中華民國第 57 屆中小學科學展覽會
作品說明書

高級中等學校組　工程學科(一)科

第二名

052318

使用 UCT 演算法建構具學習能力的對弈機器人

學校名稱：臺北市立內湖高級工業職業學校

作者：	指導老師：
職二 黃一烜	陳昭安
職二 鄧絜陽	陳永華
職二 陳皇宇	

關鍵詞：人工智慧、影像識別、機械手臂

摘要

本研究延續「使用機械手臂實現黑白棋之人機對弈」成果，改進四個部分：(一)研究使用 UCT 演算法讓機器人透過不斷對弈及學習，找到獲勝的方法；(二)使用 AI 伺服馬達改良機器手臂的穩定度及精確度；(三)使用白平衡及直方圖拓寬改善提高色光變動的容忍度；(四)將 6*6 棋盤改為正規 8*8 棋盤，實現自動取子機制。第二代機械手臂大幅改善手臂準確度，成功率從 85%提升到 100%。影像處理採直方圖拓寬有效改善環境色光變化造成的誤判。而人工智慧方面，使用遊戲樹第四層加 UCT 演算法所結合的 AI，與遊戲樹第四層對弈，在調整 UCB 公式的 C 參數後，最快可在第 9 個棋局的學習，便可持續獲勝達到 100%的勝率。整體經測試 25 賽局完整完成有 18 次，完成度達 72%。

壹、 研究動機

透過學校安排的校外教學，有幸參觀建置為工業 4.0 規格的企業「芳德鋁業」，該工廠為 IKEA 鋁製家俱的特約代工廠，大量使用機械手臂提高生產力並減少人力及人為錯誤，最重要的可以因應多樣化的智慧生產需求。運用物聯網、大數據隨時收集生產資訊，配合以視訊、機器手臂及微電腦構成的網宇實體系統 (Cyber Physical System)，讓生產流程可以隨時修正，使生產運作更加順暢，由這一個啟發加上日前資訊研究社很熱衷於人機對弈的人工智慧黑白棋遊戲，於是構想:「如果建構一套系統可以讓人與機器人或機器人與機器人對弈，分析視訊了解盤局的變化，根據變化使用微電腦運算下一步棋的決策，根據決策指揮機器手臂正確的運作落子、取棋動作、判斷輸贏及犯規等種種問題，這樣的形式似乎與工業 4.0 智慧生產觀念一致」。

圖 1 芳德鋁業使用機器手臂輔助生產實況

在「以機械手臂實現黑白棋之人機對弈」研究中機械手臂的穩定度及精度較差，希望能尋求更佳的結構及馬達，讓整體運作能夠更佳理想。而影像方面也容易受到色光改變，尋求更好的影像處理提高色光變化的容忍度。並發現就算加深了遊戲樹的層數棋力仍有上限，在本次研究嘗試 UCT 演算法進行學習以提升棋力。

為了更進一步達成工業 4.0 的規格，期望落實雲端、大數據的概念，使系統可以透過對戰不斷的提升棋力。透過資研社的訓練，成員具有 C#、PHP、Javascript、SQL Query 及 Arduino 微控板的開發能力。

貳、 研究目的

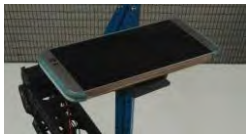
本研究延續台灣國際科展的研究成果，將進一步改良及研究的目標如下:

(一) 研究使用採用 AI 伺服馬達改良機械手臂有機構及精度不足的缺點。

(二) 研究使用白平衡演算法改良及直方圖拓寬影像處理上因色光使棋子彩度大於棋盤。

(三) 研究利用 UCT 演算法讓黑白棋機器人可以探索並學習對手的棋步以提升棋力。

參、　研究設備及器材

			
AI 伺服馬達	智慧型手機	8*8 木製棋盤	ATmega1280
		Visual C#	Microsoft SQL Server
電腦	PLC 材質機械手臂	C-Sharp	SQL Server

肆、　研究過程或方法

改良「使用機械手臂實現黑白棋之人機對弈系統」作品中，棋盤為非正規的6*6改為正規的8*8棋盤，馬達超載導致手臂不穩定出現顫抖的問題以及影像因外在光源影響導致資料異常的狀況，並且希望將人工智慧黑白棋加入學習機制達到機械學習的目的，來提高整個作品的完整性及穩定度。本研究主要分為人工智慧、影像辨識、機械手臂這三個部份分別進行，分別完成約7成後開始進行系統整合以完成成品，研究流程如下圖:

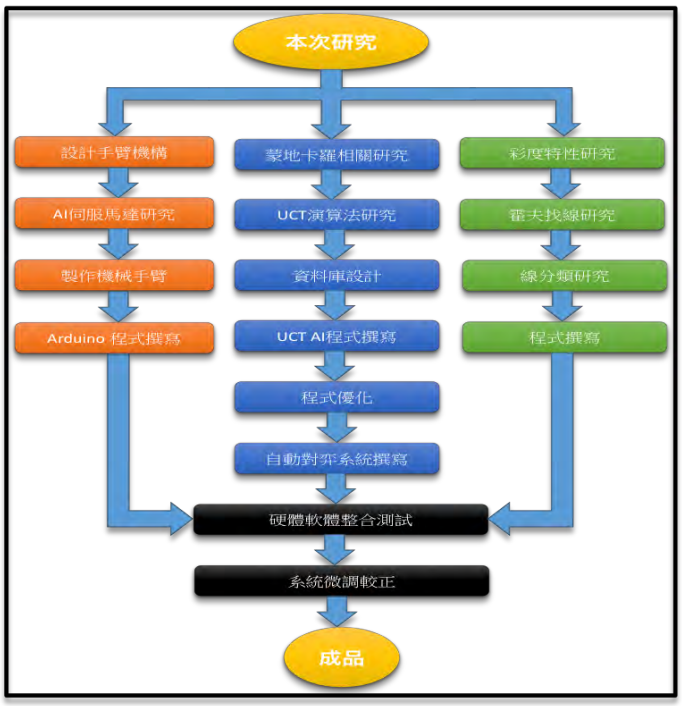


圖2 研究流程圖

一、 系統架構說明

系統架構依照硬體的功能，如圖 3 參考工業 4.0 建構網宇實體系統與雲端資料處理兩個部分。

網宇實體系統：主要是由網路攝影機、電腦控制系統與機械手臂組成一個閉迴路系統，工作平臺為黑白棋的棋局。透過電腦控制系統分析影像所得數據指揮手臂完成準確動作。符合網宇實體系統感測器和致動器裝置的整合控制系統。

一、雲端資料處理：由影像分析程式及人工智慧黑白棋程式所構成，影像分析程式主要是負責將接收到的影像處理後轉換為數據，傳送至人工智慧黑白棋程式進行運算，並依據運算完的資料進而選出最優良或最有實驗價值的位置，並且會將運算結果傳送至雲端資料庫保存，以利於下次進行搜尋時能夠增加整體的思考深度。

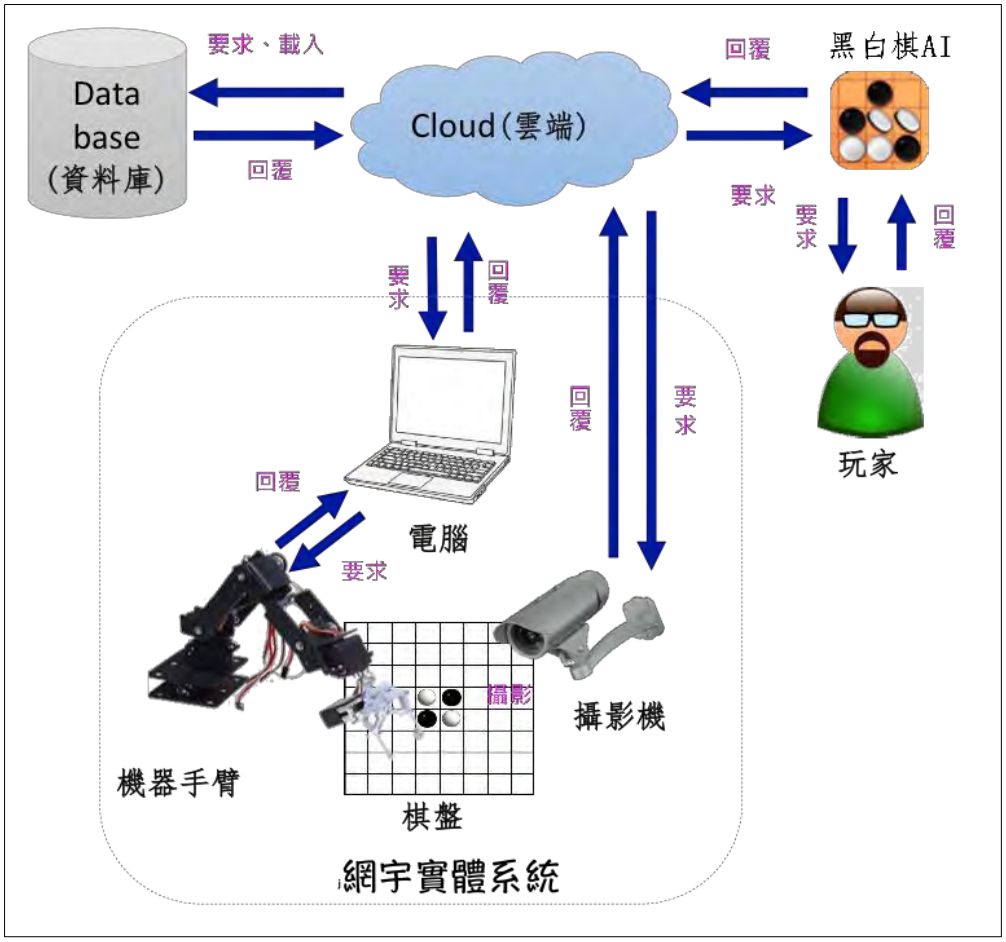


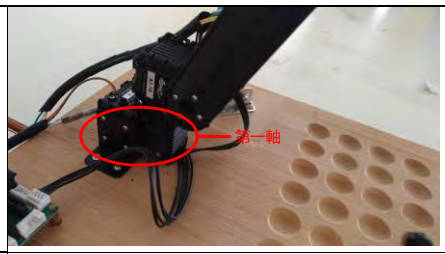
圖 3 參考工業 4.0 的系統架構示意圖

二、 機械手臂製作過程

本研究著重於將黑白棋人工智慧程式判別的結果，使用機械手臂與人進行人機對弈。

棋盤採用木板利用 CNC 進行雕刻，機械手臂的馬達以 AI 伺服馬達為主，另外夾爪採用一般的 RC 馬達。部分支架採用 3D 列印製作手臂、連接臂及夾具等部件。再利用螺絲將手臂各部分連接及固定，以建構機器手臂及棋盤與機械手臂的組合。

機械手臂必須要 5 個旋轉軸，分別定義為 A、B、C、D、E 軸。設計的說明如下：

A		1. 用途：帶動水平方向的旋轉。 2. 旋轉角度：-165~165˚
B、C		1. 兩軸及兩臂形成三個主要支點。 2. B 軸旋轉角度: 0˚~ 135˚ 3. C 軸旋轉角度 0˚~ 90˚
C		1. 使機械手臂的爪子能夠與目標物互相垂直，使機械手臂在取子時，不會輕易的碰撞到其他棋子。 2. 旋轉角度：0˚~ 90˚
D		1. 以夾子形狀的構造來夾取棋子。 2. 使用 RC 伺服馬達。 3. 旋轉角度： 60˚~120˚

另外，各個轉軸點的伺服馬達如果有過載的情形，RC 伺服馬達並不會有任何保護作用，必須立刻斷電，否則馬達很容易燒壞，使用 AI 伺服馬達可以防止過載，當感測過載的時候，馬達會立刻斷電，以保護馬達。

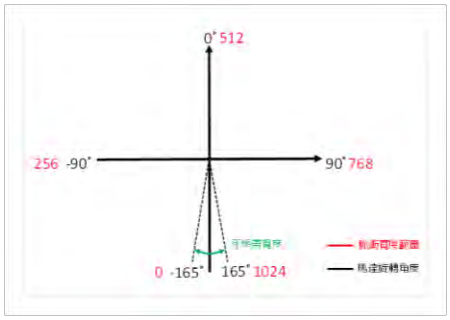
	
AI 伺服馬達正面有一個轉盤，可以固定機械臂或其他機構件。反面有兩個串列連接孔可以串接伺服馬達。	設定伺服馬達的旋轉角度必須使用 0-1023 來換算成 0~330 度的角度數值，因此其解析度為 330 / 1024 = 0.322265625°。

圖 4　AI 伺服馬達外觀、構造及角度示意圖

程式撰寫上，每顆馬達均有自己的 ID，運用 SetPositionI_JOG 函式便可驅動馬達旋轉，範例如下：

```
void setup()
{
  Serial.begin(115200);
  A1_16_Ini(115200);  ──→ 設定鮑率為115200
  SetID(254,1);  ──→ 初始化馬達編號為1
}
void loop()
{
  SetPositionI_JOG(1,200,512);  ──→ 執行馬達動作（第1馬達，0.2s，角度）
  delay(1500);
  int p1=ReadPosition(1);  ──→ 讀取馬達目前角度
  Serial.println(p1);
  SetPositionI_JOG(1,200,0);  ──→ 執行馬達動作（第1馬達，0.2s，角度）
  delay(1500);

}
```

圖 5 Arduino 撰寫程式簡易示意圖

二、　機械手臂的相關公式堆導

(一)　Z 軸旋轉角(θ)及手臂長度 (L) 計算

假設原點為了求得手臂與棋子的距離和角度，先在棋盤上方先標出與其水平、垂直的 X 軸和 Y 軸，將電腦中定義的格子示意如下圖

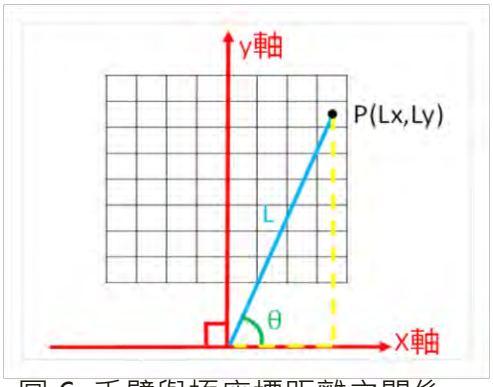


圖 6　手臂與極座標距離之關係

所示。若 P 點的位置在（Lx, Ly）的位置，利用畢氏定理推算出機械手臂中心點到目標物的直線距離 L。再將 Ly 與橫軸 Lx 利用反三角函數推算底座馬達θ 的旋轉角度。

$$L = \sqrt{Lx^2 + \mathrm{Ly}^2} \qquad \Theta = \tan^{-1} \frac{Ly}{Lx}$$

(二) 各軸馬達轉動角度換算

為求得第 2、3、4 軸伺服馬達轉動的角度θ_1、θ_2、θ_3為多少。由上面的說明，已經可以得知俯視的 L 長度，為進一步求得θ_1、θ_2、θ_3機器手臂的架構如圖 6 幾何架構，可以推算出適當的數學公式。

