以運動擷取資料最佳化程序式動畫

梁長宏
國立政治大學資訊科學系
g9606@cs.nccu.edu.tw

李蔡彥
國立政治大學資訊科學系
li@nccu.edu.tw

摘要

程序式動畫是一種根據使用者所提供的高階運動參數，自動產生動畫的方法。藉著高階的運動參數，程序式動畫比運動擷取資料有著更高的彈性。使用者可透過調整參數，輕易地讓動畫滿足情境上所需的限制。但如何調整適當的運動參數以產生擬真的動畫仍屬不易，因此程序式動畫常有在視覺上觀感不自然的問題。本研究的目標是以運動擷取的資料最佳化程序式動畫的參數，以改進所產生動畫的品質。我們的系統可以參考一段運動擷取資料，以最佳化演算法，自動調整程序式動畫的參數，搜尋能產生出與運動擷取資料最為相似的運動參數。多組最佳化過後的運動參數並可以再透過內插，重新產生出一組符合限制需求的運動參數。實驗結果顯示，我們的方法不但使程序式動畫得以保留原來彈性的優點，也改善了程序式動畫常有的視覺觀感不自然的缺點。

關鍵詞：人體動畫、程序式動畫、最佳化演算法、模擬退火法

1. 前言

1.1 研究動機與目的

人體動畫製作一直是十分具挑戰性的工作。一般而言，目前製作人體動畫最常用的製作方式可分為兩種，一種是關鍵格動畫（Keyframing），另一種是運動擷取（Motion Capture）。關鍵格動畫是由動畫師直接將關鍵格和內插資訊輸入到電腦中，電腦再依此輸入產生動畫。此種方式的優點是動畫師擁有全面的控制，因此動畫師的自由度極高，但此優點卻也是缺點，因為動畫師必須控制全部的自由度，費時費力。因此以此種方式所設計的動畫，製作成果的好壞仰賴使用者的專業程度，對一般經驗較缺乏的使用者而言，較難以使用這種方式製作出自然流暢的動畫，而且短短幾秒鐘的動畫就可能花上許多的時間。另一種人體動畫的製作方法則是以運動擷取的方式，以設備直接捕捉人體的運動後，再輸入到電腦產生動畫格。

關鍵格動畫和運動擷取都是讓使用者直接操作低階的資料（關鍵格或內插），所以難以被彈性地調整，進而被應用在不同的情境下。例如，在平地上走路和在崎嶇地形上走路，雖然是同類運動，但在運動模式上仍不太一樣，因此還是得重新調整或擷取，才能得到符合需求的動畫。近年來許多動畫技術的研究都是在如何將擷取下來的動作，做進一步的變化以加值運用在其他應用環境下。但許多這些研究都是在訊號處理（Signal Processing）的層次上，因此所能做到的變化畢竟有限。為了讓某一種運動能夠彈性地調整，只對動畫曲線做訊號層次的後處理通常是不夠的，我們必須知道該類運動的特徵，分析每一步動作背後的意義和限制（Constraints），才能根據這些資訊建構出合理的運動模型及正確的演算程序。我們稱這類以演算程序產生動畫的方式為「程序式動畫（Procedural Animation）」。

我們可以將一個程序式動畫的模組看成是一個黑箱子，這個黑箱子的輸入是多個運動參數，輸出則是一段動畫。不同種類的運動會有不同的運動參數；例如，走路的運動參數有步伐大小、步伐高度及擺手角度等。有些運動參數（如步伐大小）對使用者來說很直觀，但也有些運動參數的含意較不明顯，即使使用者了解該參數的含意，但仍難掌握該參數對於動畫的影響。因此，要設計出自然擬真的動畫，仍需使用者以嘗試錯誤法（Trial and Error）調整運動參數。

本研究的目的在以運動擷取資料做為參考，自動調整程序式動畫的運動參數，以使程序式動畫的結果儘量接近所參考的運動擷取資料。如此一來，使用者便可以只輸入一段理想的運動擷取資料，我們的系統便能自動嘗試運動參數的各種組合，自動調整出與運動擷取資料相近的動畫。另一方面，我們的系統也可以做為程序式動畫的設計輔助工具。例如，給定一段運動擷取片段（Motion Clip），若程序式動畫模組無法產生出相似的動畫，就表示當初設計的運動模型有問題，這可以讓我們重新檢視運動模型可能的潛在問題，進而改良運動的模型。

1.2 問題定義

我們對動畫程序唯一的假設是它提供多個以實數為型態的運動參數，輸入一組運動參數，動畫程序就可依照參數之要求輸出動畫。我們的目標是設計另一個獨立的最佳化系統，用來自動調整動畫程序所提供之運動參數，讓動畫程序能夠產生觀感自然的動畫。此最佳化系統對動畫程序內部如何產生動畫的演算法沒有做任何假設；換言之，對最佳

化系統來說，動畫程序是一個黑箱子，它只需處理動畫程序的輸入和輸出。

我們透過收集得來的運動擷取資料來協助最佳化系統進行最佳化。我們可以定義一個目標函數，以測量動畫程序輸出之動畫與運動擷取資料之間的差異度，然後讓最佳化系統不斷地嘗試各種運動參數組合，直到將目標函數的值降到足夠小為止。當目標函數的值愈小，就表示程序式動畫與運動擷取資料愈相似，由於運動擷取資料具有擬真的特性，所以程序式動畫愈接近運動擷取資料，就代表程序式動畫也愈具有擬真性。

