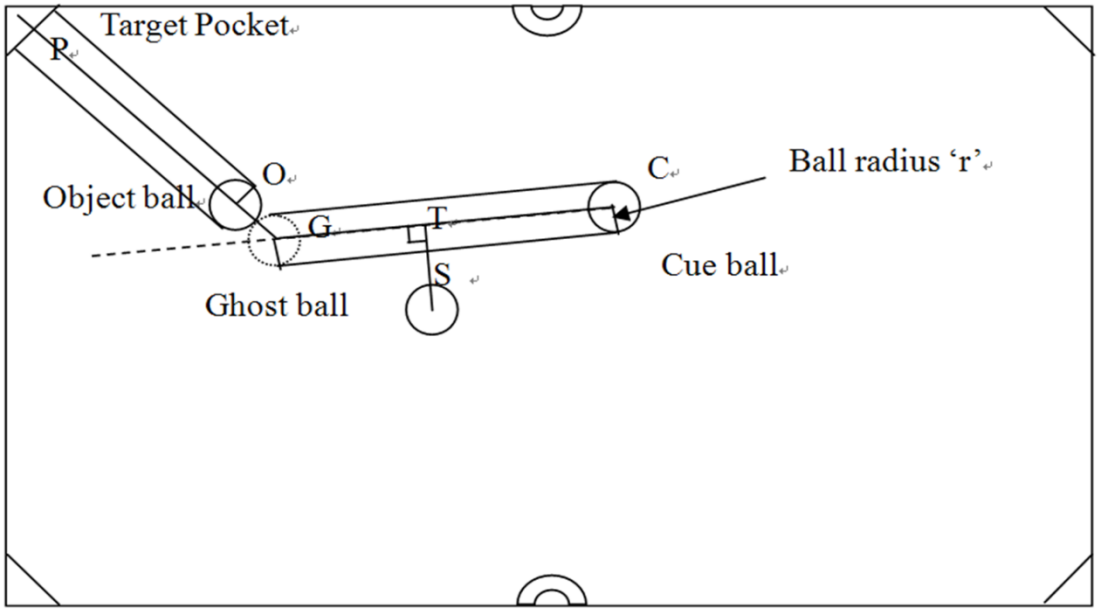

圖 16 阻擋球判斷示意圖

圖 15.中 C(母球)、O(子球)和 P(袋口)為一種擊球組合，視 CG 路徑為中心線距離中心兩側 r 之範圍內不可有其他障礙球，其中 G 為假想球、r 為球之半徑。OP 路徑同理。

我們求得與中心線垂直之向量，判斷與中心線平行距離 2r 的範圍內有無其他阻擋球（判斷球心），若兩個區間內均無阻檔即可視為一可行的擊球組合。

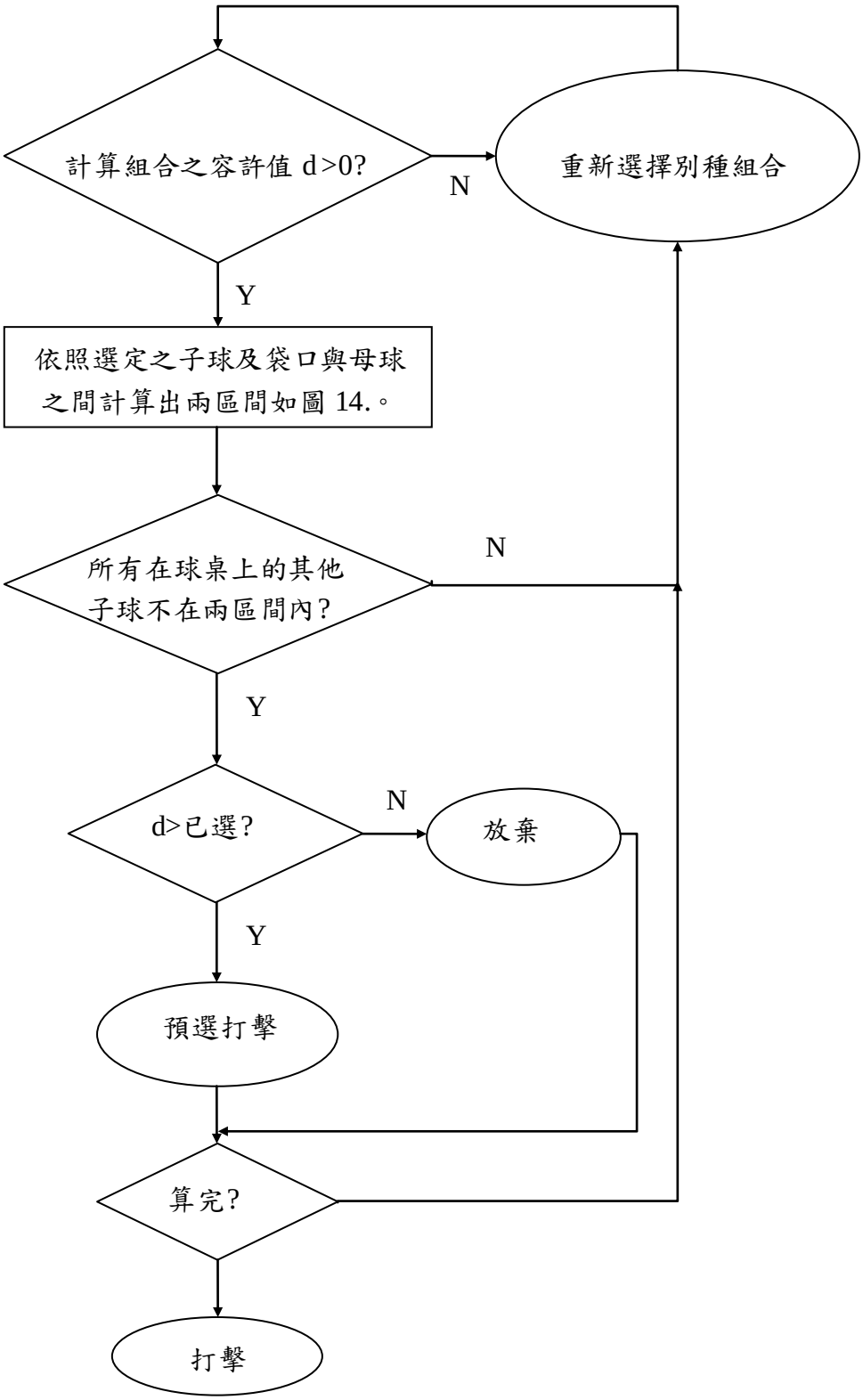
計算組合之容許值 d>0?
N
重新選擇別種組合
Y
依照選定之子球及袋口與母球之間計算出兩區間如圖 14.。
所有在球桌上的其他子球不在兩區間內?
N
Y
d>已選?
N
放棄
Y
預選打擊
算完?
打擊

