Appear in Proceedings of 2008 Computer Graphics Workshop, 2008.

以運動擷取資料改進程序式動畫品質
Enhancing Procedural Animation with Motion Capture Data

梁長宏

國立政治大學資訊科學系
台北市指南路二段 64 號
g9606@cs.nccu.edu.tw

李蔡彥

國立政治大學資訊科學系
台北市指南路二段 64 號
li@nccu.edu.tw

摘要

程序式動畫（Procedural Animation）可以依照使用者輸入的運動屬性（Motion Attributes）彈性地產生動畫。此類運動屬性雖然已經比運動擷取（Motion Capture）資料較爲高階直覺，但如何調整適當的運動屬性以產生擬真的動畫仍屬不易。因此我們提出了一個可以自動調整運動屬性的系統，參考一段運動擷取的資料，以最佳化演算法，搜尋能產生出與運動擷取資料最爲相似的運動屬性。由於運動屬性是存在於高維空間中的資料，因此我們使用攀登法（Hill Climbing）與模擬退火法（Simulated Annealing, SA）進行最佳化搜尋。爲瞭解此最佳化問題的特性，我們測試了許多 SA 的設定參數，尋找較適合此問題的參數組合，加快搜尋的速度，並提升搜尋出來的答案的品質。我們並以不同風格的運動進行測試，以瞭解動畫程序與系統參數的通用性。

Categories and Subject Descriptors

H.5 [**Information Interfaces and Presentation**]: H.5.1 Multimedia Information Systems – *Animations, Artificial, augmented, and virtual realities*. I.3 [**Computer Graphics**]: I.3.7 Three-Dimensional Graphics and Realism – *Animation, Virtual reality.*

關鍵字

人體動畫、程序式動畫、最佳化演算法、模擬退火法

1. 前言

人體動畫製作一直是十分具挑戰性的工作。一般而言，製作人體動畫最常用的製作方式可分爲兩種，一種是由動畫師直接將關鍵格和內插輸入到電腦中，電腦再依此輸入產生動畫。以此種方式所設計的動畫，製作成果的好壞仰賴動畫師的專業程度，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（motion capture）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。

以上兩種製作動畫的方法（手動和運動擷取）都是讓使用者直接操作低階的資料（關鍵格與內插），所以較難彈性地進行調整，在不同的情境下應用。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。爲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，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爲「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屬性（motion attributes），輸出則是一段動畫。運動屬性的資料結構可以組態串列（configuration list）表示，其中一個組態就代表人物的一個姿勢。不同種類的運動會有不同的運動屬性；例如，走路的屬性有步伐大小、抬腳高度及擺手角度等。有些運動屬性（如步伐大小）對使用者來說很直觀，但也有些屬性的含意較不明顯，即使使用者了解該屬性的含意，但仍難掌握該屬性對於動畫的影響。因此，要設計出擬真的動畫，仍需使用者以嘗試錯誤的方式調整運動屬性。

本研究的目的在以運動擷取資料做爲參考，自動調整程序式動畫的運動屬性，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屬性的各種組合，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爲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。

上述「找出一組運動屬性，使得程序式動畫儘量接近所參考的運動擷取片段」是一個最佳化的問題（optimization problem）。本論文以走路爲例，使用我們先前研究 [6] 時所設計的走路程序，並從 CMU Motion Capture Database [14] 下載四種不同風格的走路片段，嘗試找出走路程序的最佳運動屬性，以產生能接近運動擷取資料的擬真動畫。我們所設計的走路程序運動屬性存在於高維度的空間中，所以本研究採用有隨機特性的模擬退火法（Simulated Annealing, SA）來搜尋運動屬性空間，另外配合攀登法（hill climbing）來找尋較佳的起始組態，以加快 SA 的收斂速度。

在第二節介紹完相關研究之後，在第三節中我們會先定義我們的問題，詳述我們所使用的走路程序的運動屬性，以及最佳化的目標函數。在第四節中，我們將描述所提出的最佳化演算法。第五節則說明系統實作上的細節。第六、七節整理出實驗結果與其結論。

2. 相關研究

根據文獻 [8]，人體動畫的相關研究可大致分爲三類：動力學（dynamics）、取樣法（example-based）、程序式動畫（procedural animation）。

動力學方法是建構人體骨架的動力學模型，利用控制器（controller）控制角色以產生模擬的動畫 [3][4]。因爲動力學方法是建立在物理學模型上，所以產生出來的動畫符合物理學，具真實性，也很有彈性。但缺點是計算量大，所以不適合應用在即時環境上；更重要的是，這個方法最大的缺點是難以控制，使用者很難找出一組起始條件來產生心中所想的動畫。

