國立政治大學資訊科學系

Department of Computer Science

National Chengchi University

碩士論文

Master's Thesis

多層式動作圖

Multi-layered Motion Graph

研究生：林志忠

指導教授：李蔡彥

中華民國一百零一年七月

July 2012

致謝

轉眼間，終於寫到了這一頁。這本論文能夠完成，首先要感謝的就是指導教授，李蔡彥老師四年半來所付出的耐心與栽培，讓凡事總愛拖到最後一刻的我可以順利的通過在這期間的每一個考驗。在這專題加上研究生生涯的四年半間，我從老師身上學到了許多東西，每當碰到問題及困難時，老師總能適時的引導我回到正確的道路上。由於老師這四年半的指導，讓我無論是在專業領域上或是思考方式上，都有了成長，在此致上我由衷的感謝。此外，我也要感謝紀明德老師與戴文凱老師在口試時所提供的寶貴見，讓我能夠把論文寫的更加的完善。

在碩士班的日子裡，研究室就像我的家一樣，在這裡所結識的人們都是幫助我成長的重要人物。因此我要感謝實驗室的伙伴們：陶百、強哥、伊亮、小 cow、岳黌、君興、紅蟲、阿雷、阿傑、KC、阿 zen、映似、小泰、凱新、順貞、胖達、小愛、呆呆夫婦、頓頓、舒服、建銘、阿呈、旭峰、Mok、Helen、八方堯琪、顧氏雲集、Cyril、函娟、佩珊、小夫、游丞、宅佩、小白、多比，很開心也很驕傲的可以跟你們一同身為 IMLAB 的一員，希望未來還能夠有機會與你們一起奮鬥！

最後要感謝我的父母、弟弟與竹雅，沒有你們的支持與陪伴，該我可以全心的專注在我自己喜歡的事物上，不用為其它生活上的繁瑣之事而憂心，我愛你們。

在完成這份論文的過程中，我一直是跌跌撞撞的，而今能夠走到最後，最重要的還是感謝這一路上曾教導及幫助過我的師長與朋友們，感謝你們，我願將這份榮耀與你們一同分享。

多層式動作圖

摘要

動作擷取法是現今相當受到歡迎的角色動作產生方法，而一般多是使用已擷取好的動作，以人工的方式將數個不同的動作混合以產生出所需的動作。但想要大量產生符合需求的混合動作仍相當不容易，因此有人提出了「動作圖」這個方法。動作圖是一種根據使用者所給定的動作擷取資料集合，經過自動化的計算找出各個動作資料之間可以連接的動作片段。藉由這個自動化的程序，各個動作擷取資料可以相互連接起來，達到在不同的動作間平順轉換，且同時保有原動作擷取資料擬真特性的目的。但縱使有上述的好處，目前動作圖的技術僅能就所擷取的全身動作進行串接，品質與彈性往往決定於一開始動作擷取資料的準備，因此如何讓既有的全身動作資料得以分解再利用，以發揮最大的價值，是一個重要的問題。在本研究中，我們提出了一個階層式的動作圖結構名為**多層式動作圖**，在這個多層式動作圖的結構中，我們將身體的動作區分成數個部位，分別計算各自的動作圖後再合併成一個多層式的架構，而合併的過程中我們提出「整體動作相似度」的計算方式，以做為兩個動作是否容易轉接的比較依據。我們也提出了在不同階層間動作圖運作的規則，以使計算的複雜度及系統的可用性取得合理的平衡。此外，我們更進一步提出名為 Motion Script 的簡易語意描述語言，來輔助控制這個具有高複雜度的動作圖結構。實驗的結果顯示，我們的方法可以即時根據使用者的指令，搜尋並產生出原動作資料所沒有的動作組合。與傳統的動作圖相比，我們的方法能更進一步的發揮原動作擷取資料的價值，以有系統的方式讓動作組合自動產生更具豐富性及彈性。

Multi-Layered Motion Graph

Abstract

Motion capture is a popular method for generating realistic character animation. In most applications, a motion usually is prepared by manually blending existing captured motion clips to generate a desired motion clip. However, finding a good transition points manually for two motion clips is a time-consuming task and cannot be scaled up easily. Motion Graph is a technique that has been proposed to automate this process by finding suitable connection points and the corresponding transition motions between motion data. With this automatic procedure, motions captured separately can be smoothly connected while keeping the realism of the captured motions. However, most motion graph techniques only consider the transition of full-body motions in two motion clips, and therefore, the resulting motion .depends on the variety of motions available in the motion database. It is an important issue to be able to compose new motion clips as much as possible with given motion capture database. In this research, we propose a hierarchical motion graph structure called Multi-Layered Motion Graph. In this structure, we divide motion data into layers of parts depending on the articulated structure of human body, and then compute a motion graph for each part of the motion. We then combine these motion graphs into an interconnected hierarchical structure. In order to facilitate the composition of motions for different parts from different motion clips, we propose a new metric called Overall Motion Similarity to find reasonable composition of motions in run time. We also propose several rules about how to traverse the motion graphs in

different layers to generate feasible motions. Furthermore, we have designed a scripting language called Motion Script to facilitate the specification and search of desirable animation to be generated. Our experimental results reveal that our method is able to compose animations that the original motion graph cannot generate in real time. Compared to the traditional motion graph method, our method is able to make good use of existing motion capture library to compose new motions in a systematic way.

目錄

圖目錄

第一章

導論

1.1 研究動機與目的

角色動畫（Character Animation）的製作一直都是相當有難度的工作，也一直都是一門熱門的研究主題，如果可以簡化角色動畫產生的流程，那就可以將角色動畫應用在許多有趣的地方，例如：可以讓小朋友演出一個簡短有趣的動畫小故事，較為進階的使用者更可以創作出一個有完整劇本的動畫，或是應用在遊戲上時，遊戲玩家所操作的角色可以有較豐富的動作表現等。而現今產生角色動畫的方式大略可以分為三種，第一種方式是關鍵格動畫（Keyframing），這種方式通常由動畫師以電腦動畫設計軟體來指定角色動畫中足以表示意義的重要影格（Frame），再將影格與影格之間作內插來產生中介的影格，以產生動畫，而這些重要的影格我們就稱之為關鍵格（Key frame）。這種方式的好處是動畫師對角色有完整的控制權，經過專業訓練的動畫師甚至可以製作出假以亂真的動畫，但相對來說，想要產生品質良好的動畫就需要經驗豐富的動畫設計師，而且即使是經驗豐富的動畫設計師，短短數秒鐘的動畫也需要花上許多時間來製做，因此這種方式往往有製作成本昂貴、費時的缺點。因此，對於沒有經過專業訓練的一般使用者而言，是難以使用這種方式來創造出一個完整流暢的動畫。除了使動畫師使用電腦軟體來產生動畫以外，第二種是以樣本資料為基礎的方式來產生角色動畫，這種方式通常是使用特別的設備將人體的實際動作給捕捉下來並儲存成訊號資料，並在電腦中製作一個相

對應的角色模型，最後將儲存的訊號資料套用到角色身上來產生動畫，而這種方式稱為動作擷取（Motion Capture）。如果想要對動作擷取產生的動畫做出變化的話，通常是透過訊號處理的方式將偏好的風格特色加入使其產生變化。這個方法的優點是產生出來的動畫相當的逼真，但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。第三種是以知識為基礎的動畫（Knowledge-based Animation），這種方式通常是以模擬或是運動計劃的方式來產生動畫。但即使有良好的知識基礎或是模擬系統，要產生一個擬真的動畫仍是這種方式的一大難題。

如前所述，動作擷取是屬於以樣本資料為基礎的方式來產生角色動畫的方法，但動作擷取所產生的動畫是較不易被修改的，因為在動作擷取的資料裡，通常除了描述一個動作的各種低階屬性變化以外就沒有其它的資訊了，因此我們無法得知對於一筆特定的動作擷取資料而言，到底那些屬性才是重要的，更不用說各個屬性對於動作變化的意義。而這造成動作擷取產生的動畫非常不容易調整，而即使可以調整，其幅度也是相當的有限。因此每當新的動畫需求出現時，如果手上的動作擷取資料都無法滿足這個新的需求，就得要再擷取新的資料，才能產生符合新需求的動畫，而這昂貴且耗時的缺點，使得使用動作擷取來產生的動畫變的難以親近以及利用。不過有時新的需求或許無法由資料庫裡單一的一個動作擷取的資料所滿足，卻可以由兩個或兩個以上的動作來混成（Synthesize）後來滿足，舉例來說：如果我們想要一個動畫角色在一個指定的空間裡走一圈，雖然我們資料庫裡並沒有任何的動作擷取資料是可以直接用在這個情境下的，但我們可以透過直走和轉彎兩個動作擷取資料來混成出我們所需要的動作。因此，如果可以將足夠多個動作擷取資料適當的結合起來，那就可以創造出一個具有豐富動作的動畫角色。而動作圖就是順著這個思考脈絡而生的。

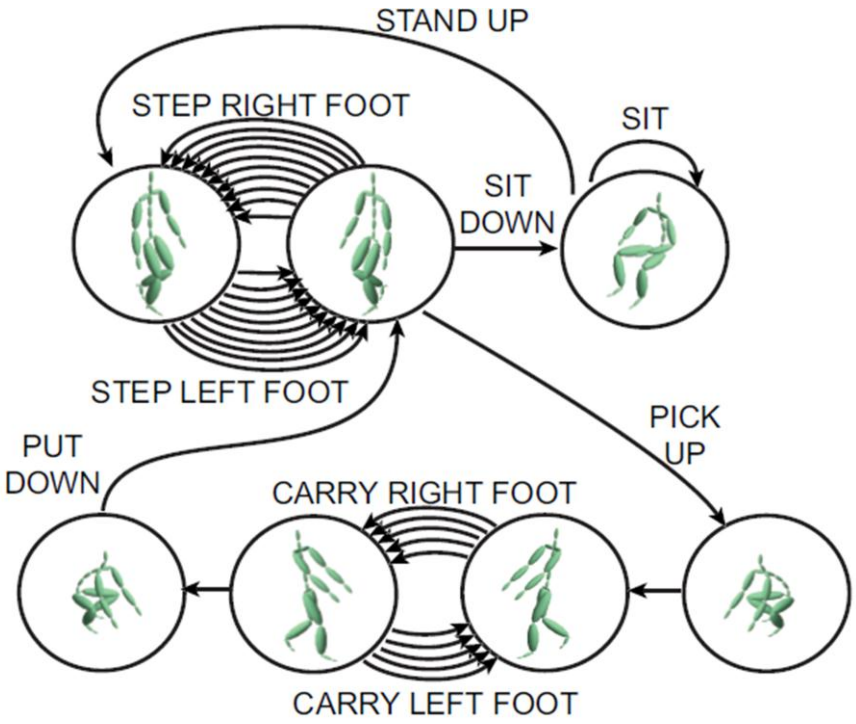


圖 1.1、動作圖示意圖[10]

動作圖是一個由許多動作片段所連接而成的有向圖（Direct Graph），而動作圖上的一條路徑就代表了一段動作。因此，要從動作圖得到我們所需要的動作就可以轉換成圖論上的一個問題，可以使用圖論或是人工智慧（AI）中的搜尋演算法來找出一條可以解決問題的路徑，而這條在動作圖上的路徑就是一個符合需求的動作。但傳統的動作圖都是以全身的動作為其礎來計算所構成的圖，因此動作圖的能力與做為其基底的動作擷取資料集合相去不遠。Jang 等人[9]提出了一個將身體各部位的動作拆解分開後，再以自動化的方式找出合理的組合後，來產生新的動作的方法來豐富原始動作擷取資料庫的方法，他們提出來的方法所得到的結果相當的不錯結果，但資料庫中的動作之間仍是彼此獨立的，無法直接的拿來產生有變化的連續動作。因此在本研究中我們提出了一個可以身體

各部位可以獨立切換動作，且能夠產生連續動作的階層式動作圖結構，舉例來說，如果以走路與原地講話這兩個動作為基礎來產生動作圖的話，而該張圖能產生的結果最多也只是走一陣子的路之後停下講話或是在原地講話之後開始走路，而不能產生走路與講話之混合動作：一邊走路一邊講話。為了達到這個目的，我們會需要一個有別於傳統的單層動作圖結講，而是一個多張包含身體各部份的多重動作圖結構，在這裡我們稱為**多層式動作圖（Multi-Layer Motion Graph）**。

大略來說，本研究的目的在於除了計算傳統的動作圖外，另外計算數張身體各部份的動作圖，最後再加以合併；透過這個多層式的動作圖結構，我們得以最大化的利用我們原始用來計算動作圖的動作擷取資料，並增加動作圖能夠產生動作的豐富性，同時亦保留了動作圖平順轉換的優點。但這個可以產生大量動作組合的結構所造成了一個問題，那就是控制上的困難。因此我們以人工的方式為每個動作（包含全身動作、上半身動作、手部等...有區分出來的部位）加上足以表答該動作特性的標籤，並在這些標籤的基礎之上建立了簡易的語意系統。

1.2 問題描述

傳統動作圖產生動作的方式是將原有的動作片段以最佳的方式相互連接起來。而我們的目標是要基於傳統動作圖建立的方法之上，建立一個帶有身體各部份子圖的多層式動作圖結構。有了這個多層式的結構後，我們希望可以產生將不同動作各取一部份後所合成的新動作，同時亦保留了動作圖可以在不同動作間平順轉換的特性。依照這個目標我們將本研究切分為兩大部份，第一部份是我們是如何產生這種新的動作圖結構，而第二個部份是我們要如何使用這新的動作圖結構來產生我們所想要的動作。

在第一部份動作圖的建立前，在資料的前處理階段我們會先以人工的方式給每個動作加上一個或是多個的標籤。在建立動作圖時，我們會先個別的產生身體各部位獨立的動作圖，之後再加以合併。在合併時，我們會以全身的動作圖做為最主要的參考，並以此為基底來建立整個多層次的結構。在整個多層式的動作圖的建立過程中，我們使用了 Kovar 等人[10]所提出的 Point cloud distance 來做為尋找動作間相似片段的距離方程式，而在找到相似的動作片段後，我們統一使用線性混合來產生連接相似動作片段間的轉換動作。在合併各個部位所獨立產生的動作圖時，我們使用骨架做為基礎，將各個部位身動部位的動作圖，依照其在骨架上的階層關係給連接起來，同時更會計算每個全身動作其所屬的部份動作與其它每個可相互轉換的部份動作之間的「整體動作相似度」，並將這個數值存下來以供使用多層式動作圖產生動作時使用。

而在第二個使用多層式動作圖來產生動作的步驟中，與以往動作圖的研究一樣，為了要使用多層式動作圖來產生符合使用者需求的動作，我們需要設計一個搜尋演算法來達到這個目的。在本研究中我們將在圖上的搜尋過程分成兩大步驟，分別是在動作種類層級的 Global search，以及在動作片段層級的 Local search。在 Global search 的部份，讀取動作圖時，會同時掃描在動作圖中每個動作的連接關係並建立一張 Metric 表，然後使用 Floyd-Warshall 演算法建立兩兩動作間最短路徑的表格，每當使用者設定其需求時，系統就會根據這張表格找同時參考「整體動作相似度」來找出一條 Global 的最短路徑，在找到這條路徑後，Local search 就派上用場了；Local search 會根據 Global search 所給予動作轉換路徑一步步的尋找最近的轉換節點，並產生動作結果。以上的步驟是我們產生動作的基礎，在這個基礎之上結合前處理時使用者給予動作的標記，我們提供簡易的語法給使用者來設定在各個狀況下所要使用動作的規則。

本論文在系統設計上同樣由二大部份所構成，分別為：動作圖建立以及動作產生。動作圖建立的部份負責產生部份動作亦可獨立轉換的多層式結構，增加動作圖在使用上能夠產生動作的豐富性。動作產生的部份負責根據使用者的指令及當下環境的狀態，並結合簡單的語法，使得多層式的動作圖在結構上複雜度提高的同時，依然能夠輕易的讓使用者得到最符合需求的動作。

1.3 論文貢獻

本論文的貢獻包含：

- **動作圖實驗平台**

我們設計一個名為 IMMograph 的動作圖實驗平台，在這個平台上，我們除了設計了自定的骨架以及動作系統，更整合了 C++上主流的 3D 繪圖引擎（OGRE）以及 GUI 繪製函式庫（QT），使我們可以快速的設計應用程式，以驗證我們所產生的動作圖之正確性。

- **多層式動作圖結構**

在計算動作圖時，我們將不只計算兩動作其相互之間全身運動方式的相似度，我們更會將身體拆分為數個部份（ex. 手、上下半身等...），分別計算其運動的相似度來形成另一張動作圖，並在建立這個結構時，會為全身動作外身體其它部位的動作圖中的動作間，計算「整體動作相似度」的數值，這個數值將做為一個指標，供我們在使用這個結構來組合出新的動作時，能有一個依據來判斷該組合是否合適。

- **以語意尋找合成動作的機制**

我們設計了一個簡易的 Script 語法，名為「Motion Script」，讓使用者可以使用高階的語法來控制我們所提出的「多層式動作圖」，不用去了解底層實際運作的方式。