圖 17 阻擋判斷流程圖

第四章實驗結果與分析

4.1 不考慮出桿誤差角度

此章節就原程式;固定以單一力道出桿擊球，與加入擊球力道演算法之程式;依照母球、子球和球袋之間的關係計算不同的出桿力道做比較如圖 16。固定力道 AI 分別在力道 300、500 和 700 時的清台桿數比較圖，當力道越大我們大多可以用較少的桿數打完一局，原因在於當力道太小時可能會有許多距離較遠的球會因為力道不足導致球沒進且將球敲散的速度也較慢，而越大的力道能夠更快的將集中在球桌上不能單獨進攻的子球敲散以利進攻也不會出現力道不足而沒進球的情況。

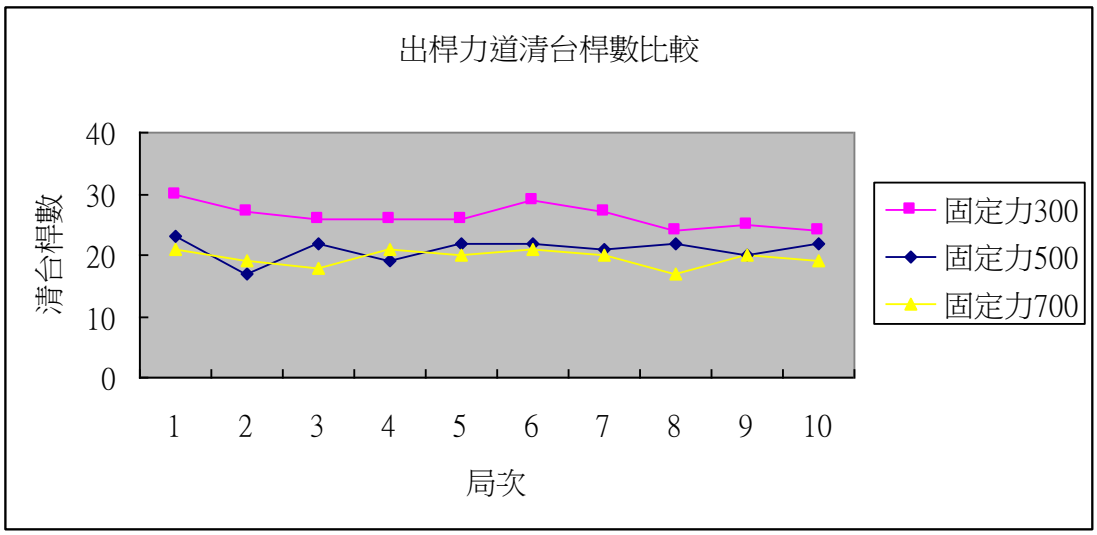

圖 18 固定力出桿力道與清檯桿數比較圖

	固定力 300	固定力 500	固定力 700
出桿平均	26.4	21	19.6

表 2 固定力清台桿數平均

但力道越大也會產生較高的洗袋犯規次數，圖 17.可以看出在單局中力道越大有較多洗袋次數的趨勢。當力道 700 時單局洗袋次數介於零至十二之間，力道 500 時單局洗袋次數介於二至六之間，力道 300 時單局洗袋次數皆於零至三之間。

