定面積可變形體的運動計畫
Motion Planning for Reshapable Objects with Size Conservation

張仁耀
政治大學資訊科學系
g9314@cs.nccu.edu.tw

李蔡彥
政治大學資訊科學系
li@nccu.edu.tw

摘要

雖然現今電腦動畫的技術比以往發達，但在產生具形體變形效果的動畫上，仍需有經驗的動畫師花費許多力氣，方能對此類物體產生特定形變且同時達到移往目標地的效果。針對這種特殊的動畫效果，我們提出了一個以運動計畫演算法為基礎的方法，根據所輸入的環境資訊，自動產生物體形變的路徑，使該物體能移往目標且盡量維持我們所設定的形狀。本研究所定義的機器人具有形狀不定但面積持恆的特性，因此在運動計畫問題的定義上，有異於一般機器人學中形狀固定之機器人的路徑計畫問題。本論文根據此方法已實作出一個動畫設計輔助系統，可以用來降低動畫師在製作此類動畫時所需的成本。我們並以數個實驗說明使用者可指定的各種參數，對計算效能及計畫結果的影響。

關鍵詞：運動計畫、電腦動畫、人工智慧、可變形物體

一、 簡介

在傳統卡通動畫中，常常可以看見如圖 1 一樣具有群體聚集效果的動畫。這些動畫透過許多微小生物或元件聚集成動物、物體或是特別的幾何形狀，藉以誇示動畫給於觀賞者的效果。卡通動畫中的群體聚集行為是動畫師在動畫的每格影格中，藉由人工手動排列產生，其過程是必須依靠動畫師的經驗和技巧，且過程也非常繁瑣耗時。不過，在電腦動畫中，類似的群體聚集動畫，可以交由電腦的繪圖軟體產生，以加速產生動畫的時間；但在聚集成特定形狀的部份，仍須依靠動畫師細心的設計與維護。

在本研究中，我們希望能透過運動計畫演算法的協助，自動產生可變形狀物體的運動軌跡與變形。一般而言，運動計畫器通常是用來為特定形狀的物體依照所輸入的環境資訊，產生一條合適的路徑，使物體能在與環境中障礙物不發生碰撞的情況下，達到我們所設定的目標地。然而，本論文所要計畫的物體不是剛體，而是像液體一般沒有一定的形狀，因此一般機器人學中運動計畫定義問題的方式在本研究中並不適用。在本論文中，我們提出將此可變形的物體離散化成小單位格點的集合，進而針對此格點集合進行運動規劃的設計。由於此格點集合相對關係的變化度高，因此整體系統的的自由度亦高，自然這個運動計畫問題的計算複雜度也相當高。此外，使用者對形狀的設定通常只是偏好，而非必要的限制。因此，如何利用此問題的特性，產生出類似液體的模擬運動，是本研究所將探討之十分具挑戰性的議題。

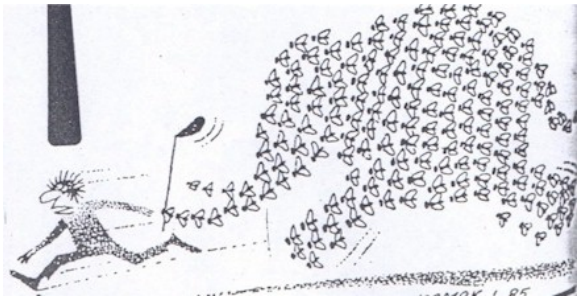


圖 2 群體聚集成動物形狀
http://courageworld.ch/images/PIC1/BeeBigElepht.JPG

二、 相關研究

在人工智慧的領域中，對於場景中可移動的物體來說，在給定起始和終點位置之後，即可利用 Latombe 在[4]所提的運動計畫(motion planning)方式為其找到一條可行的路徑。在電腦動畫中也常用來自動產生角色人物的運動行為，例如[9]利用運動計畫的方式產生拿起並帶上眼鏡的動作。

在動畫模擬中，常見的流體(fluid)運動都是透過粒子系統(Particle System)模擬[7]0[5] 產生，在整段模擬產生後，透過後製的處理計算出流體的形狀，在這過程中包含了許多模擬上的不確定性，例如遇到一些特例狀況時，必須等到系統模擬到那一個時間點才能得知，因此其產生的實驗結果不易控制。