第二章

相關研究

2.1 動作擷取

程序式動畫、關鍵格動畫或是動力學模擬的方法等來產生動畫的方式，在製作時或多或少都需要對動畫角色的運動有一定程度的認知。而動作擷取方式產生的動畫，就只是對於經由硬體設備所擷取到的動作資料在參數層面上做出變化。目前所最常見的動作擷取設備可以分為兩類，一類是使用重力加速器，而另一類是使用光學的設影機來擷取動作的變化。但無論是那一類的設備，在將動作擷取下來後都需要經過消除雜訊、矯正以及使用反向機構學來計算出未被直接測度到的關節之數據等步驟後，才可以變成可以使用的資料，而通常動作擷取資料都會以角色所有關節的旋轉或是位置來表示單一的姿勢，然後由數個連續的姿勢資料來代表一個動作。雖然使用動作擷取的方式來產生動畫可以不用對動畫角色的運動方式有所認知，但因為擷取下來的資料是直接來自真實演員的運動，因此，在所有的動畫產生方法中，動作擷取所產生動畫是最為逼真的。

為了將動作擷取的資料應用在不用的環境上以符合使用者的需求，我們需要對動作擷取的資料進行修改，但因為動作擷取資料裡不包含任何的運動結構，因此修改動作擷取資料的方式往往都是對現有的資料進行混合或延長等運算，我們通常稱修改動作擷取資料的動作為動作編輯（Motion Editing），而接下來將簡單介紹兩種動作編輯的方式。

動作混成（Motion Blending）是最基本的動作編輯方法之一，他的目的在於將使用

者提供的兩個動作擷取資料依指定的比例混合。而動作混成最常見的應用是在需要平順的連接兩段動作資料時。通常混成的方式將全身所有的關節角度以及身體的位置以線性內插的方式來做混成，而這個方法簡單，但效果只能算是差強人意，最基本混成的式子如下：

$$\mathrm{M}'(\mathrm{t}) = \mathrm{w}(t)M_1(\mathrm{t}) + (1 - \mathrm{w}(\mathrm{t}))\mathrm{M}_2(t) \tag{2.1}$$

其中$M_1(\mathrm{t})$和$\mathrm{M}_2(t)$為兩個要用來混成的動作擷取資料，w(t)是混成時的權重值，$\mathrm{M}'(\mathrm{t})$是混成出來的結果。Unuma 等人[20]的研究在內插的比例上加入了傅立葉展開（Fourier expansions）來使動作轉換時的混成更為平順，Bruderlin 和 Williams[4]則是將擷取下來的資料從原本的時域轉換到頻域後再進行動作的混成。Rose[15]在他的研究中將動作轉換時的動作混成加入了時間與空間的限制，並且計算出不同方式的動作混成所需要耗費的能量，然後以最佳化的方式找出耗費最低能量的混成方式。動作混成最大的好處就是計算的複雜度較低，但隨意的混成兩個動作是相當危險的，因為是有可能產生出視覺上不合理的動作。

而動作變形（Motion Warping）則是另一個編輯動作的基本方式，這個問題最早是由 Witkin 和 Popovic[21]所提出，在做動作變形前通常會先了解變形後目標的需求以及限制，然後去定義一個可以符合需求的目標函數，但同時也希望可以保原動作擷取資料的特性與結構，舉例來說，如果想將一動作擷取資料$\boldsymbol{C}(\boldsymbol{t})$變形成$\boldsymbol{C}'(\boldsymbol{t})$，在他們的研究中，可以將滿足目標限制的函數以如下的方式來表示：

$$\boldsymbol{C}'(\boldsymbol{t}) = \boldsymbol{a}(\boldsymbol{t})\boldsymbol{C}(\boldsymbol{t}) + \boldsymbol{b}(\boldsymbol{t}),\ \boldsymbol{t} = \boldsymbol{g}(\boldsymbol{t}') \tag{2.2}$$

其中$\boldsymbol{C}'(\boldsymbol{t})$為目標所需的結果，$\boldsymbol{C}(\boldsymbol{t})$是原本的擷取資料，a(t)是縮放的係數，b(t)是用來在做位移，而$\boldsymbol{t} = \boldsymbol{g}(\boldsymbol{t}')$是在時間上的變化。但想要找到一個符合需求的變形函數通常需要

以人工去不斷的嘗試，因此 Hsu 等人[8]提出了一個自動化的方法，使編輯者只要指定少數的時間限制，而電腦就會自動的找出一個符合限制的變形函數。

以上動作混成以及動作變形這兩個方式都是在想要編輯動作擷取產生的動畫時最常見也是最基本的方法，但在使用時仍要非常的小心，因為基本上兩個方法都是對擷取下來的資料做出訊號層面上的變化，無法保證產生出來結果的品質。因此建立在這兩個方法的基礎之上來發展出來的研究至今仍是一個相當熱門的研究題目，雖然步驟不外乎是先分析資料，然後建立轉換的模型，最後經由轉換模型將原本的資料轉換成所需的樣子，但產生的結果以及可變化的幅度已經較最原先的方法進步了不少。

2.2 程序式動畫

程序式動畫是屬於以知識為基礎的動畫產生方式。如果想要製作一個角色的動畫動作產生器的話，一般來說大致上的流程如下：首先我們需要根據生物力學上的知識來分析並了解想要產生的動作，然後將所有得到的資料匯整起來，依據分析的結果來設計該運動該有的參數、建立產生運動的規則並結合適當的控制器，最後就可以製作出產生該動作的動作模型。程序式動畫的概念根據我們知所最早由 Zeltzer[22]所提出，在他的研究中他參考了 Tomovic 與 McGhee[19]將有限狀態機（Finite State Machine）應用在生物工程學系統上的經驗，以及之前對於人類運動的研究的成果，在觀察並分析人類的走路動作後，將走路動作同樣以一個有限狀態機來定義，而狀態機中的每一個狀態都是走路動做中的一個關鍵格，而在各個關鍵格之間則是使用線性內插的方式來產生連接用的連續影格。

Bruderlin 和 Calver[2, 3]在他們的研究中更進一步的在走路程序中加入了控制器的

概念，透過使用反向鐘擺（Inverted Pendulum）及正向機構學（Forward Kinematics）這兩個控制器，使在人物行走時踏出每一步的時候，都會以有接觸到地面的腳設為身體的支撐點，重新的修正身體的移動量，以確保動畫人物不會發生像腳穿過地面等不自然的現象。此外，透過了控制器及更良好的設計方式，在他們的動作系統中使用者可以只操弄少許的簡單明瞭的高階參數就可以改變動作的樣式，而不用去接觸較為低階繁瑣的參數，而之後他們更將他們的方法進一步的應用在人類跑步的動作上。

Boulic 等人[1]在他們的研究中加入了反向機構學（Inverse Kinematics）的控制器，來做為預防動畫角色的雙腳穿過地方或是滿足特定的限制，但因為反向機構學的運算對於當時的電腦來說，是一種相當複雜的運算，因此在即時的應用環境下無法連續的使用。因此為了同時兼顧即時執行的效率以及反向機構學的控制器，他們在產生動作時仍以運算複雜度較低的正向機構學控制器為主，而在需要對影格做後處理的時後才使用方向機構學的控制器。然而，在接下來的研究中，因為電腦運算能力的快速成長，反向機構學變成了在制做程序式動畫時的必備基本工具。

Perlin[13]在他的研究中，除了設計了數個動作產生的程序，他更開始關心產生出來的動作在細節上的表現。在他的動作模型中，他加入了他先前所提出用來產生物體表面材質的 Perlin 亂數（Perlin Noise）[12]，使動畫程序產生的動畫看起來不會都是那麼的機械化，生動化了動畫動作在視覺上的感受。Chen 和 Li 除了使用了反向機構學的控制器來輔助產生走路的動作之外，也同時運用了機器人學中的運動計劃的技術來計算動畫角色在有障礙物的環境中行進的路徑。Throne 等人[18]在他們的研究中，除了設計了一個程序式動畫的產生器以外，加入了一個以手繪的方法來輸入的使用者介面。Hecker 等人[7]在他們的研究中設計了一個近似自然語言（Natural Language）的程序式程畫設計系統，使藝術家在設計運動程序時可以因為自然語言的輔助，變的更加的直覺、容易，

此外，他們並結合了在角色模型上標記的方式以及自行開發的反向機構學系統，使他們的動畫成果可以套用到未知形狀的角色身上（圖 2.1），而且最終的成果也有運用到遊戲 Spore 上。

程序式動畫發展到之已將近二十多年了，而一個動畫產生程序其生產出來動畫品質的好壞，是由其背後所使用的運動模型所決定，是一直以來不變的原則。要設計一個好的運動模型，也一直以來是一件不容易的事。如果想要設計一個運動模型，可以產生品質較高的動畫，所需要考慮的細節與變化的因素就要比較多，而這些要考慮的項目會有：運動本身的風格變化、物理學上的運動定律、使用環境可能會有的限制等。如果想要設計出一個能使用在任何狀況下的完美的運動模型，就必須要將所有的因素給窮舉出來，但這幾乎是不可能的一件事。但其實在大部份的狀況下，動畫應用的環境都不是完全沒有限制的，因此我們其實是可以例舉及考慮到可能的細節以及有可能變化的因素，依然可以在大部份的狀況下使用程序式動畫來製作出符合需求且具有彈性的動畫。

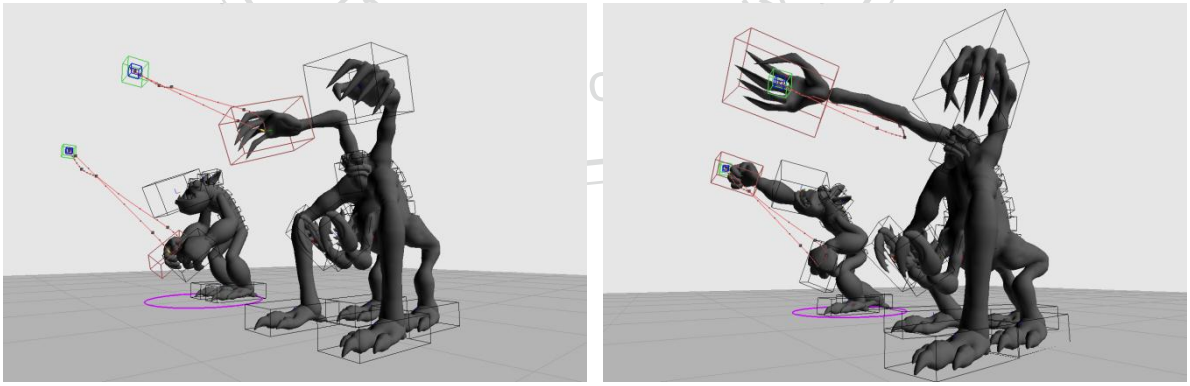


圖 2.1、將同一動作套用到不同的模型上之示意圖

2.3 動作圖

動作擷取資料彈性低的缺點是因為單一的動作難以做出大幅度的變化，但如果我們可以事先設想到可能應用的環境，將所有有可能會用到的動作擷取資料都準備好，當碰到變化時就拿出符合需求的動作資料來使用，或許也是一個解決問題的方式。而動作圖就是一個以這種思維來解決問題的答案，動作圖是將許多的動作擷取資料依照一定的規則給連接起來，所形成的一個有向圖。動作圖是一個一直都有的概念，但直到 Kovar 等人[10]在他們的研究中提出了一個自動化建立動作圖的方法後，動作圖的研究才真正的開始有系統的發展了起來。在他們的研究中，他們提出了名為 Point cloud distance 的動作相似度比較方法，並以此為基礎設計了一個自動化的程序來兩兩的比對資料庫裡所有動作片段，然後將相似的片段給找出來，在他們的研究中，所提出比對動作相似度的式子如下：

$$\min_{\theta,x_0,z_0} \sum_i w_i ||p_i - T_{\theta,x_0,z_0} p_i'||^2 \qquad (2.3)$$

其中線性轉 T_{θ,x_0,z_0} 將點 p 在水平面上（在此假設水平面為 xz 平面）旋轉 θ 角然後平移(x_0, z_0) ，此計算的索引 i 會涵蓋 Point cloud 上所有的點，權重 w_i 可依使用者決定每個點的重要性來自由設定。在比較兩兩動作間的每一個動作片段後，足夠相似的片段會利用動作編輯中的動作混成來將相似的動作片段給連接起來，形成動作圖。但這個方法在連接性以及圖的複雜度上都還有一些待解的問題。因此在這個研究的基礎之上，Zhao 和 Safonova[23]以在兩兩動作片段內插出數個新的動作的方式來增加動作圖的連接性，並且提出了依照動作類型的不同使用不同的混成方式來連換動作，以增加混成出來動作的品質。Safonova 和 Hodgins[16]提出了利用 A*演算法加快動作圖產生動作的時間，以及刪減圖上多餘節點的方法。Heck 等人[6]將程序式動作引入到動作圖之中，在這個

研究中他們提出了一個離散化的方法，在程序式動作的參數空間上取出數百個具參數組合來做為整個連續空間的代表，並套用 Kovar 等人[10]的方法來計算欲與之相連的動作之相似度，最後在參數空間上框出合理可以相連的區域並記錄下來，這個方法達到了將程序式動畫與其它動畫產生的方式做出適當的結合，來達到截長補短的效果；但全身性的動作的程序設計仍是一個難題，因此在實用上仍有一點距離。

2.4 問題與討論

在前面的介紹中，我們得知運動擷取的方法是直接將動作擷取下來，轉換成訊號資料的形式，所以比起程序式動畫與其它動畫產生動畫方式來說，運動擷取方式產生的動畫最為擬真，而且運動擷取資料編輯方式的計算量相當對較小，因此現在這種方式已經被大量的應用在動畫、電影或是即時的虛擬環境上。但因為難以經由運動擷取的資料來了解該筆資料所記錄的運動其背後的運動結構，因此編輯的幅度大多有一定的限制，即使成功的做出了超過限制的編輯，產生出來的動作在視覺上仍可能是不合理的。

另一方面，程序式動畫的根本就在運動模型的建立。而這又建立在我們對於運動本身的了解與使用環境的限制，因此程序式動畫比較適合用在運動結構較有規則的動作，如走路、跳躍等。一些比較沒有規則的動作像跳舞等就比較不適合使用動序式動畫的方式來產生。但對於那些可以使用程序式動畫產生的動作而言，其最大的好處就是可以輕易透過調整高階參數的方式來產生符合需求的動作。但要設計出一個好的動畫程序實屬不易，而且要找出產生擬真動量的參數也是一個需要不斷嘗試的工作，但它高彈性的優點與其它動畫產生的方式比較起來仍是有相當大的優勢。除此之外，因為程序式動畫具有明顯結構的高階可調整參數，可以使其它的程序可以很輕易的來調整這些參數，以達應用的目的。

而動作圖在概念上可以說是一種以量取質的方式，動作圖的出現就是為了透過事先建立大量的動作擷取資料庫，來補充原先單一運動擷取資料編輯幅度有限而彈性不足的缺點。以往以全身動作為基礎的動作圖在根本上是在事先列舉出所有有可能會用到的動作，透過大量的動作擷取資料做為輔助來提升使用時的彈性，但其實我們是很難真正將所有有可能的動作都給例舉出來的。舉例來說，同一個揮手的動作，我們就可能使用到在走路時揮手、站立時揮手、跑步時揮手或是跳著揮手等。因此我們發現雖然以全身動作為基礎的動作圖產生的動作會是最為擬真的，但這個局限也使得它無法充分的利用動作擷取資料庫裡的資源。

第三章

動作圖

動作擷取資料以離散化的方式把動作資料以連續序列的方式記錄下來，因此可以相當精確的記錄被擷取者在錄製時的動作。但通常單一筆動作擷取資料所能提供的單一或特定動作序列之動作種類是相當有限的。因此在實際使用動作擷取資料時，通常會利用多筆資料，截長補短後組合成一筆在錄製時所沒有的動作。而在這個截長補短的過程通常是以人工的方式反覆觀看不同的動作後，找出動作相似的部份後加以混合連接如圖 3.1 所示。而動作圖則是一個將此人工連接過程自動化且有效表達的方法，而動作圖上的每一個點都是動作擷取資料中的一個姿勢或動畫的一個畫格（Frame）。一個傳統的動作圖通常是由數個符合應用情境的動作擷取片所建立而成。以拳擊遊戲為例來說，用來建立動作圖的動作擷取資料就必需要有數個不同位置的出拳、對應的閃躲以及不同方向及節奏的小跳步等。當動作圖建立完成後，使用者就可以透過路徑搜尋演算法在圖上找到一條路徑的方式來產生所需的動作。在本章接下來的內容中我們將大略的說明動作圖的建立流程，並指出本研究所想要解決及改進的問題。3.1 節將描述兩動作片段間的「相似度」定義，並解釋其定義背後的想法。3.2 節將說明如何應用在上一節中定義好的「相似度」來自動建立動作圖。3.3 節的部份將討論我們在實做動作圖的過程中所發現的問題。