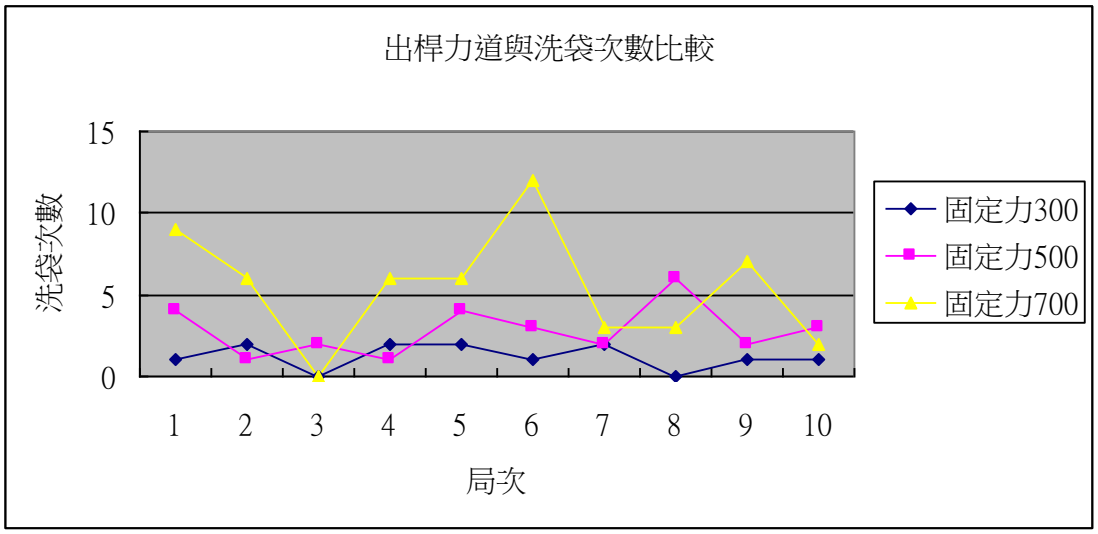

圖 19 固定力出桿力道與洗袋次數關係圖

	固定力 300	固定力 500	固定力 700
洗袋平均	1.2	2.8	5.4

表 3 固定力洗袋次數平均

圖 18.中控制力道的清檯桿數較為中庸表現上沒有非常突出，此結果也可說明控制力道出桿力道經常介於 300 至 500 之間。圖 19.為加入了控制力道 AI 的洗袋次數表現，可以看出來控制力道 AI 的表現與固定力道 300 差不多，單局洗袋次數介於零至四之間，以洗袋次數來說表現相當不錯。如此一來控制力道 AI 具備了較低的洗袋次數，並且能夠依照遠近不同的球來調整出桿的力道，不會有力道不足而沒進球的情況。