群體運動在電腦動畫中佔有極重要的地位，動畫中加入數量極多的群動運動，可造成視覺上震撼的效果。Reynolds 在[2][3]提出了根基於規則的群體運動模型，透過這模型可以讓群體運動的模擬在表現上更真實。Bayazit 在[5]利用運動計畫器搭配 Roadmap 的方式，將群體運動的行為用運動計畫的方式呈現出來。Kamphuis[1]則在運動計畫器中加上幾何形狀的形變，為群體在移動時能維持特定的群

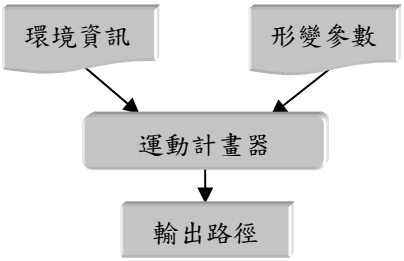


圖 2：運動計畫流程圖

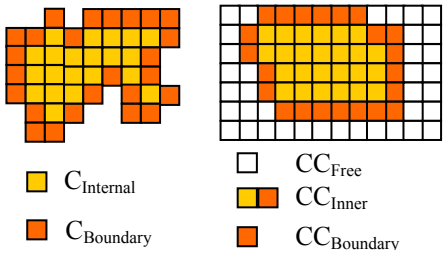


圖 3：組態與 CCMAP 狀態的範例

體形狀，彌補了在僅靠規則模擬無法控制群體形狀的缺點。

三、 系統概觀

運動計畫器是利用搜尋演算法在使用者指定的環境中，為可移動之物體找尋一條避碰路徑的一個軟體程式。在本論文中，我們將針對可產生形體變化的物體，設計一種新的運動計畫器，使其能在考慮形體的變化以及往目標地移動的要求下，產生一條合適的路徑。

圖 2 是本論文中產生可形變物體的運動計畫流程圖。此計畫器的輸入資料包括環境資訊以及物體形變的參數。在這篇論文中，我們假設環境資訊可以表示成二維平面中障礙物(Obstacle)的幾何資訊以及可形變物體(Object)的組態。這個組態資訊包括該物體的起始和目標位置，以及物體的面積大小。其中物體的面積在移動的過程中必須守恆。

物體實體幾何所存在的空間稱作工作空間(Workspace)。為了讓物體產生適當的形變，我們先將該物體在工作空間所佔面積離散化，以格點(Cell)集合表示；而格點是此運動計畫器運算的最小單位。物體由這些離散化的格點集合所組成，我們稱此集合為物體的「組態(Configuration)」。在運動計畫的過程中，我們會透過現有的組態去產生位於鄰近位置的新組態；而運動計畫器就是透過不斷的根據現有組態產生新組態的方式，達到搜尋到達目標地的路徑。在下一節中，我們會介紹產生新組態的方式。

在運動計畫結束後，我們希望能得到物體移動的一條路徑，該路徑是由一聯串的組態組合而成。在運動計畫的過程中，我們希望每次產生的新組態皆與之前產生的所有組態有一些不同，以避免產生重複拜訪的組態，而進入搜尋的無窮迴圈。因此，我們設計了一種資料結構來表示在運動計畫過程中，所有拜訪過的組態在工作空間中的集合，藉以檢查新產生的組態是否為新的合法組態。我們稱此資料結構為「組態覆蓋範圍圖(CCMAP, Configuration Coverage MAP)」。在接下來的章節中，我們會介紹碰撞偵測的作法及維護開放式組態索引圖的方法。另外，在運動計畫的過程中，我們通常必須設定目標函數，以做為搜尋演算法的評估函數，選擇最有希望的組態繼續進行。在第 5 節裡，我們也將會介紹該運動計畫的演算法以及如何利用該演算法產生所需要的路徑。

四、 問題定義

(一) 組態空間的定義

物體組態所在的空間，我們稱為組態空間(Configuration-space, C-space)。我們將所有障礙物先離散化成格點的集合，再將工作空間中該集合相對應格點的位置標註記號。這樣工作空間中沒被障礙物的集合標記到的位置，我們稱之為無碰撞(Collision-free)。如果物體在某一組態下的格點集合皆無碰撞的話，我們就稱這個組態為無碰撞的組態。