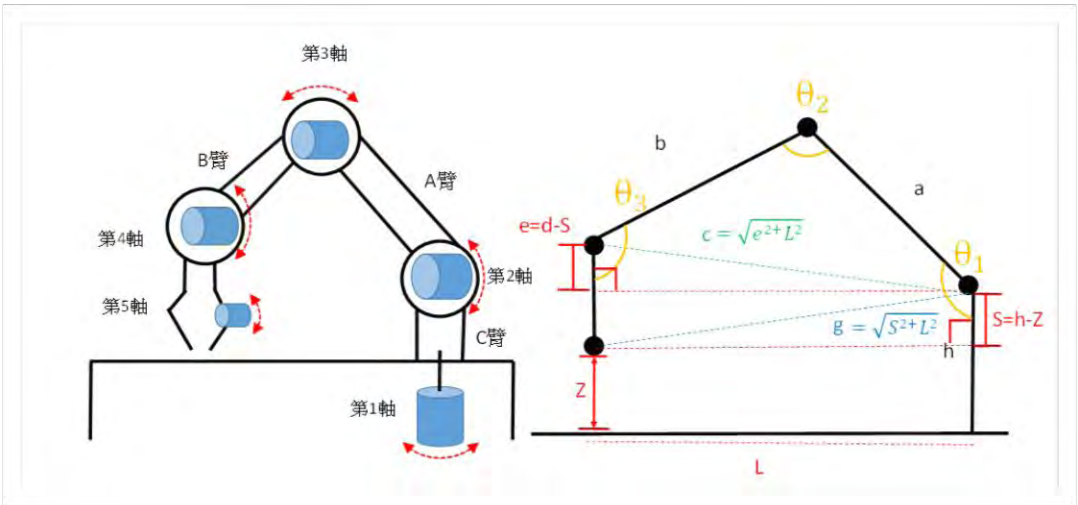


圖7 機械手臂幾何說明

由圖 7 幾何架構可以利用餘弦定理和上圖所表示出的 a 臂、b 臂、c 臂，兩者之間求出三個內角θ_1、θ_2及θ_3，公式如下：

$$\Theta_3 = \cos^{-1} \frac{c^2 + e^2 - l^2}{2ce} + \cos^{-1} \frac{b^2 + c^2 - a^2}{2bc}$$

$$\Theta_2 = \cos^{-1} \frac{a^2 + b^2 - c^2}{2ab}$$

$$\Theta_1 = 360 - \Theta_2 - \Theta_3$$

經過餘弦定理的運算後，就可以利用手臂在四個點的位置去推算其餘 60 點的位置，因為棋盤為正方形，各棋目之間的距離又相等，所以只要抓出四個點的位置，就可以使機械手臂完成所有位置的取子動作。

與前一代相比，整體架構有許多的改變，為了使系統架構更加人性化和減輕重量，

在系統上面改變了材質與使用的設備，以下是所有新增的項目：

(一) 利用智慧手機的 IPCAM 來取代以往監視用的 IPCAM，不但迎合現代人手機的高持有率，未來也可以在智慧手機上新增互動式的介面，使下棋更加有可玩性。

(二) 利用鋁合金的支架取代以往笨重的鋁擠，使其重量減輕。

(三) 利用 3D 列印 PLA 的材質，取代前代手臂較重的金屬支架。

(四) 不使用滾珠軸承的方式控制水平旋轉移動，原因 AI 伺服馬達有較高的力矩，又為減輕系統整體的重量，改由單顆 AI 伺服馬達進行動作。

(五) 以往傳統伺服馬達為並接控制，因此可能產生雜訊而使用了舵機控制板，現在因為 AI 伺服馬達採用串接的方式控制，所以目前使用 ATmega1280 Arduino 馬達控制板，直接對馬達下達命令。

(六) 為了使黑白棋遊戲更加的完整，由一開始的 6*6 的棋盤改為正式遊戲所使用的 8*8 棋盤，而考量玩家取子、落子的方便性，在棋盤上面刻有雙方棋子的放置點，讓遊戲進行時更加流暢。

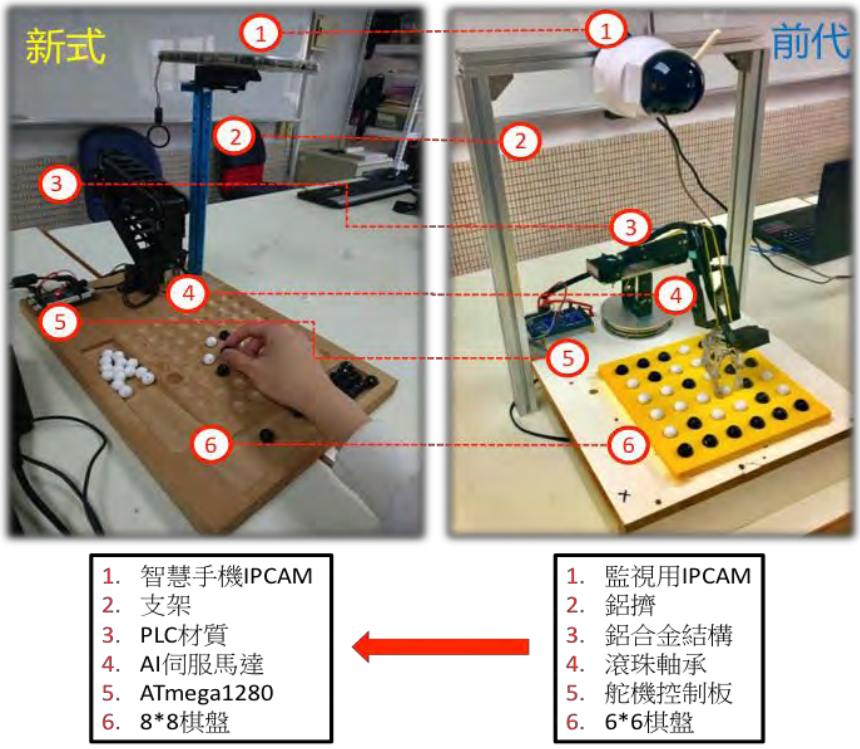


圖8 新式與前代的比較

三、 人工智慧黑白棋的設計

黑白棋又名翻轉棋，是 19 世紀末的英國人所發明的。遊戲方式是將對方的棋夾住後能將對方的棋子換成我方的棋子，在對弈的時候只要有能將對方的棋子換成己方的位置就一定要落子，若沒有能將對方的棋子換成己方的棋子時則換對方下，當雙方皆無法落子或是棋盤上沒有空位時則遊戲結束，以棋盤上棋子數多的人獲勝。

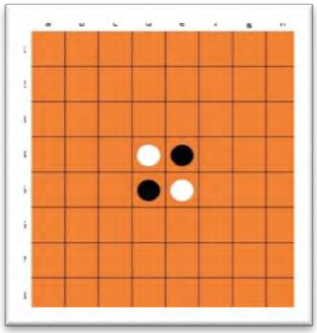


圖 7 黑白棋的棋面

(一) 位置判斷法

使用翻棋數量的方法可以在遊戲前期取得很大的棋子數量的優勢，但是因為黑白棋獨特的規則，很容易使雙方棋子的數量劇烈變化。有可能在遊戲後期可能僅用幾個回合就將大量對方棋子變成己方，從而扭轉局勢。因此，太著重於棋子數量是沒有必要的，更重要的是占據有利的位置。

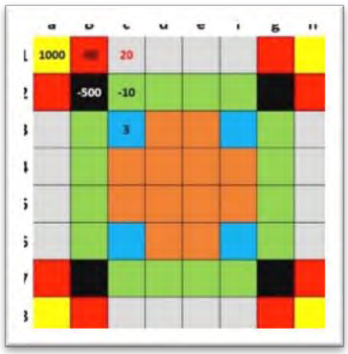


圖 8 各位置的加權分數

在撰寫這部分的人工智慧時，歸納出以下幾個重點

1. **黃色位置**：下在角落的棋子是絕對不會被對手翻面，所以它是最重要的點，而且可以確保邊上的棋子。
2. **紅色位置**：若是落子在此位置，會使黃色位置被搶走的機會變高，但是因為邊上的點只會有一個方向被翻面，所以危險性較低。
3. **黑色位置**：是棋盤上最需要小心的位置，雖然他只有一個方向會使角落被翻面，但由於棋盤中間的棋子很容易翻來翻去，使得此位置上的點很容易讓角落被搶，所以在下這個位置時要非常小心。
4. **灰色位置**：只會有一個方向會使它換顏色，而且占了這個位置後可一次翻許多棋，而且也可以提高搶到邊的機率。
5. **綠色位置**：若是落子在這裡的時候，會開始進行搶邊的動作。而這通常對先落子於此位置的人來說有些許不利。
6. **藍色位置**：剛開局時很重要的位置，因為落子於那個位置後，對方若想要將它翻面，必須要下在綠色的位置。

(二) 遊戲樹

位置判斷法從最重要的從黃色位置推導出每一部分的權重，可以使 AI 優先選擇經驗上較好的位置。但在棋局變得較激烈時，由此簡單邏輯的判斷會變得不夠精確，若直接在棋局中推測對手下一步落子位子，能更精確的掌握棋局發展進而取得勝利。

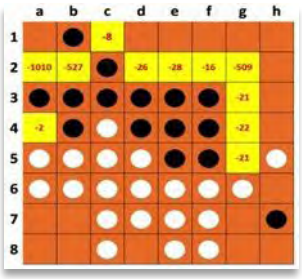


圖 10 遊戲數的加權值

遊戲樹方法會透過不斷的推測自己以及對手可能落子的點進行計算以找出最佳落子位置。以圖 11 的 a4 位置為例：AI 落子在邊上，所以這點的加權值加上 20 可以翻一粒敵方的棋子，在第二圈所以加權值減上 10。接下來要推測對手可以下的位置，假設對手會選擇加權分數最高的位置下，所以可以算出圖 4 的加權表。想要推算更後面的棋局只要重複以上的步驟就可以進行。最後再把我方的加權分數減上對方的加權分數後就可以得到一個遊戲樹推算出的加權分數。

在這個遊戲樹演算法當中，有額外加入了 MiniMax 搜尋法、α-β 剪枝演算法跟第二版位置判斷演算法以及翻棋樹演算法進去作為輔助。加入 α-β 剪枝是由於搜尋層樹的提升導致過多不必要的搜尋拖慢了整體搜尋的速度，因此加入此演算法刪剪掉較為不重要的節點以提升搜尋速度；而 MiniMax 搜尋是找出對於我方加權值高（極大）的地方以及對手加權值高（極小）的地方，判斷兩方各別的下棋走向，尋找出最為符合現實狀況的推論。

(三) UCT（Upper Confidence Bound 1 applied to trees）

以上述的遊戲樹 AI 與線上玩家進行多場對弈後發覺，在對於棋力為中階左右的玩家仍可以保持較高的勝率，不過對於較高階的玩家卻下的有些吃力，由於高階玩家時常使用長時間布局的方式將局面控制在他的理想狀態中，等待時機成熟進而一舉反攻導致我方輸棋。而我方的演算法是透過遊戲樹的搜尋來模擬對方的下法，不過若對方是以布局的方式下棋，在前期會故意削弱自己地盤的優勢以提升局末的機動性因此遊戲樹演算法無法準確預估對手行動，將一步一步掉入對手陷阱中。

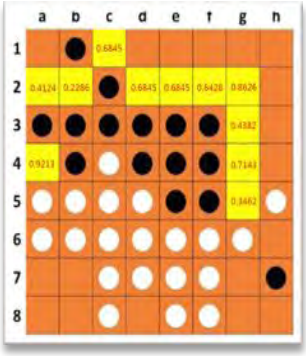


圖 11 UCT 各可落子點的權重

因此希望能夠透過學習的方式讓我方 AI 在對弈的過程中學習對手棋步，經過多次對弈後

能夠自我修正以探索出勝利棋步取得勝利。經過資料的查詢後找到了 UCT 演算法，由於 UCT 演算法的判斷依據來自於過往的對弈勝率以及探索的結果，與先前的固定依據不同，因此希望能夠過此演算法修正先前 AI 會掉入對手陷阱的問題。以下為組成 UCT 演算法的 3 種不同要素：

1. UCB（Upper Confidence Bound）

以上述的遊戲樹來說，它的加權判斷是以位置加權來做為依據，但在各種場合下，位置加權表上加權高的點不一定是好位置，也會因此容易掉入陷阱，所以透過統計學中的 UCB(上信賴界限)作為 UCT 的判斷依據，這能夠計算出下該點的價值。UCB 公式如下：

$$UCB = \frac{\omega_i}{n_i} + c\sqrt{\frac{\log \mathrm{T}}{n_i}}$$

- ω_i 代表 i 點模擬棋局獲勝次數
- n_i 代表 i 點模擬總次數
- C 為常數，當數值越高則勝率比重越低，探索慾望越大
- T 為所有點模擬的總次數，取 log 平衡測資不均的問題

UCB 公式包括獲勝機率以及模擬該位置的次數。優先選擇尚未模擬的點進行模擬，確認此點是否能夠到達理想的獲勝結果。常數值 C 主要是影響 AI 的探索慾望的參數。當 C 值提高，勝率的比重會下降，AI 探索慾望將會提高，因此探索寬度會越寬闊。找到適當 C 值來取得探索廣度與深度的比重，也同時影響了 AI 學習的速率。透過這個公式，能使 UCT 所需的資料更有參考價值且將好的點做較深的搜尋以達到不平衡的遊戲樹搜尋。

2. 蒙地卡羅（Monte Carlo）

蒙地卡羅方法是利用隨機採樣的方式及龐大的模擬資料來取得近似解。此方法剛提出時由於硬體無法跟上演算法所需的龐大數據而並未受到重視。近幾年設備的快速提升下，蒙地卡羅方法也逐漸浮出檯面。蒙地卡羅方法常運用於數學求的最佳解的問題上也常用於決策遊戲中，若將事件以及許多條件設定好，也能模擬一些無法在現實世界中所進行的實驗例如：核彈模擬、粒子碰撞等。

本研究不針對黑白棋的攻略進行 AI 設計，此外亦無較好的盤面評估方式可以判斷棋

局優劣，因此若以蒙地卡羅樹搜索 (MCTS) 進行，利用勝率的統計得出較佳的棋步，由於 MCTS 的下棋依據來自於自我探索的結果，即使撰寫者的黑白棋棋力很弱，只要程式給予黑白棋的規則，就可探索出勝率較高的下法。

3. UCT 演算法

UCT 演算法是結合了蒙地卡羅方法及 UCB 公式的遊戲樹搜尋，此演算法與一般遊戲樹搜尋的不同之處在於其加權值 UCB 會在探索時不斷的更新，因此 UCT 在選擇搜尋點時並不會平均的將每個點進行平均尋而會對較好的點進行較深的探索，因此它屬於一個不平衡的遊戲樹搜尋，也因此不必額為加上剪枝法就能自我省略不必要的可能性。下圖為 UCT 演算法實際進行的流程圖，下圖中的數字為（勝利次數/總模擬次數），在經由 UCB 公式計算後能得出一個 UCB 值，下方流程圖中的選點依據來自於此。