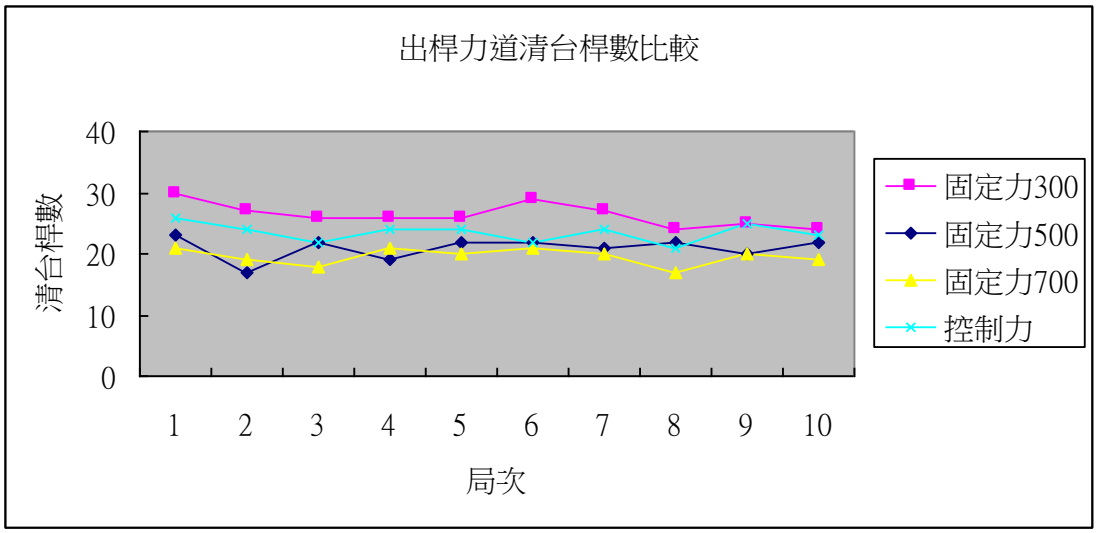

圖 20 固定力 VS 控制力清檯桿數比較圖

	固定力 300	固定力 500	固定力 700	控制力
出桿平均	26.4	21	19.6	23.5

表 4 平均桿數固定力 V.S.控制力

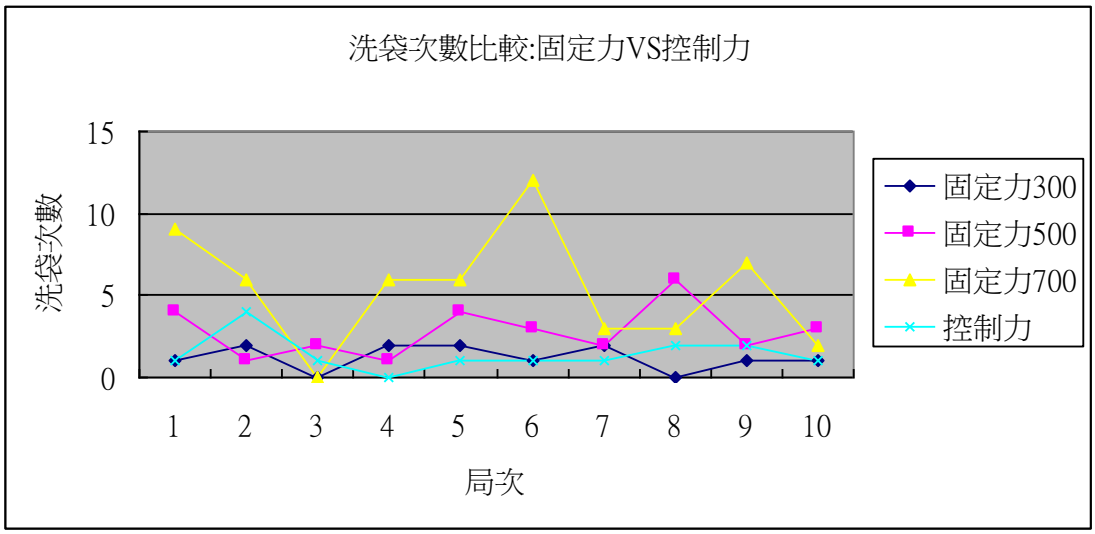

圖 21 固定力 VS 控制力洗袋次數比較圖

	固定力 300	固定力 500	固定力 700	控制力
洗袋平均	1.2	2.8	5.4	1.8

表 5 平均洗袋次數固定力 V.S.控制力

4.1.1 開球優化

開球在一局球中有著重大的影響，所以以不同的力道開球來比較每局的清台桿數，有圖 20.的結果可以看出在大部分的情況用較大的力量開球，將球撞擊的更為散開有利於打擊有更高的機率在較低的桿數下清台。

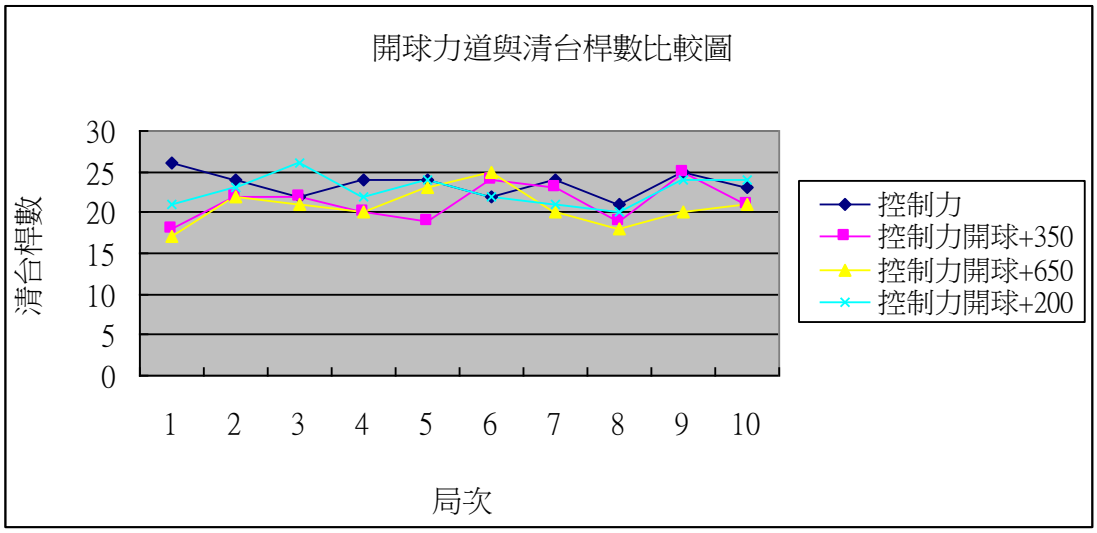

圖 22 控制力不同開球力道清檯桿數比較圖

	控制計算	外加 200	外加 350	外加 650
平均出桿	23.5	22.7	21.3	20.7

表 6 不同開球力道平均清台桿數比較表

圖 21.可以看出控制力道在清台桿數上比固定力道 700 來得多桿，但若加入了開球用較大的力道，可以看出清台桿數比控制力道來的低，有接近於固定力道 700 的現象。

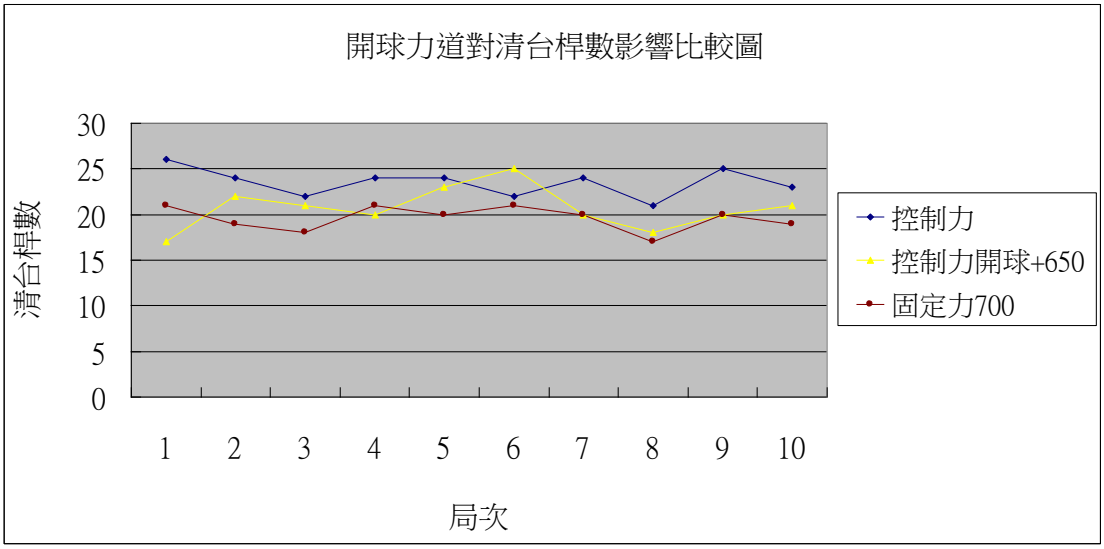

圖 23 開球力道對清台桿數影響比較圖

	控制計算	固定力 700	控制力加 650
平均出桿	23.5	19.6	20.7

表 7 平均出桿次數控制力 V.S.固定力 V.S.控制力外加 650

設固定力開球模式與"控制力開球+650"一致以示公平，所得到的結果如下圖其中可以看出清台桿數上的表現在加入開球行為後較沒有開球行為之控制力接近清台桿數表現最佳的固定力 700。