我們將物體離散化成格點的集合後，一個組態中的格點可以分成兩種狀態：$C_{Internal}$ 及 $C_{Boundary}$，藉以描述該組態中內部與邊界的格點(如圖 3 所示)。$C_{Internal}$ 的定義方式是指該格點的所有鄰居皆為組態集合中的格點，而組態集合中除了 $C_{Internal}$ 以外的格點皆為 $C_{Boundary}$。組態中的每一個格點皆具有兩個方向的獨立自由度，在考慮物體產生形變運動時，每個格點皆可能產生位置上的變化，因此整個組態空間的複雜度會隨著格點數量的增加而成指數的成長。但是在這篇論文所考慮的物體形變是連續的運動，物體在移動過程中仍須保持為單一物體；因此，為求降低格點個數對空間複雜度的影響，我們在產生新組態的方式上，僅針對 $C_{Boundary}$ 進行更新。

(二) CCMAP

CCMAP 是我們在這論文中所設計的一種資料結構，用來在工作空間上記錄已產生組態的軌跡，提供產生新組態的時候做為判斷依據。在 CCMAP 中格點可分成三種狀態，分別是 CC_{Inner}、$CC_{Boundary}$ 和 CC_{Free}(如圖 3 所示)。CC_{Inner} 是所有被記錄之組態所經過格點的聯集。CC_{Free} 則是 CC_{Inner} 在 CCMAP 中的補集合，而 $CC_{Boundary}$ 則是 CC_{Inner} 集合裡與 CC_{Free} 相鄰的子集合。

為了確保運動計畫器在搜尋時能持續有所進展，我們假設每一次產生新的組態時，該組態必須能將現有的 CCMAP 所記錄的 CC_{Inner} 集合擴大；也

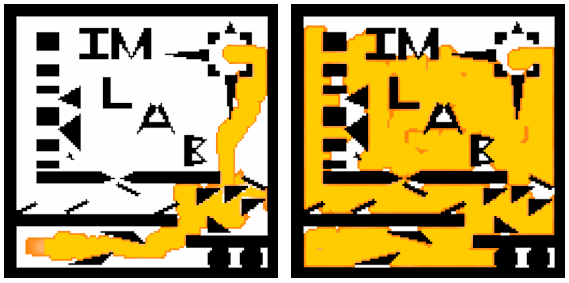


圖 4 CCMAP 範例：在相同的工作空間中在設定不同目標函數後，所得到的 CCMAP 也會有所不同

就是說，新組態的格點集合與 CC_{Inner} 集合的差集必須是非空集合。此外，新組態的每一個格點都必須不與障礙物重疊。我們稱滿足這些條件的組態為合法組態。

我們會將每一次新產生的合法組態用來更新 CCMAP，以做為下一步產生新組態時檢查的依據。CCMAP 的更新可分成兩個步驟，第一個步驟是根據組態中的 $C_{Boundary}$，更新 CCMAP 中的 $CC_{Boundary}$，使其能包含 $C_{Boundary}$；第二步驟則是將 $C_{Internal}$ 在 CCMAP 所包含的所有位置更新為 CC_{Inner} 的狀態，再將 CC_{Inner} 最外緣與 CC_{Free} 相鄰的部份設定為 $CC_{Boundary}$。

由於我們的運動計畫器是由現有組態產生新的組態，這過程中當遇到一個組態無法產生新的合法組態時，表示該組態不符合我們這一節開始所假設的前提。針對此一情況，我們將更新 CCMAP 相對應到該組態格點集合的位置，將 CCMAP 中的位置標註為 CC_{Inner}。

圖 4 為本節所提及維護 CCMAP 的方法執行後所產生的結果。左右兩圖的的差異在於目標函數設定的不同所致的，我們在下一節中會進一步探討運動計畫器目標函數的設計。其右圖淺橘色部分面積較大，由此可看出在該目標函數下，運動計畫器所需搜尋的範圍較廣。

（三） 產生形變的方法

由於可形變物體的自由度相當高，因此產生物體形變的方式也很多。圖 5 是本論文中所提出由一個組態產生新的形變組態所採用的演算法。給定一個組態 S，在產生形變之前，必須先判斷是否能產生合法組態。我們判斷的方法是在 S 的 $C_{Boundary}$ 集合的任一格點中，檢查是否有是 collision-free 的鄰居格點；如果有的話，即代表該 S 可能產生新的合法組態。在產生新組態 S'時，我們會先隨機選定在 $C_{Boundary}$ 集合中第一個外移的格點與其移動方向。再根據 S'的 $C_{Boundary}$ 中與該格點的距離，訂出其他格點的移動順序，使其能向同方向移動。移動的方式是將 $C_{Boundary}$ 中後端的格點往前端移補，而前後端的定義是根據移動的方向決定。由於此可形變物體的移動是以移補的方式進行，因此可確保此物體的

Algorithm: Generate_new_config

Input. A configuration S.
Output. A new configuration S' generated by S.