另一方面，取樣法通常參考運動擷取的資料，在其中尋求變化，以達到彈性的目的 [5]。因爲此方法直接參考運動擷取之資料，所以產生出來的動畫最爲逼真。這類研究將動畫曲線視爲訊號，然後用訊號處理的方法來產生變化。這類方法的缺點是對於運動的結構或意義一無所知，所以變化程度有限，難以適應不同的環境。[11][13]透過把許多運動擷取片段連接成 motion graph，進而在此 graph 上找一條符合要求的路徑以合成動畫。因此，motion graph 的相關研究通常也會牽涉到最佳化演算法，但是 graph 上的搜尋通常有一個明顯的目標點（goal）。例如，[13] 利用最佳優先搜尋（best first search），而 [11] 利用 A* 搜尋。我們的搜尋空間沒有目標點，所以無法利用一般的 graph 搜尋演算法來進行最佳化。

程序式動畫則是在運動中找出規則，定義出通用的關鍵格，而關鍵格之間再利用適當的數學函數或根據環境變因進行內插的計算 [1][2][6]。程序式動畫的計算量介於前兩種方法之間，所以效率佳，也具有彈性，適合應用在即時環境。然而其擬真性仰賴程序式動畫的運動模型。要設計一個好的運動模型並不容易。例如，雖然 [1] 設計了一個完整的跑步模型，但他們的跑步模型沒有考慮到轉彎。一般而言，程序式動畫最大的缺點就是在擬真性的不足。本研究希望透過程序式動畫與運動擷取資料的比對，找出程序式動畫不足所在，以改善程序式動畫的擬真性。

Spacetime constraints [12] 是一種利用數值方法進行最佳化的方法，以產生出符合時空限制的的動畫。本研究與它的差別在於，我們假設負責產生動畫的模組已經存在，因爲我們把動畫產生的責任交由程序式動畫模組負責。所以我們全部只需一個目標函數即可，而不需爲每一種運動都設計一個特定的目標函數。

表 1：走路屬性

屬性名稱	型別	範圍	鄰居範圍
Step Length	實數	[0, 2L]	±5cm
Step Height	實數	[0, 2L]	±5cm
Arm Waving Angle	實數	[0, 180]	±3°
Forward Angle	實數	[0, 90]	±3°
Backward Angle	實數	[0, 90]	±3°
Ease Exponent	實數	[1, 10]	±0.05
Land Control Point	二維實數座標點	(-2L, -2L) ~ (2L, 2L)	±5cm
Lift Control Point			
Peak Control Point 1			
Peak Control Point 2			
Phase 1 Frames	整數	[2, ∞]	無
Phase 2 Frames			
Phase 3 Frames			

3. 問題定義

根據前節所述，本研究所要解的問題是一個最佳化的問題。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。在此，我們要最佳化的自變數是運動屬性。我們以先前設計的走路程序 [6] 作爲範例進行實驗分析。此走路程序一共有 13 個屬性，其中有 3 個屬性有解析解（analytical solution），可以事先被解出來，而不用被搜尋；剩餘的 10 個屬性裡，有 4 個屬性是二維屬性，6 個是一維屬性，所以搜尋空間是 4x2+6=14 維。我們希望變化這 14 個屬性的值，使得所產生之動畫的目標函數值愈小愈好。此目標函數設計的目的在於縮小程序式動畫和運動擷取片段所產生動畫的差異；因此，我們比較兩者（程序式動畫與運動擷取片段）骨架末端關節（end effector）的位置，做爲目標函數設計的參考指標。

3.1 走路屬性

我們可以將走路動畫的產生程序看做是一個黑箱子，而此黑箱子的輸入是 13 個走路屬性，輸出是走路動畫。我們所設計的走路程序會依照輸入的屬性，計算出對應的關鍵格與內插，進而產生動畫。

在表 1 中，我們列出了走路動畫的所有運動屬性，每個屬性都有自己的資料型別及範圍；其中表格裡範圍欄中的 L 表示骨架大腿與小腿長度的和。定義屬性範圍是爲了縮小搜尋空間，當屬性的值落在範圍外時，目標函數會回傳極大值，讓搜尋演算法自然避免拜訪範圍外的值。此外，每個屬性有一個鄰居範圍，這定義了搜尋時的鄰居範圍。鄰居範圍設得愈小，搜尋就愈精確，但相對地也會使搜尋過程愈慢收斂。

這些屬性中，Step length、Step height 及 Arm waving angle 是較直觀的屬性，它們分別是走路的步伐大小、抬腳高度及擺手角度。Forward angle 是走路時前腳落地瞬間，後腳與鉛垂線所形成的角度。Backward angle 則是後腳離開地面瞬間，前腳與鉛垂線所形成的角度。Ease exponent 決定手腳緩進緩出的程度；當此值愈大，手腳的移動速度就變化得愈劇烈。四個 control point 屬性則用來描述腳踝在空中的軌跡，其軌跡由兩條貝茲曲線所組成。這些走路屬性實際上如何決定關鍵格與內插並非本論文的討論範圍，而這方面資訊在我們過去的論文 [6] 曾有詳細的說明。