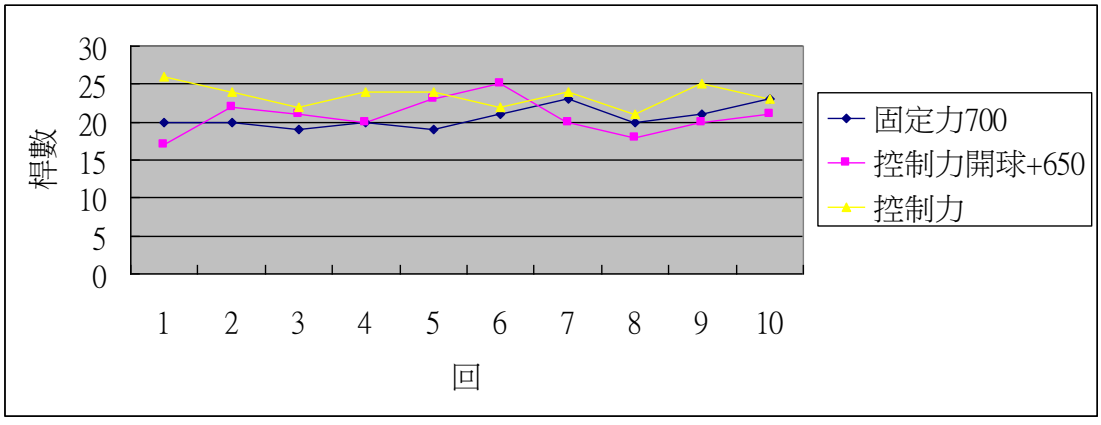

圖 24 開球模式一致桿數比較圖

	控制計算	固定力 700	控制力加 650
平均出桿	23.5	20.6	20.7

表 8 平均出桿次數控制力 V.S.固定力 V.S.控制力外加 650

黃色代表的控制力，具備固定力道 700 較低的清台桿數與固定力 300 的低洗帶次數。

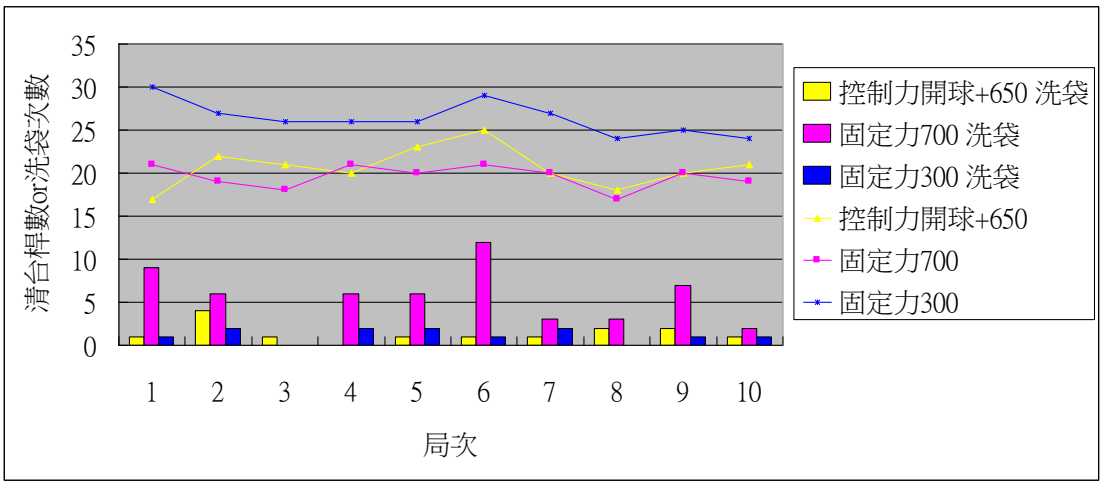

圖 25 固定力 VS 控制力比較表

	控制+開球 650	固 700	固 300	固 500	純控制力	控制+開球 350
平均出桿	20.7	19.6	26.4	21	24	**21.3**
平均洗袋	**1.4**	**5.4**	**1.2**	**2.8**	**1.8**	**1.2**

表 9 無失誤平均數據圖

4.2 考慮出桿角度失誤在正負二度之間

由於撞球實際情況包含擊球失誤，而失誤的原因可能很多但絕大多數來自於出桿不穩定，造成擊球線偏移。或者是球員本身判斷失誤，決定的擊球點一開始

就是錯誤的，而這種失誤造成的結果和前者一般，就是擊球線均不在正確的路線上。所以我們在失誤模擬上均採用亂數產生失誤角度，並不是所有失誤都會造成球無法入袋，而當失誤的角度在容許範圍之內時這次擊球仍然能夠入袋。由表10.可以看出考慮失誤時，在平均出桿次數上較無失誤的情況來的多而平均洗袋次數微幅上升的原因可能是失誤造成出桿次數上升中間伴隨洗袋情況發生。