1. Clone S' from S
2. **if** all cells in $C_{Boundary}(S')$ have no collision-free neighbors **then return nil**
3. Randomly choose C_f in $C_{Boundary}$ such that a neighbor of C_f must be collision-free
4. let v be the vector from C_f to the free neighbor
5. Add all $C_{Boundary}$ cells of S' into a list Q sorted according to the distance to C_f
6. **while** Q is not Empty
7. **begin**
8. poll the front element C_i from Q
9. **if** $C_i + v$ is free in workspace **then**
10. poll the rear element C_i from Q
11. remove C_i from S'
12. add $C_i + v$ into S'
12. **end**
13. **return** S'

圖 5 產生新組態的演算法

面積能保持不變。最後移動完畢的 S'即為我們演算法所產生的新組態。

五、 運動計畫器

（一） 運動計畫器的目標函數

我們在這一節中將介紹如何應用前面所提到的 CCMAP 來進行運動計畫。目標函數(Objective Function) F 是運動計畫搜尋演算法中選擇擴充新組態的評估參考。在本研究中，我們所設計的目標函數是由兩種決策分數組合而成，一種是物體作移動過程中與目標地距離的分數 s_{goal}，另一種則是物體形變時保持理想形狀的分數 s_{shape}。我們的目標函數即是將這兩種分數正規化後，再根據使用者指定的權重，將兩個分數進行線性組合。

$$F(S)=\alpha * s_{goal}+\beta * s_{shape} \qquad (1)$$

提高距離分數的權重α時，可讓目標函數偏好移往目標，使其產生以移往目標為優先的路徑；反之，提高β則可產生以維持物體形變為優先的路徑。

形狀分數是用來評量物體在形變後與理想形狀間的差異程度。在本論文的範例中，我們假設這個理想的形狀是一個圓形，因此我們使用以下的式子計算形狀分數。

$$s_{shape} = \sum_{C_i \text{ in } C_{Boundary}} \sum_{C_j \text{ in } C_{Boundary}} \left(C_i - C_j\right)^2 \qquad (2)$$

這個方程式在物體形狀越接近圓形時，其方程式值會越小。所以運動計畫器在僅考慮這個形狀分數的狀況下，將產生一個能讓物體盡量保持圓形的

Algorithm: Reshapable_BFP

Input. S_{init}, goal cell C_g, CCMAP, and F
Output. path P.

1. Initialize a priority queue PQ according to the objective function F
2. $SUCCESS$ = **false**;
3. Insert S_{init} into PQ
4. **while** PQ is not Empty and $SUCCESS$ = **false**
5. **begin**
6. S_i = Dequeue(PQ)
7. S' = Generate_new_config(S_i)
8. **if** S' is **nil then**
9. Update CCMAP with S_i
10. **else**
11. Insert S_i into PQ
12. Update CCMAP by S'
13. **if** S' can cover C_g **then**
14. **begin**
15. $SUCCESS$ = **true**
16. $S_{goal} = S'$
17. **end**
18. **if** S'/CC_{Inter} is not EMPTY **then**
19. Insert S' into PQ
20. **end**
21. **if** SUCCESS = **true then**
22. **return** the constructed path by tracing the configuration from S_{goal} back to S_{init}
23. **else return failure**

圖 6： 路徑搜尋演算法

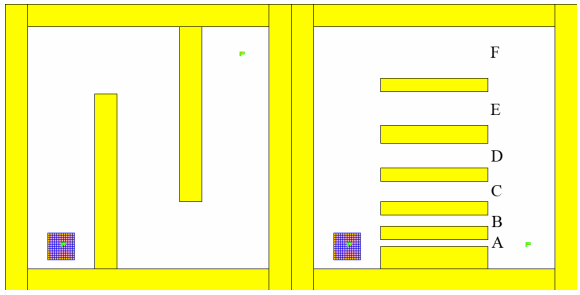
物體移動路徑。

另一方面，距離分數是用來評量一個組態與目標地的距離函數。與一般距離函數不同的是，我們必須將環境中的障礙物考慮進去，而且距離的定義必須建立在格點整體集合而非單一格點。因此，我們的作法是在工作空間中以目標點為中心，以類似 NF1 位能場的方法[4]，產生一個無區域最小值的人工位能場。此位能場將環境中的障礙物加以離散化後再藉以產生位能場，因此能反映移向目標地所需之真正距離。再者，此距離分數的計算方式是採用組態中所有格點的平均位能場，因此能將物體形狀變化後整體位置的距離加以考量。