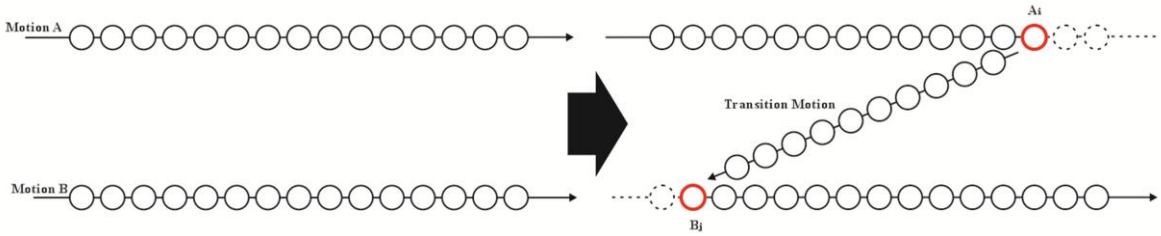


圖 3.1、產生一轉換用動作來將原本兩個分開的動作結合起來之示意圖

3.1 相似度方程式

動作圖建立的流程中，首要的問題就是動作片段間「相似度」的定義，而我們將此種用來計算兩動作片段間「相似度」的方程式稱之為「相似度方程式(Similarity Metric)」。在此我們以距離函式 $D_{ij} = D(P_i, P_j)$ 來表示相似度方程式，其中 P_i 和 P_j 分別為兩個單一的動作片段（Motion Clip，即動作擷取資料中的一個影格），而距離函式的結果越小則代表兩動作片段越相似。直覺上的想法我們會覺得直接拿兩片段所表示的姿勢來做比較就可以了，但這樣讓人感到略顯不足，因為一個良好的距離方程式所要考慮的應不僅只是兩動作片段在單一片段其靜態上其姿勢的差別，更要考慮到兩動作片段在動態上（即一小段的動作片段區間）的動作差別。而許多人也在此概念下提出了不同的方法來實做此距離函式，其中 Lee 等人[11]提出了基於 Joint angle 的距離方程式，而兩動作片段 i 與 j 的距離 D_{ij} 是基於 Joint angle 的距離方程式。其式子如下所示：

$$D_{ij} = D(P_i, P_j) = d(p_i, p_j) + w_v d(v_i, v_j) \quad (3.1)$$

其中 $d(p_i, p_j)$ 代表兩片段關節角度上的差異，而 $d(v_i, v_j)$ 則是兩片段關節速度上的差異，而 w_v 則是用來調整速度重要性的權重值。 $d(p_i, p_j)$ 的計算方式如下所示：

$$d(p_i, p_j) = ||p_{i,0} - p_{j,0}||^2 + \sum_{k=1}^{m} w_k ||\log(q_{j,k}^{-1} q_{i,k})||^2 \quad (3.2)$$

其中 $p_{i,0}$ 與 $p_{j,0}$ 分別是 root 點在第動作片段 i 與動作片段 j 時的位置，$q_{i,k}$ 與 $q_{j,k}$ 分別是關節 k 在第 i 個動作片段與第 j 個動作片段時的旋轉量。而在本研究中我們所採用的是 Kovar 等人[10]的所提出的 Point Cloud Distance 方法。在他們的想法中，一般動畫的動畫角色所顯示出來的樣子是由其內在的骨架所決定，因此這個方法認為比較兩動作片段即是比較兩個由骨架所驅動的姿勢。綜合以上，想要計算兩動作間的距離就是將比較兩個由骨架所驅動的點雲（Point Cloud），而在此我們將骨架的各個關節點之世界座標所集成的點來構成我們所使用的點雲。在實際要計算動作片段 P_i 和動作片段 P_j 的距離 $D(P_i, P_j)$ 時，需要同時將與這兩個動作片段有關的兩長度為 k 的動作區間考慮進來，其分別是一段以 P_i 為起點的區間 $[P_i, P_{i+k-1}]$ 而另一段則是以 P_j 為終點的區間 $[P_{j-k+1}, P_j]$ 如圖 3.2 所示，同時我們發現：

$$D(P_i, P_j) = D(P_{j-k+1}, P_{i+k-1}) \quad (3.3)$$

意即動作片段 P_i 與 P_j 的相似度計算方式和 P_{j-k+1} 與 P_{i+k-1} 是一樣的。透過比較兩動作片段區這個方法，使他們可以將動態上的動作差別考慮到方程式中。此外這個用來計算距離的動作區間與最後用來混合產生出來做為連接用的動作是一樣的，因此根據 Mizuguchi 等人的建議我們將 k 的值設為 20（當動作的 FPS 為 60 時約為 1/3 秒）。

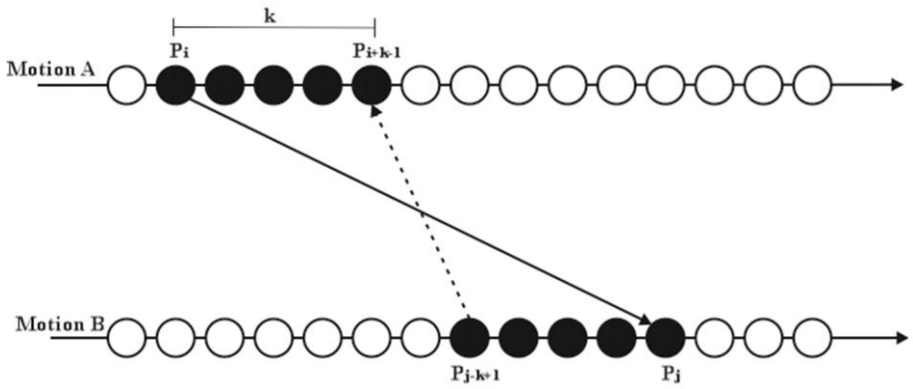


圖 3.2、由圖所示我們得知 $D(P_i, P_j)$ 與 $D(P_{j-k+1}, P_{i+k-1})$ 計算時所使用的動作區間的一樣的，因此 $D(P_i, P_j) = D(P_{j-k+1}, P_{i+k-1})$

當在兩動作區間中取出動作並轉成點雲之後，最後還有一個問題需要解決。動作一開始在擷取時可能兩個動畫角色在方向及位置上都是不相同的，因此在開始計算前，我們需要將一個點雲經過旋轉與平移後與另一點雲對齊（Align）。最後動作片段 P_i 和動作片段 P_j 的距離計算的方式就是將兩點雲中所有相互對應的點之歐幾里得距離（Euclidean distance）加總起來。以上的描述做成公式後就如下面的式子所示：

$$D(P_i, P_j) = \min_{\theta, x_0, z_0} \sum_i w_i ||p_i - T_{\theta, x_0, z_0} p_i'||^2 \quad (3.4)$$

其中線性轉換 T_{θ, x_0, z_0} 將點 p 在水平面上（在此假設水平面為 xz 平面）旋轉 θ 角然後平移 (x_0, z_0) ，此計算的索引 i 會涵蓋 Point cloud 上所有的點，權重 w_i 可依使用者決定每個點的重要性來自由設定。權重的設定對不同類型的動作時會有不同的最佳設定，但在尋找比較同一組動作之間所有的動作片段時，其權重的設定必須一致。最後計算出來的距離結果，其距離越大者則兩動作片段越不相似，反之亦然；但是計算出來的距離數值是沒有實質上的絕對意義。換言之，我們只能從距離數值的大小來得知兩組動作片

之間那一組連接起來後會是相對起來品質比較好的轉換，而不能經由一個絕對的數值來決定何者是品質好的連接。

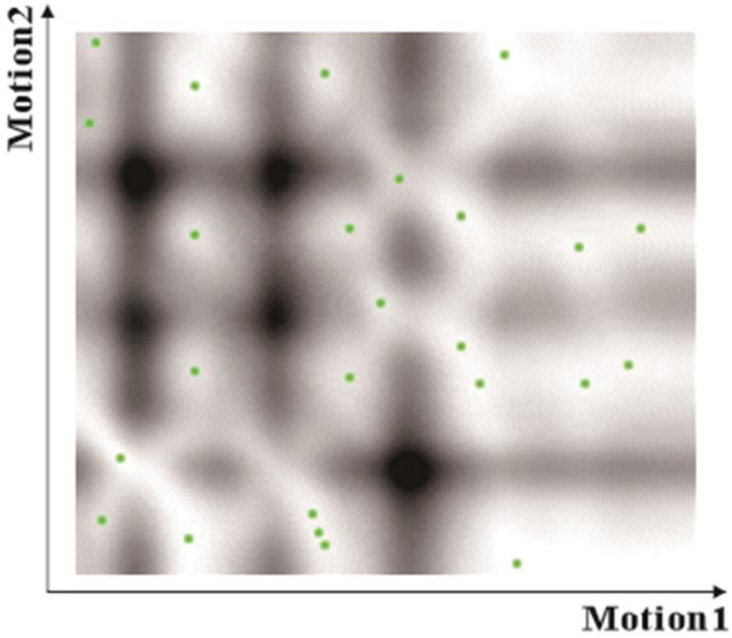


圖 3.3、動作距離及候選連接點選擇示意圖，顏色越深代表距離值越高，而圖上綠色的點為候選轉換點

3.2 動作圖建立

在產生動作圖之前使用者必需先收集及擷取所有在應用情境中會使用到的動作，整理後形成一個做為基底的動作擷取資料集合。動作擷取資料中的每一個影格在圖上皆是一個獨立的點，而點和點之間的有向邊連接則代表這兩點間可以平順的轉換。原始動作擷取資料中相鄰的兩個影格必定是可以平順的轉換的，因此在起始時我們會將到原動

作擷取資料中相鄰的點加上連接（圖 3.4(B)），接來下為了找到預設以外的連接，我們使用前面所提到的 Point cloud distance 來計算所有動作片段之間兩兩的相似度。在計算完兩動作中所有動作片段之間的相似度之後，我們以一個二維的矩陣來表示其結果，如圖 3.3 所示。我們接著將矩陣上各區的 Local Minima（圖 3.3 中的綠點）取出，做為我們的候選轉換點（Candidate transition point），並透過指定一個相似度的門檻值，如果在候選轉換點中兩動作片段 P_i 和動作片段 P_j 的相似度 $D(P_i, P_j)$ 的值小於我們所指定的門檻值，則我們將此兩動作片段視為足夠相似可以來創造連接的組合。由前所述 $D(P_i, P_j) = D(P_{j-k+1}, P_{i+k-1})$ ，如果當 $D(P_i, P_j)$ 小於我們所指定的門檻值時，我們會分別創造兩個有向連接（圖 3.4(C)），分別是從 P_i 到 P_j 以及從 P_{j-k+1} 到 P_{i+k-1} ，並為這兩個連接計算一個轉換花費所使用的的式子如下：

$$t = 1 - e^{(-s/\sigma)} \qquad (3.5)$$

其中 t 是指前面的轉換花費，s 是兩動作片段的相似值，σ為選取連接點用的門檻值，而 t 的範圍介於 0 和 1 之間（0 為最平順的轉換而 1 是最不順的轉換）。當所有動作之間的連接建立完成後，最後一個步驟就是將這張圖去蕪存菁，去除掉那些會使我們在使用時進入死路（Dead End）的結點，即找出這張圖的最大強連接點集合（Strongly connected component, SCC），而這最後所找到的最大強連接點集合所構成的圖就是我們最後所要的動作圖（圖 3.4(D)）。

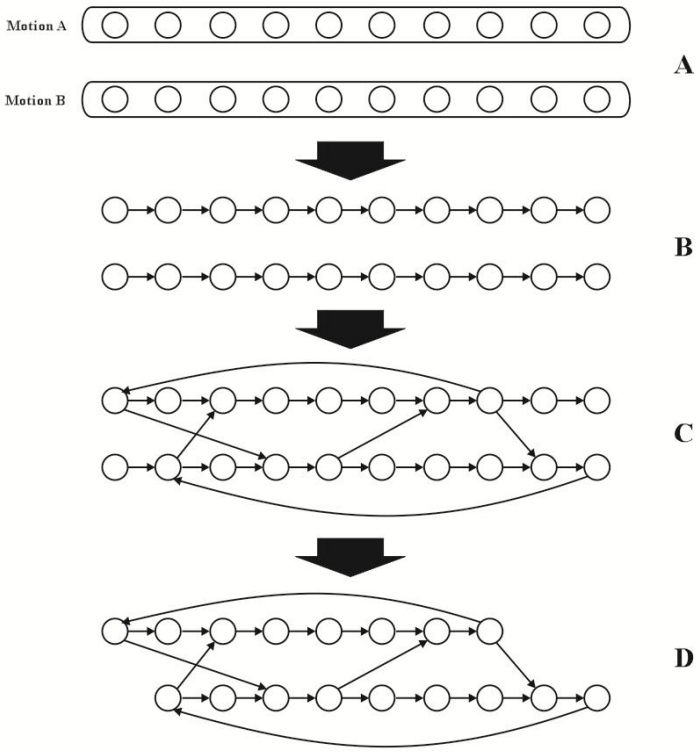


圖 3.4、動作圖建立各階段示意圖

3.3 問題

動作圖為動作的混合提供了一個相當具有保證性的自動化技術，同時動作圖具有結構簡單以及能夠使用簡易的路徑搜尋演算法產生長動作的能力，但動作圖仍有許多在使用上或在建立時待解的問題，因此至今仍相當的受到研究者們的歡迎，並應用在動畫的自動產生之系統等的議題上。

在本論文中我們所面對的問題是，在使用傳統動作圖時，當你想要從一動作切換至

另一個動作，你必須全身的動作一起切換，即便同樣是邊走路邊揮手這一類的動作，僅只是想要從揮左手換成揮右手也是一樣。或是當你的動作圖中有一個走路動作以及一個原地講話的動作時，可能就會有人想說何不有個一邊走路一邊講的動作，但在傳統的動作圖中，如果一開始用來建立圖的動作擷取資料庫中沒有一邊走路一邊講話這個動作的話，使用者是無法經由使用動作圖來產生的。以上這兩個例子都指向了同一個問題：動作圖除了可以在不同的動作間可以混合出用來平順轉換的轉換動作外，並無法自動的混合不同身體部份的動作。因此在本研究中，我們企圖以為身體其他部位也計算動作圖的方式，使身體的部位可以根據該部位的動作圖來進行動作的轉換，以達將身體不同的部位間用不同動作來組合出新動作的目的。而難題將會是如何判斷出那些是好的動作組合。

第四章

多層式動作圖之建構

傳統動作圖所能合成的動作是大幅度依賴動作資料庫的內容。基本上傳統動作圖是一個描述多個不同動作之間的連接關係圖，並不具有能夠組合出新動作的能力。因此如果想要使用傳統動作圖產生出一個變化豐富的連續動作時，動作圖之中勢必要包含相對大量的動作種類。但是，如果我們能夠改變現有的動作圖結構，使其具有能力可以藉由重新組合動作資料庫中的部份動作來產生出新的動作，那我們就可以用常見的數個基本動作，搭配一些部份變化的動作即能夠在不需要大量動作資料庫的狀態下，建立出一個有能力產生高度變化的動作圖。舉例來說，假設原先動作資料庫有以下三個動作，分別是走路、原地站立且雙手插腰以及對著人講話，我們希望除了能夠產生原先資料庫中就具有的三個動作外，更能產生邊走路邊講話、手插腰上走路等能從不同資料組合部份動作的新全身動作。

在本論文中，從 Chen [5] 的研究所得到的靈感，我們提出了一個由多張動作圖所構成的階層式動作圖結構，稱之為**多層式動作圖**。經由使用多層式動作圖，我們就可以如前面所舉例的，僅切換部份身體的動作（ex. 僅切換上半身的動作或左手的動作等...）來產生出新的動作，同時因為使用與傳統動作圖相彷的資料結構，所以得以與傳統動作圖可以平順的在不同的動作間轉換。為了更具體描述我們在使用上的主要想法，我們將

以下面兩張動作圖所構成的一個已完成的**多層式動作圖**為例說明。 $MG_{全身}$ （Full Body Motion Graph）以及 $MG_{上半身}$（Upper Body Motion Graph）（如圖 4.1 所示），其中 $MG_{全身}$ 是一張由全身的動作所建立而成的動作圖（即傳統的動作圖），而 $MG_{上半身}$ 則是由動作擷取資料庫中的上半身動作所計算建立而成的動作圖。我們基本是想法是， $MG_{全身}$ 的用途如同傳統的動作圖一樣，經由在其上使用路徑搜尋演算法，我們可以產生由一連串 $MG_{全身}$ 中有包含的動作所組成的新動作，而當我們僅想要改變一個動作的上半身動作並保持其當下下半的動作時（ex. 將走路其上半身的動作由擺手改成講話的動作），就將上半身產生動作的方式切換到 $MG_{上半身}$ 來做動作間轉換的搜尋與產生，並將此產生出來的上半身動作加以覆蓋至 $MG_{全身}$ 產生的動作的上半身後，即為我們所想要的結果。

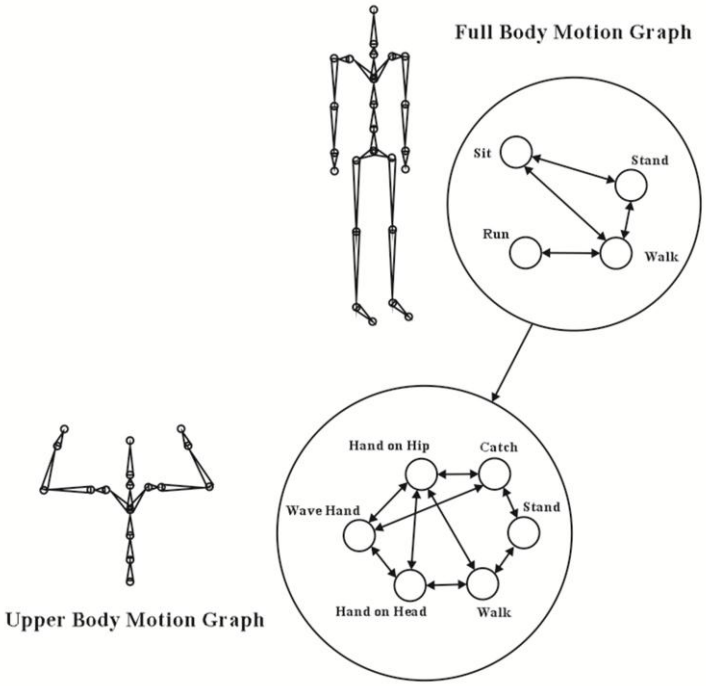


圖 4.1、由上半身及全身動作圖所構成的兩層式動作圖

4.1 多層式動作圖建構概論

大體上來說，我們的演算法會先使用傳統動作圖來產生一張做為主幹的動作圖（$MG_{全身}$），然後依照應用情境的需求來產生一個至數個部份動作的動作圖（ex. $MG_{上半身}$ 、 $MG_{左手}$ 等…），最後將產生出來的多個動作圖依照骨架的階層關係給連接起來，產生**多層式動作圖**。而在連接的過程中我們會為所有相互可以替換的部份動作間計算一個「整體動作相似度」的數值。