我們的走路程序將走路分成三個階段（phase），這三個階段的時間長度分別由屬性 phase 1 frames、phase 2 frames 與 phase 3 frames 決定。階段之間的畫格稱爲關鍵格，當使用者調整這三個屬性時，就會改變關鍵格的時間點。這三個特殊屬性因爲不會被我們搜尋，所以我們沒有定義它們的鄰居範圍。在做最佳化搜尋之前，我們會先調整這三個屬性，使得程序式走路動畫的關鍵格對齊運動擷取的走路片段。關鍵格對齊的步驟將會在第四節做深入的描述。

3.2 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，而一個組態即可代表人物的一個姿勢。我們的目標函數定義如下：

$$f(A)=\sum_{i=0}^{(n-1)/s} d(P_A[i\cdot s], M[i\cdot s]) \qquad (1)$$

其中 P_A 表示用屬性組合 A 所產生的程序式動畫，M 表示運動擷取片段 ，$P_A[k]$ 就代表程序式動畫的第 k 格組態，$M[k]$ 就代表運動擷取片段的第 k 格組態。d 是距離函數，計算兩段動畫的差異度。n 爲 P 與 M 的最小畫格數，即 $n = min(P.length, M.length)$。$s$ 是取樣週期，表示每幾格要計算一次距離函數，將 s 設成 1 可以得到最精確的目標函數，但會使搜尋速度變緩慢。爲了加快搜尋的效率，我們把 s 設定成 12。由於我們實驗的動畫取樣率爲 120 fps，因此在最佳化過程中每秒的取樣約爲 10 次。

距離函數 d 的定義如下：

$$\begin{aligned} d(p,q) = {} & w_1 \cdot \overline{head_p head_q}^2 \\ & + w_2 \cdot (\overline{leftElbow_p leftElbow_q}^2 + \overline{lrightElbow_p rightElbow_q}^2) \\ & + w_3 \cdot (\overline{leftWrist_p leftWrist_q}^2 + \overline{rightWrist_p rightWrist_q}^2) \\ & + w_4 \cdot (\overline{leftAnkle_p leftAnkle_q}^2 + \overline{rightAnkle_p rightAnkle_q}^2) \\ & + w_5 \cdot (\overline{leftFoot_p leftFoot_q}^2 + \overline{rightFoot_p rightFoot_q}^2) \end{aligned} \qquad (2)$$

其中 p 和 q 表示兩個要測量距離的組態。$JointName_c$ 表示關節 $JointName$ 在組態 c 相對於骨架根關節（root joint）的位移。例如，$leftWrist_p$ 就表示左手腕在組態 p 相對於根關節的位移。$\overline{AB}$ 表示兩個 3D 座標點 A 與 B 的距離，例如，(2) 式中的 $\overline{leftElbow_p leftElbow_q}$ 就表示組態 p 的左手肘相對於根關節的位置，與組態 q 的左手肘相對於根關節的位置的距離。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）是動作最爲明顯的部位，所以我們會給予它們較大的權重。

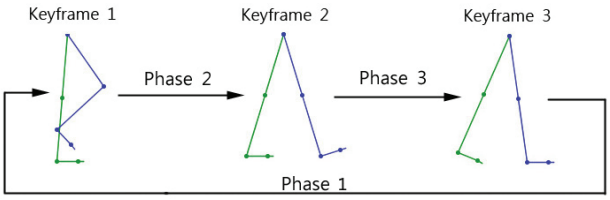


圖 1：走路的關鍵格

爲了保持屬性的值落在有效範圍內，在計算 (1) 式之前，我們會參照表 1 的屬性範圍檢查屬性值是否合法，若值在範圍外，則略過 (1) 式，直接回傳一個極大值，且屬性值距離邊界愈遠，所回傳的數值就愈大。

4. 最佳化方法

我們的最佳化方法分爲三個步驟。第一步是對齊程序式動畫與運動擷取資料的關鍵格。第二步是利用攀登法（hill climbing）[10] 做單維度的區域搜尋（local search），即找出每個單一屬性的最佳解。第三步利用前一步找出的解作爲起始點，再以 SA [10] 做進一步的最佳化。由於 SA 是一種全域搜尋（global search）演算法，所以它可以跳脫 local minimum，修正攀登法所找出的次佳解，逼近真正的最佳解。

4.1 關鍵格的對齊

我們的走路程序有三個關鍵格，如圖 1，第一關鍵格（K1）定義在雙腳交錯的瞬間，第二關鍵格（K2）爲前腳跟落地的瞬間，第三關鍵格（K3）爲後腳尖離地的瞬間，K3 之後則左右腳互換，再回到 K1。

