# 第四章

# 舞獅的路徑規劃

介紹完本系統的各元件之後，本章將對舞獅演員運動計畫器的部分做詳細的介紹。4.1 節中我們會對運動計畫過程中的組態及組態空間做定義；4.2 節介紹組態合法性的定義；4.3 節將介紹舞獅在運動計畫過程中有哪些動作可符合我們的規範；4.4 節將介紹我們所使用的舞獅運動計畫演算法；4.5 節介紹在計畫出舞獅人物的運動路徑後，如何決定虛擬人物的重心位置。

## 4.1 組態的定義

在我們的研究中，舞獅是由兩個虛擬人物所組成。前方的是主要演員(head dancer)，引領舞獅前進的方向和控制獅頭的動作；後方的則是輔助演員(tail dancer)，跟隨主要演員前進。輔助演員手的位置是固定在主要演員的腰上，而前後兩演員共有四隻腳。我們對一個組態(configuration)的定義是這四隻腳所踩的樁的的 *X-Y-Z* 座標值或編號。換句話說，也就是虛擬場景中前後兩演員所站的那些梅花樁，因此一個組態可表示為：

$$\boldsymbol{C_i}: (S_a, S_b, S_c, S_d)$$

$S_i$ 表示其中一支樁的序號。組態定義完成後，接著我們描述組態空間(configuration space)。在本研究的舞獅問題中，組態空間是根據虛擬人物腳的數量和場景中梅花樁的數量而決定的。前後兩演員共有四隻腳，因此組態空間的維度為四維。假設梅花樁的數量為 $n$，問題的複雜度為 $n$ 的四次方。因此我們的運動計畫，便是在此組態空間中搜尋合法的組態。

## 4.2 組態的合法性

在上述龐大的組態空間中要找到一組連續且合法的路徑，似乎不是一件簡單的事。我們以兩個階段的方式逐一去除不合法的組態。第一步是在系統初始梅花樁環境時，透過梅花樁之間相鄰關係的建立來大幅減少原本要拜訪的的鄰居數目。假設原本在找尋下一個組態時所需要拜訪的目標為 $n$ 個，但是建立梅花樁的鄰居關係後，搜尋時所需拜訪的目標只剩下各個梅花樁的鄰居而已。當梅花樁個數很多時，可以透過鄰居關係的限制條件減少許多的運算時間。第二步則是透過一些符合虛擬人物運動習性的規則，更準確的篩選掉一些不合法的組態。我們所使用的判斷準則包含以下原則：

1. 兩演員的任兩隻腳不能同時落在相同的梅花樁上。

2. 輔助演員落腳點的相對位置不能超前於主要演員的落腳點。

3. 舞獅演員本身的雙腳距離不能分太開，必須符合人物模型的限制。

4. 兩組相鄰腳步的高度差，不能超過某個限制。

5. 一組態中兩演員的腳步之間有距離限制，避免違反輔助演員手部與主要演員的連結限制。

6. 先後的相鄰兩組態，主要演員的面向必須與前進方向一致。換言之，就是不能倒退走。

第一點的作法是，系統在初始梅花樁時，會給每個梅花樁一個獨一無二的編號，因此，在搜尋時只要限制每一演員的兩腳所踩的梅花樁編號不能相同即可確保；在實作第二點準則之前，我們首先去除兩演員腳部座標連線有相交的組態，這邊所用到的座標，皆是將 3D 模型投影在二維平面上。而實作第二點準則的方法，可以參照圖 4.1 所示。一開始我們將演員腳部座標的位置向量化，主要演員右腳到左腳的向量，定義為向量 A。輔助演員右腳到左腳的向量，定義為向量 B。輔助演員兩腳座標連線中點位置到主要演員兩腳座標連線中點位置的向量，定義為向量 C。向量 A 自身旋轉負九十度，定義為向量 A'。