依照以上的說明，整個多層式動作圖的建立過程大略可分為四大步驟：

- 第一步，對要用來計算動作圖的動作擷取資料進行前處理，在前處理的過程中需要以人工的方式對動作擷取的資料做出基本的分類，並對每個動作加上足以表示該動作特性的標籤（tag）。
- 第二步，全身動作、部份動作（要將動作切為幾個部份由使用者決定）等個別以自動化的演算法來尋找每組動作擷取片段之間適合連接的地方。
- 第三步，在個別的圖上建立轉換用的連接，並且刪除會導致在各別圖上運行時進入死路的結點。
- 第四步，將個別產生出來的圖加以連接，並計算「整體動作相似度」數值，產生多層式動作圖。

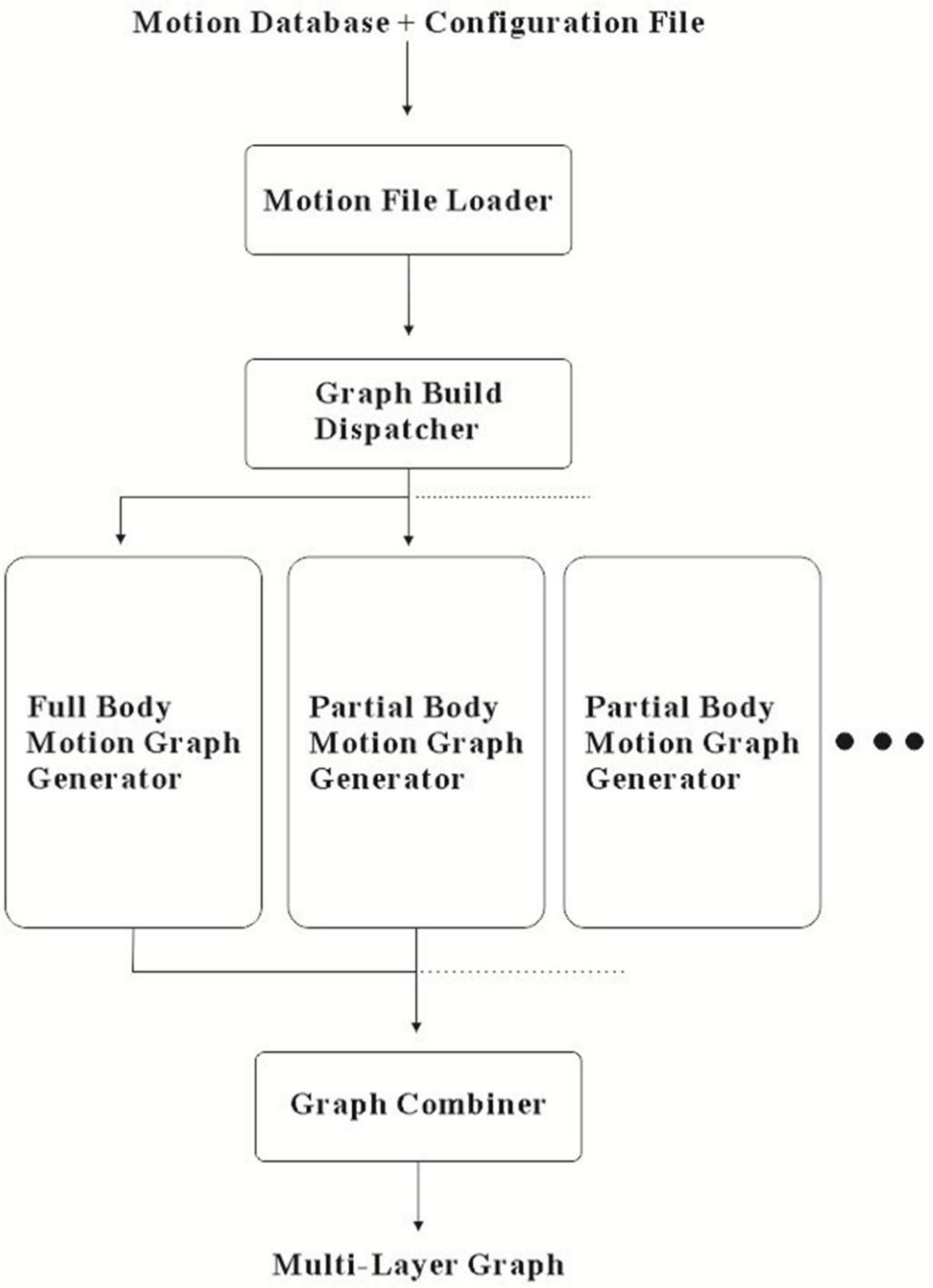

Motion Database + Configuration File
Motion File Loader
Graph Build Dispatcher
Full Body Motion Graph Generator
Partial Body Motion Graph Generator
Partial Body Motion Graph Generator
Graph Combiner
Multi-Layer Graph

圖 4.2、系統架構圖

4.2 多層式動作圖建立系統

而在本章接下來內容中，4.2.1 節我們會先介紹依照 4.1 節的建立步驟所設計出來的多層式動作圖所建立的 IMMGC （Intelligent Media Lab's Motion Graph Computation System），並在緊接的章節中對整個架構的流程做出細部的說明。

4.2.1 動作圖建立系統 IMMGC 總覽

在本研究中，我們設計了一個名為 IMMGC 的多層式動作圖建立架構，而 IMMGC 的系統架構如圖 4.2 所示，圖中每一個方塊為一個獨立的模組，而模組與模組之間的箭頭代表了資料傳遞的方式。這個系統的目的如前所述，是要產生一個部份動作可以獨立切換且具強連接性（SCC, Strongly Connected Component）的多層式動作圖。此架構主要由動作讀取器（Motion File Loader）、動作圖建立分派器（Graph Build Dispatcher）、動作圖產生器（Motion Graph Generator）和動作圖合併器（Graph Combiner）所組成。動作讀取模組（Motion File Loader）負責將動作資料以及設定檔（Configuration File）轉換成適合用來計算的容器內，而設定檔內儲存了前處理階段時所設定的各項屬性，像各個動作的標籤以及分群的資料等；動作圖建立分派器會根據使用者所給予的設定檔，負責將讀取進來的動作以及個別的設定派送給對應的動作圖產生器，各個動作圖產生器在接收了這些資料後就會開始著手產生動作圖，並將結果傳給動作圖合併器，最後合併器收到各個動作圖產生器的結果，再以我們所制定的規則連接後，就是我們所要的多層式動作圖了。

4.2.2 資料前處理

在資料前處理階段總共有兩個部份需以人工的方式來處理，分別是為每個動作加上具代表性的標籤。在本研究中，因加入了部份動作後，會使動作圖能切換的動作組合的數量，因原始動作資料的數量增加，而以等比級數的方式成長，大大的增加了使用者在使用時的複雜度。為了解決這個問題，我們提出了加入簡易的語意來輔助使用者控制。透過語意的輔助，使用者可以自行的創造複合式或是更加高階的控制指令。所以前處理的第一步是以人工的方式將各個動作的全身動作、上半身動作、下半身動作等分別加上適當的的標籤如：走路（WALK）、跑 （RUN）、跳（JUMP）、揮手（WAVE_HAND）等...。

為了去除一些不合理的動作轉換以及增加動作轉換時的視覺順暢性，在前處理的階段我們會以人工的方式來標記允許相互轉換的動作，我們借由這個人工介入的輔助來避免一些不合理連接的產生，但要如何在這些動作間找到適合的連接點仍是採用自動化的演算法。

4.2.3 動作圖產生器細部架構

動作圖產生的細部架構如圖 4.3 所示，整個動作圖產生的架構主要是基於 Rafidah 等人[14]所建議的動作圖計算基本架構所延展設計而成，其主要的模組有轉換點偵測器（Candidate Transition Point Detector）、動作距離計算器（Distance Metric Computation Method）、連接點產生器（Transition Generator）以及圖形清理器（Graph Cleaner）。

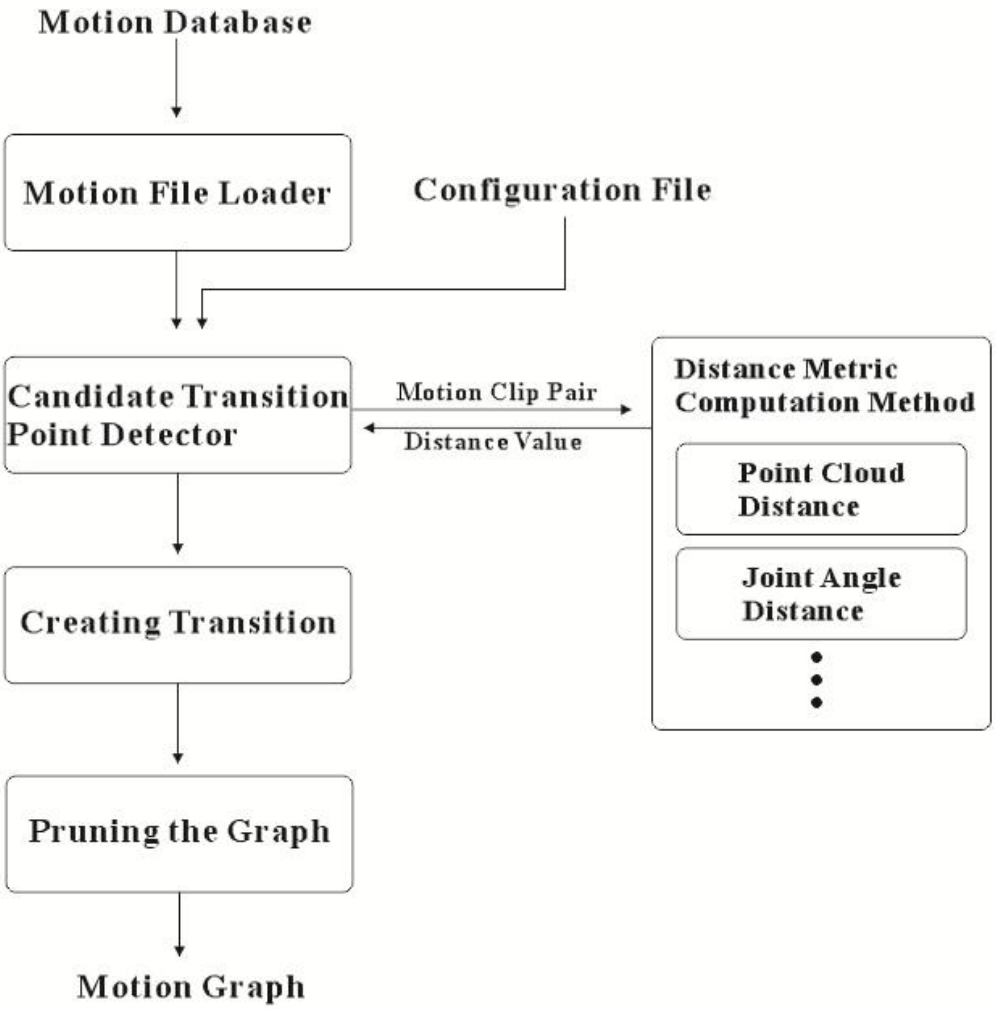


圖 4.3、動作圖計算細部架構示意圖

4.2.3.1 動作相似度比較

在這裡比較兩兩動作片段相似度的方式是透過計算兩兩動作片段間距離的方式，距離越大則兩動作片段越不相似，反之亦然。在接收了動作圖建立派送器所讀取的動作資料以及設定資料後，轉換點偵測模組將讀取進來的動作，參照前處理時所寫下的設定檔，

整合後把每對動作一一分派給距離計算模組，而距離計算模組則會依照使用者設定的需求來選擇適當的距離計算方式。在這裡我們實作了 Point Cloud Distance（式 2.3）來做為我們計算動作相似度距離的方法。

4.2.3.2 動作連接節點選擇

如在第三章所提到的，我們無法透過一個絕對的門檻值來二分出那些是好的接點組合而那些不是，因此我們參考了 Kovar 等人[10]的方法來設計選擇連接點的方法。當我們計算完兩動作間所有動作片段的相似度並將其圖形化後，我們可以得到一個如圖 3.3 所示的分佈圖，圖上的每一個點都代表了對動作片段的組合，圖中兩軸分別是兩個用來比較相似度的動作，點的顏色越深代表該對動作片段是較不相似的（距離值高），淺色反之。在有了這個二維的相似度分佈的資料後，我們先選出各個位於區域低點的點做為候選的轉換點，形成一個候選轉換點的集合，然後由使用者來決定要從這個候選轉換點的集合中選出那些來做為正式的轉換；若選出來的點多，則最後圖的連接性佳，但有可能會出現品質較差的轉換，反之若只選出其中最好的幾個轉換點，則整體運行的品質會較為良好，但圖的連接性會變的較差。以上的步驟化為系統上的流程就是，當候選點偵測模組得到了距離計算模組回傳的結果後，會選出一個候選的轉換點集合並傳給製造轉換的模組，然後製造轉換的模組再依照使用者所給定的門檻從候選的轉換點的集合中，選出合乎規定的轉換點並加以連接。在以往的研究中，當找到一個合乎規定的轉換點 (A_i, B_j) 後，建立連接時通常會將 A_i 到 $A_{(i+k-1)}$ 的動作片段與$B_{(j-k+1)}$ 到 B_j 的動作片給混合起來產生 A_i 到 B_j 的轉換動作（Transition Motion）（如圖 4.4(a)所示），但在本的研究中，多層式動作圖中是一個由多張動作圖所結合而成的結構，如果在建立圖時

就將轉換的動作都一同產生好的話，那最後將產生一個相當龐大的資料結構。因此，本研究在產生 (A_i, B_j) 間的連接時，會 A_i 上寫上特別的標記（如圖 4.4(b)所示），待運行需要用到此連接時，再根據這個標記將連接的動作即時產生出來。

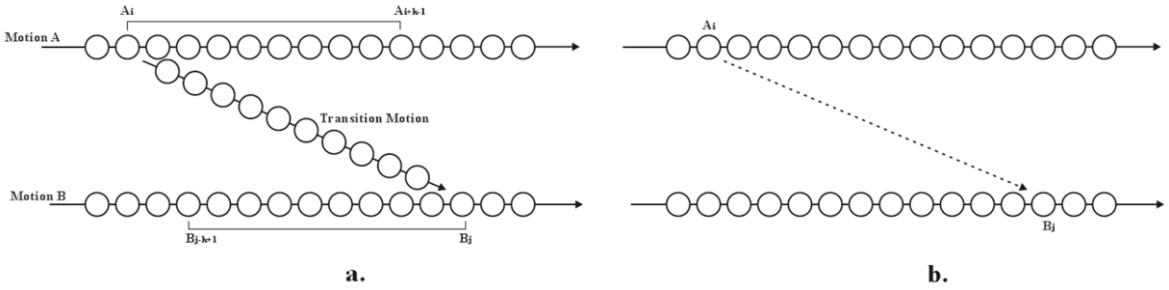


圖 4.4、動作連接示意圖

4.2.3.3 動作圖清理

截至目前為止，我們已經可以自動的產生出一個基本的動作圖了，但要順利的經由動作圖產生動作仍有一些問題需要解決，因為在目前這個階段的動作圖中仍存在一些會讓在圖上搜尋或是 Random walk 時進入死路（Dead-End）的不必要節點，因此最後的修剪器模組所負責的工作就是要將所有會導致死路的結點刪除，而在本系統中我們使用 Tarjan's Algorithm[17]來做為刪除死路的方法；Tarjan's Algorithm 是 Robert Tarjan 所提出的強連接尋找法，Tarjan's Algorithm 一開始會任選圖上的一結點進行深度優先搜索（Depth First Search），在搜索過程中會將經過的節點都存放到一堆疊（Stack）中，當搜索時如果遇到一個已經被搜索過的點的話，會先檢查該結點是否是任一強連接的根結點並將其從堆疊中刪除。如果該結點是任一強連接的根結點的話，則該結點與會堆疊中其它的點形成了一個強連接。在這個搜索完成後，最大強連接所構成的圖就是我們所要的動作圖了。

```
algorithm: Tarjan's Algorithm
input: graph G = (V, E)
output: set of strongly connected components (sets of vertices)

index := 0
S := empty
for each v in V do
   if (v.index is undefined)
      strongconnect(v)
   end if
repeat

function strongconnect(v)
   // Set the depth index for v to the smallest unused index
   v.index := index
   v.lowlink := index
   index := index + 1
   S.push(v)

   // Consider successors of v
   for each (v, w) in E do
      if (w.index is undefined) then
         // Successor w has not yet been visited; recurse on it
         strongconnect(w)
         v.lowlink := min(v.lowlink, w.lowlink)
      else if (w is in S) then
         // Successor w is in stack S and hence in the current SCC
         v.lowlink := min(v.lowlink, w.index)
      end if
   end for

   // If v is a root node, pop the stack and generate an SCC
   if (v.lowlink = v.index) then
      start a new strongly connected component
      repeat
         w := S.pop()
         add w to current strongly connected component
      until (w = v)
      output the current strongly connected component
   end if
```