在做運動屬性的搜尋之前，我們可以先調整每個階段的時間長度，使得程序式走路動畫的 Ki（i=1, 2, 3）對齊運動擷取走路片段的 Ki。程序式動畫的關鍵格時間點可以直接由 phase 1 frames 、phase 2 frames 及 phase 3 frames 這三個屬性得到。然而，我們一開始所拿到的運動擷取資料並非剛好是一個走路週期，所以我們先自行先切出一個週期的走路片段，然後再人工標記出運動擷取資料的關鍵格時間點。當我們知道運動擷取資料的關鍵格位置後，也就可以知道它在這三個階段中的畫格數。我們只需把這些畫格數設定到走路程序中，即可得到運動擷取資料對齊的關鍵格設定。

function Hill-Climbing(*f*) **returns** a state that is local minimum
input: *f*, an objective function
local variables: *current*, a state
current ← an arbitrary starting state
for each motion attribute *A* **do**
loop do
    neighbors ← all neighbors of current by adjusting *A*
    bestNeighbor ← a state of *neighbors*, which has the
                lowest *f* value among *neighbors*
    if (*f(bestNeighbor)* ≤ *f(current)*) **then**

```
        current ← bestNeighbor
    else break
return current
```

圖 2：單維度搜尋演算法

```
function Sim-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 3：SA 演算法

4.2 單維度的區域搜尋

一個好的起始點可以提早 SA 的收斂時間，換句話說，就是可以讓 SA 早一點找到好的答案。由於我們的運動屬性彼此的相關性不高，因此我們先對每個屬性做單維度的區域搜尋，找出一個不錯的起始點，然後才用 SA 的方法搜尋。我們使用攀登法進行單維度的區域搜尋，雖然此演算法名爲「攀登」，但在此問題中其實我們是讓它「滑落」，使目標函數漸漸下降到最低點。此演算法如圖 2 所示，我們可以將所有要被搜尋的走路屬性視爲一個狀態，針對狀態的每一個維度，不斷地找出可使目標函數下降的鄰居，直到周圍鄰居都比目前狀態還高爲止。

4.3 模擬退火法

模擬退火法（SA）是一種模擬自然界物質降溫現象的演算法。自然界物質從高溫降到低溫是一漸近的過程，倘若溫度降得太快，物質分子的排列無法達到最穩定的狀態，所以爲了讓物質分子的排列達到最穩定的狀態，溫度必須漸漸降低。當溫度較高時，分子向四周移動的距離較大，當溫度愈來愈低，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 SA 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，SA 會試著擾動目前的狀態，若擾動結果的能量（即目標函數的值）比原來低，則接受擾動結果成爲目前的狀態。若擾動結果的能量比原來高，則接受擾動結果的機率爲 $exp(-C\Delta E/T)$，其中 $exp(A)$ 表示自然對數的 A 次方，ΔE 是擾動結果與原來狀態的能量差（爲一正數），T 是當前的溫度，而 C 則是一個使用者定義的常數。由於能量與溫度的單位不同，所以 C 可避免機率值太小或太大。C 值愈大，機率就會被壓得愈低，所以我們把它稱爲機率抑制常數（probability suppress）。C 值的大小跟運動類型、骨架尺寸與起始溫度有關。以我們的目標動作爲例，我們發現把 C 定爲 2000 可以得到較好的搜尋效果。

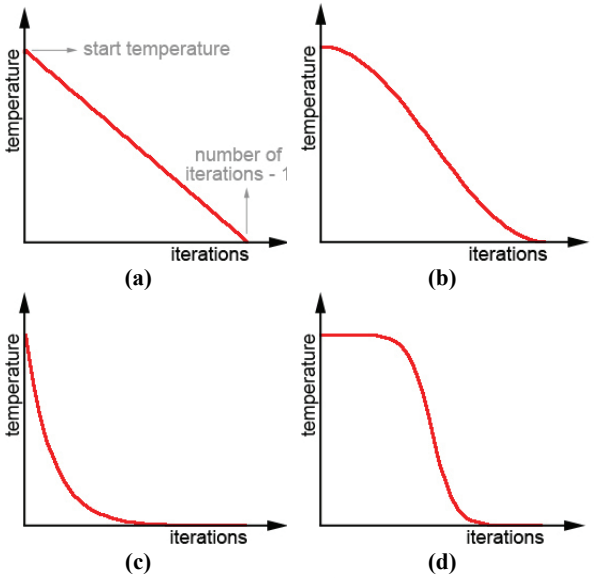


圖 4：SA cooling schedule 的例子。分別是(a)線性函數、(b)餘弦函數、(c)指數函數以及(d) tanh 函數。

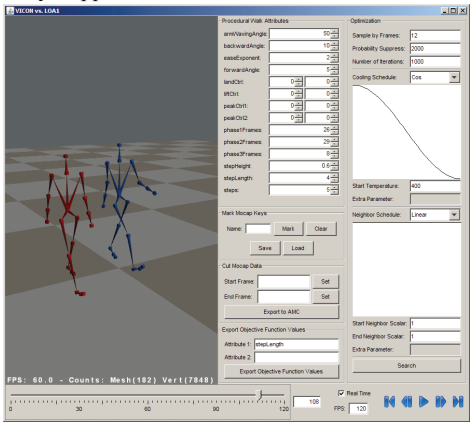