這些向量定義好之後，我們計算向量 A'與向量 C 的內積，並限制它們的內積必須大於零，目的是要讓這兩向量夾角的絕對值小於九十度，如式子 4.1 所示。

$$\vec{A} \cdot \vec{C}' > 0 \tag{4.1}$$

站在虛擬人物的角度來看，是要讓輔助演員的相對位置必須在主要演員後方。圖 4.1 中我們舉了幾個例子來說明上述判斷組態合法性的方法。(a)圖和(c)圖為這個方法判斷出來合法的組態，從圖中得知向量 A'和向量 C 的夾角都小於九十度，而(b)圖中輔助演員的相對位置已經跑到主要演員的前方，所以計算出來的夾角會大於九十度，為不合法的組態。(d)圖中由腳部座標向量得知，主要演員與輔助演員呈現背對狀態，但是經由上述的方法會判斷為合法組態，因此我們多加了一個判斷準則，限定向量 A

與向量 B 的夾角必須小於九十度，如式子 4.2 所示，確保主要演員與輔助演員在一定範圍的角度內面向是一致的。

$$\vec{A} \cdot \vec{B} > 0 \tag{4.2}$$

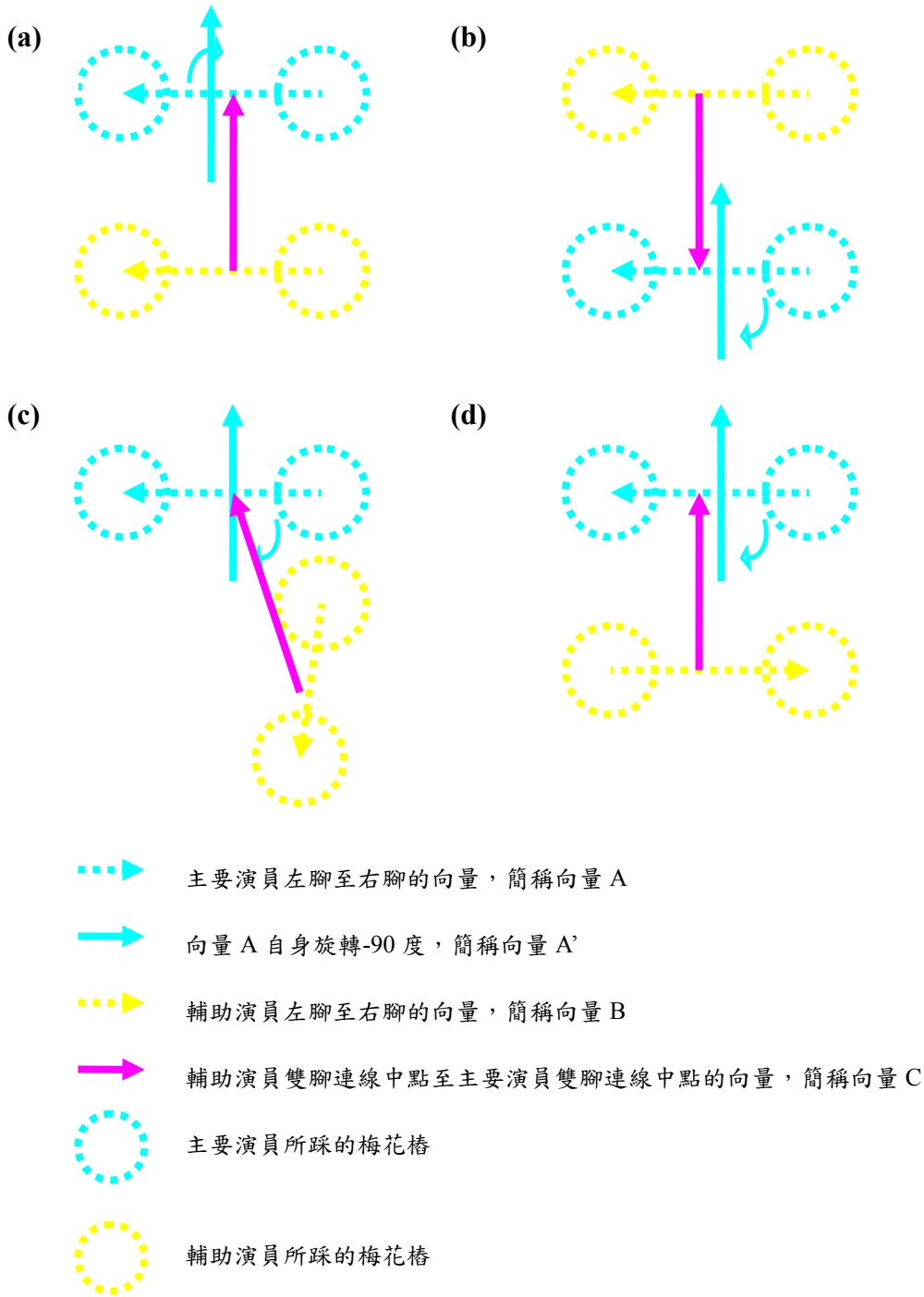


圖 4.1：組態合法性示意圖：(a)(c)為合法組態，(b)(d)為不合法組態

第三點準則的目的是要讓演員雙腳的距離不能分太開，造成不符合虛擬人物模型規範的情況，因此我們會加上距離限制，上限為 $d_u$，下限為 $d_l$，雙腳距離必須在此範圍內才算合法；第四點準則的目的是避免一個演員雙腳距離高度差過大的情形，在這邊我們會以一個高度限制 $h$，作為高度差的限制。第五點準則說明的是輔助演員的雙手，系統預設是扶在主要演員的腰上；因此，我們在規劃路徑是不能違反這個限制。所以我們的作法便是在兩演員腳部之間的距離。加上限制，不能超過上限 $r_u$，也不能小於下限 $r_l$。