下圖為 UCT 演算法實際進行的流程圖，說明如下：

(1) 選點：透過 UCB 公式計算出各個可落子節點的 UCB 值，選出值最高的節點進行下一層的延伸探索。

(2) 擴點：當已經沒有可以模擬的節點時，優先對無資料的節點進行模擬。

(3) 模擬：亂數模擬直到分出勝負。

(4) 更新完節：將勝負結果反向的更新到剛剛落子的節點，新增一筆勝/負的資料 (反向傳播)。

以上的四個步驟稱作 1 次輪迴，每一個節點的模擬都會衍伸上千或上萬次的輪迴模擬，讓數據更完整更具參考價值，藉不斷的刷新 UCB 值來取得較佳的落子點。

由於 UCT 演算法需要及龐大的資料以進行分析，而目前的設備無法符合蒙地卡羅的需求，因此透過資料庫的結合，將對弈過程中的棋譜、探索資料、棋譜及棋局結果等資訊上傳至資料庫，藉由分析過往對弈的數據進行近一步的棋局分析來達到經驗傳承的效果讓 UCT 演算法具有學習及修正的能力。透過這個方式，UCT 演算法能在對弈的過程中自我探索、學習如何躲避陷阱，以達到這次人工智慧所期許的主要目標。

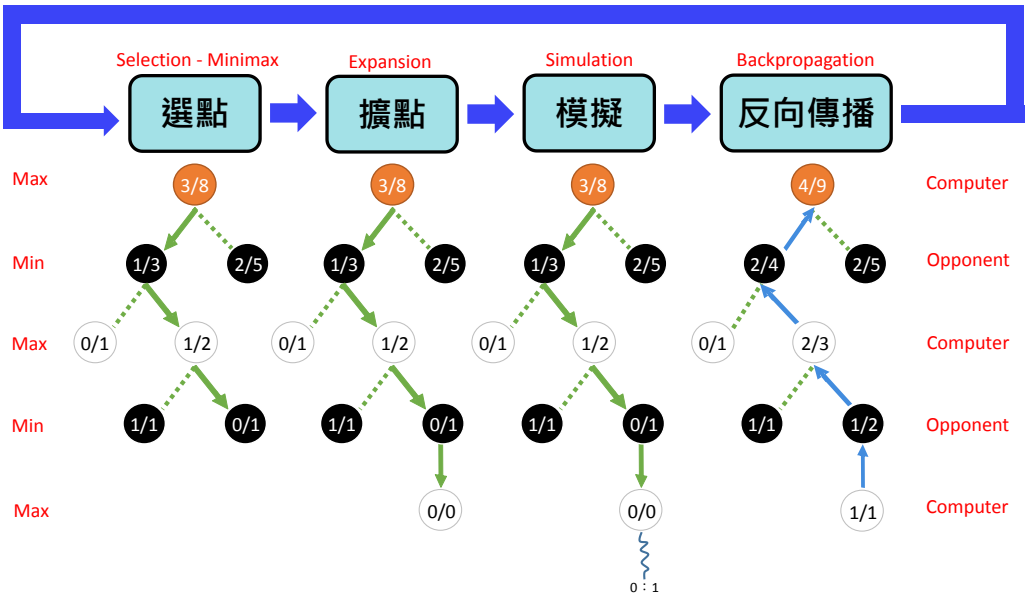


圖 12 運用蒙地卡羅樹搜尋的 UCT 流程

4. RAVE 搜尋法

RAVE 搜尋法是建構在蒙地卡羅樹之上的搜尋法，其搜尋流程與 UCT 類似，唯差別在於最後的更新資料以及加權值的不同，在對弈時或許 A 點（棋盤上某一點）在模擬後得到敗場但是 A 點在其他棋局模擬時得到了不少勝場，所以推測 A 點或許也是個好點，因此在更新資料時多了一個「點」的權重。在黑白棋中有金角銀邊的觀念，也因如此，認為 RAVE 搜尋用於黑白棋或許是一個不錯的方法。透過此方法能讓 AI 更快速的學習到黑白棋的特性（金角銀邊）進而提升 AI 的學習速率，強化 AI 的學習。

四、 影像分析

當棋盤上影像從雲端擷取後，必須進行影像處理，藉由判斷棋盤上棋子的位置、黑色、白色，來建構出一個棋局的陣列，供後續人工智慧的部份進行分析，以決定下一步的落子、取子、補子及違規，透過這些結果指揮機器手臂完成必要的動作。

(一) 邊緣偵測

整個黑白棋對弈的流程中，有一部分需要進行影像處理轉成程式所需要的資料。影像處理時所需要最為重要的為圓形偵測以確保棋盤上目前對弈資訊，但在進行影像處理前，需先進行灰階化，因為只透過灰階明亮判別邊緣比含有色光資訊的邊緣，判斷上含有色光的部分反而還要在多上一些 RGB 的視覺亮度做運算，所以轉灰階能夠更加直觀的去做這類類比影像資訊轉數位資訊的影像處理系統。

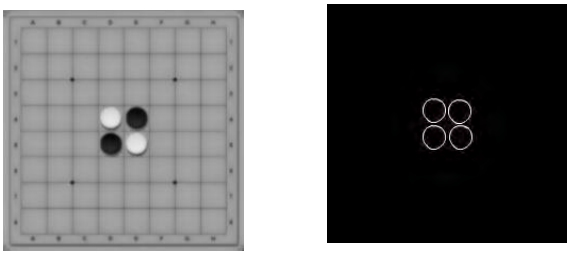
(a) 灰階化的圖　　(b) 邊緣偵測後的圖

圖 13 邊緣偵測

通過灰階化後會因為背景有細小雜物而讓邊緣偵測錯誤的情況，將整張圖片進行高斯模糊後，細小的雜物會被模糊與背景同化，對於要偵測的大目標並不造成影響，而達到降噪的效果。降噪後進行邊緣偵測，當上面黑灰的圖畫成反明度的量圖時，會發現會有一個點的明度有巨大的變化，此時產生變化的 pixel 便是所需要找的邊緣。

(二) 圓形偵測

使用的圓形偵測方式為霍夫找圓(HoughCircles)，霍夫找圓的找圓方式是利用圓方程式 $(x-a)^2+(y-b)^2=r^2$，其中(a，b)為圓心座標，r 為圓的半徑，圓形半徑是由作品所使用的棋子所產生，用這個二維數據，讓(a，b)在影像座標內不斷改變位置，最後當這三維數據組的點數，超過訂定的閾值時就判斷為圓，而將圓繪製。



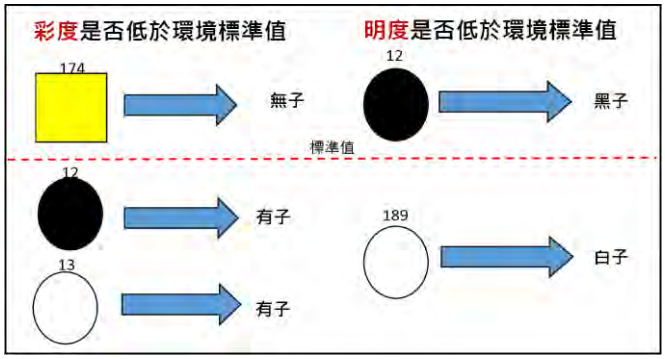
(a) 灰階化的棋盤　　(b)圓型偵測後的圖

圖 14 圓形偵測

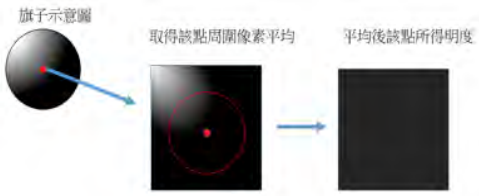
將圓型偵測後繪圖的圓形畫在預設的黑色影像上，利用設定固定距離搜尋剛才的繪圖的結果，與高斯模糊後的黑白明度做比對讓程式知道黑白棋棋盤中目前哪一個棋在哪一個位置上，匯集成數位資料輸入於程式中傳進資料庫與程式，以便給人工智慧去判斷。

(三) 顏色偵測

在確立位置後可取得該圓的中心點 XY 目標點，可以透過將圖轉成 Bitmap 格式來取得該目標點的飽和度、明度來辨別，先利用該位置的飽和度辨別該點是否有棋，因有棋的情況下，只會有黑白等彩度偏低的顏色，所以當取該點的彩度有高於所設定的值，則可以先行判斷出此處為無子。假如此處有子，則利用明度偵測判斷該棋為黑棋或者白棋，來輸出成指定的圖片格式。



利用彩度判斷該點是否有子，明度判別黑子、白子



取棋子中心點的周圍明度平均降低干擾

圖 15 黑白子的判別方法

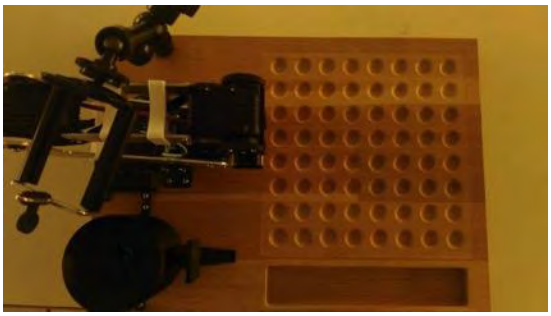
(四) 白平衡演算法

從之前的研究結果得知，單純利用彩度與明度來判斷會因為環境色光的不同而導致棋子彩度大於棋盤彩度的情況，使影像就算調整彩度與明度的閥值也會有資料錯誤的情況，所以加入了白平衡演算法利用平均的 RGB 色光比例，增減每格像素的 RGB 色光來

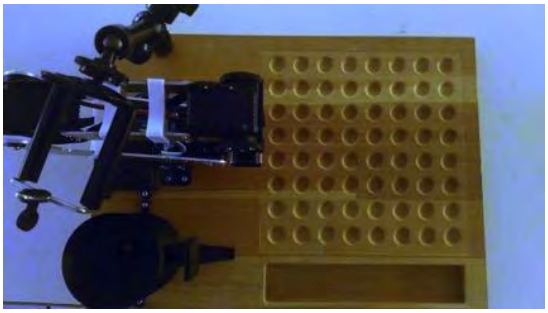
減少色光不均所造成的影響，以維持棋盤與棋子的彩度差異性。

(五) 直方圖拓寬

此為影像增強的方法之一。雖然白平衡已經進行了對於色光比例的降噪，但本身影像的差異度並不會提高，故使用直方圖拓寬，針對每個點進行全局統計，運用全局統計的結果，可將每一像素把原強度範圍擴增至最大強度範圍，使影像的差異度等比例的提高，以利有效判斷範圍可以拓展。



原始影像



經白平衡及拓寬後

伍、 研究結果

一、 機器手臂測試

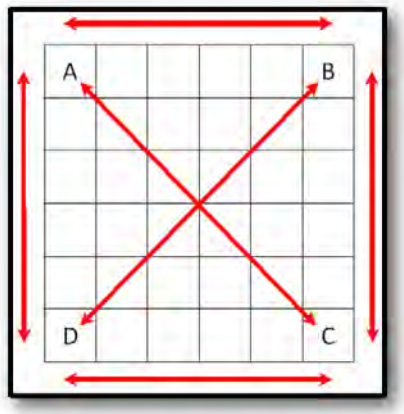
實驗一 使用 AI 伺服馬達版的機器手臂，進行 4 點 6 方向的來回夾取測試，以了解機器手臂的穩定性及精確的程度。

實驗說明：

(一) 為瞭解機器手臂是否可靠，經調整結構穩定、彈力平衡力矩、減輕重量已達最佳化後，進行 4 點間 6 種來回取子、落子動作測試成功的次數，評估機械手臂重現率。

(二) 以玩家視角的位置，分別在棋盤上的四個角落標出四點，左上角為 A 點，以順時針的順序依序為 A→B→C→D。

(三) 每次進行實驗都以兩點反覆落子與取子為一次動作。完整完成視為成功，若中間有動作失敗，則視該次測試為失敗。每一個動作都進行 100 次，利用上方的網路攝影機記錄每一次的動作過程。



動作	第一代 RC 伺服馬達版		第二代 AI 伺服馬達	
	成功次數 (測試 50 次)	成功率	成功次數 (測試 100 次)	成功率
A<-->B	45	90%	100	100%
B<-->C	46	92%	100	100%
C<-->D	48	96%	100	100%
A<-->D	47	94%	98	98%
A<-->C	43	86%	100	100%
B<-->D	45	90%	100	100%

4 點 6 方向測試路徑規畫	測試結果

圖 18 六組測試的規畫及測試結果

測試結果：

(一) 以最後的實驗結果得知，手臂的重現率達 98%以上。

(二) 都能夠完成 100 的反覆取動作，而在 A 點與 D 點有兩次取子失敗，原因在於夾子設定的力道不夠大，經由調校之後，都可以完成 100 取子。

討論：

(一) 與 RC 伺服馬達版的機械手臂的取子成功率實驗相比 ，使用 AI 伺服馬達後整體的穩定度和精度都有大幅的提升，成功率有 98%以上的結果。

(二) RC 伺服馬達版有抖動或是過載的現象，使用 AI 伺服馬達來控制，加上整體結構重量的減輕，將原本的機械手臂更加穩定，能夠完成精準的動作。

二、 系統整合測試

實驗一 評估黑白棋是否會依照規則落子，並邀請 20 位同學協助測試評估各種演算法對人的勝率，並區別初學及有經驗者有有何差別。

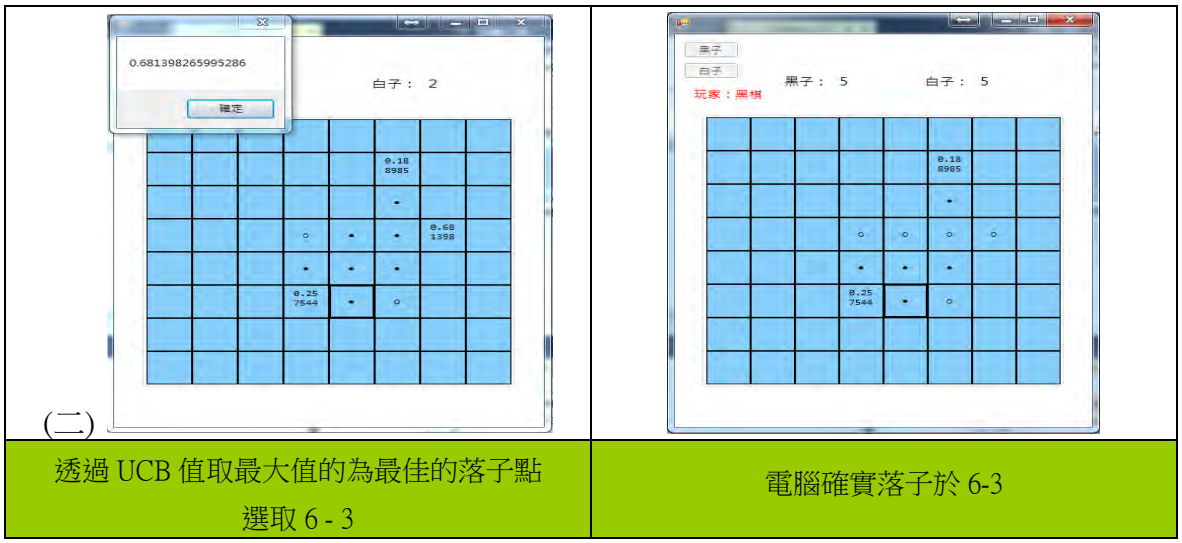
實驗說明：

(一) 觀察黑白棋程式是否會照規則落子，將權值顯示在版面上比對，確定是否正確。

(二) 為了瞭解位置判斷、遊戲樹及 UCT 三種演算法所設計的人工智慧黑白程式與人對弈的獲勝率，邀集 20 位校內同學，包含新手與有玩過經驗的分別針對這三組程式各玩 5 次。