每提供一個運動擷取片段，最佳化系統可以幫我們找出一組運動參數，使得動畫程序可以產生與該片段相似的動畫（即自然的動畫）。從許多運動擷取片段，我們可以最佳化出多組運動參數，這些運動參數都有一個共同的特性，就是它們都能讓動畫程序產生出自然的動畫。我們希望從這些已知能產生自然動畫的運動參數，再內插出所需的新運動參數，令輸出之動畫能夠滿足情境上所需的限制。

本論文可分為三個子系統：動畫程序、最佳化系統與運動參數之內插。動畫程序負責產生角色的動作，但不保證其動作的自然與否。最佳化系統則負責用來找出能產生自然動畫的運動參數。最後經由運動參數的內插，我們得以從已最佳化的運動參數中，再合成出所需之新的運動參數。

2. 相關研究

角色動畫的相關研究可大致分為三類：程序式動畫、動力學方法（Dynamics-Based）以及運動擷取方法（Motion Capture-Based）。

程序式動畫是一種以知識為基礎（Knowledge-Based）產生動畫的方法。以人體動畫來說，程序式動畫通常會參考生物力學上的知識，以設計動作的模型。例如，Bruderlin 和 Calvert [1] 利用此方法設計出人類跑步的運動模型。Perlin [6] 提出如何將 Perlin Noise 用在動畫之中，讓角色的動作更加生動。Chen 和 Li [2] 利用反向關節產生走路的局部動作，也使用運動規劃（Motion Planning）技術計算人物在有障礙物之環境中的行進路徑。Throne 等人 [8] 提出了以手繪軌跡來產生動畫的人機介面，除了在底層動畫是以程序方法產生之外，他們還另外處理了手繪軌跡辨識的問題。

動力學方法是建構人體骨架的動力學模型，以物理學上的知識來模擬物體在空間中的運動。此種方法大多會利用控制器（Controller）來控制即時物理模擬環境中的角色。Faloutsos 等人 [4] 提出了一個兩階層式的控制器架構，以解不同控制器連接的問題。控制器的設計至今仍是一大難題，控制器參數的設定與動作本質有很大的關係，不同種的動作就需要不同的控制器參數。Coros 等人 [3] 採用資料驅動（Data-Driven）的方法，先以正向動力學模擬出大量成功行走之控制器參數，在執行期間就利用這些資料內插出符合要求的新參數，並即時地去控制角色，使角色在走路過程仍可以滿足情境限制（如跨越地上之裂縫）。

運動擷取是一種捕捉真實人類動作的技術。由於捕捉下來的動作資料來自真實的人類運動，所以比起程序式動畫或動力學動畫，它們在觀感上最為自然擬真。但動作資料必須做某種程度的修改，才能夠利用動作資料合成出有彈性的動畫。Witkin 和 Popovic [9] 提出動作扭曲（Motion Warping）的概念，將滿足限制的新訊號描述成原訊號的線性組合，修改原來的動作來滿足需求。Kovar 等人 [5] 提出利用動作圖（Motion Graph）來合成角色動作，他們從動作資料庫當中找出相似的兩個動作片段並將其連接在一起，形成動作圖。如此一來，動作圖上的一條路徑就表示角色的動作。

程序式動畫最大的優勢是它所提供的高階參數，這讓使用者或其他程式模組可以輕易地控制動畫的生成，使其滿足情境所要求之限制。因此，相較於動力學方法和運動擷取方法，程序式動畫的控制性較佳，加上它計算量小，讓程序式動畫適合被應用在遊戲和虛擬環境上。然而，程序式動畫主要問題之一是運動模型的設計；運動模型好壞決定動畫的品質，而運動模型並不容易設計，所以程序式動畫常有觀感不自然的缺點。本研究的目標之一便是解決這個問題，我們提出利用最佳化方法，並參考運動擷取資料，讓系統自動幫我們調整出可以較好的參數組合，以產生觀感上較為自然的動畫。

3. 走路動畫之設計

設計一個運動程序有兩大步驟。第一，定義提供使用者的高階運動參數，以走路來說，可能包括步伐大小、抬腳高度、臀部擺動和擺手角度等。第二，定義關鍵格和內插，以及它們與運動參數之間的關係。在這個章節裡，我們將敘述我們是如何設計一個足以產生不同風格走路的動畫程序。

3.1 走路的運動參數

走路是一種循環性運動，每個走路週期可獨立處理。在我們的走路程序裡，我們用 197 個型態為實數的參數定義一個走路週期，表 1 列出了其中幾個運動參數。每個運動參數皆控制動畫程序所產生的關鍵格或內插。例如，Hip Twist 會影響角色在走路時臀部扭擺的程度。

對於每個運動參數，我們皆定義了有效範圍與鄰居範圍。有效範圍是用來限制參數的最小和最大值，預防使用者輸入的值超過界限。除此之外，定義範圍也可大幅縮小搜尋空間，避免最佳化程式去拜訪我們已知不合法的狀態（最佳化方法在第四章有進一步說明）。鄰居範圍是定義給最佳化程式參考之用。在最佳化過程裡的每一回合中，系統會從

表 1：走路的部分運動參數

參數名稱	有效範圍	鄰居範圍
Step Length	[0, 兩倍腿長]	±腿長/160
Step Height	[0, 腿長]	±腿長/80
Hip Tilt	[-45˚, 45˚]	±2˚
Hip Twist	[-60˚, 60˚]	±2˚
Down Sink	[-100%, 100%]	±2%
Pass Sink	[-100%, 100%]	±2%
High Up	[-100%, 100%]	±2%
Spine Down Lean	[-60˚, 90˚]	±2˚
Spine Pass Lean	[-60˚, 90˚]	±2˚
Spine Contact Lean	[-60˚, 90˚]	±2˚
Phase Frame 1~3	[2, ∞]	無