經過上述五點判斷準則，已經排除虛擬人物腳步交錯或是踩不到的組態值，而第三、第四和第五點判斷準則同時也是我們決定殘餘的候選組態中那個是較好組態的根據。挑選最好組態的原因與我們所使用的運動計畫演算法有關，之後的章節會再詳細介紹。舉例來說，演員雙腳距離較近的組態、前後兩演員距離較適中的組態等，都可能是我們認定為較好的姿勢，同時在排序候選組態的分數上，可以獲得較佳的被選取順位，原因我們在下一節會再詳細說明。

第六點準則是確保搜尋出來的組態不能讓虛擬演員後退，作法是透過檢查主要演員的面向與前進的方向是否同向。此作法類似前面的方法，首先算出先後相鄰兩組態的前進向量 D，再與主要演員的面向向量 E 做內積，限制夾角必須小於九十度即可，如式子 4.3 所示。

$$\vec{D} \cdot \vec{E} > 0 \tag{4.3}$$

## 4.3 虛擬位能場及目標函數

在運動計畫器中虛擬位能場的計算與 Latombe[7]所提的 NF1 演算法類似，而此虛擬位能場(potential field)[7]的目的是來導引搜尋的方向。虛擬位能場是由終點組態出發，因此終點的位能最低。根據場景中梅花樁距離終點的距離漸漸擴散出去，直到所有與終點有連通的梅花樁皆被拜訪過為止。圖 4.2 為我們在場景中建立位能場的示意圖，圖中白色的樁為位能最低點，色調越深的綠色代表位能越高。一個場景中，我們