測量結果：

(一) UCT 演算法：這個方法由於透過經驗及勝率來做為落子依據，因此不容易掉入對手所設下的陷阱。以 UCB 公式計算最具探索價值的節點，從下圖左所示共有 0.18、0.68 及 0.25 三組 UCB 值，對畫框顯示最高的是 0.68，也就是最佳探索價值點為 6 － 3 的位置，因此下一步為右圖中的 6-3 點已被電腦落子，並取下兩個黑子。蒙地卡羅樹是透過隨機採樣的方式進行模擬，因此資料不足時等同於亂下，在此實驗中勝率會較低。

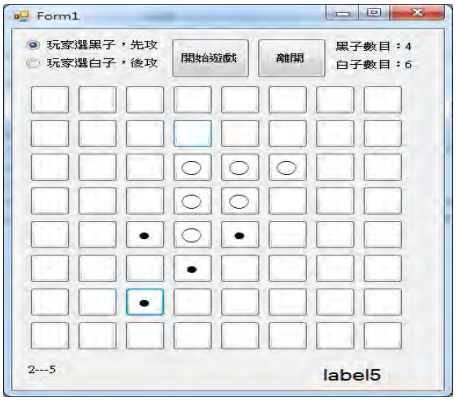


(二)

透過 UCB 值取最大值的為最佳的落子點選取 6 - 3	電腦確實落子於 6-3

圖 18 觀察採用 UCT 演算法是否會照運算結果落子

(三) 單論翻棋加位置判斷：計算加權值的黑白棋 AI 也會按照所訂定的規則落子，而且它在與新手的對弈中勝率達到 85%，並且會以最安全的位置為落子的點。但也因此十分容易中有黑白棋經驗的人所設的陷阱，所以對於有玩過黑白棋的人勝率只剩下 60% 。



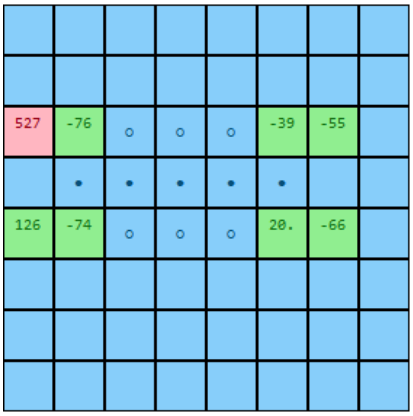
若是黑子落子於 3-1 的位置

根據規則電腦會落子於 0-2

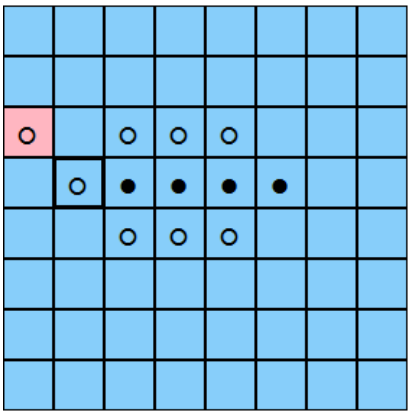
電腦確實落子於 0-2

圖 19 以位置判斷計算加全值的黑白棋ＡＩ是否會依照數量來落子

(四) 遊戲樹：透過多回合的分析能夠比對手想得更加深入因此有很高的機率，遊戲樹並不全然按照所訂定的規則走，而且推算到很多步以後的棋，也因此通常執行時間有會稍長，但是新手幾乎贏不了它，即使是有玩過黑白棋的人也時常會輸掉棋局。



經過計算後得知 0-2 加權值最高



電腦確實落子於 0-2

圖 20 以遊戲樹為基礎所撰寫的黑白棋ＡＩ是否會依照數量來落子

討論：

(一) 以勝率來看，遊戲樹的勝率最高、UCT 其次、位置判斷最低。

(二) 速度上在 UCT 模擬輪迴在經過各種優化後已經可以達到 2000 次以上的模擬圈數，不過

由於它必須做資料的分析再經由計算才能得到結果，時間上仍是 UCT 最久。

(三) 遊戲樹層數越高可能性越多，藉剪枝演算法剪去沒有獲勝機會的分支可使速度提升。

(四) 由表 1 遊戲樹具有最高的勝率，由於 UCT 演算法需要經過長時間的學習才能有較為準確的數據能供演算法進行判斷，因此在資料不足的狀況下 UCT 的棋力不足。在這階段尚無法知道 UCT 是否可以勝過遊戲樹，須待更充分的學習才可以證實。

表 1 人與黑白棋進行 100 次對弈的結果統計 (20 人各玩 5 次結果)

使用何種方式運算	對手為第一次玩的人勝率	對手為有玩過的人勝率
單論翻棋加位置判斷	85%	60%
以遊戲樹(4 階)為基礎	100%	95%
UCT 演算法	70%	40%

實驗二 讓 UCT 與遊戲樹第四階（簡稱 T4）進行對弈，了解是否可以學習？評估系統是否在充分學習之後， 可以勝過 T4。

實驗說明：

(一) UCT 從第一步到最後一步的可能性極為龐大，若從頭學到尾學習，需要很多時間。

(二) 實驗將使用 T4 下完前面的 N 步，以 T4(N) 表示，再將剩餘的棋步採用 UCT 進行探索及學習。藉以限制 UCT 的學習範圍，縮短學習時間。

(三) 測試了 3 種範圍分別為先以 T4(20)、T4(30)及 T4(40)。為了測試大量棋局，使用 T4(N)+UCT 與 T4 進行了 160 場對弈，觀察其學習效率以選取較適當的學習範圍。

(四) UCB 公式中的 C 值為 0.6，讓系統地學習慾望適中。

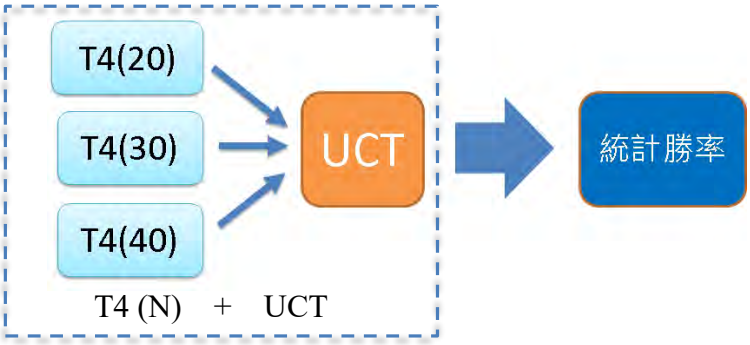


圖 21 評估各種不同組合勝率的實驗流程

實驗結果及討論：

分析 T4(N)+UCT, C=0.6 的實驗統計（如圖 22） 結果說明所示：

(一) T4(40)+UCT：UCT 只剩下 20 步，可學習的步數不多，再加上刻意挑選的列式棋局可至勝的步數很少，因此在這個實驗中大約學習 20 場後就已經達到 100%的勝率了。由於學習前面的佈局均交給遊戲樹處理，剩下 20 步讓 UCT 學習。雖然很快達到 100%的勝率，但學得太少對於系統的提升來說幫助不大，也無法測試學習的成果。

(二) T4(30) +UCT：在 50~60 場左右完成棋局探索並在 80 場後完成學習，達到 100%的勝率。30 步為棋局中盤左右，學習時間約 70~80 場內就能夠完成學習，而學習的數量比較多，選擇此範圍做為下一個實驗的範圍。

(三) T4(20) +UCT：大約 110 場才完成學習，學習時間較久，才可達到 100%的勝率。剩下的 40 步進行 UCT 的學習，學習時間有點長且範圍有點大因此在這不做採用。

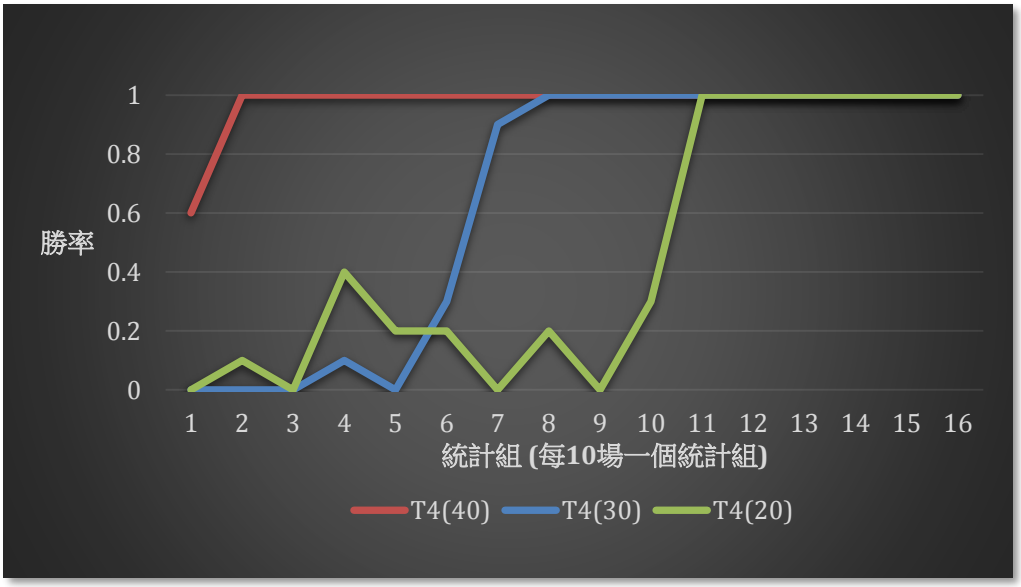


圖 22 使用 T4(N)+UCT , C=0.6 的測試結果

實驗三 透過改變 C 值觀察其學習曲線以找出適當 C 值

實驗說明：

(一) 由於 UCB 公式中的 C 值，會影響系統的探索慾望，影響學習探索的深度與廣度。

(二) 透過 5 種 C 值分別為 0.2、0.6、0.8、1 及 1.414 與 4 階遊戲樹進行 160 場的對弈。

(三) 以 T4(30) + UCT 鎖定學習範圍從中局開始，觀察其學習曲線找出最適當的參數值。

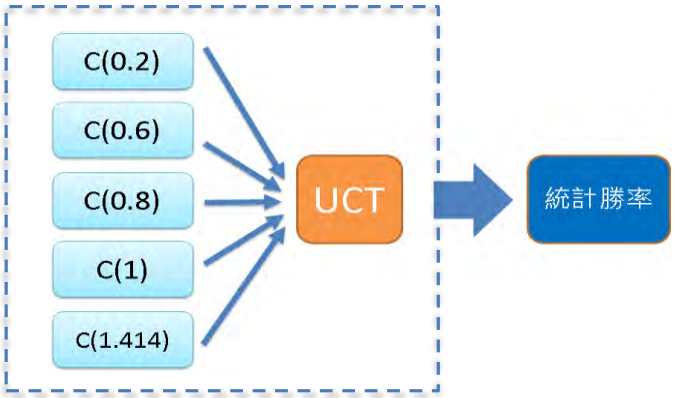


圖 23 評估 T4(30)+UCT 在不同 C 值學習效果的實驗流程

實驗結果及討論：

由圖 24 可比較各種 C 值在 T4(30)+UCT 上的學習效果，結論如下：

(一) C=0.2：曲線中可以看到它很快得完成學習並達到 100%的勝率，因為他缺乏探索慾望，當發現了勝利的步數後便不會再去探索其他點。以學習的角度來講，探索慾望太低廣度狹窄，可能遺漏許多更好的點，這並不是一個好的範圍。

(二) C=0.8：相較於 0.6 有更廣的學習寬度，加上學習效率不比 0.6 慢，因此 0.8 比較符合目前的演算法。

(三) C=1&1.414：這兩個範圍在 160 場的棋局中並沒有完成學習的趨勢，只是因為它的探索慾望強大，需要非常長的學習時間，雖然探索上會比其他來的周全但也會因此降低效率，因此目前不採用這個組合。

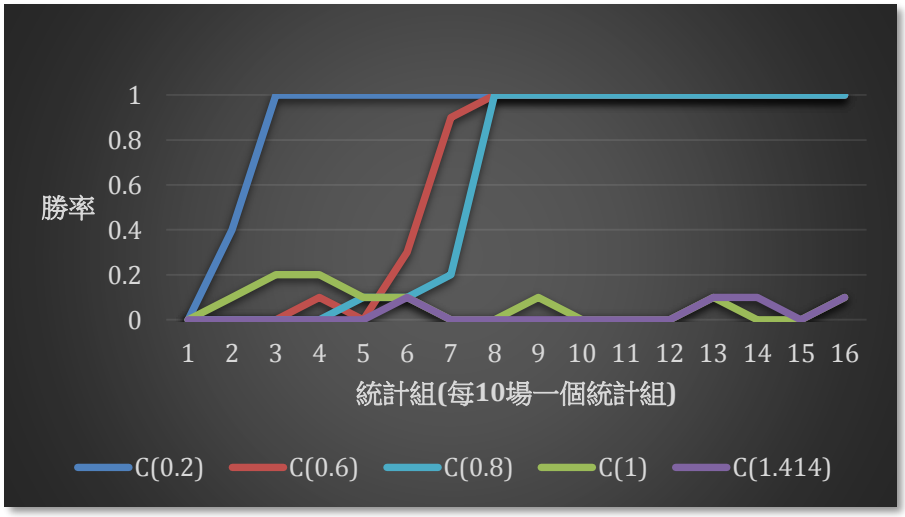


圖 24 評估不同 C 值對學習效率的影響

實驗四 確認透過影像調整、識別及分析流程，是否可轉為規畫的棋局數位資料。

實驗說明：

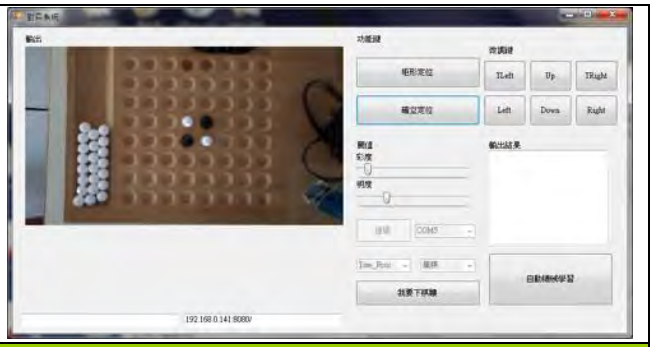
(一) 將影像透過 HTTP 的協定，從 IPCAM 上進行下載。

(二) 棋盤的四個角落依序點兩下，定出四點再推算出其他 60 點的位置。

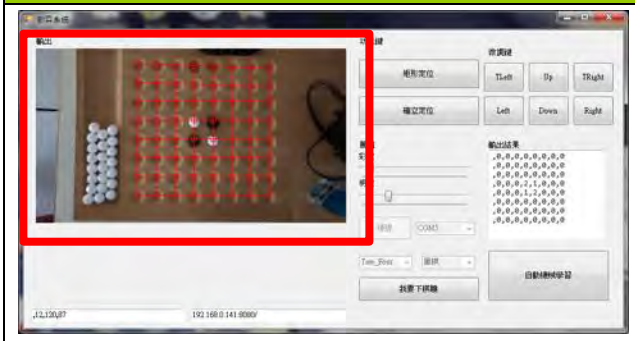
(三) 調整彩度及明度之閥值使棋局上落子的分佈，與數位資料輸出結果一致。數位資料定義無子為 0，黑子為 1，白子為 2。