目前狀態的鄰居範圍中，隨機挑選出一個狀態做為下一回合的起點，在此定義鄰居範圍就是為了讓最佳化程式知道「鄰居」的定義。

3.2 關鍵格與內插

我們將一個走路週期分成四個關鍵格：後腳離地瞬間（Down）、兩腳交錯瞬間（Pass）、重心最高瞬間（High）與前腳踏地瞬間（Contact）。兩個關鍵格之間稱為階段（Phase），雖然我們設計了四個關鍵格，但我們讓階段數依舊保持為三。如圖 1 所示，四個關鍵格依序為 Down、Pass、High 及 Contact。Down 到 Pass 之間的轉變稱為第一階段（Phase 1），Pass 到 Contact 稱為第二階段（Phase 2），Contact 到 Down 稱為第三階段（Phase 3）。這個三個階段時間長度（畫格數）會直接由參數 Phase Frames 1 至 3 分別指定，即 Phase Frames 1 是第一階段的時間長，其餘以此類推。其中第二階段較為特別，它中間另夾有一個 High 關鍵格，對此，我們目前假設 High 關鍵格永遠位於 Pass 和 Contact 的中央，即 Pass 到 High 的時間長度等於 High 到 Contact 的時間長度。

圖 2 (a)-(d) 是走路四個關鍵格下半身的定義。圖 2 (a) 是 Contact 關鍵格，此時是前腳腳跟接觸到地面的瞬間。兩腳一前一後，前腳腳尖翹起某個固定角度，後腳腳跟稍微提起，前後腳跟之間的距離等於步伐大小。圖 2 (b) 是 Down 關鍵格，此時角色的重心在最低點。參考前一個 Contact 關鍵格，將骨盆往前移動步伐大小的四分之一，往下移動參數 Down Sink 所指示的距離，而前腳掌完全平貼地面，後腳腳跟則往上提。圖 2 (c) 是 Pass 關鍵格，此時雙腳交錯，一腳懸空，一腳支撐。利用腿完全伸直時的高度做為參考，將骨盆往下移動參數 Pass Sink 所指示的距離。另外讓懸空腳位於支撐腳的上方，向上抬起 0.75 倍的步伐高度。圖 2 (d) 是

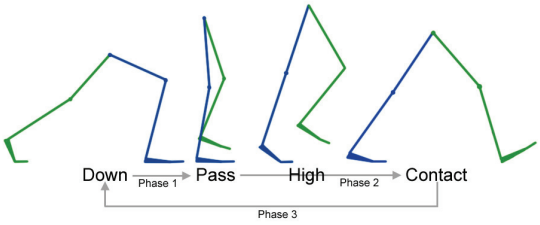
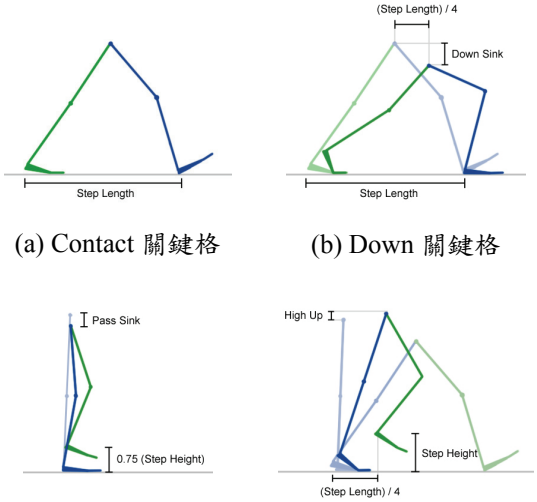


圖 1：走路的四個關鍵格



(a) Contact 關鍵格　(b) Down 關鍵格

(c) Pass 關鍵格　(d) High 關鍵格

圖 2：走路四個關鍵格與運動參數的關係

High 關鍵格，此時角色的重心達到最高點。骨盆往前移動步伐大小的四分之一，往上移動參數 High Up 所指示之距離。懸空腳往前移動步伐大小的四分之一，往上移動到步伐高度的位置。

關鍵格之間的區間則必須透過內插來補足。位置性質的內插（如手腕的移動）我們是透過貝茲曲線去產生關鍵格間的平滑軌跡，而旋轉（如臀部的扭擺）我們則透過 Quaternion 一般常用的 slerp 內插法。我們另設有控制移動或旋轉速度的運動參數，透過調這些參數，使用者可以控制關鍵格之間各部位的運動速度，以達到加速或減速的效果。

4. 程序式動畫之最佳化

我們的最佳化流程可分為三個步驟。首先，在執行期之前，我們會先對運動擷取資料做前處理，包括片段裁剪、骨架轉換及關鍵格標記。經過前處理之後，在最佳化的執行期時，我們會先對齊程序式動畫與運動擷取資料的關鍵格，然後再利用模擬退火法（Simulated Annealing），最佳化程序式動畫所提供的運動參數，使得程序產生之動畫儘可能接近給定的運動擷取片段。