（二） 運動計畫器演算法

在設定好環境的相關資訊之後，我們必須根據物體的起始與目標位置，產生物體的運動路徑。在運動計畫器中，我們的目標函數是希望透過設定方程式(1)中不同權重的 α 與 β 值，產生以物體移動為優先或是物體形變程度為優先的路徑。

圖 6 是我們所提出來的搜尋路徑演算法。此演算法是根據每個組態的目標函數所得到分數高低，決定該由哪個組態產生新的組態，藉以找到連上目標點的組態，及其間所產生的物體運動路徑。



(a) 場景一 **(b) 場景二**

圖 7 測試用場景

我們使用了優先佇列紀錄已產生組態的目標函數分數，在每一個回合取出一個分數最低的組態，以嘗試產生新組態。如果沒辦法產生新的合法組態，就會更新 CCMAP 中對應位置的狀態。反之，就繼續放入優先佇列中等待產生新的組態。這個動作會持續到涵蓋目標格點的組態出現或是優先佇列已空卻無法產生新組態的時候停止。在涵蓋目標格點的組態找到後，我們可透過追蹤每個組態的原生組態(parent)，回溯得到一條路徑。

此演算法與一般最佳優先的搜尋演算法最大不同的地方，在於第 7 行至第 11 行的程序。一個組態是否為合法的組態，在於其是否有部分的邊界格點也在 CCMAP 的邊界格點上，如此才能產生一個新的鄰居組態。如果一個組態由於 CCMAP 擴展的結果，已不在 CCMAP 的邊界上，則此組態將從 PQ 裡被移除；反之，我們將會把此舊組態（S_i）繼續放回 PQ 中（第 11 行），藉以產生其他新的組態。

六、 實驗與討論

我們設計了兩個實驗來檢驗我們運動計畫器的能力。第一個實驗是利用圖 7 的場景一來測試運動計畫器對於不同面積大小的物體，在做運動計畫時所需花費的時間比較；第二個實驗則是利用場景二來測試相同面積的物體，在目標函數權重值不同的情況下，運動計畫器所表現出來的效果也不同。場景二中的 A、B、C、D、E、F，分別代表不同寬度的狹縫。

（一） 不同面積的運動計畫比較

圖 8 是設定不同的物體面積後，新產生一個組態所需花費的時間折線圖。由此圖可以得知當我們增加物體的面積時，產生一個組態的時間也會相對應的增加，但時間趨近於線性且穩定的成長，不會產生在前文中所提到，因為自由度增高而導致空間與時間複雜度成指數成長的狀況。對於不同的權重值，形狀分數的權重值越低，計畫所需的平均時間會越長，這是由於該權重值低會讓運動計畫器在產生新組態時以移往目標為優先，導致所產生的組態形狀與障礙物的接觸面積較大，增加每次產生新組態時對障礙物的碰撞偵測次數。