圖 4.5、Tarjan's Algorithm

4.2.4 合併建立多層式動作圖

到目前這個階段，我們已將各個動作圖個別建立完成了，而接下來的工作就是要將個別建立的動作圖連接起來。在多層式動作圖的概念中，動作圖之間合併的定義是：以在骨架上的階層關係將兩動作圖連接建立父子關係。所以如兩動作圖之間在骨架上相互沒有階層關係則無法合併。舉例來說，如果現在有一張全身動作的動作圖與上半身動作的動作圖，在骨架上的關係是上半身的骨架是全身的子階層，因此我們會將上半身的動作圖做為全身的動作圖的子圖而與之連接；但如果現在除了前述的兩張圖外又加入了一個左手動作的動作圖，左手動作在此最接近的父階層為上半身，因此會先做為到上半身動作圖的子圖與之連接。更為複雜的連接結果如圖 4.6 所示；又如果現在分別有一左手動作的動作圖與一右手動作的動作圖，這兩個動作在骨架階層上並無父子關係，故無法連接產生多層式動作圖。

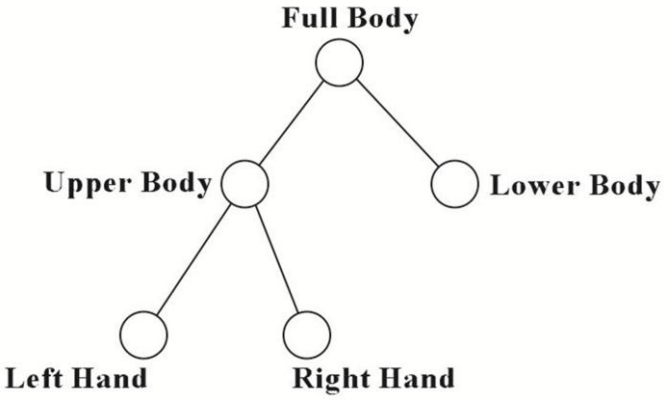


圖 4.6、較為複雜之多層式動作圖簡易完成示意圖，圖上的每一個圈圈皆代表一張動

在合併動作圖時，之所以要子階層的圖參考父階層的圖上的動作後再與之合併，是因為如果我們由一父圖的動作上，其一部份身體的動作切換到子圖上運行時，除了該身體部份外，仍是繼續使用原來的動作，因此我們必需確保身體部份動作在子圖上運行選擇動作時，能夠選擇與原本動作較為相似動作，否則就有可能會組合出不自然動作的結果。而在這裡我們在設計上使用了兩個方法來降低這個問題發生的機會，第一個方法是在前面所提到過的，在前處理階段以人工標記的方式大略的去除掉一些不可能的組合；而第二個方法是為相互可以轉換的兩兩動作間（全身動作間、上半身動作間等...）計算一個名為「整體動作相似度」的數值，而計算出來的值會以表格的方式記錄下來，做為使用多層式動作圖時選擇動作的參考。「整體動作相似度」數值的概念如接下來所述：當想要用一個部份身體的動作來替換到一動作所對應到的部份身體的動作時(ex.將走路動作的上半身替換成講話動作)，是需要一個指標來讓系統判斷這個替換適不適合的，而這就是「整體動作相似度」數值的用途。所以「整體動作相似度」是用來讓系統判斷一動作適不適合用來替換另一動作的數值指標，因此以下兩點是計算這個數值時所要考慮的：

- 連接性
- 動作相似度

由於我們切換部份身體動作的方式是經由使用該身體部位的動作圖來切換的，與欲切換的動作相間的連接點的數量是否夠多，連接的品質好不好都是要考慮的，而這兩者所對應到的就是「連接性」；而另一點要考量的便是，當身體的一部份的動作用別的動作來

取代時，必須要考慮用來取代的動作是否能與身體餘下部份的動作相容，不相容的話便有可能會產生出有視覺上缺陷的動作，因此在取代時便會希望用來取代的動作能與原動作有一定的相似度，而這便是我們提到的第二點「動作相似度」。基本上這兩點都可以透過我們在尋找兩動作間適合連接的位置時所計算的動作距離來衡量，所以直覺上會想說將兩動作相對應的動作距離值給加總加起來，如下式所示：

$$\sum_i D_{i,i} \qquad (4.2)$$

其中 i 為影格的編號，$D_{i,i}$為動作 A 與動作 B 在第 i 個影格間的動作距離，計算之示意圖如下：

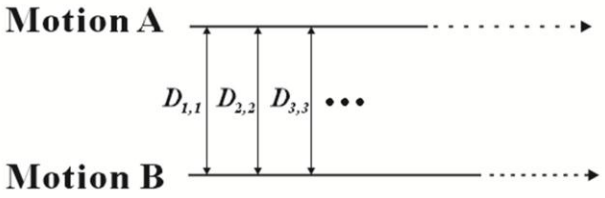


圖 4.7、整體動作相似度初步想法計算示意圖

如果當我們取代動作時都是使用索引值是一樣的影格來取代的話，那上面的式子是可行的，但實際上在運作時，我們是使用動作圖來尋找轉換的方式，因此在原動作與用來取來的動作在索引值上必定會有一個偏移量（offset）(如圖 4.8 所示)，到這裡我們便出現了我們的第二個想法，在這個想法中，我們大略的將偏移量分成數個級距，並為每一個級距都計算一個相似度值，待使用時再依照實際的狀況來查尋所對應的偏移量的相似度值便可。但在這但又出現了另一個問題，因此我們的動作彼此間長度都是不一樣的，且

大多的動作都是可以自我循環（Self-Loop）的，一但動作循環後，偏移量便會產生變化，因此這個想法也是不可行的。

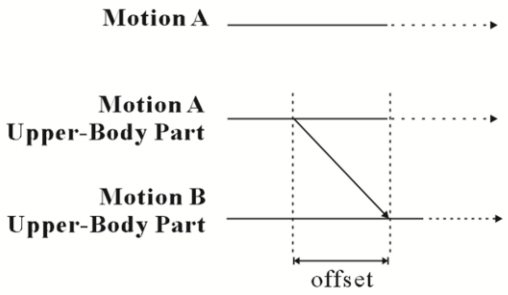


圖 4.8、動作取代偏移量示意圖

綜合並檢討前述的想法後，我們將動作以每個區段大小的 s 的方式切分成數個區段，這部份是使用了第二個想法中將偏移量分成數個級距的想法，而因為實際使用時偏移量是會一直變化的，故需要將所有的偏移量都一同列入考量，此外，因為動作間長度不一，因此最後需要取一個平均來使不同動作組合間的「整體動作相似度」能夠相互比較。而最後，我們的「整體動作相似度」其定義如下：

$$\frac{\sum_{i=1}^{m/s}\sum_{j=1}^{n/s} minD_{i,j}}{(m/s)\times(n/s)} \qquad (4.2)$$

其中 m 與 n 分別為動作 A 與動作 B 的影格數，s 是每個區段的大小，而$minD_{i,j}$為兩動作區段中所有影格間使用式(3.4)計算出之距離的最小值。計算之示意圖如下圖 4.9 所示：

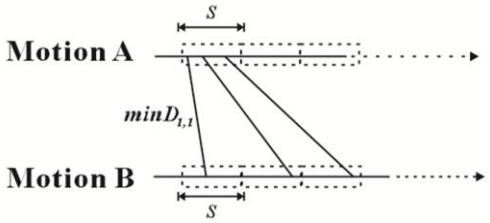


圖 4.9、整體動作相似度計算方式示意圖

最後我們以圖 4.1 中舉的$MG_{全身}$ 及 $MG_{上半身}$ 為例說明。如圖 4.1 所示，並有兩個動作圖，上半身動作的動作圖（之後以$MG_{上半身}$代替）與全身動作的動作圖（之後以$MG_{全身}$ 代替）。如上一個例子所述，依階層關係我們會將 $MG_{上半身}$ 做為子圖連接至$MG_{全身}$ 上。在連接的過程中，會發現Walk和Stand的上半身動作有被包含在$MG_{上半身}$中，因此系統會個別計算 Walk 及 Stand 的上半身動作與其它所有有相連的上半身動作的「整體動作相似度」，並將這些數值存到表格內以備之後查詢。到了這裡，會發現我們的方法有一個很大的限制，那就是當兩個在骨架上有階層關係的動作圖合併成多層式動作圖後，父層上的一個動作其部份動作能夠切換至子層的圖上做轉換，只有當其部份動作在建圖時有被包含在子圖上才行，這個限制在下一章的內容中將會再進一步說明。

第五章

使用多層式動作圖

由於多層式動作圖的資料結構與傳統動作圖相當的相似，所以如同使用傳統動作圖一樣，在多層式動作圖上的任一條路徑都可以轉換成一段連續的動作。在搜尋的演算法方面，從最簡易的隨機搜尋、最短路徑搜尋、A*或是以最佳化演算法方式都可以用來進行搜尋。在此我們設計了一個使用多層式動作圖來產生動作的系統架構，如圖 5.1 所示。

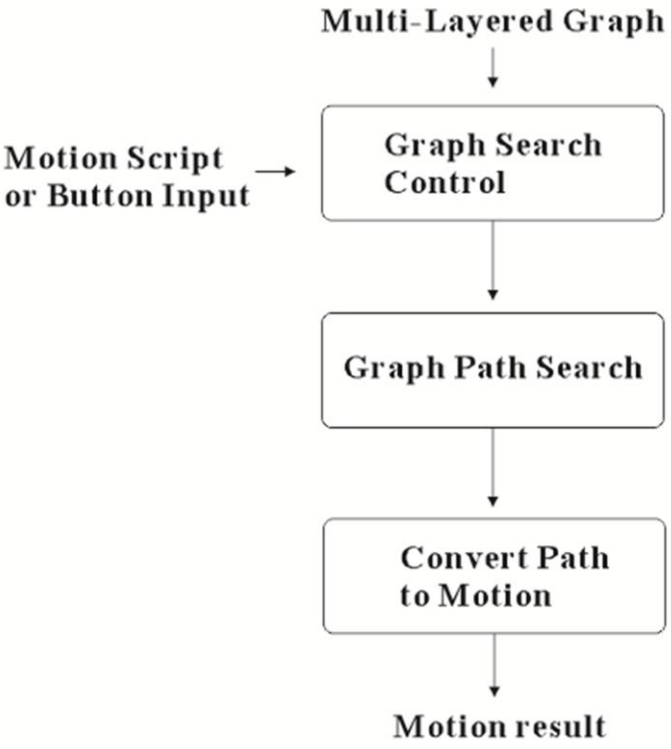


圖 5.1、多層式動作圖動作產生架構

整個多層式動作圖動作產生架構主要由三個模組所組成，分別是動作搜尋控制（Graph Search Control）、動作路徑搜尋（Graph Path Search）以及路徑轉換（Coverting Path to Motion）。 如一開始所提到的，整個搜尋的過程被分成了兩大步驟，分別是在動作類層級的 Global Search，以及在動作片段層級的 Local Search。而這兩大步驟與上述模組的對應分別是Global Search對應到動作搜尋控制,以及Local Search對應到動作路徑搜尋。其中動作路徑搜尋與路徑轉換這兩部份，與傳統動作圖產生動作的流程大略上是一樣的，因此在此不再多做贅述。多層式動作圖在使用上與傳統動作圖最不一樣的地方，在於如何根據使用者的需求，依照一定的規則轉換成連續且較小的搜尋指令，並正確的分配到對應的動作圖上，而這就是我們所提到的 Global Search（也是動作搜尋控制模組）所要做的事，因此在本章接下來的內容將整個 Global Search 所做的事進行說明。

5.1 多層式動作圖階層切換規則

經由上一章的介紹後我們得知，多層式動作圖的資料結構是由多個動作圖以階層的方式組合而成。與傳統動作圖最不一樣的地方在於搜尋時需要在不同階層間的動作圖切換，因此我們在沒有制定一些基本的使用規則之前較難定義搜尋工作如何進行，以合理的產生動作。所以在建立多層式動作圖的搜尋演算法之前，制訂適當的使用規則是相當重要的。在本節接下來的內容中，我們將進一步描述整體運作的方式，以說明多層動作圖在不同階層動作圖之間切換的使用規則。

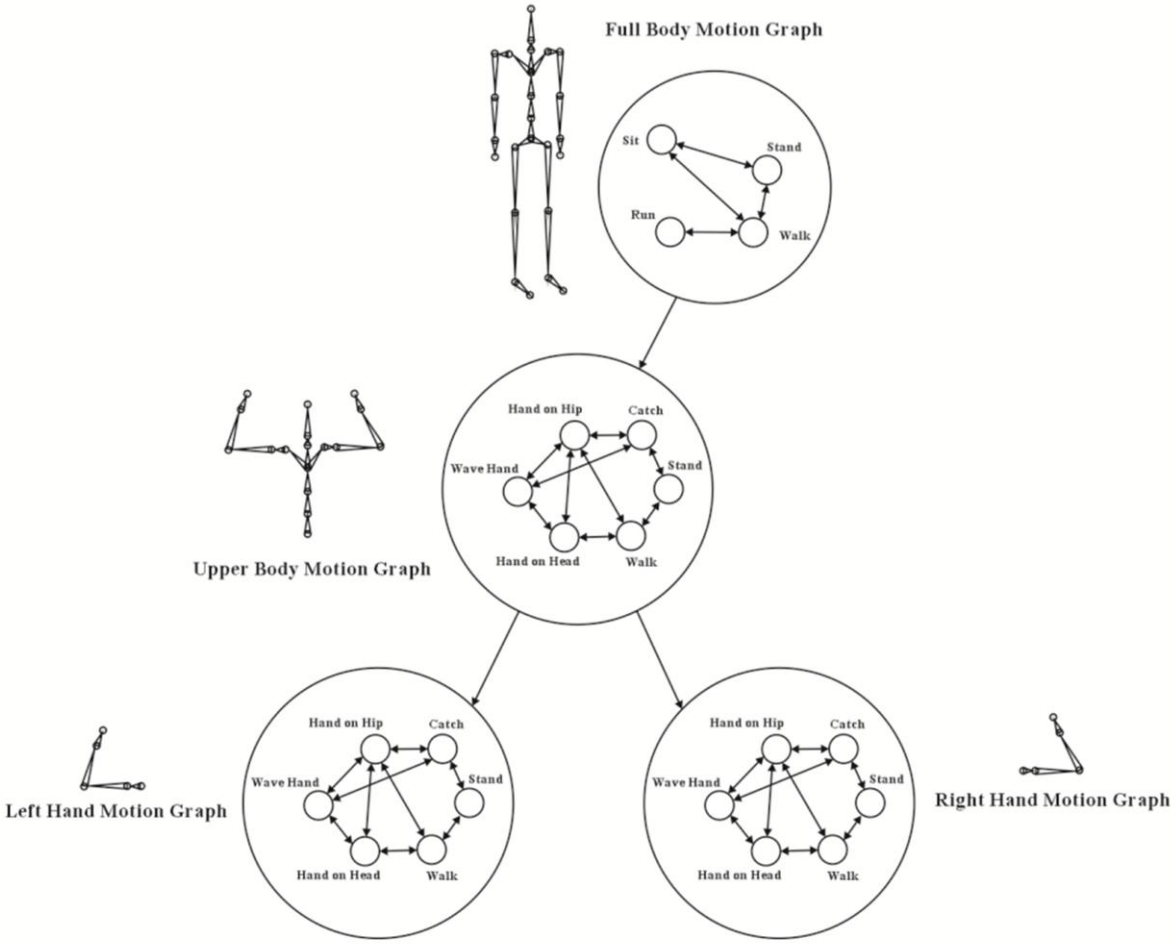


圖 5.2、說明範例所做用之多層式動作圖

多層式動作圖在使用時的基本流程如下：多層式動作圖階層中的每一個節點都代表一個傳統的動作圖。以圖 5.2 之組成為例，動作圖起始在多層式動作圖的根節點（圖 5.2 中的 Full Body 動作圖）上的任一點（動作），此時只有用到單一層的動作圖，因此在使用上如同傳統的動作圖一樣，當使用者想保留其下半身的動作並只改變其上半身的動作