(四) 送出棋局資料，根據 AI 判別進行下一步。紀錄 100 步，觀察影像判斷的穩定度。

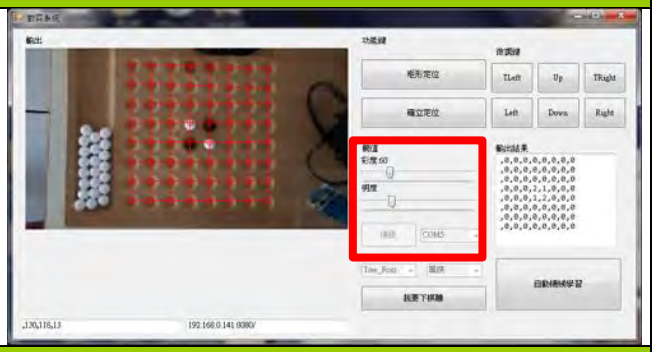
測量結果：以下為影像調整、識別及分析的操作流程。



步驟一：輸入 IP、選擇 AI 及玩家棋子並連線 （左），等待後影像傳至畫面（右）。



步驟二：四點定位，並推算出其餘 60 點。



步驟三：調整閥值使輸出與棋局影像一致。

步驟四：點擊其按鈕將輸出資料送出到 AI 進行判讀。

之後進行 100 步的對弈，測試影像的穩定度，總共出錯了 2 次，此兩次的原因分別為：

(一) 人工定位的運算誤差。

(二) 閥值非位於可判斷區間之中，導致判斷錯誤發生。

討論：目前影像處理系統的誤差來源其中來自於定位的不精確，以及條閥值的可變性，未來希望能利用霍夫找圓進行定位及利用區間內相同結果的特性來自動條閥值，來減少這類人為誤差提高準確性。

實驗五 測試自動白平衡後，藉由彩度可正確判讀的範圍大小，瞭解對色光變化的容忍度。

實驗說明：

(一) 將棋盤上的影像攝影下來，並完成定位的程序。

(二) 明度固定調動彩度，使棋局畫面與輸出陣列的結果一致，紀錄最小彩度與最高彩度。

(三) 將原始影像直接判讀（未白平衡），調整可正確判斷範圍的最小彩度及最高彩度。

(四) 把原始影像調整白平衡後進行判讀，調整可正確判斷範圍的最小彩度及最高彩度。

(五) 判別對色光變化的容忍程度：範圍越大，代表容忍度越高，越適合各種環境。

測量結果：

彩度	無白平衡	**有白平衡**
最低		
最高		
容忍度	108(最高) － 63(最低) = 45	102(最高) － 37(最低) = 65

討論： 自動白平衡後確實能減少色光對於影像的影響，並提高可判斷區間之範圍。但自動白平衡並無法提升棋子與棋盤間的對比，只能恢復棋盤與棋子間原本的對比，所以在正常色光比例下提昇寬度的效果很有限。

實驗六 觀察採用直方圖拓寬提高對比，是否可提高白平衡影像對色光變化的容忍度。

實驗說明：

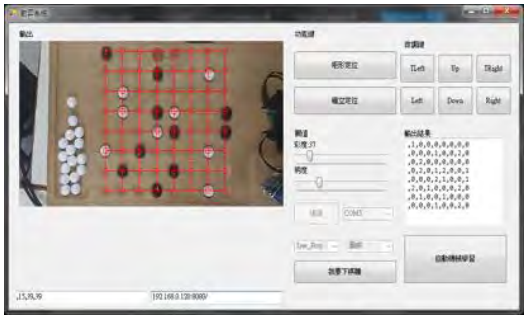
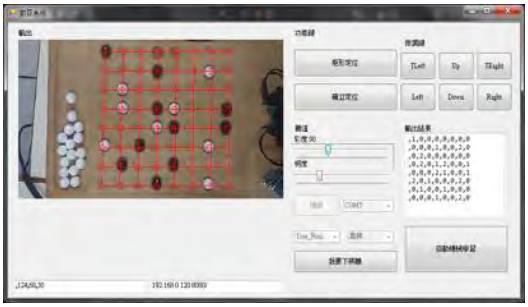
(一) 將棋盤上的影像攝影下來

(二) 將原始影像進行自動白平衡，觀察可判斷範圍的寬度

(三) 將原始影像進行自動白平衡，並加上直方圖拓寬強化，並觀察可判斷範圍之寬度

(四) 兩者數值進行比較來確立直方圖拓寬之效果

測量結果：

彩度	未直方圖拓寬	有直方圖拓寬
最低		
最高		
容忍度	102(最高) － 37(最低) = 65	90(最高) － 18(最低) = 72

討論：直方圖拓寬確實能使原本棋盤與棋子的對比加寬，但加寬的幅度很有限，此影像強化方法只能用在範圍小的地方，如為內容較廣之區域幾乎看不出強化成果，所以未來打算加上直方圖等化來進行進一步的影像強化，來提升判斷有效範圍。

實驗七 智慧黑白棋系統測試：統合所有系統，進行人機對弈的測試

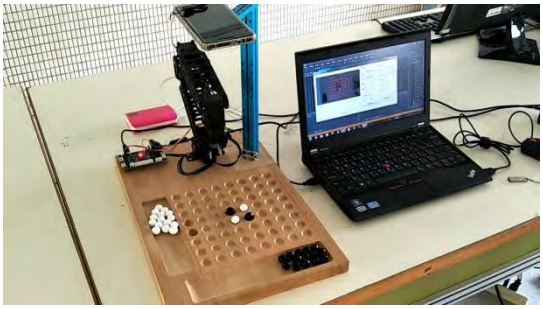
實驗說明：

(一) 電腦透過 http 協定擷取攝影機影像，使用霍夫找圓以及顏色判斷的處理，將影像轉為數位資料傳送至電腦。

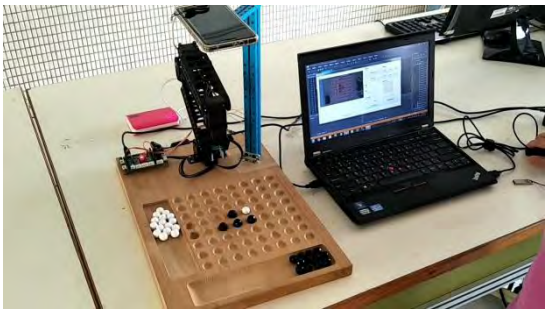
(二) 將數據經過黑白棋程式的計算，並將計算出來的最佳位置傳送至馬達控制器進行定位。

圖 23 人機對弈實況

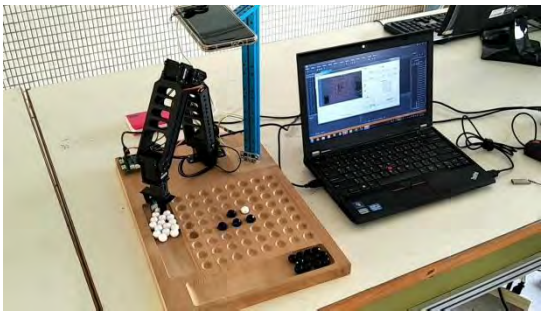
(三) 馬達控制器接收程式所傳的指令，以進行機械手臂夾棋和放棋的動作。



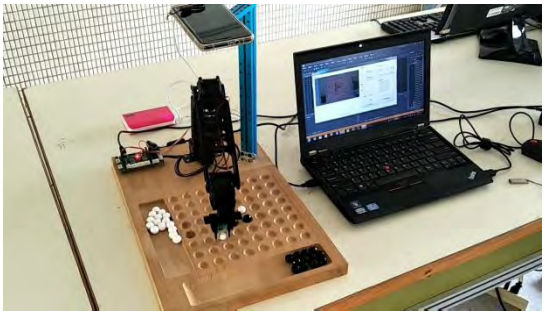
原始棋局



由玩家先手完成落子及取子動作後的棋局

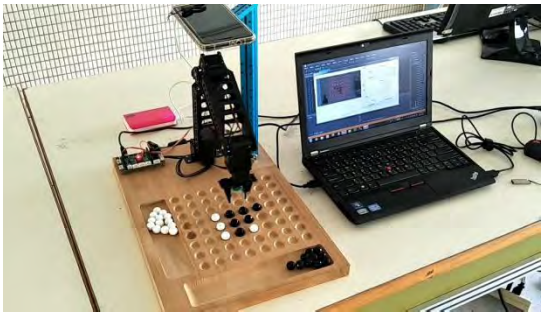


機器手臂進行取子



機械手臂進行落子

機器手臂完成換子後的棋局



進行三手後的棋局

圖 24 夾棋動作與各系統互相協同

總整 25 次測試結果：

表 2　5 位玩家各 5 場對弈之數據統計表

玩家	影像處理正確	手臂運行正確	AI 及手臂運行時間	總時間	AI 勝利	完成完整棋局
玩家一	V	V	16:54	39:21	V	V
			17:42	37:36	V	
	V		18:26	35:12	V	
	V	V	17:21	30:51	V	V
	V	V	19:05	28:32	V	V
平均(五場)	80%	60%	17:54	34:18	100%	60%
玩家二	V	V	17:36	30:35	V	V
	V	V	17:04	32:24		V
	V		19:23	28:42	V	
	V	V	18:42	34:28	V	V
	V	V	19:28	32:03	V	V
平均(五場)	100%	80%	18:27	31:38	80%	80%
玩家三	V		16:32	35:26	V	
		V	18:43	30:16	V	
	V		17:36	31:20	V	
	V	V	20:13	34:18	V	V
	V	V	17:25	29:14		V
平均(五場)	80%	60%	18:06	32:07	80%	40%
玩家四	V	V	16:32	25:42		V
	V	V	18:43	26:37		V
	V	V	18:26	30:24	V	V
	V		16:25	32:23		
	V	V	17:53	28:16	V	V
平均(五場)	100%	80%	17:36	28:40	40%	80%
玩家五 (線上玩家)	V	V	17:25	35:24	V	V
	V	V	16:38	32:54	V	V
	V	V	19:13	40:18		V
	V	V	16:25	41:23		V
	V	V	17:31	43:16		V
平均(五場)	100%	100%	17:24	38:39	40%	100%
總平均	92%	60%	17:53	33:04	68%	72%

結果與討論：

(一) 在本次實驗中，影像的錯誤多半為人為調整閥值的不穩定性，辨別棋盤資訊的閥值是由玩家自行調整，所以就算經由影像強化後，容忍度已經調整的比原先還廣，也會因為人為調整不良而導致判斷錯誤，所以未來打算以自動調整閥值來去減少人為錯失的機率。

(二) 以上所有手臂運行錯誤皆為取子通道設計不良導致的偶發性卡子，手臂在進行落子及取子時十分穩定，並無任何一次誤夾或誤放，因此需再次設計取子通道以減少運行錯誤。

(三) 玩家四與線上玩家(玩家五)對於黑白棋有一定研究，因此 AI 勝率較低，在與線上玩家對弈時，發現線上玩家會以特定戰術佈局來提升後期機動性，我方 AI 在最後 3 場皆因線上玩家已了解我方下法而導致輸棋，且戰術皆為同一種，從此可得知我方 AI 弱點在於盤末機動性掌握並不周全，導致後期遭到逆轉。

(四) 經過今次研究的改良結果，影像已經穩定於 9 成以上，且手臂落子準確度達 100%，在 AI 方面則需將棋局機動性考量在內，避免後期因敵方機動性過高而遭到逆轉。由目前整體完整度可得知，重新設計取子機構為當務之急，若改善此問題，則完成度可達 92%。

陸、 討論

(一) 若是在比較陰暗的地放下棋的話那是否會因為光線不足而導致影像判斷失準?

A：在本次實驗中，加入自動曝光及自動白平衡，並以直方圖拓寬進行強化後，已能夠在陰暗的或者有色光的環境下（如傍晚、凌晨等）進行正確的判斷，但環境容忍度還是會因環境因素而增減。

(二) UCT 演算法該如何提升棋力?

A：由於目前 AI 在進行棋局模擬時只能模擬 2000 次左右，對與此演算法來說次數偏低，因此修改資料結構及搜尋流程以提升整體運行速度是目前提升棋力的主要方法。

(三) 如果給老人或兒童進行遊玩會有危險性嗎?