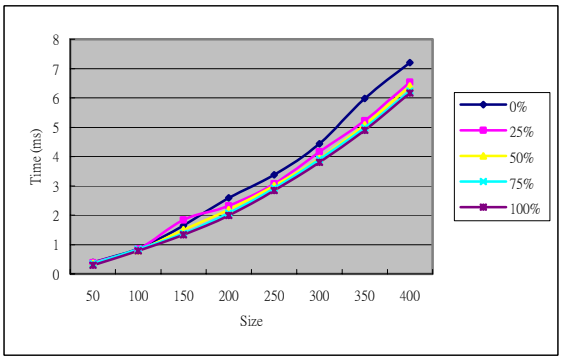


圖 8 不同面積產生新組態所花之時間比較圖

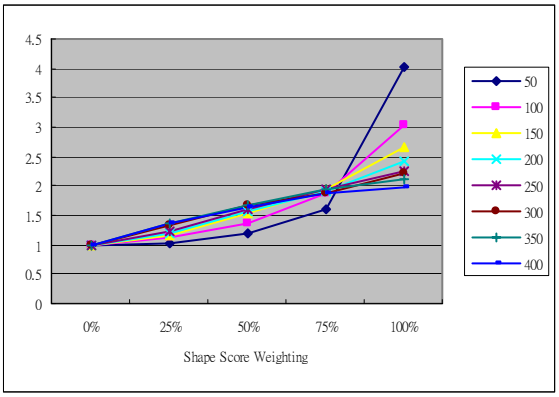


圖 9 正規化後的產生組態/路徑長度數比

圖 3 則是表示在運動計畫的過程中，程式所產生的組態數目與最後產生路徑所用的組態數目比值。這是用來評估每一次所產生的組態在做運動計畫的時候，所產生的效益；比值越低代表所產生的組態利用率越高，運動計畫器在執行上會越有效率。我們以 0%權重值的數據作基準，將各個數值等量縮小作正規化的動作。由圖可以看出當權重值逐漸調高，該比值也是隨著增高，且在約 75%的地方有劇烈變化的轉折臨界點；這個臨界點的位置會與形狀分數和距離分數所表現出來的分數比值有關。在此圖中，我們可以發現其比值會因為物體面積增加而升高；這是因為物體面積越大，要維持較高的形狀分數比較困難，因此在產生的組態中，最後能成為移動路徑中的組態者較少。在臨界點前，距離分數對運動計畫器的影響較大；產生較短的路徑會比產生較好的形狀更為重要，因此比值容易隨著物體面積增加而往上升。在臨界點之後，形狀分數對運動計畫器的影響較大，導致比值會因物體面積的增加而呈反比的狀況。

（二） 不同權重值的運動計畫比較

在實驗二中，我們依序測試了採用不同權重值的目標函數，以瞭解其對運動計畫器產生路徑所造成的影響。圖 10 為不同權重所造成的分數變化圖。

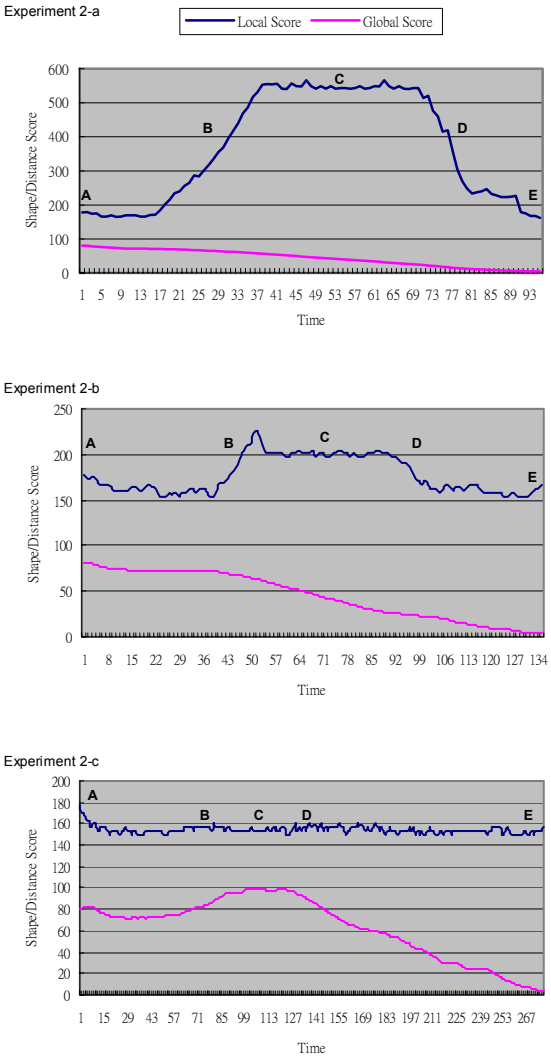


圖 10： 不同權重所造成的分數變化圖

我們設定了實驗 2-a、2-b、2-c 三個測試，其(形狀分數/距離分數)的權重分別設定為(0%/100%)、(25%/75%)、(75%/25%)，而圖表上的 A、B、C、D、E 分別為我們標示出來比較有代表性的時間位置。可在圖 11 看到相對應的物體形狀變化。其中第三個設定(75%/25%)是根據前一小節的實驗所得出的臨界數值。

在實驗 2-a 中，我們可以看見形狀分數所表現出來的曲線在 B-C-D 段明顯高於其他段，但距離分數則是整段遞減；這是由於我們將形狀分數的權重設定為 0%，因此運動計畫器不會考慮物體形變成什麼樣子，而僅參考距離分數值。這使得物體在移動時選了下方狹路的最短路徑，造成形狀因為穿越狹路被拉長，因而導致形狀分數呈現劇烈變化。

在實驗 2-b 中，可以看見形狀分數在 B-C-D 段較其他位置高；這也是因為物體在該時刻進入狹路的關係，因為我們在設定運動計畫器時，將形狀分數的權重值設為 25%；因此計畫產生出的路徑中，