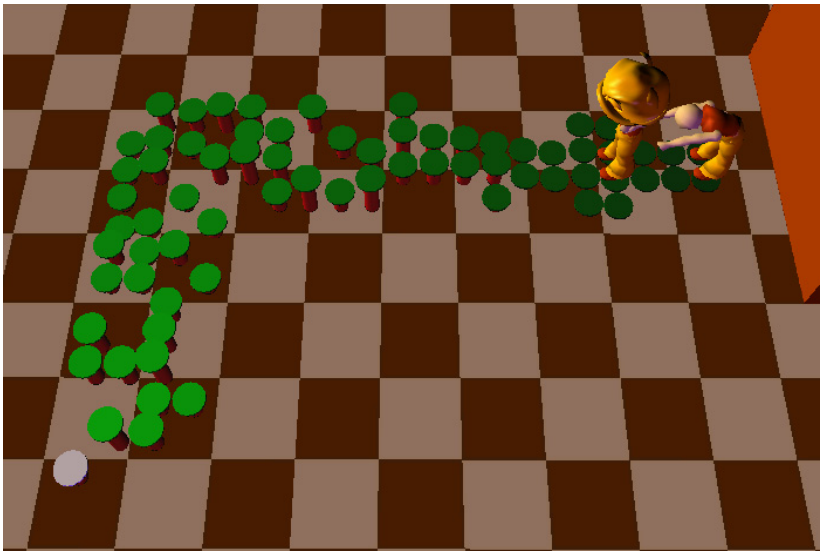

圖 4.2：虛擬位能場示意圖

會替一個組態的四個成員(即四隻腳)分別建立一個獨立的虛擬位能場。

虛擬位能場在搜尋過程中，只能導引搜尋前進的方向，不能決定該組態的好壞。因此我們根據 4.2 節組態合法性準則中的第三、四及五點和虛擬位能場來建立我們的目標函數。目標函數的輸入值是組態 $q$，如式子 4.4 所示，包括兩演員四隻腳所踩梅花樁的座標資訊，$S_{HLF}$ 與 $S_{HRF}$ 分別代表主要演員的左腳與右腳，$S_{TLF}$ 與 $S_{TRF}$ 分別代表輔

助演員的左腳與右腳。目標函數包含四個子函數，而子函數所獲得的值越小，表示在搜尋過程中有較高的優先權。式子 4.5 中主要演員及輔助演員雙腳的距離，分別表示為 $Dis\tan ce(S_{HLF}, S_{HRF})$ 與 $Dis\tan ce(S_{TLF}, S_{TRF})$；而主要演員和輔助演員的距離，則表示為 $Dis\tan ce(S_H, S_T)$，其中 $S_H$ 和 $S_T$ 代表主要演員與輔助演員骨盆的座標。如圖 4.3 所示這邊我們為舞獅演員定義一個自然姿勢。而演員姿勢與自然姿勢中的雙腳距離及雙人距離差距越大，所獲得的優先權值就會比較低。式子 4.5 所反映的就是 4.2 節中所提到要獲得較佳的姿勢。在說明式子 4.6 之前，我們先定義一個位能場百分率 $P$ 代表與終點距離的百分比。我們把整個場景中位能場值最大值到位能場最小值（終點組態）的區間正規化到 1~0 之間，以百分比的方式呈現。所以式子 4.6 描述的是在位能場百分率 $P$ 下，對一開始搜尋的組態而言，主要演員面向與起始組態主要演員面向夾角較小的會得到比較高的優先權值；式子 4.7 描述的則是在位能場百分率(1-*P)*下，對較接近終點的那些組態而言，主要演員面向與終點組態主要演員面向夾角較小的的會得到較高的優先權值。換言之，式子 4.6 和 4.7 根據組態在路徑中的位置決定，計算的是此組態與起始和終點越符合的程度。式子 4.8 則是描述一個組態的位能場成本總和，代表與終點組態的距離。最後將式子 4.5~4.8 根據個別函式分數分佈的範圍加以正規化後，再乘上權重後加總起來，如式子 4.9 所示。我們可以根據場景的不同或使用者偏好來調整這四個子函數的權重 $a_i$，以搜尋出各類型的路徑。

$$q = (S_{HLF}, S_{HRF}, S_{TLF}, S_{TRF}) \tag{4.4}$$

$$f_1(q) = Dis\tan ce(S_{HLF}, S_{HRF}) + Dis\tan ce(S_{TLF}, S_{TRF}) + Dis\tan ce(S_H, S_T) \tag{4.5}$$