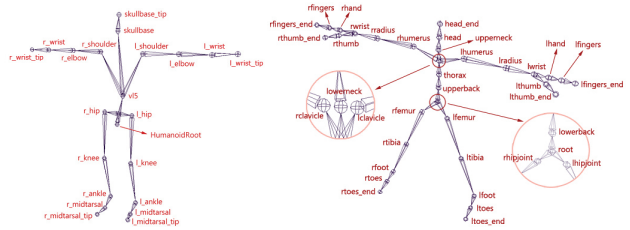


圖 5：實驗平台的使用者介面，紅色骨架的走路動作爲由動畫程序產生，而藍色骨架則是套用運動擷取的走路動作。

我們所使用的 SA 演算法如圖 3 所示，輸入包括 Cooling schedule、目標函數及起始狀態。Cooling schedule 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。Cooling schedule 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 SA 搜尋的起點，可以來自於上一步的單維度搜尋，或是任意一個點。

Cooling schedule 對 SA 搜尋的速度與解答有相當大的影響，但 Cooling schedule 的好壞並非絕對，不同的問題適用不同的 cooling schedule。圖 4 (a)-(d) 是我們設計的其中四種典型的



(a) (b)

圖 6：(a) LOA1 骨架與 (b) Vicon 骨架

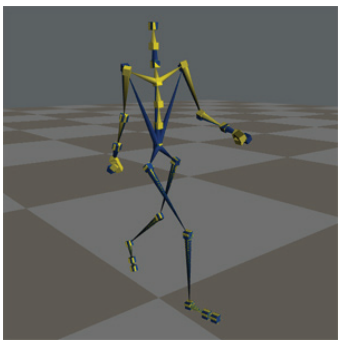


圖 7：retargeting 的問題定義，其中黃色骨架爲 Vicon 格式，藍色骨架爲 LOA1 格式。

cooling schedule，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），第二個參數是搜尋的回合數（number of iterations）。如圖 4 (a) 所示，cooling schedule 必須從座標 (0, start temperature) 開始，然後在過程中溫度逐漸遞減，最後通過座標 (number of iterations - 1, 0)。

5. 系統實作

我們的系統以 Java 程式語言實作，並建構在我們先前的動畫實驗平台 IMHAP [7] 之上。圖 5 是我們實驗平台的操作介面截圖，介面左方各有一個紅色與藍色人體骨架，紅色骨架的走路動作係由動畫程序產生，而藍色骨架則套用運動擷取的走路動作。介面右方則是一些最佳化的相關設定。

我們的走路程序所接受的人體骨架必須符合 VRML [17] 裡 H-Anim [15] 所規範的 Levels of Articulation 1 (LOA1)，而運動擷取資料裡採用的骨架格式則是 Vicon [16]。如圖 6 所示，(a) 是 LOA1 骨架，(b) 是 Vicon 骨架，這兩者骨架的關節名稱與數目都不一樣。爲了計算兩個組態的距離函數（(2) 式），我們將運動擷取動畫從 Vicon 骨架 retarget 成 LOA1 骨架。

首先，我們先建造一個與 Vicon 骨架肢體長度都相同的 LOA1 骨架，然後找出兩種骨架的的共同關節，例如，手腕、手肘、膝蓋及腳踝等，完成這些關節的對應後，我們利用反向關節（inverse kinematics），使得 LOA1 骨架上的關節能夠與 Vicon 骨架的對應關節在同一個位置上。圖 7 是 retargeting 的結果，其中黃色骨架是 Vicon 格式，而藍色骨架則是 LOA1 格式，做完 retargeting 後，程序式動畫和運動擷取片段使用的

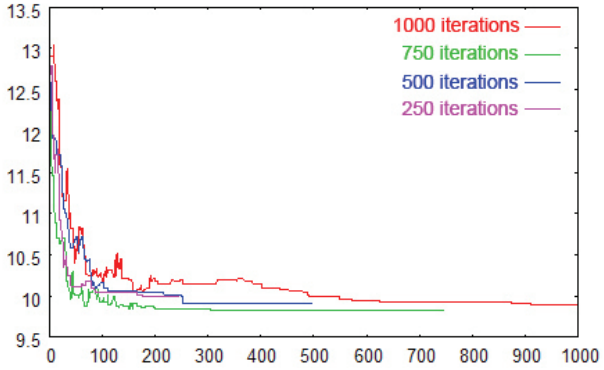


圖 8：回合數對 SA 的影響。圖中橫軸爲時間（第幾回合），縱軸爲目標函數的值。

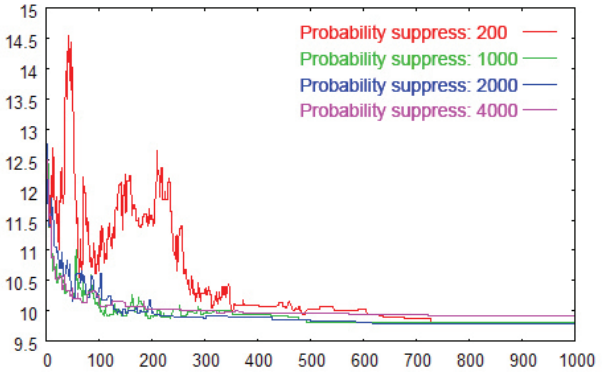


圖 9：機率抑制常數對 SA 的影響。可以看出當此參數愈大，SA 上下振盪得愈劇烈。

都是同一個骨架，便可以利用 (1) 式計算出它們的差異度。

6. 實驗結果

我們的最佳化演算法主要有五個參數可供調整：

■ **起始溫度（start temperature）**指定 SA 一開始的溫度，當起始溫度愈高，在搜尋前期愈可能向能量高的地方爬升。

■ **回合數（number of iterations）**指定溫度要經過多少回合才降爲 0 度，在每一回合都會拜訪一個新鄰居。這個參數直接影響了搜尋速度；回合數愈少，搜尋愈快。我們使用 2GHz 雙核心 CPU 的 PC 測試，大約每秒可以執行 60~80 回合。但回合數愈少，搜尋出來的結果也就愈差。如圖 8，我們的實驗發現回合數至少要 500 以上，SA 才能發揮功效，而當回合數超過 700 時，所增加的效益有限（即不見得能找到更好的答案）。