4.1 運動擷取資料的前處理

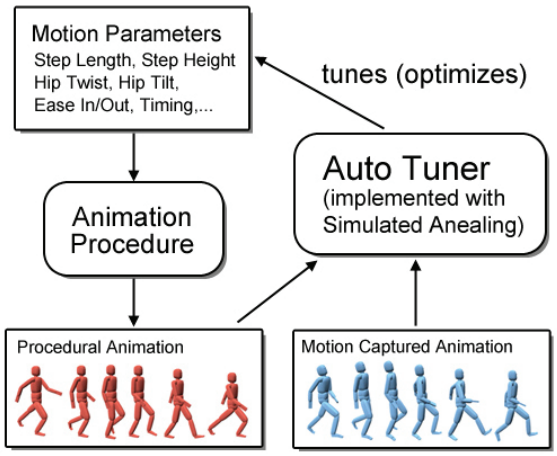


圖 3：最佳化系統架構圖

我們從 CMU Motion Capture Database [10] 挑選並取得的 13 個步伐大小高低皆不同走路動作片段。由於每個原始片段所包含的走路週期數皆不等，而在我們的最佳化程序只需兩個走路週期（即角色行走兩步），所以在前處理的第一步，我們必須先用人工的方式，將每個片段都剪裁成兩個走路週期。

為了對齊程序式動畫和運動擷取片段的關鍵格，在前處理的第二步，我們用人工的方式，為這些運動擷取片段標出關鍵格的位置。方法是先載入運動擷取資料到螢幕上以動畫播放，然後直接以人工方式判斷關鍵格出現的時間點，並標示之。在第三章中，雖然提到我們將一個走路週期切割成四個關鍵格（Contact、Down、Pass 和 High），但由於我們假設 High 關鍵格出現的時間點位於 Pass 和 Contact 的正中間，所以無需特別標出，我們只需標出其他三個較為重要的關鍵格：Contact、Down 和 Pass。

4.2 最佳化方法

我們要解決的最佳化問題是「找出一組運動參數，使得程序式動畫儘可能地接近所參考的運動擷取片段」。一般而言，最佳化問題的兩個要素是最佳化對象與目標函數。如圖 3 所示，我們有一個自動調整運動參數的模組（Auto Tuner），它可持續不斷地去調整動畫程序所提供的運動參數，然後計算程序所產生出的動畫與運動擷取片段之間的差異度，做為它下一步調整運動參數的考量因素，目的是使兩段動畫的差異度儘可能地達到最小。因此，我們的最佳化對象（自變數）是動畫程序的 197 個運動參數（表 1），而目標函數是程序式動畫與運動擷取片段的差異度。

由於運動參數為高維空間中的資料，有著許多局部解，為了克服這些局部解，我們必須藉助機率搜尋演算法來解決此問題。一般來說，模擬退火法是容易實作且效率佳的演算法 [8]，因此我們利用模擬退火法來最佳化程序式動畫的運動參數。模擬退火法是一種全域搜尋演算法，其特點是利用機率跳脫局部解，以逼近真正的最佳解。但在執行模擬退火法之前，我們希望程序式動畫與運動擷取片段關鍵格的時間點一致，所以我們會先對齊兩者的關鍵格，然後才以模擬退火法進行最佳化。

4.2.1 關鍵格對齊

在真正進行最佳化搜尋前，我們會先對齊程序式動畫與運動擷取片段的關鍵格，使兩者關鍵格出現的時間點一致。程序式動畫的關鍵格位置可從運動參數 Phase Frames 1 至 3 得知，而運動擷取片段的關鍵格位置則是透過 4.1 節所述的人工標記法得知。有了兩者的關鍵格位置資訊，我們便可調整程序式動畫裡用來控制階段時間長度的參數 Phase Frames 1 至 3，使它的關鍵格位置與運動擷取片段一致。

4.2.2 目標函數

一個好的目標函數應該要能測量出一段程序式動畫和一段運動擷取片段的差異度。我們將一段動畫表示成一個組態串列，而一個組態即代表角色的一個姿勢。假設以運動參數組合 A 所產生之程序式動畫以 P_A 表示，已經過剪裁處理的運動擷取片段以 M 表示，目標函數定義如下：

$$f(A)=\sum_{i\in keyframes}\begin{pmatrix} w_1\, d(P_A[i],\ M[i])+ \\ w_2\left|\begin{matrix} d(P_A[i-1],P_A[i+1])- \\ d(M[i-1],M[i+1]) \end{matrix}\right| \end{pmatrix} \qquad (1)$$

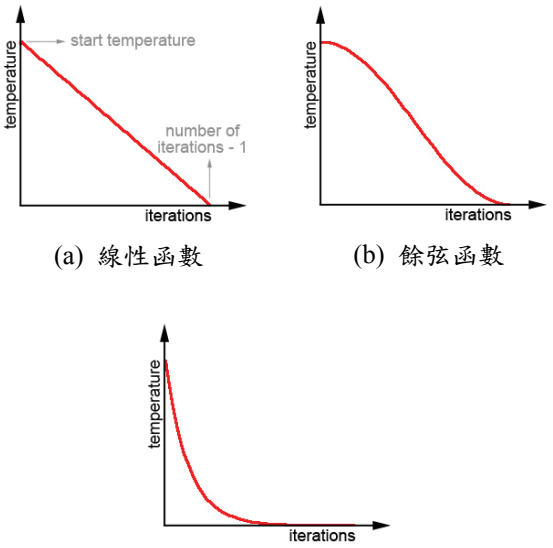
其中 $P_A[k]$ 代表程序式動畫的第 k 格組態（或姿勢），$M[k]$ 代表運動擷取片段的第 k 格組態。d 是距離函數，用來計算兩個組態的差異度。式子可分成兩項，前項 $d(P_A[i],\ M[i])$ 用來測量 P_A 和 M 在關鍵格 i 角色姿勢的差異度，後項 $|d(P_A[i\text{-}1],\ P_A[i\text{+}1])$ - $d(M[i\text{-}1],\ M[i\text{+}1])|$ 用來測量 P_A 和 M 在關鍵格 i 角色各部位運動速度的差異度。其中 $d(P_A[i\text{-}1],\ P_A[i\text{+}1])$ 與 $d(M[i\text{-}1],\ M[i\text{+}1])$ 分別用來表示 P_A 和 M 在關鍵格 i 一階導數的量度（Magnitude），用來近似物理意義上的瞬時速度。w_1 和 w_2 為權重值，在我們目前的實驗中，兩者值皆為 1。