$$f_2(q) = P \times InitialAngle(q) \tag{4.6}$$

$$f_3(q) = (1-P) \times GoalAngle(q) \tag{4.7}$$

$$f_4(q) = Potential(q) \tag{4.8}$$

$$GetCost(q) = \sum_i a_i \times f_i(q) \tag{4.9}$$


圖 4.3：虛擬人物距離示意圖，A 線段表示主要演員雙腳距離，B 線段表示輔助演員雙腳距離，C 線段表示主要演員與輔助演員之間的距離

## 4.4 舞獅運動的型態

將我們系統中運動計畫過程的組態定義清楚後，接下來將定義舞獅人物有哪些移動類型，及它們出現的時機。這些合法移動大致分為三類：第一種為行走型態，第二種為跳躍型態，第三種為轉身型態。

行走型態是舞獅演員最主要的運動型態，當舞獅人物由一個組態行走至下一個組態時並不是一步完成的，而是經過兩次的步伐轉換。更明確的說，例如主要演員的第一個動作是跨出左腳，輔助演員搭配的是跨出右腳，以上算一個步伐。接著主要演員跨出右腳時和輔助演員跨出左腳，這算是第二個步伐。這兩次的步伐轉換便是一個行走型態的過程。步伐轉換過程中，我們也會限制兩腳位置改變所表示的組態必須是合法組態。

跳躍型態一般會出現在兩種狀況下：第一種是發生在當舞獅人物，無法透過行走由一個組態到另一個組態的時候，也就是說場景中有一段梅花樁之間的距離過大。這時候可以藉由舞獅人物的跳躍來完成。第二種情形是當跳躍這個組態在優先權評選上超過行走組態，也會發生跳躍的情形。跳躍的步伐是在同一個時間點同時移動兩隻腳，而且兩隻腳可以落在不同梅花樁上。

最後一種為轉身型態，轉身動作跟前兩種動作不太一樣，它不是組成舞獅演員動畫的主要動作之一，轉身主要是發生在當前後兩個組態中主要演員的面向不同的時候，必須改變動畫中演員面對的方向，讓動畫看起來比較自然而制訂。例如當演員走向梅花樁的盡頭時，假設要回到原出發點，就必須透過轉身來改變獅頭的方向，才能繼續進行後續的動作，或者是當梅花樁的場景呈彎曲的形式分佈時，轉彎的地方就必須透過轉身來達成連續的動作。

## 4.5 路徑規劃演算法

我們所使用路徑規劃的方法是最佳化優先搜尋演算法(Best-First Search Algorithm)，該方法的特性是在搜尋過程的每一回合裡，我們將該回合所擷取出來的合法組態，放置到一個分數佇列中。分數的計算方式是根據 4.3 節中所定義的式子 4.9，再依據搜尋的特性作優先權排序，每回合從這個佇列中挑出最佳的組態。在拜訪每一組態的鄰居時，我們會建立一條連結到上一個父親組態。最後，追蹤終點組態到起點組態的連結所產生的組態集合就是舞獅運動的路徑。圖 4.4 是我們所使用的路徑規劃演算法虛擬碼。一開始先設定舞獅人物運動的初始位置 $q_i$ 和終點位置 $q_g$ 的組態值。$L$ 是一個開放式的陣列(Openlist)用來存放搜尋過程中所產生的合法組態，並依照 4.2 節中的式子 4.6 計算出各組態的分數，依此分數的不同放在陣列中不同的位置。起始和終點組態設定好之後，接著將 $q_i$ 插入 $L$ 中，並且標記該組態已經拜訪過以及紀錄該組態的分數。第 5 行 *while* 迴圈的終止條件是當已經搜尋到路徑至 $q_i$ 或是 $L$ 內已經沒有任何組態，也就是說在起點和終點之間找不到一組合法且連續的組態。進入 *while* 迴圈之後，$q$ 所指引的是陣列 $L$ 中分數最低的組態，如果本身是第一回合，所指的就是起始組態，第 7 行 *for* 迴圈所做的是將所有與 $q$ 相鄰且合法的組態 $q'$ 挑選出來，並加到佇列 $L$ 中，這些組態必須是未被拜訪過的，並將該組態建立一條連結至 $q$。第 12 行所做的是更新 $C_{min}$ 的值，確保它永遠指向 $L$ 中擁有最低分數的組態。當拜訪到的某個組態等於終點組態，第 13 行的條件便會成立，搜尋終止。最後由運動計畫所產生的路徑便可透過追蹤 $q_g$ 到 $q_i$ 的所有組態來完成，如演算法中第 16 行所示。假如陣列 $L$ 已經變為空陣列，且尚未產生一條 $q_i$ 到 $q_g$ 的鍊結，則表示在這組態空間中找不到一條合法的路徑。