■ **機率抑制常數（probability suppress）**用來縮放 SA 爬升的機率。如同 4.3 節所描述的，當此值愈大，SA 冒險向能量高處爬升的機率就愈小，此情況如圖 9 所示。

■ 利用**單維度的區域搜尋**先找出一 local minimum 的答案，作爲 SA 的搜尋起點。如圖 10，紅色曲線表示在 SA 之前做了

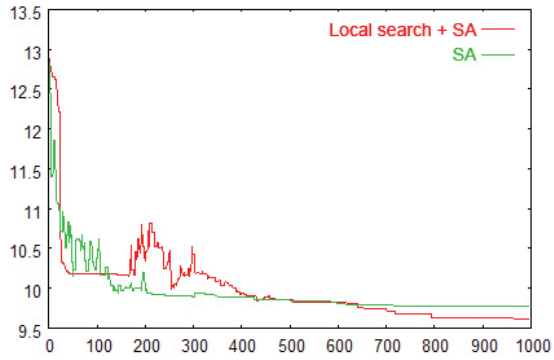


圖 10：有無單維度區域搜尋的差別。紅色曲線的前 165 回合爲單維度區域搜尋，後 835 回合才利用 SA 搜尋。綠色曲線則完全是 SA。

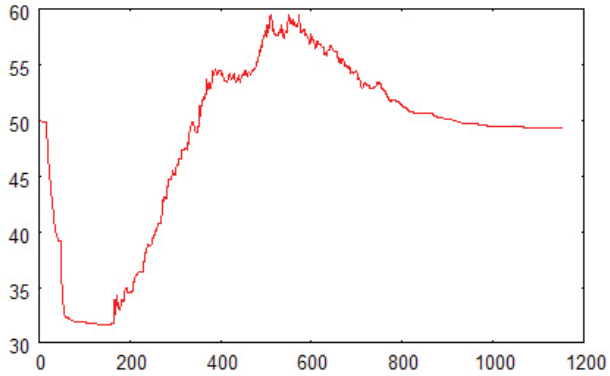


圖 11：搜尋失敗的例子。不當的起始溫度和機率抑制常數，反而會使得 SA 爬升之後無法下降。

單維度區域搜尋，讓它前 165 回合能夠急劇下降。如果運動屬性彼此都完全獨立，則區域搜尋可以幫我們找出近似最佳解的答案。但我們的走路屬性並非完全彼此獨立，所以之後我們仍需再利用 SA 去脫離 local minimum。

■ **Cooling schedule** 是 SA 的重點所在，我們設計了幾種 cooling schedule 作爲實驗，在此我們列出其中三種，我們分別把它們稱爲線性函數（linear）、餘弦函數（cos）及指數函數（power），它們的形狀分別是圖 4 的 (a)-(c)，而定義分別爲 (3)(4)(5) 式：

$$T_i = T_0 - i \cdot T_0 / (n-1) \tag{3}$$

$$T_i = 0.5\, T_0 (1 + \cos \frac{i\pi}{n-1}) \tag{4}$$

$$T_i = (T_0 + 0.1)(\frac{0.1}{T_0 + 0.1})^{i/(n-1)} - 0.1 \tag{5}$$

其中 T_i 表示第 i 回合的溫度，T_0 爲起始溫度，n 表示搜尋的回合數。

我們的實驗結果顯示，在這五個參數中，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，

表 2：實驗數據，紅色標出部分爲該走路風格的平均收斂所需回合數最小者。

運動擷取資料	搜尋設定		收斂所需回合數		收斂值	
	L?	CS	平均	標準差	平均	標準差
風格一	是	Cos	922.2	42.4	9.90	0.12
		Power	549.6	149.1	9.82	0.07
		Linear	957.1	23.4	9.95	0.09
	否	Cos	863.0	56.7	9.84	0.06
		Power	469.4	103.0	9.88	0.10
		Linear	940.6	37.0	9.90	0.08
風格二	是	Cos	909.7	56.4	15.13	0.07
		Power	496.3	136.9	15.10	0.03
		Linear	948.7	29.3	15.13	0.06
	否	Cos	891.8	65.5	15.29	0.12
		Power	500.0	134.2	15.23	0.04
		Linear	943.8	33.8	15.31	0.11
風格三	是	Cos	884.7	37.4	16.42	0.03
		Power	598.3	128.8	16.38	0.02
		Linear	955.6	23.2	16.49	0.09
	否	Cos	875.9	48.8	16.73	0.10
		Power	450.3	116.1	16.76	0.08
		Linear	946.9	33.7	16.69	0.06
風格四	是	Cos	914.2	34.4	18.32	0.05
		Power	590.5	65.7	18.29	0.03
		Linear	947.9	21.9	18.33	0.07
	否	Cos	912.3	47.1	18.64	0.17
		Power	840.3	117.2	18.62	0.09
		Linear	955.4	37.0	18.59	0.13

不但無法讓目標函數愈來愈低，還有可能會使它攀高；圖 11 就是一個搜尋失敗的例子。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。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 左右對此範例是最佳的組合。