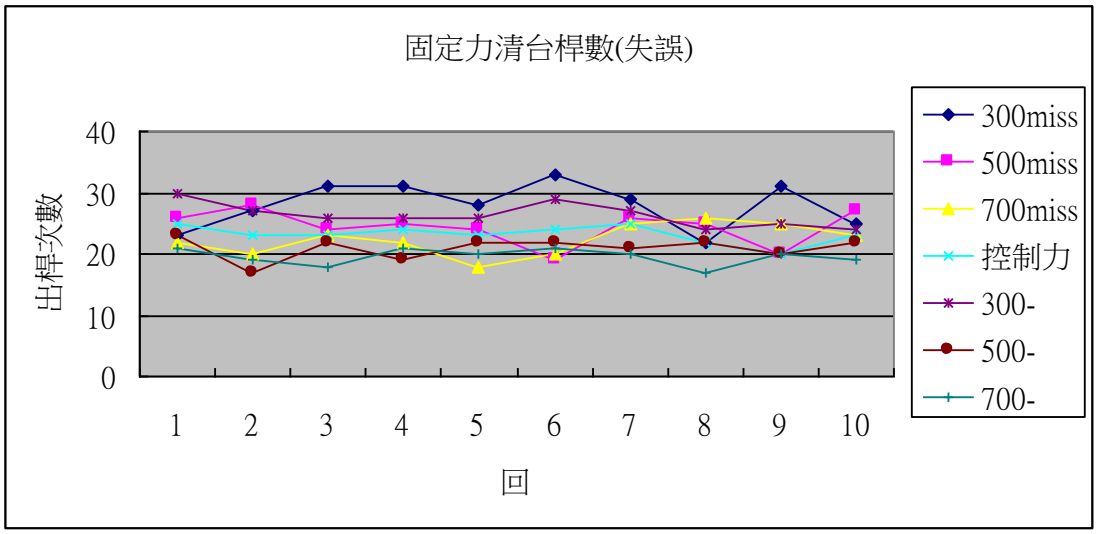

圖 26 考慮失誤清檯桿數總表

	力 300	力 500	力 700	控制力	300miss	500miss	700miss
平均出桿	26.4	21	19.6	23.2	28	24.4	22.4