**Algorithm:** Chinese Lion Dance BFP

**Input:** start & goal configuration $q_i$, and $q_g$
**Output:** path $P$
**begin**
1 PathFound = **false**;
2 append $q_i$ to $L$;
3 mark $q_i$ as visited;
4 $C_{min}$ = GetCost($q_i$);
5 **while** !PathFound or !Empty($L$)
6 $q$ = removeFirst($L$);
7 **do for** each configuration $q'$ adjacent to $q$
8 **if** $q'$ is a legal and unvisited configuration
9 **then**
10 insert $q'$ to $L$ with a pointer toward $q$;
11 mark $q'$ as visited;
12 **if** GetCost($q'$) < $C_{min}$ **then** $C_{min}$ =GetCost($q'$);
13 **if** $x = q_g$ **then** PathFound = **true**;
14 **if** PathFound **then**
15 return $P$ by tracing the pointers from $q_g$ to $q_i$;
**end**

圖 4.4：運動計畫演算法

## 4.6 人物重心位置的決定

路徑規劃完成後，在產生動畫之前，我們尚須決定虛擬人物的骨盆座標。舞獅路徑規劃完成後，產生的是虛擬人物落腳點的座標，我們必須再估算出舞獅演員骨盆的位置，才能符合下一階段產生動畫時所需求的參數。因為單純只知道虛擬人物雙腳的座標，要來推算人物重心的高度和膝蓋彎曲的程度並不容易，而且虛擬人物雙腳在梅花樁上時不一定位於同一個水平面，要快速求出重心的位置尚無一定的方法。我們採

用一個大略的算法，首先計算出虛擬人物雙腳的距離 D，以虛擬人物自然站立時骨盆到腳的距離 R 做半徑畫一個圓，如圖 4.5 所示，在圓中以 D 為底邊做等腰三角形，而三角形的高就是我們估算虛擬人物重心的高度。因此，只要輸入虛擬人物雙腳距離 D，便可以式子 4.10 推算出人物重心的高度；而重心位置的水平分量，是與兩腳中點位置的水平分量相同。

$$H = GetPelvisHeight(D) = \sqrt{R^2 - \left(\frac{D}{2}\right)^2} \tag{4.10}$$

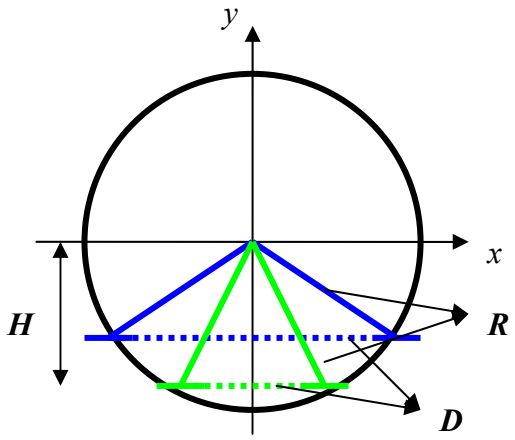


圖 4.5：虛擬人物重心推算示意圖