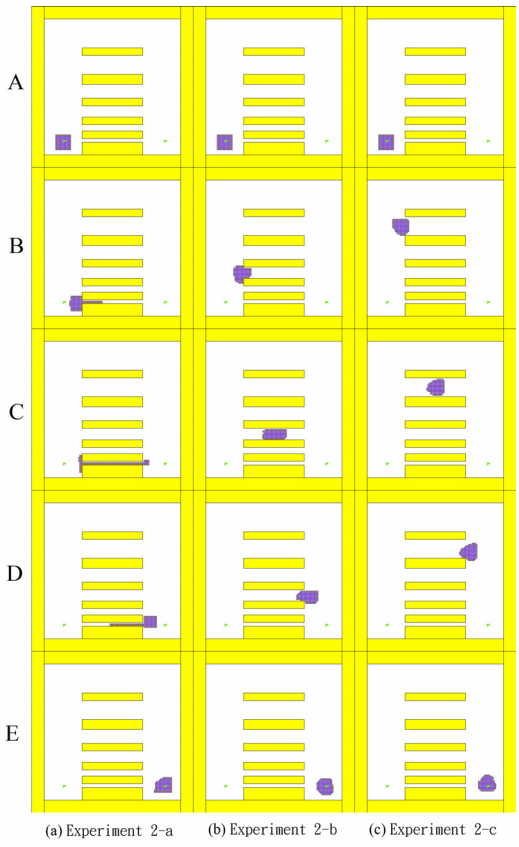


圖 11：實驗裡的關鍵點截圖

形狀的變形不像實驗 2-a 那麼劇烈。也因為有設定參考形狀的分數，所以在 A-B 段可以看見距離分數有下降後稍微增加的趨勢；這是因為運動計畫器要維持較好的物體形狀，所以選擇上方較寬的狹路。

在實驗 2-c 中，整段形狀分數雖然呈現劇烈的起伏變化，但其變化幅度不大，不過距離分數卻在 B-C-D 呈現了明顯的增加；這是因為我們將形狀分數的權重設定為 75%的關係，使得運動計畫器在整段路徑都維持了很好的物體形狀；可是卻選擇繞遠路（距離分數較高）的方式做路徑規畫。

在這個實驗中，可以看出我們所提出的運動計畫器演算法，可以依據我們所調整的權重值，影響運動計畫中對物體運動方式的偏好。

（三） 複雜場景實驗

在這裡，我們設計了圖 12 這個較複雜的場景來實驗本論文所提出的方法。我們所使用的電腦配備有 Intel Pentium IV 2.8GHz CPU 和 1.5GBytes 的記憶體。我們使用 JAVA 5.0 做為軟體開發環境。這實驗場景的大小為 256*256，考慮的物體面積為 1024 單位。圖 12(b)為我們實作的程式執行後，所得到的 CCMAP。在執行過程中，共產生 15375 個組態，其中有 7748 個組態被用作產生新組態，最後所得的物體路徑，是由 1213 個組態所構成。我們的程式花了 302 秒做完整個場警的運動計畫，平

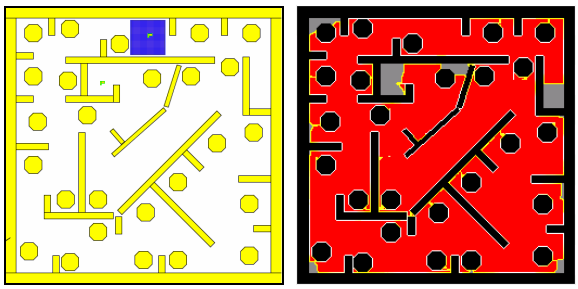


圖 12：(a) 實驗所使用的場景，藍色部分為起始位置，下方旗標為目的地 (b) 實驗所得之 CCMAP

均每個組態的產生時間約為 20 毫秒。圖 13 是此計畫器所得到的結果截圖，可以看見本論文所提出的方法能讓物體在移動的過程中，遇到較為狹窄的地方時，能夠改變形狀以利物體通過，而在較為寬敞的地方時，能盡量維持最佳的形狀分數。圖 14 是我們將程式之路徑結果輸出至 3D 動畫軟體繪圖所得結果。