表 10 平均出桿次數比較表

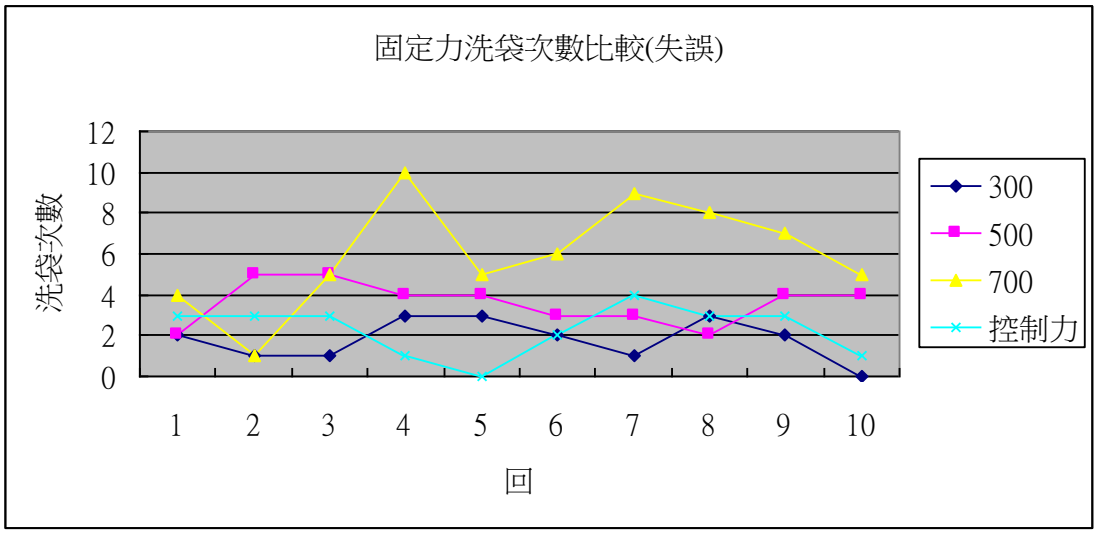

圖 27 考慮失誤洗袋次數表

	力 300	力 500	力 700	控制力	300miss	500miss	700miss
平均洗袋	1.2	2.8	5.4	2.3	1.8	3.6	6

表 11 平均洗袋次數比較表

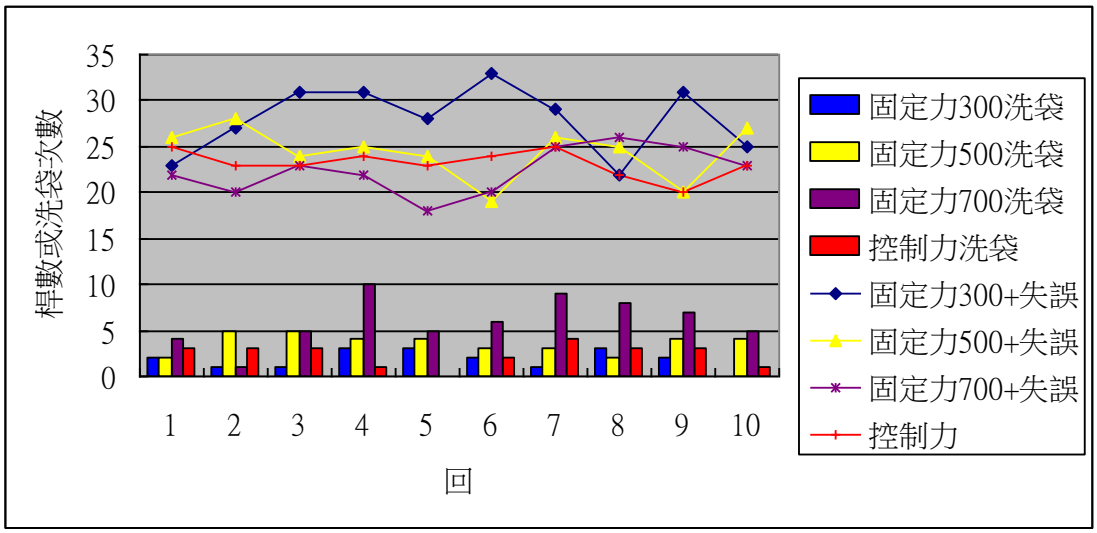

圖 28 考慮失誤洗袋次數與清檯桿數總表

	力 300	力 500	力 700	控制力	300miss	500miss	700miss
平均出桿	26.4	21	19.6	23.2	28	24.4	**22.4**
平均洗袋	**1.2**	**2.8**	**5.4**	**2.3**	**1.8**	**3.6**	**6**

表 12 考慮失誤的平均出桿與洗袋次數

4.3 不同 AI 換手間的連桿情況及平均連桿數

控制力	2	0	7		4	**3**	3	0		2	2	3	**2**	1	0		1	2	1	**1**
固定力 350	0	0	0		2	**1**	3	0		1	1		**1**	0	0		2	8		**3**
控制力	1	0	0		2	1	1	**1**		1	0	2	4	1	**2**		3	0	**1.5**	
固定力 350	0	1	0		2	4	3	**2**		0	0	1	2	4	**1**		0	12	**6**	
控制力	0	3	1	**1.33**		3	2	0	**1.67**		3	0	1	1	**1**		3	8	**5.5**	
固定力 350	3	3	5	**3.67**		3	2	5	**3.33**		2	4	4		**3**		0	4	**2**	
控制力	3	0	1		0	8	**2**	1		0	0	0	1	3	8	**1.86**				
固定力 350	0	2	1		0		**1**	0		0	1	0	0	1		**0.33**				

表 13 控制力 V.S.固定力 350 控制力先攻

	勝	負	勝率
控制力	6	6	50%
固定力 350	6	6	50%

表 14 控制力先攻勝負數據

表 13.紀錄”控制力”與”固定力 350”比賽十二局由控制力先開球打擊，開球”控制力”打進了兩球後失誤，換”固定力 350”出桿打進零顆球，第二次出桿兩者均無進球記零顆，第三次出桿”控制力”進了七顆”固定力 350”零顆，此局最後一次分別進了四顆及兩顆，紅色的數字代表此局”控制力”單次出桿進球數平均為三

顆，”固定力 350”為一顆。我們的設定先攻為不利的情況，因為開球不好掌握進球會將球衝散，表 14.可以看出控制力在不利的情況下與對手打平。

固定力 350	0	3	1	**1.3**	0	0	1	1	3	**1**	0	0	0	7	0	**1.4**								
控制力	3	3	5	**3.7**	3	5	0	2		**2.5**	0	3	4	0	1	**1.6**								
固定力 350	0	1	2	2	**1.25**	0	1	3	7	**2.75**	0	0	2	1	**0.75**									
控制力	5	0	5		**3.33**	3	0	1		**1.33**	6	0	1	5	**3**									
固定力 350	0	1	6	**2.3**	0	0	0	3	**0.75**	0	0	0	1	2	**0.6**									
控制力	3	5		**4**	0	0	12		**4**	5	0	3	4		**3**									
固定力 350	0	2	3	0	**1.25**	0	4	2	**2**	0	3	**1.5**												
控制力	3	2	1	4	**2.5**	8	1		**4.5**	10	2	**6**												

表 15 控制力 V.S.固定力 350 固定力 350 先攻

	勝	負	勝率
固定力 350	1	11	8%
控制力	11	1	92%

表 16 固定力 350 先攻勝負數據

表 15.16.為”固定力 350”先攻之數據資料，控制力在有利條件下勝率為 92%。

控制力	2	0	7	4	3.25	固定力350	0	2	3	0	**1.25**	
固定力350	0	0	0	2	0.5	控制力	3	2	1	4	**2.5**	
控制力	1	3	1	1.6666667	固定力350	0	1	3	7	**2.75**		
固定力350	2	3	5	3.3333333	控制力	3	0	1		**1.33**		
控制力	3	2	0	1.6666667	固定力350	0	0	0	7	0	**1.4**	
固定力350	3	2	5	3.3333333	控制力	0	3	4	0	1	**1.6**	
控制力	1	0	2	4	1	1.6	固定力350	0	1	2	2	**1.25**
固定力350	0	0	1	2	4	1.4	控制力	5	0	5		**3.333**
控制力	3	0	1	1	1.25	固定力350	0	0	0	1	2	**0.6**
固定力350	2	4	4		3.3333333	控制力	5	0	3	4		**3**
控制力	3	8	5.5	固定力350	0	3	1	**1**				
固定力350	0	4	2	控制力	3	3	5	**4**				
控制力	3	0	2	2	3	2						
固定力350	3	0	1	1		1.25						