距離函數 d 用來測量兩任意組態的差異度，它的定義如下：

$$d(p,q)=\sum_i w_i \left\| T_p(e_i)-T_q(e_i) \right\|^2 \qquad (2)$$

```
function Simulated-Annealing (schedule, f, S)
        returns a solution state
input: schedule, a mapping from time to
                temperature
       f, an objective function
       S, a starting state
local variables:  current, a state
                  next, a state
                  T, a temperature
current ← S
for t ← 0 to ∞ do
    T ← schedule(t)
    if ( T ≤ 0 ) then return current
    next ← a randomly selected neighbor of
current
    ΔE ← f(next) – f(current)
    if ( ΔE < 0 ) then current ← next
    else current ← next only with probability
exp(-CΔE/T)
```

圖 4：模擬退火演算法



(a) 線性函數　(b) 餘弦函數

(c) 指數函數

圖 5：四種典型的冷卻進度表

其中 p 和 q 表示兩個要被測量距離的組態。e_i 是某個關節 i，以我們的系統為例，我們讓 e_i 依序為頸部（skullbase）、左右手肘（elbow）、左右手腕（wrist）、左右腳踝（ankle）及左右腳（midtarsal）。$T_p(e_i)$ 表示關節 e_i 在組態 p 相對於骨架根關節（Root Joint）的位移，是三維空間中的一個座標點。$\|T_p(e_i) - T_q(e_i)\|$ 是兩座標點 $T_p(e_i)$ 和 $T_q(e_i)$ 間的歐幾里得距離（Euclidean Distance）。根據部位的不同，每一段距離我們都乘上一個權重 w_i，一般來說，由於四肢的末端（手腕與腳踝）是動作最為明顯的部位，所以我們會給予它們較大的權重，其他非末端之部位（如手肘和膝蓋）則被給予較小的權重。

4.2.3 模擬退火法

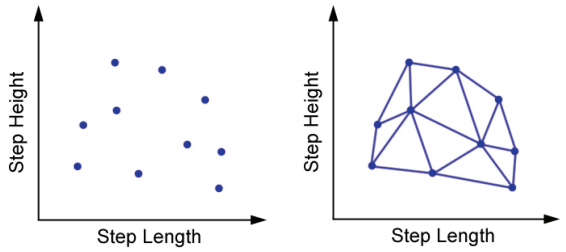
模擬退火法是一種模擬自然界物質降溫現象的最佳化演算法 [7]。自然界物質從高溫降到低溫是一個漸進的過程，倘若溫度降得太快，物質分子的排列無法達到能量最低、最穩定的狀態，所以在自然界中，為了讓物質分子的排列達到最穩定的狀態，溫度是逐漸降低的。當溫度較高時，分子向四周移動的距離較大，降溫過程中總能量可能會升高，但隨著溫度愈來愈低，分子失去了能量，移動的距離就愈來愈小，最後所有分子都固定不動時，物質就達到穩定狀態。

在模擬退火法搜尋過程中，溫度值會隨著搜尋時間逐漸降低。搜尋剛起步時溫度較高，此演算法會嘗試擾動目前的狀態，若擾動後狀態的能量（即目標函數的值）比原來低，則接受擾動結果成為目前的狀態，並繼續拜訪下一個狀態。若擾動結果的能量比原來高，則接受擾動結果的機率為 $\exp(-C\Delta E/T)$，其中 $\exp(A)$ 表示數學常數 e 的 A 次方，ΔE 是擾動結果與原來狀態的能量差（為一正數），T 是當前的溫度，而 C 則是一個使用者定義的常數。由於能量 ΔE 與溫度 T 所用的單位不同，所以常數 C 可用來調節兩者單位之差距，以避免機率值太小或太大。C 值愈大，機率就會被壓制得愈低，所以我們可以把它稱為機率抑制常數（Probability Suppress），其值的大小與骨架尺寸和起始溫度有關，因此不同的情況適用不同的 C 值。

我們參考 [7] 並依需求稍微修改的模擬退火演算法如圖 4 所示，輸入包括一個冷卻進度表（Cooling Schedule）、目標函數及起始狀態。冷卻進度表是一個從時間映射到溫度的函數。在實作上，我們以回合數來代表時間。冷卻進度表必須是一個遞減函數，且必須在有限時間內讓溫度對應到 0 度，如此才能使此演算法終止。起始狀態則是模擬退火法搜尋的起點，可以是預設的參數組合，或是使用者指定之任意一點。

在每一回合裡，模擬退火法會從目前狀態的鄰居範圍中，隨機選出一個狀態做為下一個可能拜訪的狀態。因為 Phase Frames 1 至 3 這三個運動參數在 4.2.1 節所提到之對齊關鍵格時就會被處理，所以在最佳化時可排除這三個參數，這使得一個狀態包含除 Phase Frames 1 至 3 之外的 194 個運動參數。我們在表 1 定義了這 194 個運動參數的鄰居範圍，所以在模擬退火法的每個回合裡，我們會依據所定義的鄰居範圍，隨機擾動（加或減）目前狀態中的每個運動參數，然後視新狀態與目前狀態的能量差，再決定是否要接受新狀態成為目前狀態。