七、 結論

在這篇論文中，我們將運動計畫器應用在可形變物體的運動自動產生上。我們利用將該物體離散化的方式，設計了組態的模型來表示物體所具有的形狀、面積與位置。我們提出了 CCMAP 的觀念來輔助運動計畫器在產生新的組態時，能保證持續有所進展。我們也利用了形狀分數與距離分數的概念設計目標函數，以做為最佳優先搜尋演算法的搜尋條件。藉由改變此目標函式，可以讓運動計畫器產生不同特性的物體運動。例如，透過設定較低的形狀分數權重，可產生黏滯程度較低的物體運動，而設定較高的權重則產生形狀維持較好的物體運動。我們也以數個實驗證明了本論文所提運動計畫演算法的有效性，並分析了不同參數對計算效能或產生結果的影響。

八、 致謝

此研究在國科會 NSC 93-2213-E-004-001 計畫的支助下完成，特此致謝。

九、 參考文獻

[1] A. Kamphuis and M. H. Overmars, “Finding Path for Coherent Geoups using Clearance,” *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* , pp.193-202, 2004.

[2] C. W. Reynolds, “Flocks, Herds, and Schools: A Distributed Behavioral Model,” *Computer Graphics: ACM SIGGRAPH*, pp.25-34, 1987.

[3] C. W. Reynolds. “Steering behaviors for autonomous characters,” *1999 Game Developers Conference*, pp.763-782, 1999.

[4] J.C. Latombe, *Robot Motion Planning*, Kluwer,

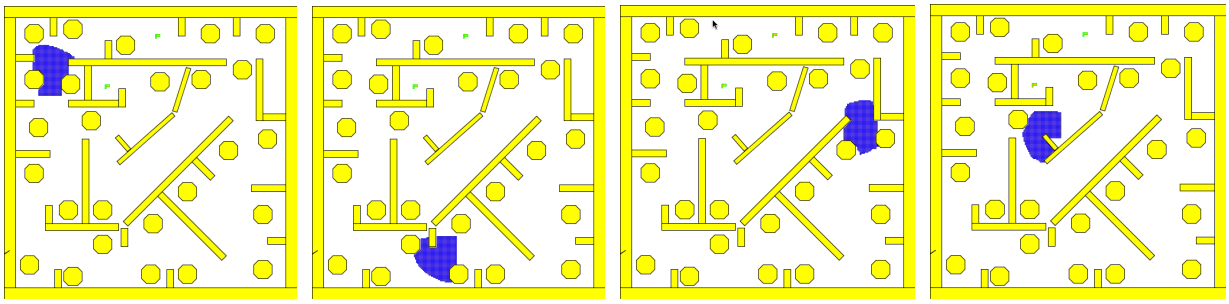


圖 13： 實驗所得路徑之截圖，依序為圖形移動順序

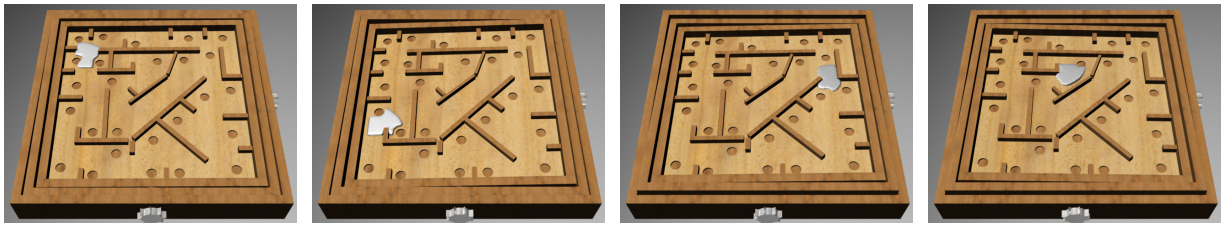


圖 14： 將實驗結果輸出至 3D 動畫軟體繪圖所得的結果

Boston, MA, 1991.

[5] N. Foster and R. Fedkiw, “Practical Animation of Liquids,” *ACM SIGGRAPH Symposium on Computer Animation*, 2001.

[6] O. B. Bayazit, J.-M. Lien, and N. M. Amato, “Better flocking behaviors in complex environments using global roadmaps,” *Proc. of the 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp.528-534, 1995.

[7] N. Rasmussen, D. Enright, D. Nguyen, S. Marino, N. Sumner, W. Geiger, S. Hoon, R. Fedkiw, “Directable Photorealistic Liquids,” *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp.193-202, 2004.

[8] S. Patel, J. Cohen, A. Chu, and F. Pighin. “Fluid Simulation via Disjoint Translating Grids,” *ACM SIGGRAPH*, Los Angeles, 2005.

[9] Y. Koga, K. Kondo, J.J. Kuffner, and J.C. Latombe, “Planning motions with intentions,” *Proc. of SIGGRAPH '94*, pp. 395-408, 1994.