A：以本研究採用的 AI 伺服馬達還不足以造成嚴重傷害，但希望使用者可以正常使用機械手臂。AI 馬達有偵測電流流量，藉此或許可以設計碰觸到使用者時，解除動力的設計，避免敲擊到使用者。

(四) 為何不在每個棋目上安置感測器而是用影像來進行判別？

A：如在每個棋目上安置感測器，每副棋盤將安置高達 64 個感測器，且只能用於單一遊戲上，成本較高且自由度也較低，若以影像來進行判別，則適用於任何棋類遊戲，不只成本低且自由度也遠比安裝感測器還高上許多，因此決定利用影像進行處理判別。

柒、 結論

本研究改良第一代的黑白棋人機對弈系統，第二代在人工智慧、視覺及機械手臂方面的研究成果如下：

一、人工智慧程式可透過遊戲樹及 UCT 演算法計算出勝率最高的棋步落棋，並將位置座標傳至機械手臂完成動作。

二、人工智慧方面透過 T4(N)+UCT 與 T4 對弈，經測試確實能透過學習找出必勝下法，使用 T4(30)+UCT C=0.8 以下均能破解 T4 的棋路，一旦學習完成，勝率可達到 100%。

三、機械手臂更換為 AI 伺服馬達後，大幅提升機械手臂的穩定性及安全性。

四、視覺方面透過曝光與白平衡的調整確實可降低色光影響，且直方圖拓寬也能夠強化差異之間的對比，以避免識別不出棋盤與棋子彩度之間的差異而造成的誤判。

五、實際測驗人工智慧方面，面對不同的玩家可以達到 68%的勝率，棋力約為中階左右。

六、機械手臂可以連續及順利的達成指定的動作以完成棋局。採用 T4 與人的對弈，總計五位玩家各五場對弈，總共 25 場對弈，有 18 場完整完成棋局。若改善取球機制，可以達到更高的完成度。

本研究未來展望及應用領域如下：

一、 目前以一隻手臂和玩家進行對弈，若加入機器間的協定規則，可以進行機器與機器間的對弈競賽。

二、 透過雲端科技進行異地對弈，讓下棋機器人可以成為遠端下棋者的實體分身。

三、 在老年化社會的趨勢下，可做為「居家陪伴的下棋機器人」。透過大數據除了可以供系統學習使用，亦有分析下棋者的行為模式的可能性，是否可透過分析做為例如失智症相關醫療照護研究有待後續研究。

捌、　參考資料及其他

陳俊嶧(2010)。**一個蒙地卡羅之電腦圍棋程式之設計**。國立交通大學 資訊科學與工程研究所碩士論文

Cameron Browne(2012)。**A Survey of Monet Carlo Tree Search Methods**。IEEE

周冠羽(2016)。 **國立交通大學_電控工程研究所_應用於機器人之基於影像人員活動偵測_論文研讀心得**。取自http://coolmandiary.blogspot.tw/search/label/EmguCV_%E9%96%8B%E7%99%BC

鍾國亮(2012）。**影像處理與電腦視覺(第五版)**。台北市：台灣東華書局股份有限公司。

莊順斌(2005)。**以改良式Sobel邊緣偵測法與密度分群為核心之產品外觀檢測系統—以車牌辨識為例**。國立屏東科技大學碩士論文。

陳閩雄(2003)。**一個應用托勒密定理的隨機圓形偵測演算法**。國立臺北科技大學資訊工程系碩士論文。

張任佐、李昀龍(2015)。**具有手眼協調能力的高爾夫機器球童**。中華民國第55屆中小學科學展覽會。

王小科、王軍(2015)。**王者歸來-C#完全開發範例集**。佳魁資訊。

成和平(2004)。**百玩不厭的黑白棋《國內第一本黑白棋攻略專書》**。宇河文化出版有限司。

【評語】052318

本作品以 UCT 演算法建構具學習能力的黑白棋對弈機器人。相較於前期作品，作品主要改良演算法以提高勝率，以及使用 AI 伺服馬達以提升機器人運作時之定位精度。針對這兩個主要改良點，可進行量化分析，以佐證成效。

軟體對弈能力為本作品重要內容之一，在 UCT 演算法之外，團隊可評估其他類型的演算法，並評斷其在對弈上的效能。

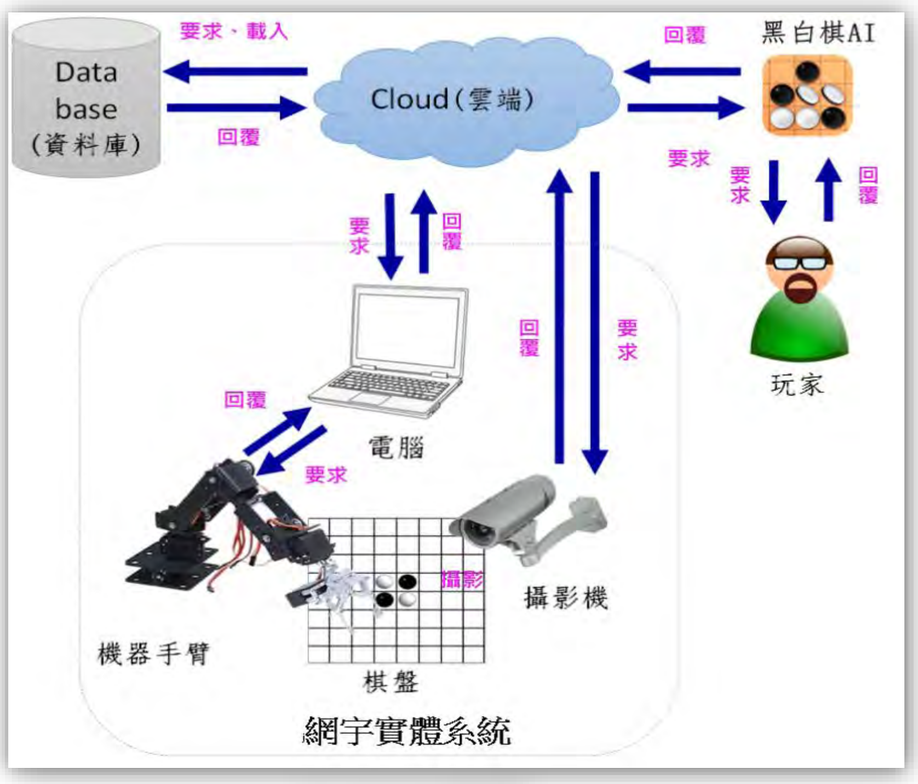
在軟體即可完成對弈功能之際，以機械手臂完成對弈的任務具有獨特性和差異性。團隊也可探索在機械手臂之外，若以其他類型的「實體」來對弈，能否更進一步提升怡情效果。

作品海報

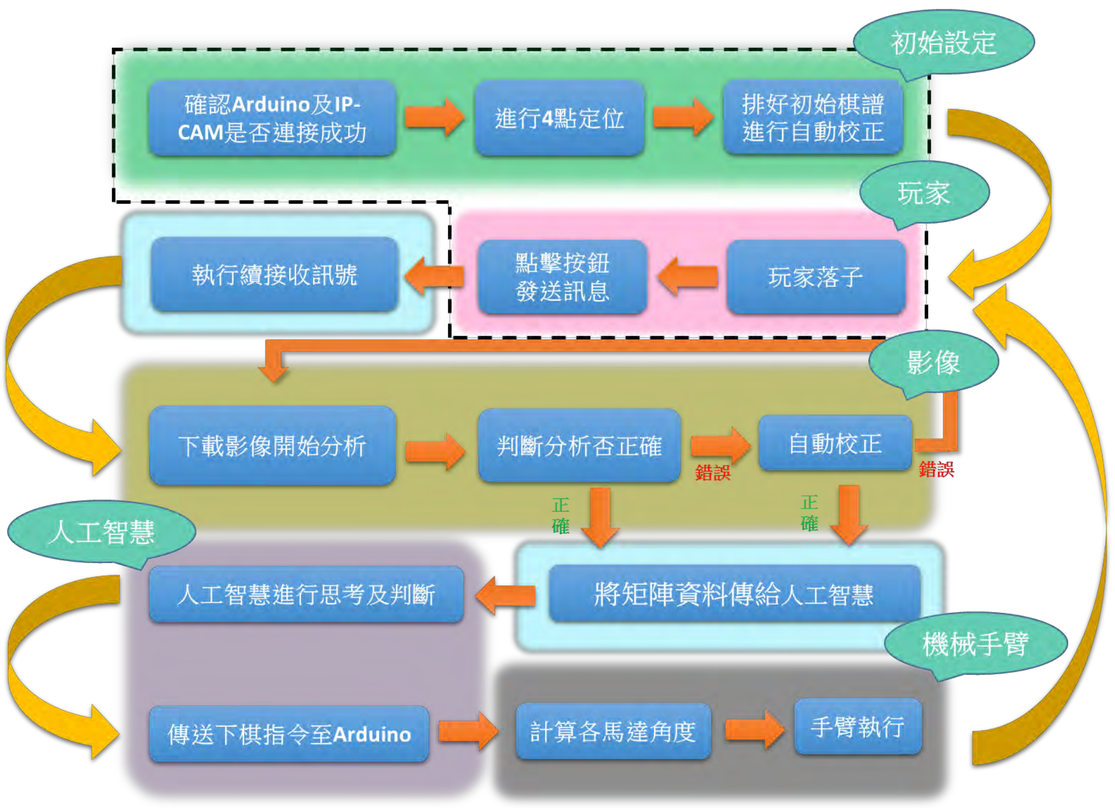
摘要

本研究為了建構具有學習能力的自動對弈系統，並往以下三個方向進行改進:(一)研究使用UCT（UCB For Tree Search）演算法讓機器人可以透過對弈進行學習，找到獲勝的方法；(二)使用 AI 伺服馬達、改良機械手臂的架構以增加機器手臂的穩定度及精確度；(三)使用白平衡與直方圖拓寬改善色光變動造成的誤判。使用AI伺服馬達並利用3D列印自製手臂結構，第二代機械手臂大幅改善手臂準確度，經測試重現率從85%提升到100%。此外使用白平衡後，盤面棋局在適度調整之後，因為光源改變或附近人員走動造成的誤判降低許多，為求100%正確，本研究自創棋譜式影像自動校正法經實驗，影像正確率確實可提高至99%以上，此方法亦可自動判讀人為的錯誤棋局提醒使用者或機器人修正錯誤。人工智慧方面，UCT搭配資料庫，實驗結果證實以遊戲樹八層為對手，最終可以達到100%的勝率。並在研究中加入必勝步搜尋，以提升棋力。