冷卻進度表對模擬退火法搜尋的速度與答案可能會有相當大的影響，但冷卻進度表的好壞並無絕對，不同性質的問題適用不同的冷卻進度表。圖 5 (a)-(c) 是四種典型的冷卻進度表，它們有兩個共同參數可供使用者調整，第一個參數是起始溫度（Start Temperature），第二個參數是搜尋回合數



(a) 步伐大小-步伐高度的二維平面　(b) Delaunay 三角剖分

圖 6：運動參數內插平面的建造過程

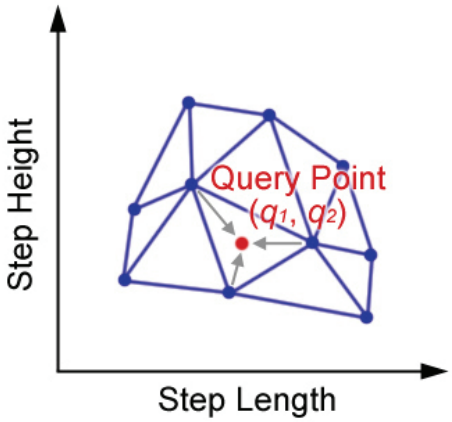


圖 7：運動參數的內插

（Number of Iterations）。以圖 5 (a) 為例，冷卻進度表必須從座標 (0, 起始溫度) 開始，然後在過程中溫度逐漸遞減，最後停留在座標 (回合數-1, 0)。

模擬退火法提供了四個參數可供調整：起始溫度、搜尋回合數、機率抑制常數與冷卻進度表。我們的實驗結果顯示，在四個模擬退火法的參數之中，起始溫度和機率抑制常數的組合特別重要。如果這兩個參數若組合得不正確，不但無法讓目標函數愈來愈低，還有可能會使它攀高。好的參數組合必須經過多次的微調及多次的實驗才能找到。以我們的情況為例，最後我們發現把起始溫度定在 400 左右，而機率抑制常數則定在 2000 左右，對此我們這個問題是較佳的參數組合。此外，以我們的問題來說，我們發現冷卻進度表對最後找出的答案影響不大，影響的是能量收斂的速度。經過實驗，我們歸納出採用指數函數（圖 5 (a)）類型的冷卻進度表，可以讓能量最快收斂，儘早找到答案。

5. 運動參數之內插

在第四章所介紹的最佳化其目的是改善程序式動畫，使其動作更加自然。然而，程序式動畫的最大優點是有彈性，使用者可以控制高階的運動參數，製作出符合所需限制的動畫。以我們的例子來說，我們希望能只需控制角色步伐大小和高度，程序就能自動幫我們產生出自然的走路動畫。第四章所介紹的最佳化方法可以幫我們找出一組可以產生自然動畫的運動參數，但是由於運動參數彼此可能不獨立，所以當我們去修改部分參數（如步伐大小和高度）以符合需求時，可能會造成動作不自然。因此，這一章我們提出將許多最佳化後的運動參數集結起來，當使用者做出部分參數的要求時，我們從這些最佳化後的運動參數去內插出新的、符合使用者要求的參數組合。藉此以保持程序式動畫固有彈性強的優點，同時也能增加動畫的自然度。

5.1 運動參數內插平面

我們的走路程序有 197 個可供調整的運動參數，但我們希望只提供少數較為高階的參數給使用者調整，其他較低階但又決定動作自然與否的參數，則以最佳化後的參數內插，內插過程可完全自動化，無需使用者介入。在我們的應用裡，我們的高階參數只有兩個：步伐大小和步伐高度。我們將這兩個運動參數稱為「限制參數」，它們控制角色雙腳的落點等動作的限制及其大致結構。而步伐大小和步伐高度以外的其他 195 個參數稱為「風格參數」，它們控制動作風格，即動作裡較為細微的元素。我們希望使用者可以直接設定步伐大小和步伐高度（限制參數），以控制角色的腳步位置，而其餘的 195 個風格參數則以內插自動產生。

我們將每一組最佳化後的運動參數各用一個向量 $A_i(a_1, a_2, a_3, \ldots, a_{197})$ 表示，其中 i 是運動擷取片段的索引，在我們的實驗中，我們一共使用了 13 個走路的運動擷取片段，所以 $i = 1, 2, \ldots, 13$。假設 a_1 和 a_2 分別表示步伐大小和步伐高度，我們在二維座標平面上，點出 13 個最佳化後的 (a_1, a_2)。我們希望利用這些點，幫我們內插出其餘參數 $(a_3, \ldots, a_{197})$。為了達到這個目標，在前處理時，我們將每個最佳過後的運動參數前兩項 (a_1, a_2) 視為一個點，並將其分佈在一個二維座標平面上，其中橫軸代表步伐大小，縱軸代表步伐高度，如圖 6 (a)。然後再對這些點做 Delaunay 三角剖分（Delaunay Triangulation），以便之後內插步驟的進行，圖 6 (b) 即是 (a) 的 Delaunay 三角剖分。