時，系統會啟始化上半身的動作圖（在此指圖 5.2 中的 Upper Body 動作圖），並開始由上半身的動作交由上半身的動作圖來產生。因此，第一條使用規則為：

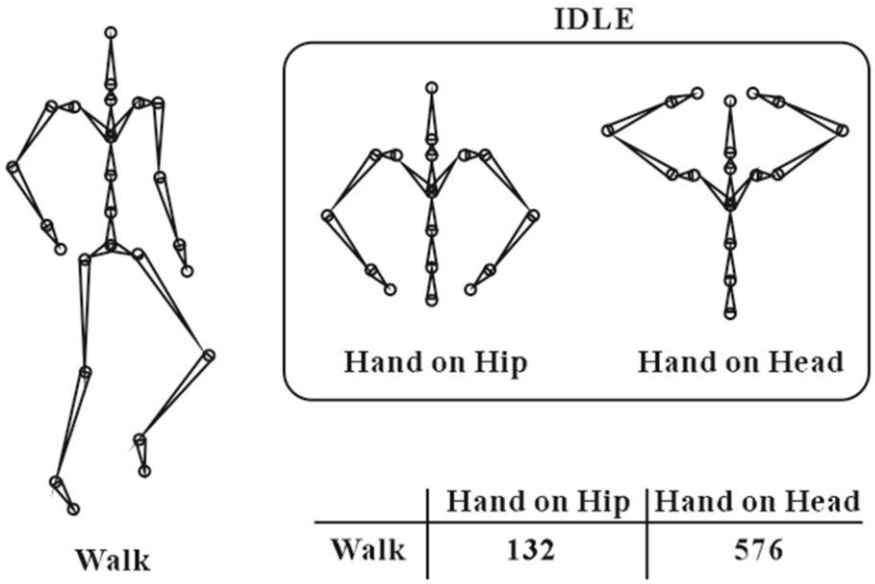
- **一個動作欲只切換其骨架結構上的子動作時，其原始之子動作必須被包含在所對應之子動作圖上。**

如圖 5.2 的多層式動作圖，在全身動作圖的四個動作中，欲使用上半身動作圖來轉換上半身的動作時，只有 Stand 和 Walk 這兩個動作的上半身是可以轉換的，因為這兩個動作的上半身動作有被包含在上半身動作圖中。更進一步，如果現在在上半身的動作圖之下還有一右手的子動作圖時，在上半身部份轉換到所需求的動作之後，如果希望可以保留此時上半身的動作，並只改變其右手（下一層）的動作，也需套用同樣的規則來檢查。接續前面的例子，假設現在的動作狀態是全身 Walk ，上半身 Hand on Head，如果現在想要將全身動作轉換成 Stand，此時就出現了第二條規則：

- **當父層的動作圖要轉換動作時，會先「結束」所有正在使用中的子動作圖後，再進行轉換。**

根據這個規則，當現在要將全身的動作轉換成 Stand 前，需要先「結束」其上半身的動作圖，而在這裡「結束」的意思是將子動作圖上的動作轉換至與父動作圖上相同的動作，並且同步父子動作圖上的動作後，就可以將產生身體架構部份動作的工作交還給父圖。因此要將全身的動作轉換至 Stand 前，需要先將上半身的動作由 Hand on Head 轉換回 Walk，並且需與全身的 Walk 同步後，全身動作才轉換至 Stand。然而，在有限的時間內要在子圖上要找到一條「剛剛好」能回到與父圖相同的動作，並與之同步的轉換路徑其實是相當困難的。因此我們引入了兩個做法來使這個目的能夠更容易達成。其一，搜尋

時允許在一定的程度內進行動作變形；其二，在使用其一的方法後如仍無法在一定的路徑長度內找到路徑，則強制即時混合動作出一條新的路徑來使動作能夠同步化。



	Hand on Hip	Hand on Head
Walk	132	576

圖 5.3、動作選擇示意

5.2 動作選擇

有了前面所制訂的規則後，在開始使用多層式動作圖之前，仍有一個問題需要解決，那就是當一部份身體的動作交由另一張子動作圖獨立運作來產生動作時，有可能子動作圖會產生與原動作差異相當大的結果，造成產生出一個相當怪異的動作組合。因此如何在子動作圖上找到既符合需求又與身體其餘部份搭配起來最佳的動作，是在開始使用前所要解決的最後一個問題。在本研究中，我們提出一個以動作標籤、計算「整體動作相似度」及使用簡易語意所組合而成的方法來解決這個問題。透過使用動作標籤與簡易的語意，使用者得以一次指定一群相互相類似的動作，而經由計算「整體動作相似度」，

便可以根據計算出來的數值從這群動作中找到一個最佳且符合需求的動作。我們再次以上一章圖 5.2 中的兩動作圖所組成多層式動作圖為例，當全身的動作在 walk 時，想要將上半身的動作轉換到有 IDLE 標籤的動作上（假設在此上半身的動作中有 IDLE 標籤的有 Hand on Hip 及 Hand on Head 這兩個動作，如圖 5.3 所示），會先從事先將算好的表格中尋找 Walk 的上半身動作與所有具有 IDLE 標籤的動作之間的整體動作相似度數值，並取數值最小的做為我們所要轉換的目標動作，以圖 5.3 為例的話最後會選擇 Hand on Hip 做為轉換的目標動作。此外，我們也將動作相似度延伸使用在動作和動作間轉換的選擇。

Full-Body Graph Motions		Upper-Body Graph Motions	
Idle_01	IDLE	Idle_01	IDLE
Idle_02	IDLE	Walk_01	SWING_HAND
Walk_01	WALK	Walk_02	SWING_HAND
Walk_02	WALK	Walk_03	WAVE_HAND
Walk_03	WALK, WAVE_HAND	Music_01	Music, IDLE
Music_01	Music, IDLE	Talk_01	TALK, IDLE
Talk_01	TALK, IDLE	Talk_02	TALK
Talk_02	TALK		

圖 5.4 示範 Motion Script 能力用的動作之標籤設定

5.3 Motion Script

為了方便使用者以語意方式提出對動作的要求，我們設計了一個語意式動畫腳本語言（在本研究中稱為 Motion Script），透過簡易語意的描述，對系統下達動作的命令。此腳本語言的 BNF 定義如圖 5.5 所示。

```
<motion-command> ::= <hybrid-command> | <hybrid-command> "for" <time>
 <hybrid-command> ::= <simple-command>
                        | <simple-command> "and" <hybrid-command >
                        | <simple-command> "with" <hybrid-command>
<simple-command> ::= <graph-name> <motion-term> | <motion-term>
      <motion-term>::= <motion-tag> | <motion-name>
            <time> ::= <constant> "sec" | <constant> "min"
        <constant> ::= <digit> | <digit> <constant>
           <digit> ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
```

圖 5.5、Motion Script 之 BNF 定義式

在腳本語言 Motion Script 中，最基礎指定動作的方式是透過 <motion-term> 來指定，而<motion-term>又可以分為<motion-name>和<motion-tag>兩種。以閒置動作為例，在動作圖上可能有三個不同的閒置動作其名稱分別為：Idle_01、Idle_02，且這三個動作皆具有「IDLE」的動作標籤。使用時可以用直接指定這些動作的名稱（ex.「idle01」）的方式，來指定所想要轉換到的特定閒置動作，而這裡的名稱就是<motion-name>。但有時使用情境並不在意所轉換到的具有 IDLE 標籤的動作是那一個，那就可以用指定動作標籤的指令，例如：

Motion Script>>> IDLE

以圖 5.4 之設定為例，便會選取 Idle_01 或是 Idle_02 這兩者的其中一個。但如果所指定的標籤在根圖上並不存在呢?例如：

Motion Script>>> SWING_HAND

那我們就會將此指令派送至較下層的動作圖裡來檢查是否有符合需求的動作。以圖 5.4 之設定為例，這裡會選擇上半身圖中的 Walk_02 或 Walk_03 做為目標動作。使用指定動作標籤的方式一次可能選定多個可能的動作，並交由系統自動從中選出一個當下最好的目標動作進行轉換，在這個動作標籤指的就是<motion-tag>。此外，我們也可以在<motion-tag>前加上動作圖的名稱來指明我們欲在其上進行動作轉換的動作圖，例如：

Motion Script>>> Upper-Body TALK

在這個指令中，Upper-Body 為一動作圖的名稱，使用這個指令會在 Upper-Body 動作圖上尋找 TALK 標籤的動作，以圖 5.4 之設定為例便是尋找上半身圖中的 Talk_01 或是 Talk_02 來做為目標動作，當指名了欲在其上進行動作轉換的動作圖的名稱後，在指定的動作圖上找不到結果，並不會如同之前會將指令派送至較下層的動作圖做搜尋。如果想要進行更為複雜的轉換的話，Motion Script 也提供了「with」和「and」這兩個詞來讓我們可以組合出複合式的指令，像是：

Motion Script>>> WALK with Upper-Body TALK

使用這個指令就可以找到一個全身動作為 WALK 標籤的動作且上半身為動作標籤為 TALK 的動作組合，以圖 5.4 之設定為例，有可能會產生 Walk_01、Walk_02、Walk_03 與 Talk_01、Talk_02 所組合出的六種可能性中的一種，而另一個例子：

Motion Script>>> TALK and IDLE

這個指令則的目的是可以找到一個具有 TALK 及 IDLE 這兩個標籤的動作組合，以圖 5.4 之設定為例，會找到 Talk_01 做為目標；但如果無法在單一層的動作圖裡找到一個同時具有這兩個標籤的動作，如下面的例子：

Motion Script>>> WALK and TALK

那就會將這兩個標籤分開到可以同時運行的兩個動作圖上做尋找，看是否可以用組合不同身體部份動作的方式來符合指令的需求，以圖 5.4 之設定為例，便會找到由 WALK 及 TALK 所組合出來的六個動作中的一個來做為目標動作。最後，Motion Script 更具有可以指定每個動作指令執行的時間，例如：

Motion Script>>> WALK for 10 sec

這個指令會在執行「WALK」指令 10 秒後再執行接續在其後的 Motion Script。

在前面對 Motion Script 有了簡單的介紹後，對於動作標籤的使用上有兩點需要注意：其一、擁有同一個動作標籤的動作，必須在其動作的意義上具有一定的相似程度；其二、使用的動作標籤在字詞上的意義，也需與被指定該標籤的動作具有一定的關連程度。如果在一開始給予各個動作標籤時沒有注意到上面這兩件事，則可能造成在使用 Motion Scripts 時，會產生一些預期之外的結果。但如果標籤有做出良好的設定，Motion Script 與多層式動作圖可說是相輔相成的最佳組合，這點由「and」的例子（TALK and IDLE）可以了解到；在上面「and」的例子中，雖然下的是一個相當簡單的語句，但可以充份的利用多層式動作圖的特性，在單一層中找不找結果時，可以在可以同時運行的不同層間做進一步的嘗試。

第六章

實驗結果與討論

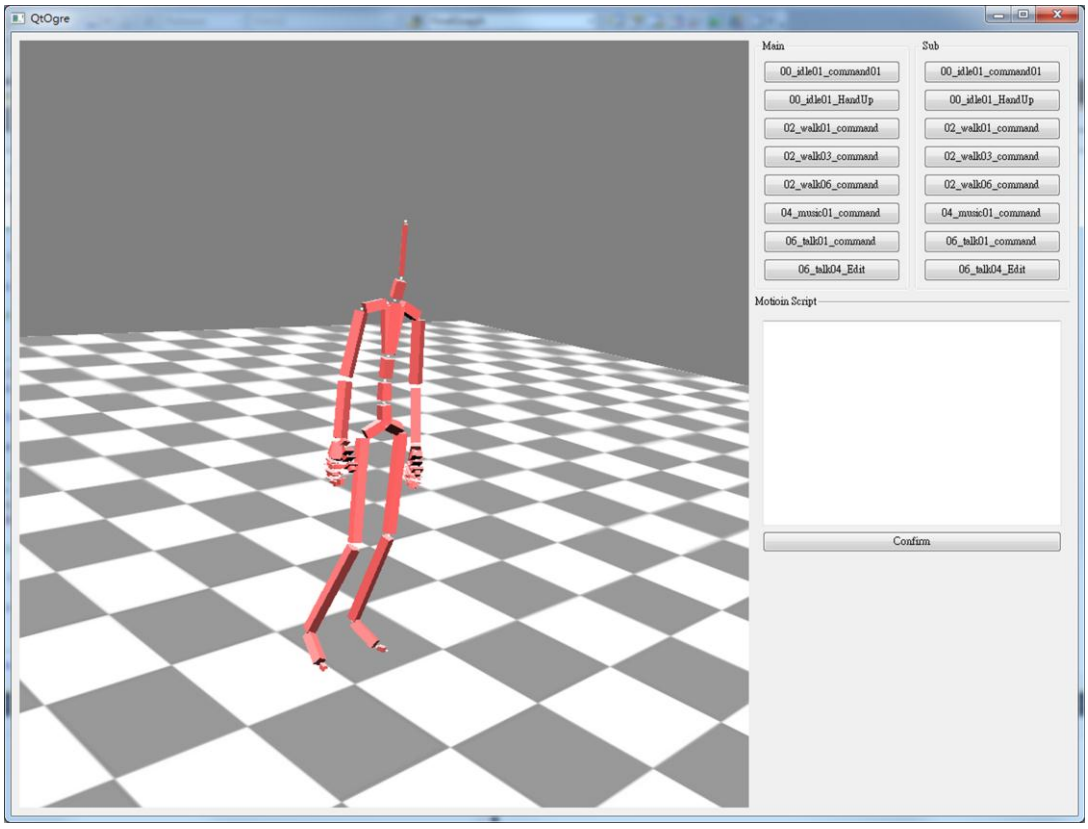


圖 6.1、實作程式之介面

在本章的內容中，我們將會對前面章節所提出的多層式動作圖，進行實作，並提供圖形介面供使用者輸入及指定動作。接著，我們將會透過實驗來進行比較與測試。

6.1 系統實作

我們目前所使用的電腦設備 CPU 為 Intel Core i5 2.53Ghz，記憶體大小為 4GB，顯示晶片為 nVidia GeForce 420m。我們使用 C++做為系統開發語言來實現前面所提到的架構。在多層式動作圖建立的部份，為了加速前處理的速度，我們設計了可以平行化計算的演算法，並使用 Intel TBB（Intel Threading Building Blocks）[24]做為平行計算用的函式庫。在多層式動作圖的使用上，我們以 OGRE(Open-source Graphics Rendering Engines）[25]做為 3D 繪圖引擎，搭配 Qt[26]來產生 GUI 介面，實作多層式動作圖及動作產生應用程式。而我們實驗中使用了 8 個不同風格的動作擷取資料（共約 3000 影格）來建立多層式動作圖，所使用到的動作擷取資料格式為 BVH（Biovision Hierarchy），實驗所使用的動作擷取資料皆使用相同的骨架結構，如圖 6.2 所示。另外，在路徑搜尋演算法上，我們使用了 BFS 演算法。

圖 6.1 為目前我們所實作出來運用多層式動作圖來產生動作的應用程式，其操縱方式與一般的第三人稱 3D 遊戲相仿。畫面中央的紅色骨架人物為呈現動作結果的虛擬人物，為方便觀察產生動作之結果，我們將人物的位置鎖定在畫面的中央，使用者可以用滑鼠的三個鍵的來控制觀察的角度及遠近。介面右上方的按鈕區提供了使用者以按鈕的方式來選擇動作，其中每一排的按鈕對應到多層式動作圖上的一個動作圖（在圖 6.1 的例子中，使用了由兩張動作圖所構成的多層式動作圖，故有兩排按鈕供使用）。而介面右中的輸入框讓使用者可以輸入我們所定義之 Motion Script（客製化的動畫腳本）的方式來指定所需要的動作。

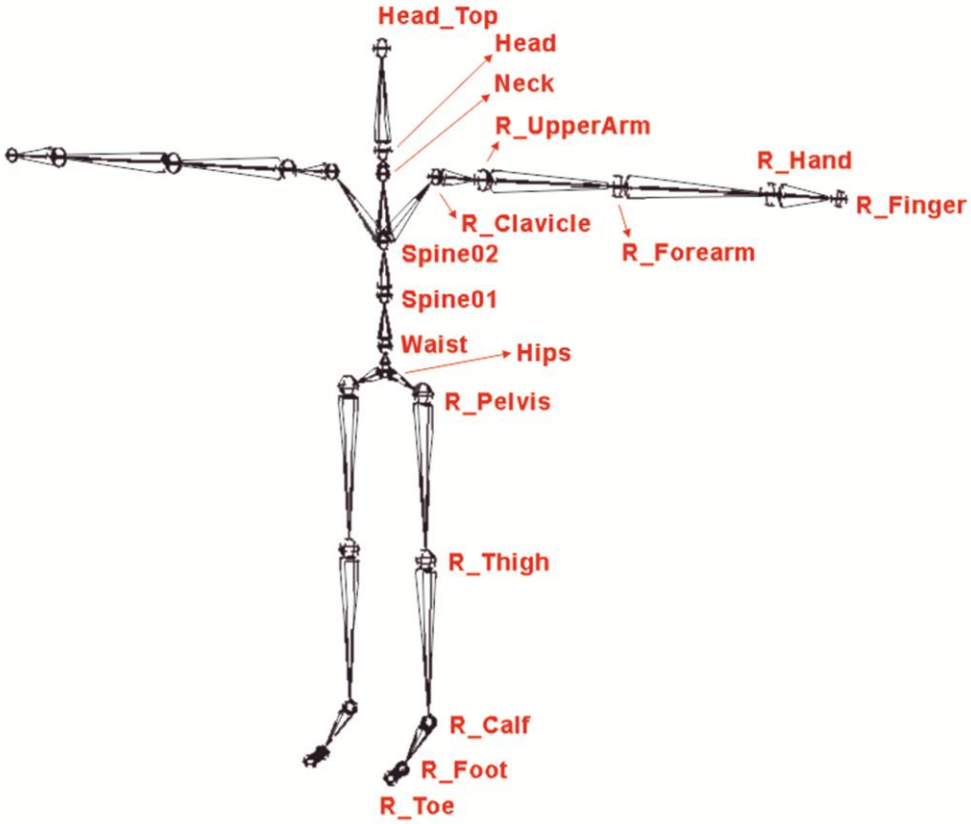


圖 6.2、實驗所使用的骨架

6.2 整體動作相似度驗證

由前面的敘述可以得知，使用一般的搜尋演算法僅能在動作轉換處找到一個最為平順的轉換路徑。然而在本研究中，在切換身體一部份的動作時若使用同樣的方法，可能會產生出切換過後的子部份動作與身體其餘部份動作相互不協調的結果。因此我們提出了透過計算整體動作相似度的數值（式 4.2），來做為在切換子部份動作時的選擇依據；