研究流程及方法



系統架構圖

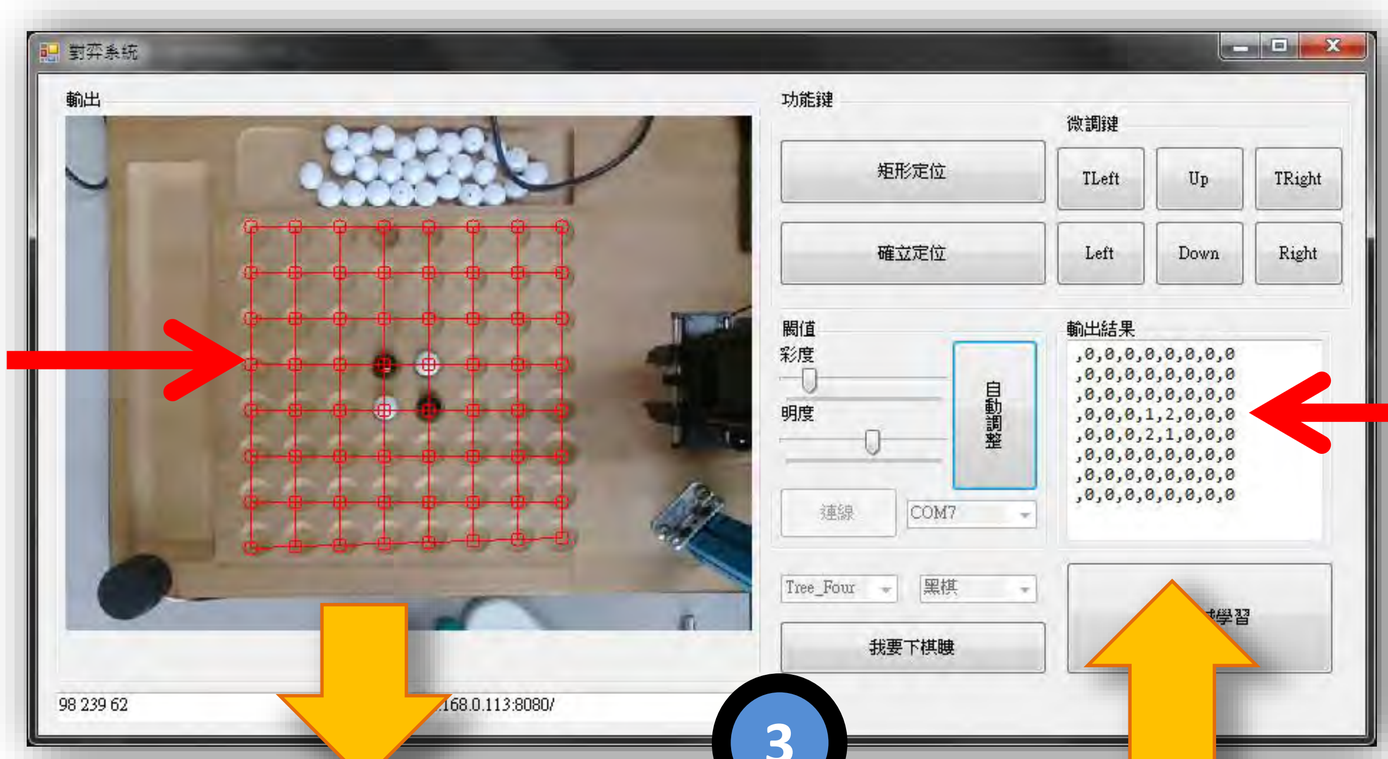


系統運作流程圖

影像處理

1

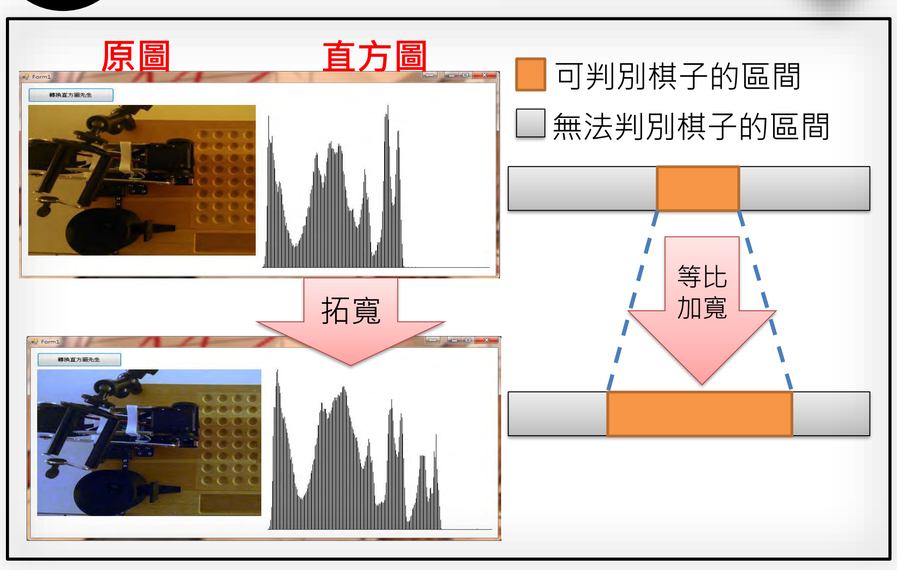
點選棋盤四個角落的棋目，運用四點定位將所有棋子的位置計算出來



4

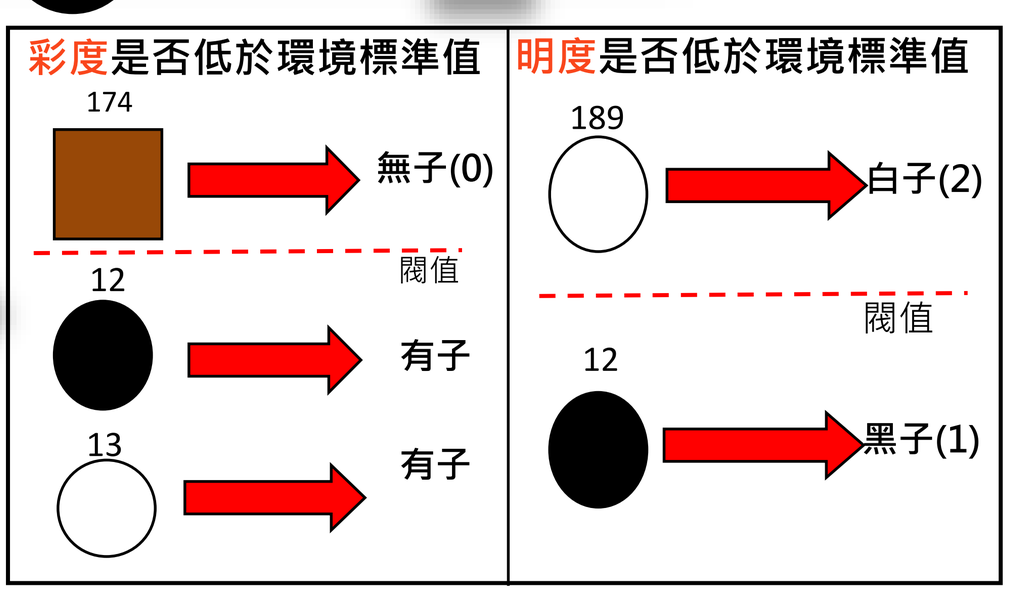
構成棋局陣列資料

2



運用白平衡+直方圖拓寬，延展可辨別棋子的區間

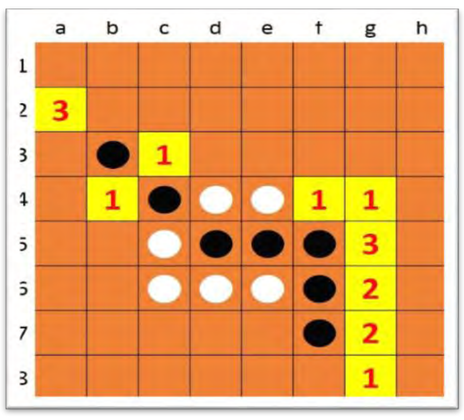
3



透過彩度辨別有子無子，明度辨別白子黑子

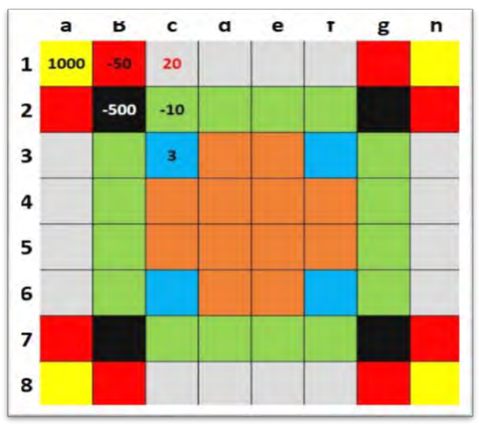
黑白棋人工智慧

第一代研究



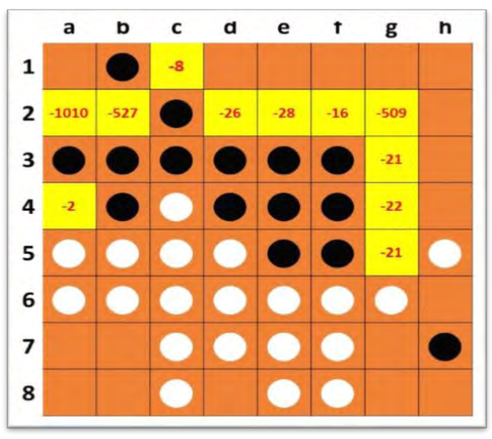
單論翻棋數量:評估各個落子位置可以翻多少的棋子，並選取可翻最多棋的位置落子。

第二代研究



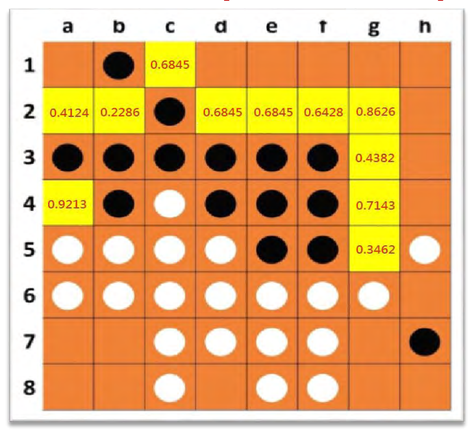
位置判斷:由於黑白棋邊與角落的位置重要性較高，因此給他相對高的加權分數，以利於人工智慧進行判斷。

第三代研究



遊戲樹:先假設落子位置後，利用左邊的演算法計算對手的加權分數，選出所有的點之中最低分的為落子位置。

第四代研究(研究重點)



UCT:此演算法為遊戲樹的一種，再選點時，先以隨機採樣的方式模擬完此棋局，並將得到的結果回傳，最後選出勝率最高的點落子。

UCT演算法(Upper-Confidence-Bound for Tree Search)

蒙地卡羅方法：透過亂數模擬的方式進行搜尋棋局，若模擬次數越龐大，則越會接近實際狀況。我們將此模擬結果的勝負作為回傳值並帶入以下UCB公式。

UCB公式：UCB（*Upper confidence Bound*）它是在統計學中計算上信賴區間的一個公式，在棋類中的應用多半是為了解決探索的深度與廣度間不均的問題， 透過此公式能夠讓蒙地卡羅的採樣能夠更加平均， 數據能夠更有參考性。

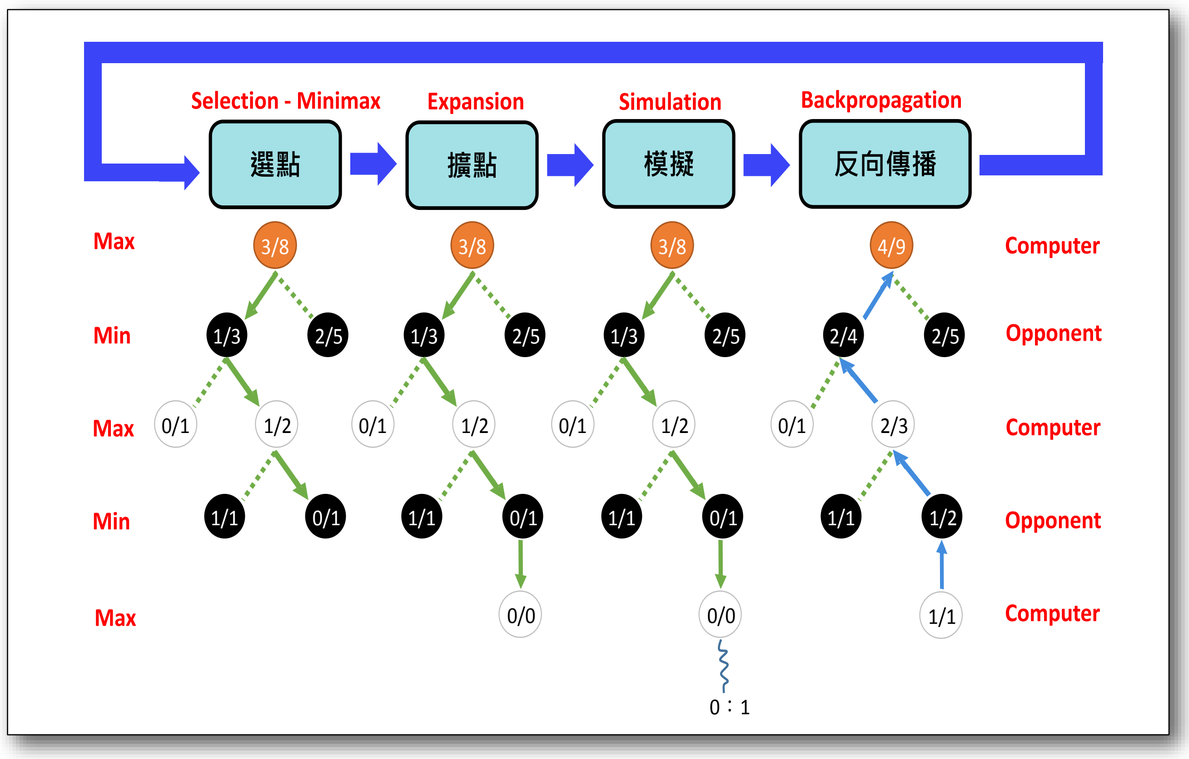
$$UCB = \frac{\omega_i}{n_i} + c\sqrt{\frac{\log T}{n_i}}$$

ω_i ：i 點模擬棋局獲勝次數

n_i ：i 點模擬總次數

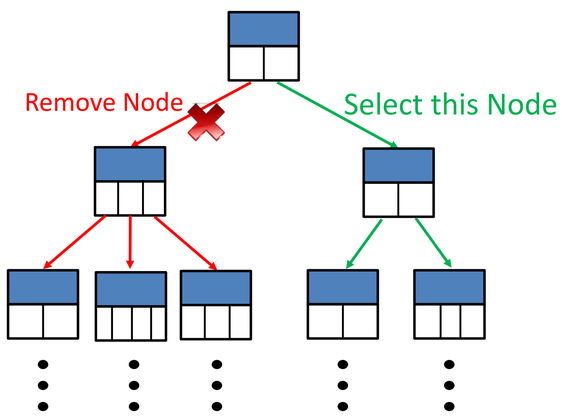
C ： 常數，數值越高勝率所佔的比重越低

T ：所有可落子點的模擬總次數，取 log 平衡測資不均的問題



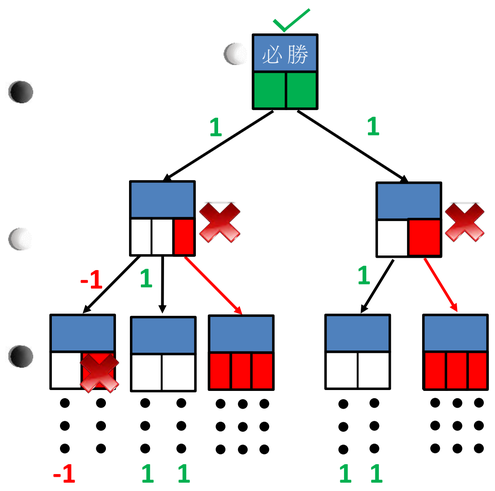
蒙地卡羅樹搜尋流程圖

Tree資料結構：為了在時間內進行**10000次的搜尋**，因此必須**提高速度**。本研究將拜訪過的節點，以鏈結結構（**Linked List**）建立遊戲樹，並將較不好的分枝捨去，留下分數最高的分枝。

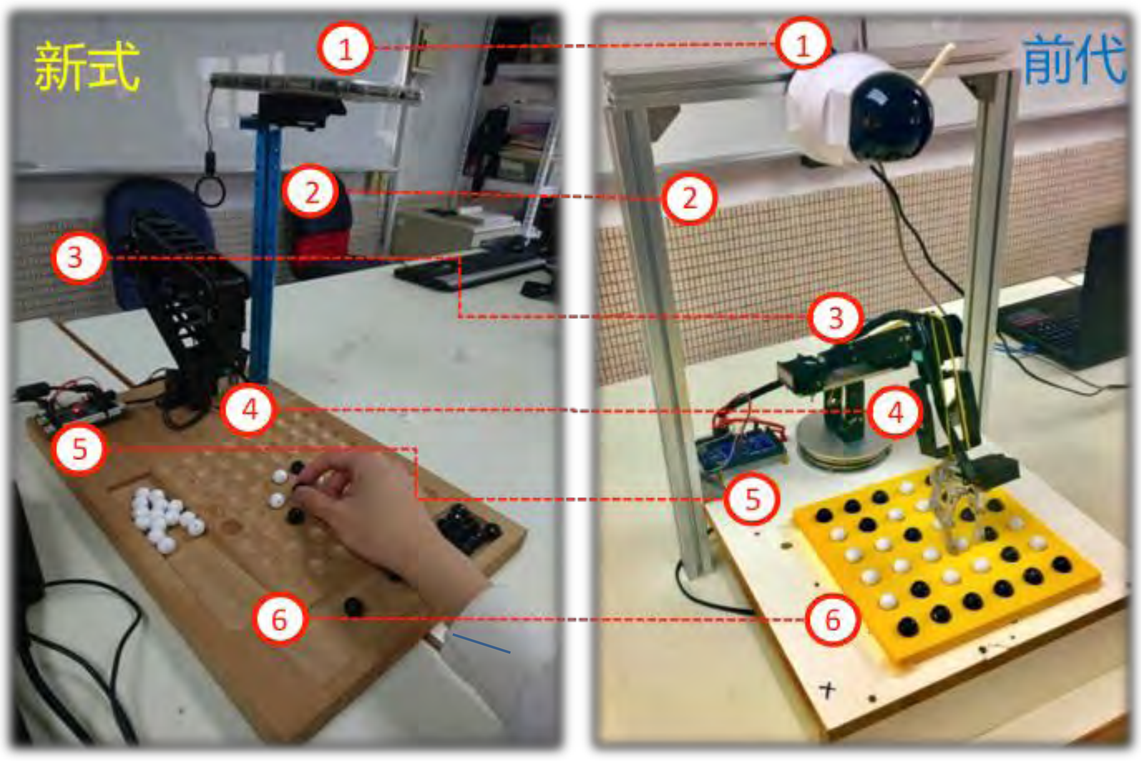


必勝步搜尋：若搜尋所有節點以找出能夠**必勝的路徑**，目前電腦要在1分鐘內跑完，由於可能性過多只能跑4-5層。因此我們透過**Alpha–beta pruning剪枝法**，將不會影響判斷的節點刪除，成功將層數提升至**16-17**層。