有了 Delaunay 三角剖分的結果之後，我們就可以利用這些三角形來幫助我們做運動參數的內插。如圖 7 所示，假設情境所要求的步伐大小與高度為 $Q(q_1, q_2)$，我們在 Delaunay 三角剖分的結果裡，找出一個三角形 T 包含點 Q，利用三角形 T 的三個頂點 $A(a_1, a_2)$、$B(b_1, b_2)$ 及 $C(c_1, c_2)$，此三頂點為二維平面上之座標，再加上除了步伐大小和高度之外的一個運動參數，就可變成三維座標點，即 $A_i(a_1, a_2, a_i)$、$B_i(b_1, b_2, b_i)$ 及 $C_i(c_1, c_2, c_i)$。由於我們的運動參數總共有 197 個，所以 i 值範圍介於 3 到 197 之間。假設通過點 Q 且平行於 Y 軸（上下方向）

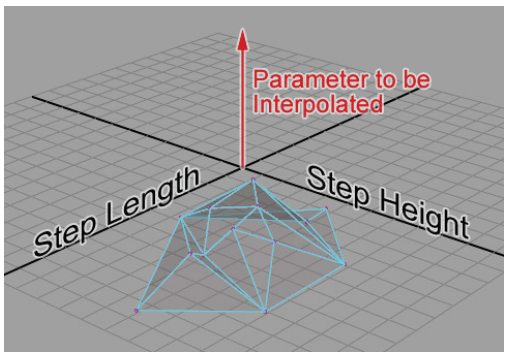


圖 8：以三維空間圖視覺化運動參數的內插

的直線為 L，三個三維座標點 A_i, B_i, C_i 定義出的三維空間中之平面為 E，如圖 8 所示。令 $i = 3, 4, \ldots, 197$，反覆計算出直線 L 與平面 E 之交點 $Q_i(q_1, q_2, q_i)$，我們就可以求出點 Q 的其餘 195 個運動參數。

5.2 風格平面

運動參數內插平面可以再進一步依動作風格分類，以提供使用者對動作風格的控制。然而，所有最佳化過後的參數都被放置在同一個二維座標平面，會造成使用者無法選擇所要產生的動作風格。為了提供動作風格選用的彈性，我們使多個內插平面共同存在，其中每一個平面僅含一種產生特定動作風格的運動參數，並各自獨立建立自己的 Delaunay 三角剖分，我們稱之為「風格平面」。如此一來，我們便可將程序式動畫的風格控制在某一類，或改變到另一風格。

風格平面的建立方法如 5.1 節所述，差別僅在於在平面建立之前，我們會對最佳化後的運動參數依據其運動風格加以分類，並將它們放在不同的內插平面上。以圖 9 為例，圖中包含了三個風格平面，它們分別是從三種不同風格的走路運動擷取片段最佳化出來的運動參數，在圖中它們分別以紅、藍、綠表示。紅色平面為一般風格，藍色平面為快樂風格，綠色平面為偷偷摸摸風格。每個風格平面的兩個維度仍是走路的步伐大小和高度。若我們希望角色的動作風格控制在「快樂」，則我們只需利用快樂風格的平面（藍）進行運動參數的內插。若我們希望角色的動作風格由一般轉為偷偷摸摸，我們只要將運動參數的內插，從原本一般風格平面（紅）轉到在偷偷摸摸風格平面（綠）上做即可。

6. 實驗結果

我們實作出一個可以自動產生走路動作的應用程式，使用者可以透過圖形介面指定每一步的步伐大小和高度。此應用程式可以用來解決踏腳石問題（Stepping-Stone Problem），如圖 10，我們希望限制角色的腳步落在綠色圓柱體上，並將腳抬高以避開藍色圓柱體。我們所設計之走路程序本身即可

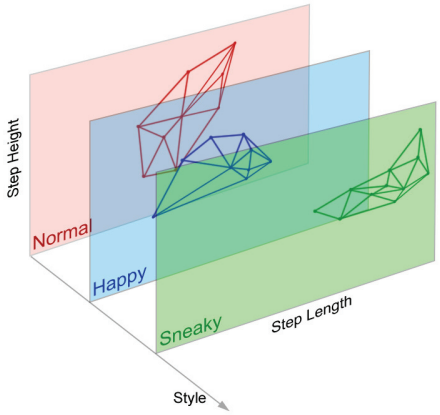


圖 9：風格平面

解決踏腳石問題，但使用任意的運動參數組合，如圖 10 上排的動畫截圖，所產生之動作較為死板、機械化，沒有真人在走路的感覺。經過前面所提到的最佳化與內插，所產生出來的動畫如圖 10 下排動畫截圖，動作較為擬真、自然，看起來更有真人在走路的感覺，而且角色的腳步仍舊符合限制需求。

為了進一步賦予使用者控制動作風格的權力，在接下來的實驗，我們加入了前面所提到的風格平面。我們挑選出七個不同風格的走路運動擷取片段，然後用它們分別最佳化程序式動畫，最後形成的七個風格平面如圖 11 所示。每個風格平面左上角皆標記有風格名稱，平面的橫軸代表步伐大小，縱軸代表步伐高度，藍線則是 Delaunay 三角剖分的結果。使用者可以用滑鼠點擊風格平面來產生指定的步伐，每一次點擊代表一個步伐，所點擊的位置將決定步伐的大小、高度和風格。圖中紅點及其上的編號顯示了使用者點擊的順序。使用者必須注意點擊的位置不可落在三角形之外，否則系統將無法利用已最佳化之點去內插使用者所指定的點。圖 12 是利用上述之七個風格平面產生之動畫，圖中地板上的藍色文字規定了角色在該區域必須使用的動作風格。