但由於整體動作相似度所提供的是一個期望值的概念，因此動作選擇上所找到的會是一個平均的最佳解。

在此我們設計了一個實驗，以驗證我們所提出的整體動作相似度數值，在大部份的狀況下都能使我們找到較好的動作組合結果。我們使用了相同的動作組合分別計算了兩張由全身及上半身構成的兩層式多層式動作圖，其中一張多層式動作圖在計算的過程中，我們不對其所有上半身的動作之間進行整體動作相似度的計算。然後去比較這兩者在實際使用時所選擇的上身動作，與原全身動作的上半身動作的差異，比較是方式是去加總所有「用來取代的動作片段」與對應的「原動作的動作片段」之間的 Point Cloud Distance；加總出來的值高，則代表用來取代的片段與原片段較不相似，反之則較相似。由於轉換的結果當下所在動作圖上的位置有關係，因此同一組動作間的轉換的實驗我們都會進行 20 次。而每一次實驗我們分別想要觀查，短期的使用狀況下、一般使用的狀況下以及動作至少有一次以上循環的狀況下，因此我們讓動作維持 3 秒、10 秒、20 秒（實驗用的擷取動作影格數都在 300 上下、FTP 為 30，時間上約為 10 秒，因此維持 20 秒可以確保動作至少循環一次以上）來分別觀查其加總的距離值。

	Talk_01	Talk_02
Walk_01	187.252	427.763

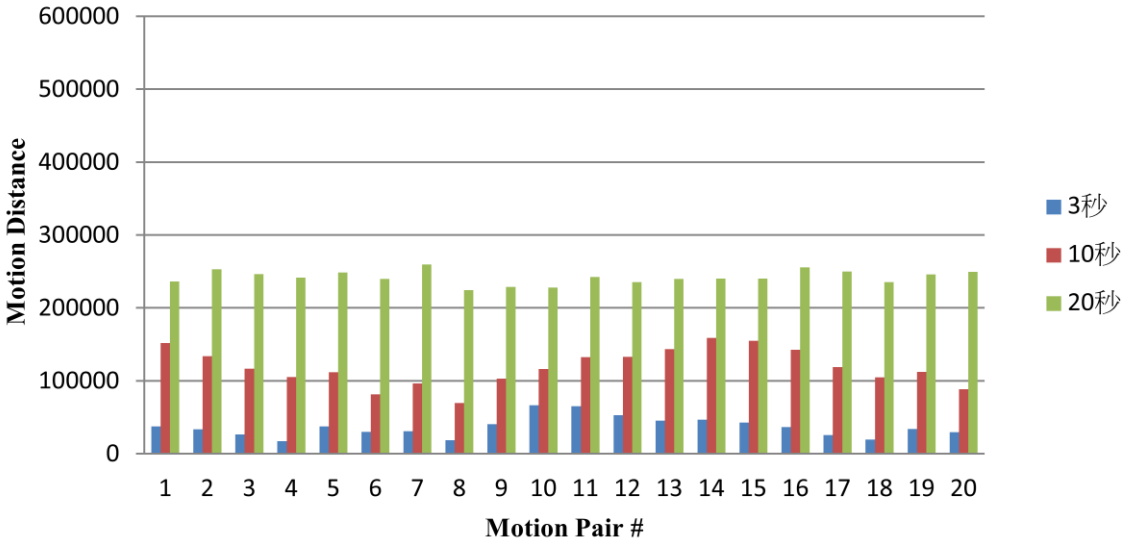
圖 6.3、具有 TALK 標籤的兩動作與 Walk_01 上半身動作的整體動作相似度數值

6.2.1 驗證實驗

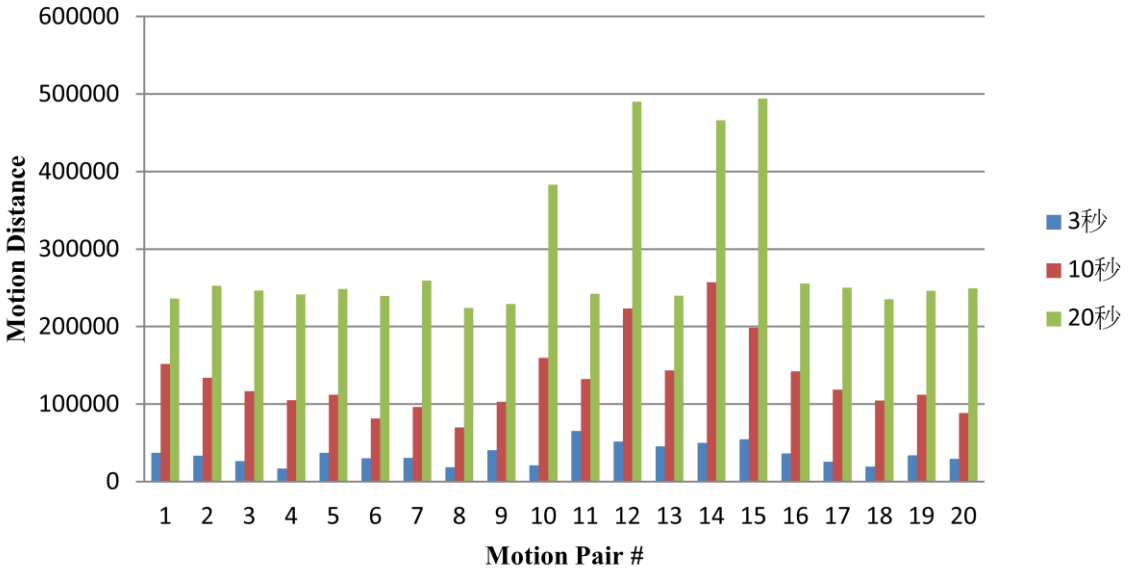
在本實驗中我們所使用的語法為「Walk_01 with Upper-Body TALK」，上半身的動作以標籤的方式來指定一群具有 TALK 標籤的動作來供系統選擇，其中上半身中具有

TALK 標籤的動作共有 Talk_01 與 Talk_02 兩個，而其 Walk_01 與這兩個動作在上半身部份的整體動作相似度值分別如下圖 6.3 所示：

實驗的數據結果如圖 6.4 所示，圖 6.4 中圖(a)與圖(b)互為同一次語法指令下，使用整體動作相似度與沒有使用的狀況下之結果，其中圖 6.4(a)為使用了整體動作相似度的結果，而圖 6.4(b)則是沒有使用的結果。從 6.4(a)的數據結果可以看到使用了整體動作相似度做為選擇的參考後，20 筆資料在短期與正常使上的動作距離值上互有好壞，在長期的表現上則彼此差異不大。但綜合來說每次產生動作的表現都算是相當的平均。而從 6.4(b)沒有使用整體動作相似度的數據結果中可以看到一些不一樣的東西，在查看實際選擇動作的結果（如圖 6.5 所示）後發現除了編號 10、12、14、15 外，大部份所選擇的動作與使用了整體動作相似度是一樣的。而大部份所選到一樣的動作這件事是可以被預見的，如前面所提到過的，整體動作相似度提供了一個巨觀的期望值在兩動作的相似度上，由整體動作相似度的式子（式 4.2）可以看到，整體動作相似度的值其實是由許多區部的值所加總而成的。因此，兩動作其在巨觀的期望值要好的話，勢必其大部份的區域間的表現都要不錯才行，所以在一半以上的情況下找到了與使用了整體動作相似度相同的結果是合理的。而編號 10、12、14、15 之所以會選擇了不同動作的原因，可以由短期（3 秒）的距離數值來觀察到；這四個轉換在其短期的距離數值表現上都相當的不錯，但中長期的表現都不是很好，這是因為一但沒有了整體動作相似度的指示，搜尋演算法便只能根據區域的資訊來做搜尋，一但出現了區域最低值（Local Minima），便有可能會陷入。經由以上的結果驗證了我們所提出的整體動作相度數值，可以做為一個適合的相似度指標。



(a)



(b)

圖 6.4、整體動作相似度驗證實驗結果

編號	動作選擇結果		編號	動作選擇結果	
	使用	沒使用		使用	沒使用
1	Talk_01	Talk_01	11	Talk_01	Talk_01
2	Talk_01	Talk_01	12	Talk_01	Talk_02
3	Talk_01	Talk_01	13	Talk_01	Talk_01
4	Talk_01	Talk_01	14	Talk_01	Talk_02
5	Talk_01	Talk_01	15	Talk_01	Talk_02
6	Talk_01	Talk_01	16	Talk_01	Talk_01
7	Talk_01	Talk_01	17	Talk_01	Talk_01
8	Talk_01	Talk_01	18	Talk_01	Talk_01
9	Talk_01	Talk_01	19	Talk_01	Talk_01
10	Talk_01	Talk_02	20	Talk_01	Talk_01

圖 6.5、整體動作相似度驗證實驗之動作選擇結果

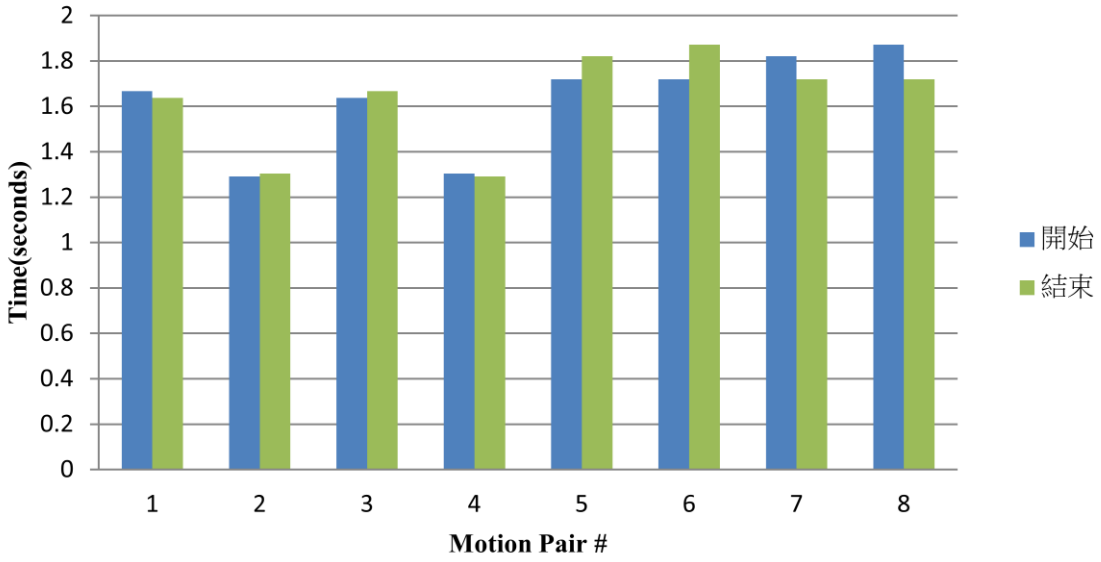


圖 6.6、單層的多層式動作圖其效率測試結果

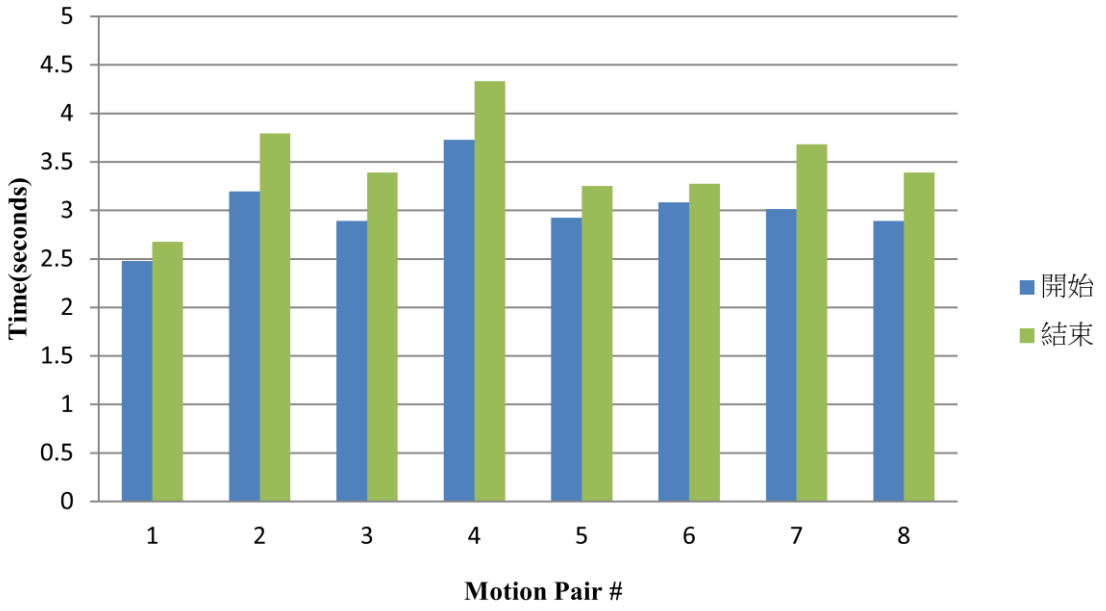


圖 6.7、二層的多層式動作圖其效率測試結果

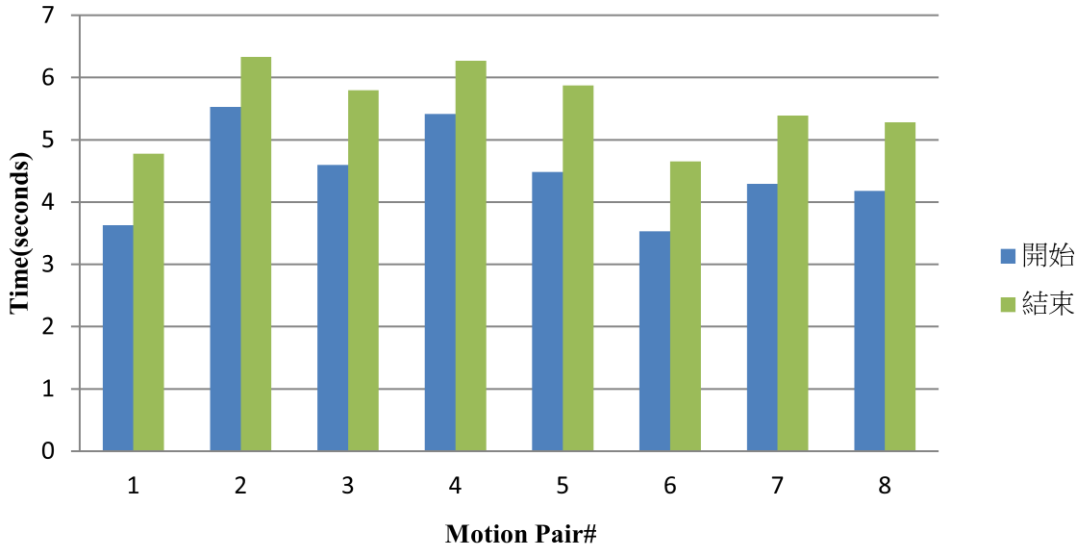


圖 6.8、三層的多層式動作圖其效率測試結果

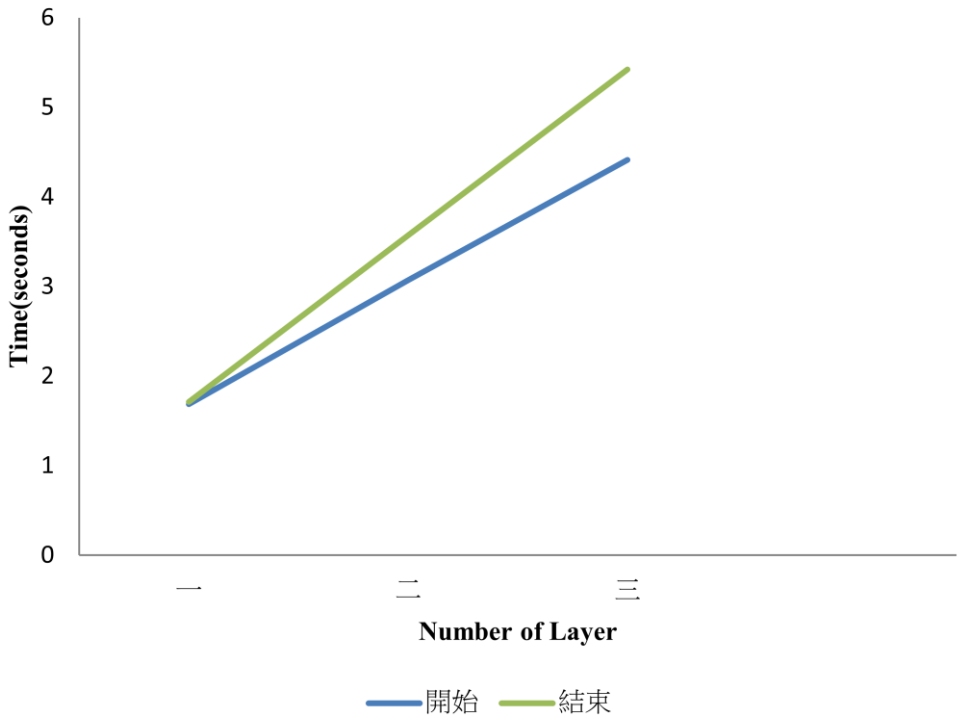


圖 6.9、層數與動作產生效率的關係

6.3 動作轉換效率測試

在衡量一個動作圖的品值是否良好上，除了轉換動作是否平順外，另一個使用者最容易感受到的便是動作轉換所需的時間。動作間轉換所花的時間（亦為在動作圖上的路徑長度，因動作圖上的一條路徑可轉換成一連串的動作影格，在動作的 FPS 固定的情況下，可換算出所需的時間）多的話，會造成使用者在使用時有指令沒有送達的錯覺。我們在第五章中提出了多層式動作圖產生動作的方法，其中提到了整個搜尋演算法是階層式的，所以使用多層式動作圖來產生動作，從下指令到動作變成所需的動作，其轉換變化過程時間上的花費，理論上會隨著所使用到的動作圖圖層數量呈線性的關係成長（因每一層皆要做一次搜尋與轉換）。