爲了找出最佳的搜尋設定，我們做了一連串的實驗。我們準備了四種不同風格的運動擷取走路片段，針對每一種走路風格，嘗試六種不同的搜尋設定值。這六種設定值是由「是否做單維度的區域搜尋」及 cooling schedule 兩個參數組合而成。每個設定值我們都重複做了 10 次實驗，再去計算每次實驗的收斂所需回合數及收斂值。10 次實驗就有 10 個收斂所需回合數和 10 個收斂值，我們取其平均和標準差，最終的實驗結果如表 2 所示。

我們對收斂的定義如下：若從第 i 回合開始的 200 回合以內的值，與第 i 回合值的差都落在 [-0.02, 0.02] 裡，則稱該曲線在第 i 回合開始收斂，而第 i 回合的值爲其收斂值。又如果 i 爲該曲線開始收斂的最小回合數，則我們稱 i 爲收斂所需回合數。

表 2 中搜尋設定的 “L?” 欄代表 local，表示是否有做單維度的區域搜尋以找尋有效的初始設定，“CS” 欄則表示採用的 cooling schedule。從表中的平均收斂值可看出，這六種搜尋設定最後找出的答案都差不多，因此其差異主要是在搜尋速度。從平均收斂回合數可明顯看出，指數函數（power）比其

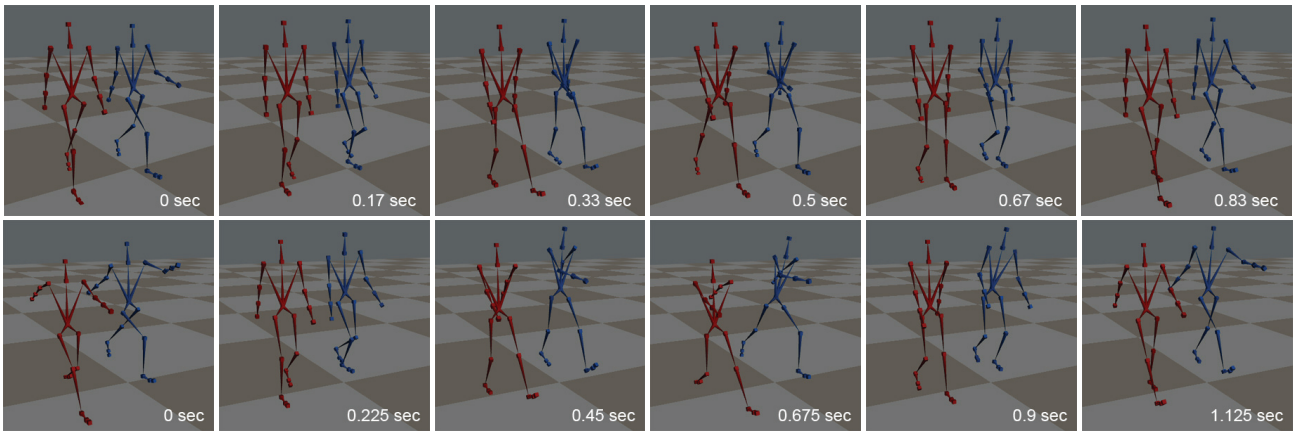


圖 12：風格一（上排）與風格四（下排）的走路最佳化結果。紅色骨架代表程序式動畫，藍色骨架則是運動擷取片段。

他 cooling schedule 函數來得好，以它作爲 cooling schedule 可以讓 SA 收斂得最快。但我們可以發現指數函數會放大收斂所需回合數的標準差，這表示會使得搜尋速度難以預測，可能會時快時慢。

單維度區域搜尋的有無在風格一到三並無太大的影響，甚至在某些例子中，有單維度區域搜尋反而使收斂速度稍微變慢。然而，以風格四作爲例子時，有單維度區域搜尋再搭配上指數函數的 cooling schedule，其收斂所需回合數比其他設定大約少 200~300 回合，減少幅度相對比其他風格大。這是因爲風格四的走路動作較爲特殊的關係，風格一到三的走路步伐都屬正常，差別在於走路的節奏，由於我們的走路程序是以預設屬性所產生的動畫，已經跟風格一到三的動畫有些接近了，所以單維度區域搜尋的功效有限。而風格四則有較大的步伐，與走路程序預設屬性所產生的動畫有很大的差異。這使得單維度區域搜尋可以讓目標函數的值急速下降，爲 SA 找到一個好的起始點，SA 因而能提早收斂。

圖 12 的上下排分別爲風格一與風格四最後找出的答案，圖片中左側的紅色骨架代表程序式走路動畫，右側的藍色骨架代表運動擷取的走路片段。風格一走路步伐正常，軀幹沒有劇烈的擺動，因此我們可以搜尋出與相似運動擷取相似的程序式動畫。風格四步伐較大，軀幹有劇烈擺動，而我們的走路程序當初並沒有考量軀幹擺動的問題，所以找出來的答案雖然身體重心、擺手角度及步伐大小看來是正確的，但是視覺上並不自然，跟運動擷取片段也就不太相像。

7. 結論

本研究中，我們設計了一個可以最佳化的動畫程序，找出一組適當的運動屬性，使所產生出來的動畫接近運動擷取動畫的擬真品質。我們利用攀登法（hill climbing）先對每個單一屬性做單維度區域搜尋，找出 local minimum 作爲 SA 的起始點。若運動屬性彼此獨立，單維度區域搜尋在起始點不理想時所得到效益最高，它可以幫助 SA 更快找到解答。然而，在起始點理想的情況下，使用單維度區域搜尋反而會稍微延後收斂的時間。

我們實驗了 SA 各項參數，嘗試找出最佳的設定。我們發現 cooling schedule 和單維度區域搜尋並不會影響最後的答案，能影響答案的是起始溫度和機率抑制常數。另外，我們發現使用指數函數（即 (5) 式）作爲 cooling schedule 可以讓 SA 最快收斂，加快搜尋的速度。

在目前所測是的某些例子中，所搜尋出來的答案跟運動擷取片段有一些較大的差異（如風格四的走路），原因可能是我們的走路程序的通用性不夠，因此無法產生出該風格的走路動畫。這也是本研究的動機之一：評估程序式動畫的能力。我們可以反過來分析程序式動畫與運動擷取資料中是那一部分使得目標函數無法降低，其分析結果將可以做爲改良程序式動畫模組的一個研究方向。

8. 致謝