7. 結論與未來發展

在本論文中，我們嘗試以人工智慧的技術，提升動畫製作的彈性及品質。我們提出以模擬退火法為最佳化演算法，參考運動擷取資料，改善程序式動畫的品質。在得到最佳化的運動參數後，為保留程序式動畫原來彈性高的優點，我們進一步對最佳化過後的參數進行內插。我們利用 Delaunay 三角剖分從已知可產生較為自然動作的運動參數，內插出新的、可產生自然動作的運動參數。我們藉此結合程序式動畫和運動擷取各自的優點，使得程序式動畫不但容易修改以符合情境需求，而動作在視覺觀感上也更加自然。

目前我們僅以走路動畫程序為例，驗證我們最

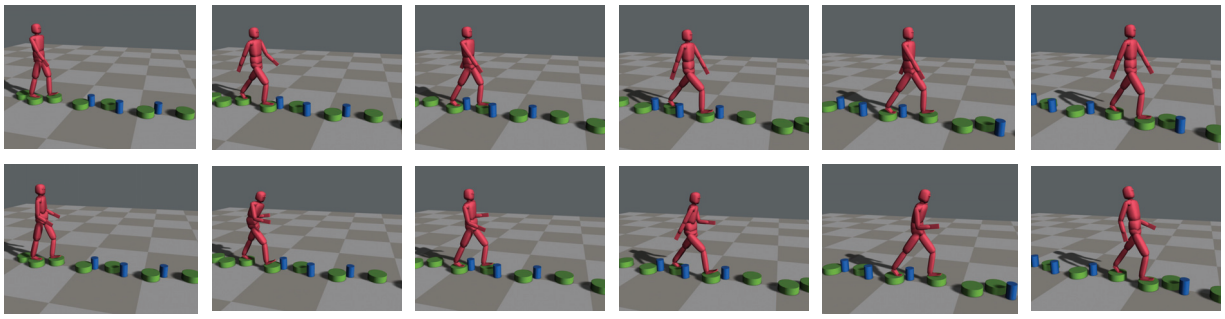


圖 10：最佳化前（上排）後（下排）的程序式動畫比較

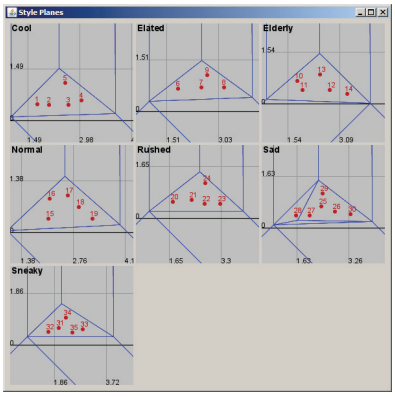


圖 11：用來指定步伐的風格平面

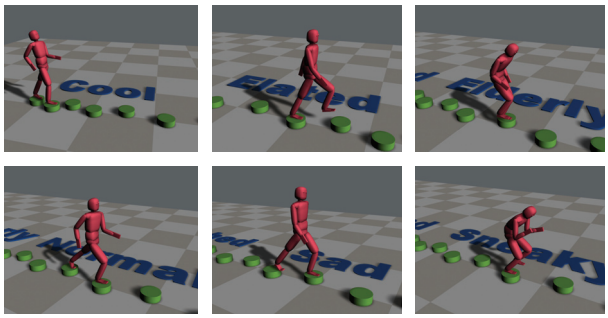


圖 12：利用風格平面控制走路步伐限制及風格

佳化方法的有效性，但由於我們將動畫程序視為黑箱子，最佳化系統只要求它必須提供實數型態的運動參數，沒有做其他任何假設，所以理論上我們所提出的最佳化方法可以用在其他不同種動作上。未來我們希望加入其他種類的動作，創造出更豐富的應用情境。此外，我們目前使用二維座標平面與 Delaunay 三角剖分來進行運動參數的內插。在未來我們希望可以參考 [3] 的作法，設計出一個更為一般化的內插機制，允許在更高維度的空間中做參數內插。

致謝

此研究在國科會 NSC 97-2221-E-004-010 計畫的支助下完成，特此致謝。

參考文獻

[1] A. Bruderlin and T. Calvert, “Knowledge-Driven, Interactive Animation of Human Running,” in *Proceedings of the Conference on Graphics Interface '96*, 1996, pp.213-221.

[2] P.F. Chen and T.Y. Li, “Generating Humanoid Lower-Body Motions with Real-Time Planning,” in *Proceedings of Computer Graphics Workshop 2002*, 2002.

[3] S. Coros, P. Beaudoin, K.K. Yin, and M. van de Pann, “Synthesis of Constrained Walking Skills,” *ACM Transactions on Graphics*, Vol.27, Issue.5, 2008.

[4] P. Faloutsos, M. van de Panne, and D. Terzopoulos, “Composable Controllers for Physics-Based Character Animation,” in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, 2001, pp.251-260.

[5] L. Kovar, M. Gleicher, and F. Pighin, “Motion Graphs,” *ACM Transactions on Graphics*, Vol.21, Issue.3, 2002, pp.473-482.

[6] K. Perlin, “Building Virtual Actors Who Can Really Act,” in *Proceedings of the 2nd International Conference on Virtual Storytelling*, 2003, pp.127-134.

[7] S.J. Russell and P. Norvig, “Simulated Annealing Search,” *Artificial Intelligence: A Modern Approach* (*2nd Edition*), Prentice Hall, 2002, pp.115-116.

[8] M. Thorne, D. Burke, M. van de Panne, “Motion Doodles: An Interface for Sketching Character Motion,” *ACM Transactions on Graphics*, Vol.23, Issue.3, 2004, pp.424-431.

[9] A. Witkin and Z. Popovic, “Motion Warping,” in *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, 1995, pp.105-108.

[10] CMU Motion Capture Database: http://mocap.cs.cmu.edu