表 17 輪流出桿數據資料

	勝	負	勝率
固定力 350	4	8	33%
控制力	8	4	67%

表 18 輪流出桿勝負數據

表 17.18.為”固定力 350”與”控制力”輪流開球之數據資料，可以看出條件公平一致的情況下控制力取得較高的勝率。

第五章結論

以 Jeff Lander 所製作 3D 撞球碰撞雛形增加了袋口、球數、落袋、瞄準、失誤換手、位置計算及力道計算等許多功能，使之成為更完整的撞球模型。利用此模型來模擬並探討加入擊球策略與沒有擊球策略之球局，會產生怎樣的影響。其中之擊球策略我們試著加入大力開球將子球衝散，並且導入母球之位置分析；計算下一球母球停在什麼位置利於打擊，並配合力道計算確實的將子球打進球袋也在此前提之下將母球送往進攻下一球最有利的位置上。從結果上可以看出加入這些擊球策略對賽局的影響是正面的，清台所需要的平均出桿次數下降的同時也具備較低的平均洗袋次數，這與沒有加入這些擊球策略之賽局產生的結果；球局中能夠以較低的桿數清台往往伴隨的較高的洗袋次數，反之若有著較低洗袋次數經常在清台桿數上是偏高的。比較之後加入擊球策略的確獲得較好的結果，因為規則上計分方式是要將犯規洗袋次數也計算在內，同時較低的清台桿數代表較高的平均連桿數代表平均得分較高。

未來可能的方向有將擊球位置分析的功能更加精進，修正路徑阻擋清況判斷之 bug 以增進模擬之準確與一般性，也可將容許值的選擇策略增進為兩層式，不單單考慮單一最大的組合而由兩層之合為最大的組合當成進攻組合，作更進一步的探討。

參考文獻

[1]http://zh.wikipedia.org/wiki/%E6%92%9E%E7%90%83

[2] M.Smith, “PickPocket: A computer billiards shark”, Artificial Intelligence, 171(2007) 1069-1091.

[3] L.B. Larsen, P.M. Jensen, K. Kammersgaard, L. Kromann, “The automated pool trainer—a multi modal system for learning the game of pool,” Proceeding of the International Conference on Intelligent Multimedia and Distance Education, 2001.

[4] S. C. Chua, E. K. Wong and V. C. Koo, “Pool Balls Identification and Calibration for a Pool Robot”, Proceedings of the International Conference on Robotics, Vision, Information and Signal Processing (ROVISP 2003), Penang, Malaysia, pp. 312-315, January 2003.

[5] W. Leckie, M. Greenspan, “Pool physics simulation by event prediction 1: Motion transitions”, International Computer Games Association Journal 28 (4) (2005) 214–222.

[6] W.Leckie, M. Greenspan, ”Pool physics simulation by event prediction 2: Collisions”, International Computer Games Association Journal 29 (1) (2006) 24–31.

[7] C. Shih, “Aiming strategy error analysis and verification of abilliard training system”, Knowledge Based System, Nov. 2009.

[8] B.R. Cheng, J. T. Li, J. S. Yang, “Design of the Neural-Fuzzy Compensator for a Billiard Robot”, Proceedings of the 2004 IEEE International Conference on Networking, Sensing & Control, Taipei, Taiwan, March 21-23, 2004

[9] P. Grogono, Mathematics for Snooker Simulation, personal communication, 2001.

[11] M. Jouaneh, P. Carnevale, “The development of an autonomous robotic system

for playing mini-golf," IEEE Robot. Autom. Mag. 10 (2) (2003) 56–60.

[12] K. Hashimoto, T. Noritsugu, "Modeling and control of robotic yoyo with visual feedback," Proceedings of the IEEE International Conference on Robotics and Automation Vol. 3 (1996) 2650–2655.

[13] H. Nakai, Y. Taniguchi, M. Uenohara, T. Yoshimi, H. Ogawa, F. Ozaki, J. Oaki, H. Sato, Y. Asari, K. Maeda, H. Banba, T. Okada, K. Tatsuno, E. Tanaka, O. Vamaguchi, M. Tacyhimon, "Volleyball playing robot", Proceedings of the IEEE International Conference on Robotics and Automation Vol. 2 (1998) 1083–1089.

[14] J. Hoffman, E. Malstrom, "Teachinga miniature robotic manipulator to play chess," Robotica 1 (4) (1983) 197–203.

[15] F.C.A. Groen, G.A. den Boer, A. van Inge, R. Stam, "Chess playing robot," IEEE Trans. Instrumen. Measure. 41 (6) (1992) 911–914.

[16] L. Acosta, J.J. Rodrigo, J.A. Mendez, G.N. Marichal, M. Sigut, "Ping-pong player prototype: a pc-based, low-cost, ping-pong robot," IEEE Robot. Autom. Mag. 10 (4) (2003) 44–52.

[17] C.H. Shih , P.AHsiung, C.H. Wan , C.S.Koong , T.K. Liu, Y.F. Yang, C.H. Lin, C.C. Chu "Integration of a vision-based tracking platform,visual instruction, and error analysis models for an efficient billiard training system", 2009.

[18] W.Leckie, M. Greenspan, "Pool physics simulation by event prediction 1: Motion transitions", International Computer Games Association Journal 28 (4) (2005) 214–222.

[19] W.Leckie, M. Greenspan,"Pool physics simulation by event prediction 2: Collisions", International Computer Games Association Journal 29 (1) (2006) 24–31.

[20] C.W. Chou, M.C.Tien, and J.L. Wu, "Billiards wizard: a tutoring system for broadcasting nine-ball billiards videos", IEEE International Conference on Acoustics, Speech, and Signal Processing, April, 1921-1924, 2009.

[21] 石志雄, 楊翔宇, 王景岳, 傅裕瑒, “智慧型互動式撞球訓練平台建置”, 2009.

[22] C.Y. Lin, C.Z. Lin, C.H. Shih “A low cost billiard tutoring system based on optimal cue repositioning control and planning”, 2010