此研究在國科會 NSC 96-2221-E-004-008 計畫及國立政治大學頂尖大學計畫的支助下完成，特此致謝。

9. 參考文獻

[1] A. Bruderlin and T. W. Calvert. Knowledge-Driven, Interactive Animation of Human Running. *Graphics Interface 1996*, pp. 213-221, 1996.

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

[3] P. Faloutsos, M. van de Panne, and D. Terzopoulos. Composable Controllers for Physics-Based Character Animation. *Proceedings of ACM SIGGRAPH*, 2001.

[4] J.K. Hodgins, W.L. Wooten, D.C. Brogan, and J.F. O'Brien. Animating Human Athletics. *Proceedings of ACM SIGGRAPH*, 1995.

[5] L. Kovar, M. Gleicher, and F. Pighin. Motion Graphs. *Proceedings of ACM SIGGRAPH*, 2002.

[6] C.H. Liang and T.Y. Li. Simulating Human Low-Posture Motions with Procedural Animation. *Proceedings of 2007 Computer Graphics Workshop*, Taiwan, 2007.

[7] C.H. Liang, P.C. Tao, and T.Y. Li. IMHAP – An Experimental Platform for Humanoid Procedural Animation. *Proceedings of Third International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, 2007.

[8] F. Multon, L. France, M-P. Cani-Gascuel, and G. Debunne. Computer Animation of Human Walking: a Survey. *Journal of Visualization and Computer Animation*, 10:39-54, 1999.

[9] M. van de Panne, From Footprints to Animation. *Computer Graphics Forum*, 16(4): 211-223, October 1997.

[10] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003.

[11] A. Safonova and J.K. Hodgins. Construction and Optimal Search of Interpolated Motion Graphs. *Proceedings of ACM SIGGRAPH*, 2007.

[12] A. Witkin and M. Kass. Spacetime Constraints. *Computer Graphics*, 22(4):159–168, August 1988.

[13] A. Witkin and Z. Popovic. Motion Warping. *Proceedings of ACM SIGGRAPH*, *Addison Wesley*, pp.105-108, August 1995.

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

[15] Humanoid Animation Working Group (H-Anim): http://www.h-anim.org

[16] Vicon, http://www.vicon.com

[17] VRML, http://www.web3d.org/x3d/specifications/vrml