必勝步搜尋

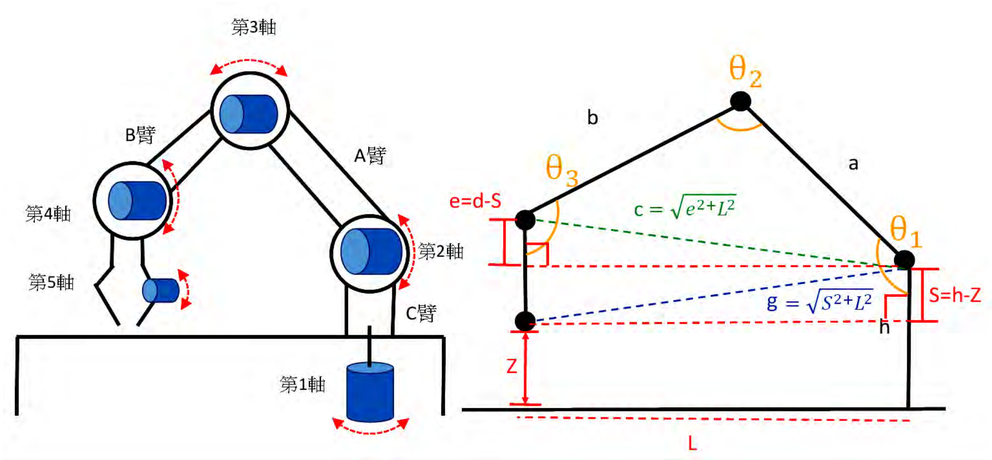


機械手臂設計



1. 智慧手機IPCAM
2. 支架
3. PLC材質
4. AI伺服馬達
5. ATmega1280
6. 8*8棋盤

1. 監視用IPCAM
2. 鋁擠
3. 鋁合金結構
4. 滾珠軸承
5. 舵機控制板
6. 6*6棋盤

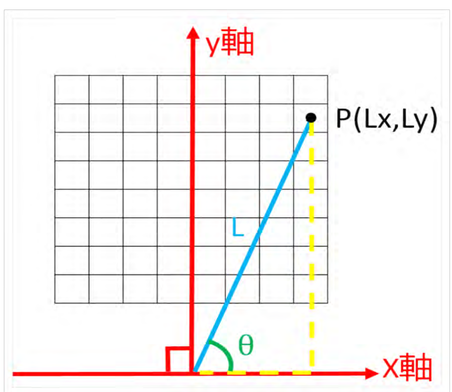


利用餘弦定理**推算各馬達的旋轉角度**:

$$\theta_3 = \cos^{-1}\frac{c^2+e^2-l^2}{2ce} + \cos^{-1}\frac{b^2+c^2-a^2}{2bc}$$

$$\theta_2 = \cos^{-1}\frac{a^2+b^2-c^2}{2ab}$$

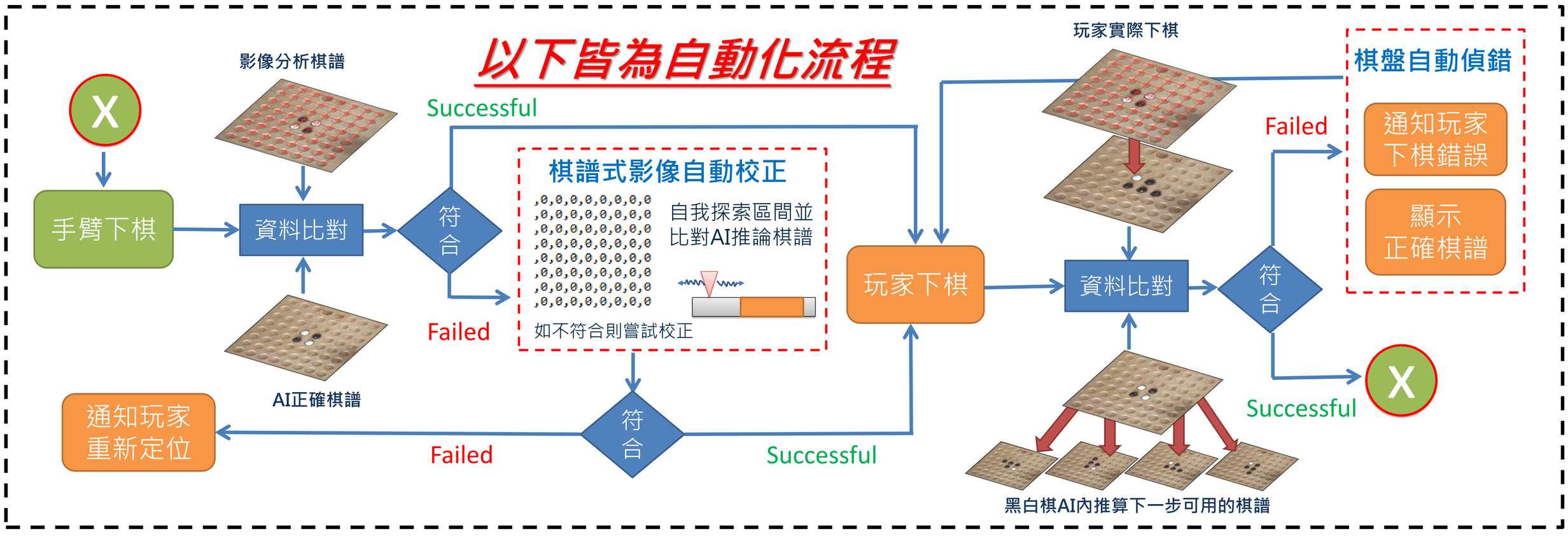
$$\theta_1 = 360 - \theta_2 - \theta_3$$



利用X長度與Y長度並利用**畢氏定理**推算出L長，並且利用餘弦定理來推算**底座馬達**的旋轉角度。

公式: $L = \sqrt{x^2 + y^2}$

影像自動校正及自動偵錯處理



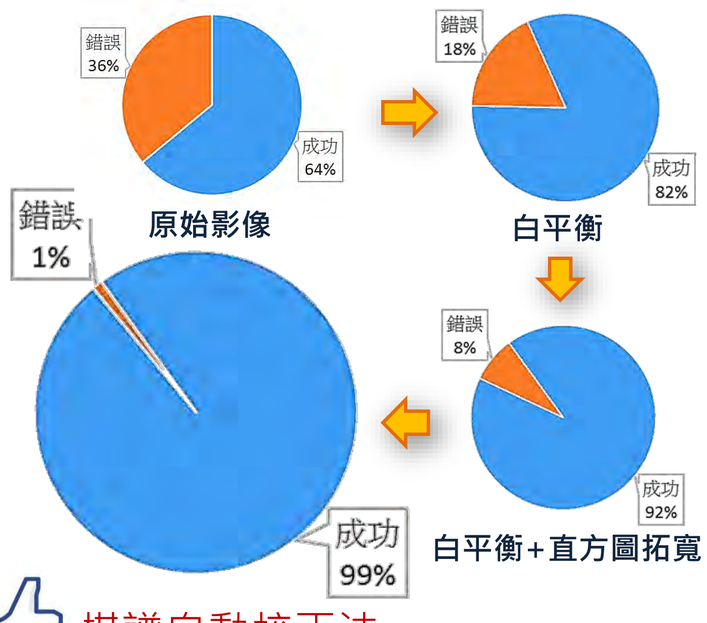
研究結果

影像穩定度測試

實驗說明：分別針對強化前後的影像進行判斷及觀察。各實驗100次，將其結果進行分析比較。

實驗結果：

影像處理	正確率	主要干擾因素分析
原始影像	64%	環境光改變、人影擾動
白平衡	82%	人影擾動
白平衡+直方圖拓寬	92%	人工調閥值的誤差
棋譜自動校正法	99%	偶而傳回錯誤的數值



棋譜自動校正法

人工智慧實驗

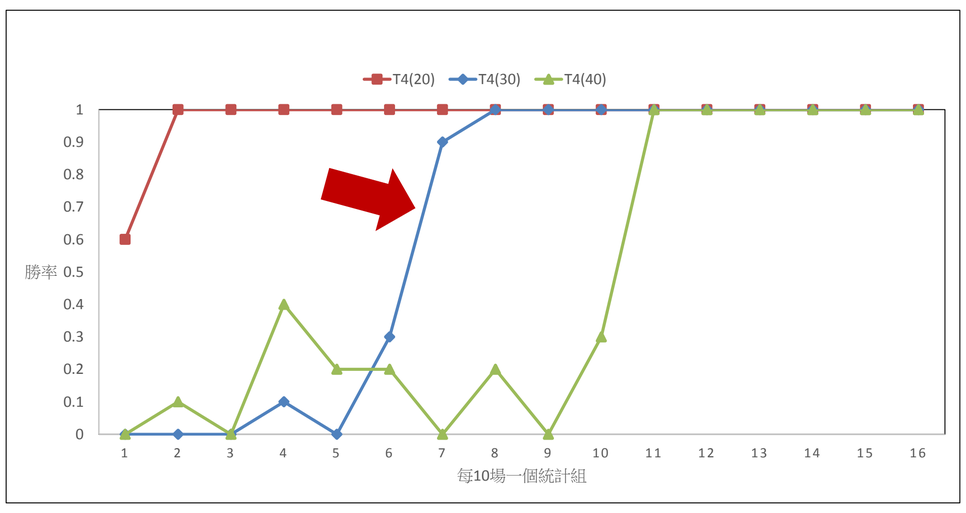
實驗一：證明UCT的學習能力及不同學習範圍的學習效率

實驗說明：

1. 為證實搭配資料庫確實能具學習能力，採用先4層遊戲樹(T4)後UCT的方式縮短學習時間。 c值(探索意願) 固定為0.6
2. T4(n)+UCT代表T4走n步，再進行UCT進行探索及學習。
3. n 值越大探索範圍就越少。測試不同的範圍的學習效率有何不同。

n= 20, 30, 40

T4(n) → UCT → 統計勝率



1. 此三個實驗組合，均可以學到如何勝過T4的能力，證實UCT確實具有學習能力。
2. T4(40)在很快的時間便找到獲勝的方面，但可能失去學習其他步數的機會，T(20)要花更久的時間學會，因此折衷採用T4(30)+UCT的組合。

採用T4(30)

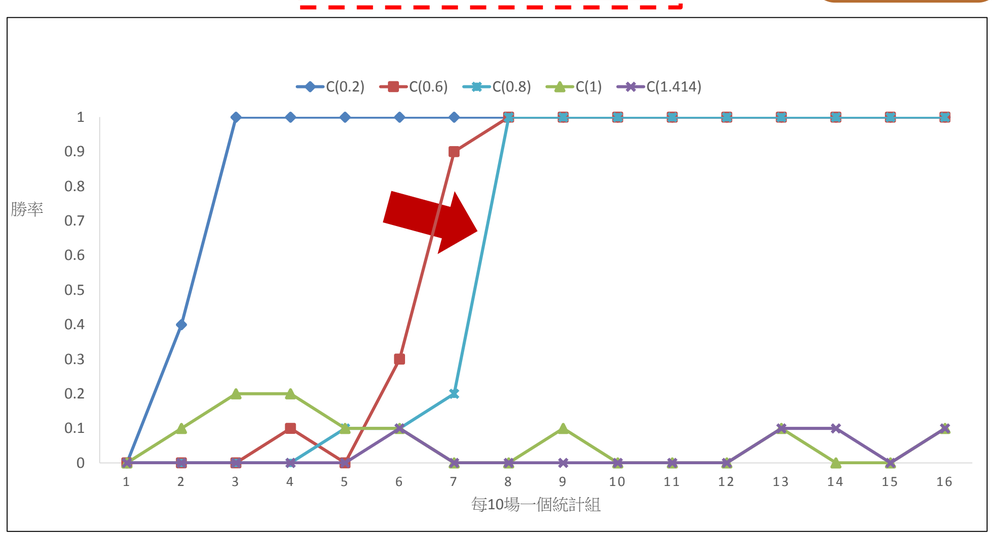
實驗二：測試不同探索意願 (c值) 的學習效率

實驗說明：

1. 採用T(30)+UCT。
2. 刻意挑選劣勢的局面讓UCT進行探索並學習。
3. 探討UCB公式的常數C，對於學習效率的影響，以找出適當的C值。

n=0.2, 0.6, 0.8, 1, 1.414

C(n) → T4(30) + UCT → 統計勝率



1. 設定C值，確實可以表現出具體的探索慾望。
2. C(0.2)學習快速，但可能失去學習很多步數的機會，C(1.0)與C(1.414)探索慾望過於強烈，以致160場內仍在探索階段，C(0.8)在80場便可以完成，不論學習廣度或速度都很適中。

實驗三：將資料結構改為Tree結構後演算效能提升的程度

實驗說明：

使用Visual Studio 診斷工具，可以分析每個執行緒的速度，分析原UCT及改為Tree結構的UCT所占用的執行緒時間，比較其效能。

原UCT (10000圈)

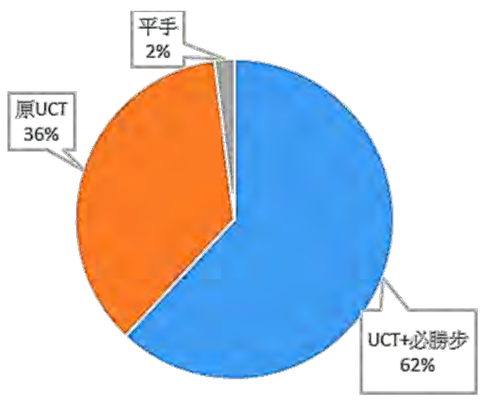
Tree 結構版 UCT(10000圈)

1. 可明顯看出Tree結構UCT不會因資料增加而變慢，速度穩定。
2. 秒數上第一次快了16.8倍，第二次快了63.6倍。

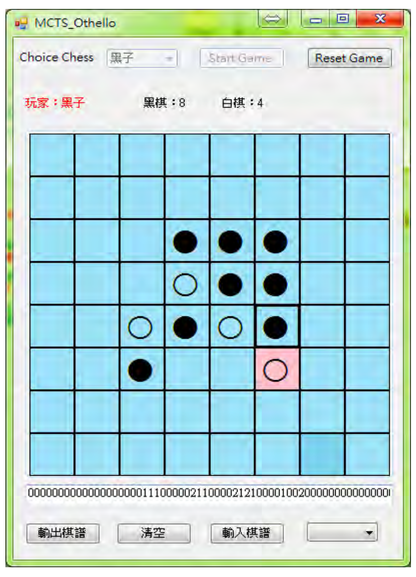
實驗四：採用必勝步與未採用的UCT對戰比較勝負差別

實驗說明：

為了瞭解加入必勝步勝率是否提升，分別使用必勝步的UCT與UCT 進行100次對弈，驗證其棋力是否在UCT之上。



原UCT與加入必勝步UCT進行100次對弈



自動對弈畫面

結論與應用

透過25場的對弈測試，記錄影像錯誤、手臂錯誤、對弈時間、AI思考時間等資料，並進行分析得出以下結論。

一. 因取子通道設計不良，導致偶發性卡子使機械手臂運行成功率下降。

二. 影像利用自動白平衡及直方圖拓寬等強化處理後，資料正確率從原本的64%提升至92%，證實此方法確實能夠提升分析成功率。

三. 採用自創的棋譜式影像自動校正法正確率可以達到99%以上。

四. 透過Tree資料結構改良後，UCT在可運行10000次迴圈數下 (原100次)，思考時間大幅下降且結果可靠。

五. 改良作品後，整體時間從一局45分鐘降至30分鐘、大幅縮短手臂運行時間、增加影像分析穩定度及強化AI智能提升棋力。

未來展望及應用領域：

一、老年化社會的趨勢下，可做為居家陪伴的下棋機器人。並透過大數據分析下棋者的行為模式，為失智醫療的早期診斷之應用。

二、透過雲端科技可以進行異地對弈，讓下棋機器人可以成為遠端下棋者的實體分身。

三、使整個機器人的流程更加自動化，減少設定與初始的步驟，使一般人也能夠輕鬆上手。

四、希望可以透過AI與AI間的自我對奕以自我提升棋力。