在這個效率測試的實驗中，我們分別對層數是一層、二層以及三層的多層式動作圖進行了測試。由於每個動作之間的連接性都不盡相同（連接性與動作轉換所需的時間息息相關，連接性好的動作間，搜尋出來轉換用的路徑大多都較短，轉為動作後撥放所需的時間較少），為了得到比較客觀的數據，我們在每一層的測試中，皆任取 8~10 個不同的轉換組合進行測試，而每一個轉換組合皆由兩個動作所構成，分別為起始動作及目標動作，而起始動作皆為多層式動作圖其根動作圖上的一個動作。舉例說明所使用到的測試組合如下：

一層：（Idle01, Walk01）

二層：（Idle01, Walk01 with Upper-Body Talk01）

三層：（Idle01, Walk01 with Upper-Body Talk01 with Left-Hand Wave01）

此外，轉換的所需的時間更與當下所在動作圖上的位置（在不同的位置上，搜尋到的路徑長度皆不盡相同）有關係，因此同一組動作間的轉換的測試我們都會進行 20 次，測試轉換至目標（開始）與由目標轉換回來（結束）的效率，最後將其取平均值後做為我們的綜合比較用的實驗數據。

圖 6.6、圖 6.7 及圖 6.8 為我們分別對層數是一層、二層以及三層的多層式動作圖的數據結果。其中橫軸編號為所使用動作轉換組合之編號；縱軸為轉換所需的時間。每個轉換組合對應到兩個直條方塊，藍色方塊為由起始動作轉換到目標動作從下指令到轉換成功所需的時間，紅色方塊為由目標動作回到起始動作從下指令到轉換成功所需的時間。

圖 6.6 為層數是一層的多層式動作圖的數據結果。單層的多層式動作圖與傳統的動作圖並沒有什麼差別，從圖中可以看到每一組轉換中，轉換至目標動作與由目標動作轉回所需的時間是相當接近的。圖 6.7 為使用層數是二層的多層式動作圖的數據結果，比較圖 6. 6 與圖 6.7 後我們可以看出以下兩點。第一點，整體轉換所需的時間皆比單層的多層式動作圖為多，且約接近兩倍左右。會造成這點的原因是，相較於單層的多層式動作圖，兩層的多層式動作圖主層與其子層皆分別需要一次轉換，因此相較於單層的多層式動作圖其所需的時間平均起來約為兩倍。第二點，由目標動作轉回所需的時間皆比轉換至目標動作所需的時間多，而在單層的多層式動作圖的測試數據中則無這個現象。而造成這點的主要原因是，當使用到複數層的動作圖時，轉換回起始動作（如前所述，起始動作皆為多層式動作圖其根動作圖中的一個動作）就會需要進行「同步化」這個動作，而這通當需要較長的動作才能達到這個目的。我們更進一步加入層數是三層的多層式動作圖之測試數據來比較（見圖 6.8），平均起來所需的時間為單層的三倍，證實了我們前面對於第一點所解釋的原因。而同樣的，由目標動作轉回所需的時間皆比轉換至目標動作所需的時間多，也證實了我們在前面對於第二點所解釋的原因。

最後將這三個的測試數據取平均後整合在一起的結果如圖 6.9 所示。從圖 6.9 可以看到轉換過程時間上的花費與圖層數量呈線性的關係成長，且由目標動作轉回起始動作所需的時間的成長又較多，證實了我們一開始的推論。此外，當使用兩層圖時，平均轉換所花費的時間就需要 3 秒左右，尚可令人接受。但當使用到三層圖時平均轉換的時間更成長到了 4.4 秒左右，已是有感的等待了。因此，我們建議應用時主要以兩層式的多層式動作圖為主。

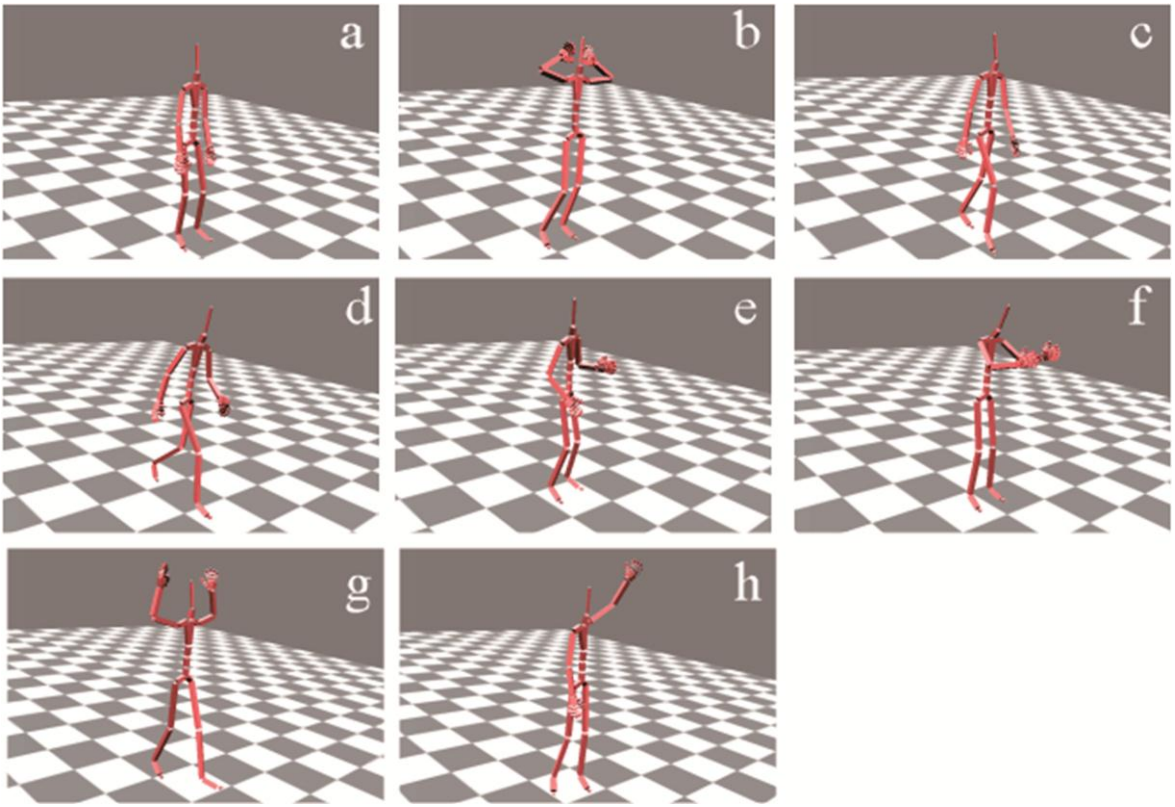


圖 6.10 實驗用動作資料庫裡所含的動作，動作分別為：a.雙手下擺的閒置動作、b.雙手抱頭的閒置動作、c.走路動作、d.較快的走路動作、e.原地全身律動的動作、f.講話動作、g.雙手舉起的走路動作、h.手高舉的講話動作

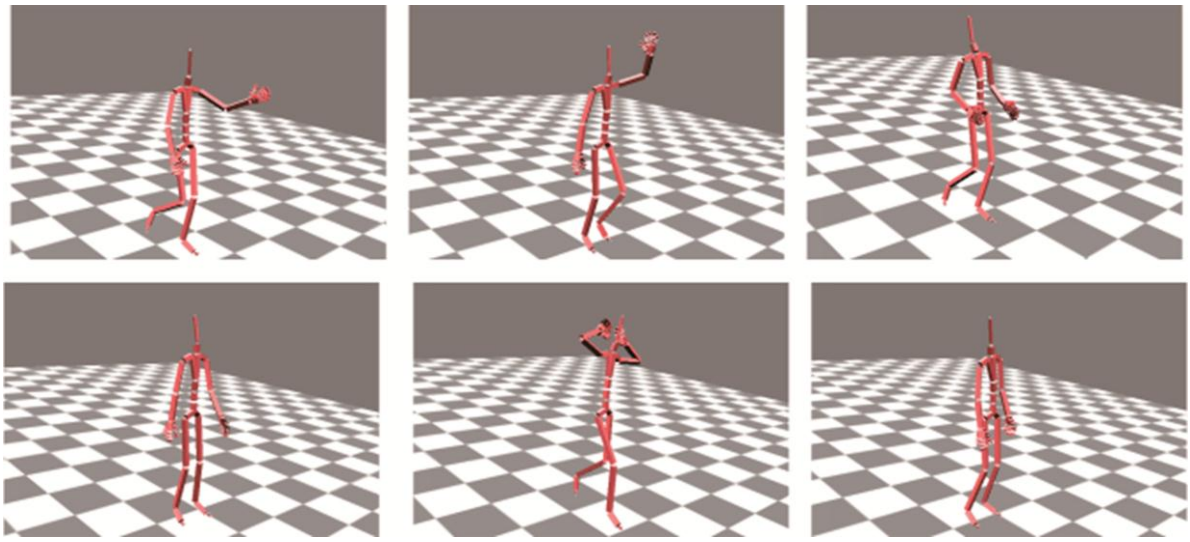


圖 6.11、使用本研究的方法後可額外產生的動作（將身體拆分成全身與上半身的狀況下），動作分別為：1.邊走路走講話的動作（c+f）、2.手位置較高的邊走路邊講話的動作(c+h)、3.邊走路身體邊律動的動作(c+e)、4.站在原地雙手擺動的動作(a+c)、5.手抱頭走路的動作(c+b)、6.僅下半身律動的動作(e+a)（英文代號皆對應至圖 6.10 的動作，其中 c+f 代表動作由 c 及 f 所對應到的動作所組成）

6.4 綜合應用範例

如前面所提到過的，我們實作出了一個可以運用多層式動作圖來產生動作的應用程式，使用者可以透過圖形介面上的按鈕或是輸入 Motion Script 的方式來控制動作的產生。當熟悉上敘兩種控制方式後，使用者便可以利用這個應用程式產生一些原先動作資料庫裡所不具有的動作，如圖 6.11。圖 6.10 中所示的八個動作為實驗用動作資料庫所含有的動作，當使用這些動作來產生傳統的動作圖時，所能制造出來的動作僅是在這八個動作間轉換的連續動作。但在使用本研究的方法後，在圖 6.12 的設定之下，由全身以及上半身動作的動作圖所組成的多層式動作圖，可以額外的產生 49 種組合出來的動作，而其中在視覺上可用的動作有 24 種。圖 6.11 中所示為最佳的六個結果，如圖所示，我

們的研究可以透過組合不同動作的方式，可以更有效率的利用資料庫裡所含有的動作，但又同時保有了動作圖可以在動作間平順轉換的特性。

Full-Body Graph Motions		Upper-Body Graph Motions	
動作名稱	動作標籤	動作名稱	動作標籤
Idle_01 (a)	IDLE	Idle_01	IDLE
Idle_02 (b)	IDLE	Walk_01	SWING_HAND
Walk_01 (c)	WALK	Walk_02	SWING_HAND
Walk_02 (d)	WALK	Walk_03	WAVE_HAND
Walk_03 (g)	WALK, WAVE_HAND	Music_01	Music, IDLE
Music_01 (e)	Music, IDLE	Talk_01	TALK, IDLE
Talk_01 (f)	TALK, IDLE	Talk_02	TALK
Talk_02 (h)	TALK		

圖 6.12 測試用的動作之標籤設定，每個動作名稱後的英文代號皆對應至圖 6.8 的動作

編號	Motion Script
1	IDLE
2	SWING_HAND
3	WALK with Upper-Body TALK
4	TALK and IDLE
5	WALK and TALK

圖 6.13 測試所使用到的 Motion Script

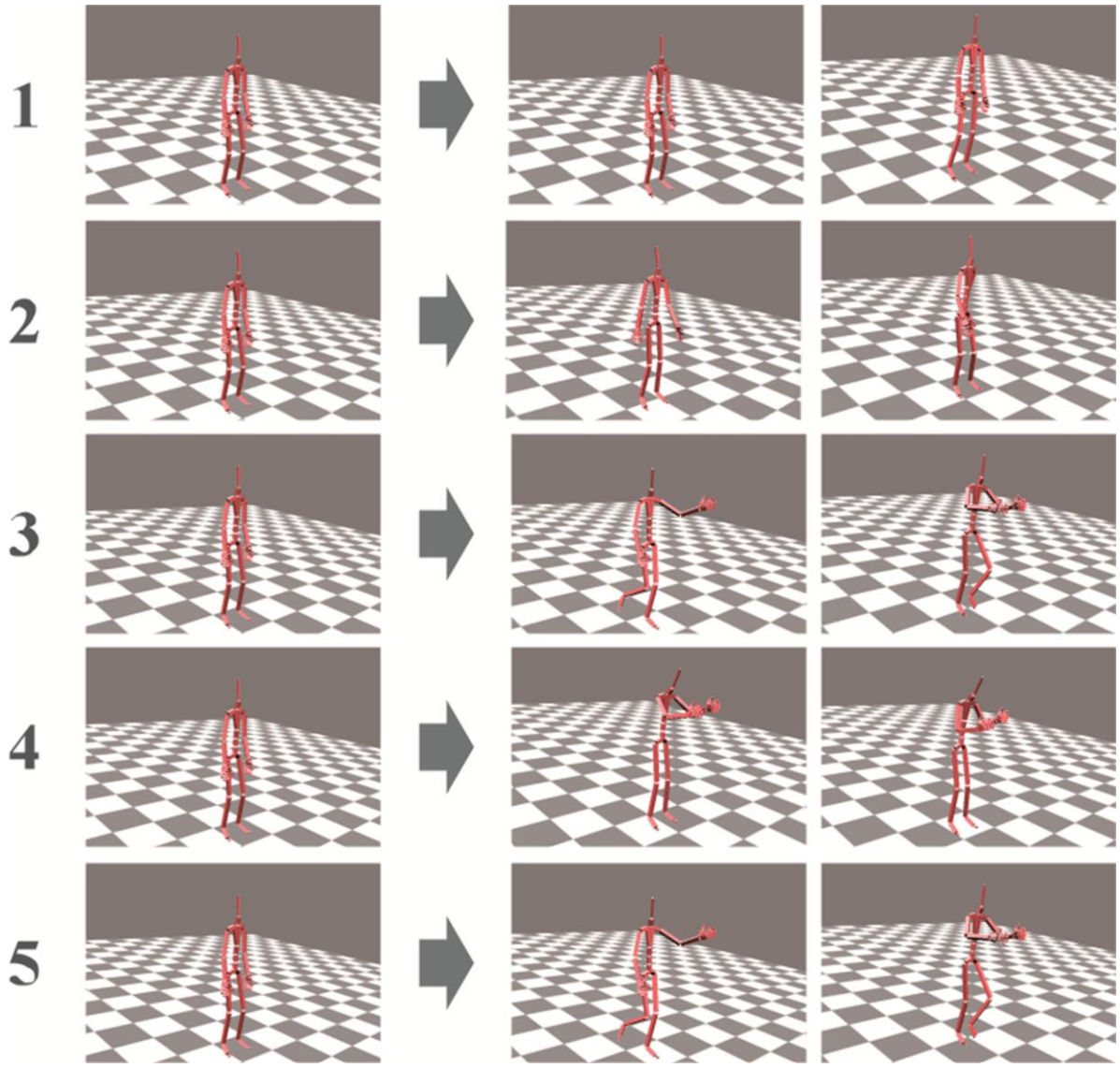


圖 6.14、Motion Script 測試結果截圖，數字對應到圖 6.11 中的編號

6.4.1 Motion Script 測試

為了展現Motion Script在實際使用上的效果，我們設計了一個測試。在這個測試中，我們使用了總共八個不同的動作（其動作的設定如圖 6.12 所示）所組成的兩張分別為全身以及上半身動作的動作圖，並由這兩個動作圖組成一個兩層式的多層式動作圖。所有測試皆以 Idle_01 做為起始動作，而根據在此所使用的多層式動作圖的特性，我們挑選了 5 個能夠展現我們搜尋能力的 Motion Script 來做測試，如圖 6.13 所示。

圖 6.14 所展示的截圖結果中，編號 1 及編號 2 為一對照的組合，編號 1 所使用的 Script 為「IDLE」，因在全身的圖上便有所需要的結果（Idle_01, Idle_02），在區域搜尋後選擇了一樣的 Idle_01 動作做為轉換目標。而編號 2 的 Scritp 是「SWING_HAND」，在這個例子中因為系統無法在根圖中找到所需要的結果，因此將「SWING_HAND」這個指令派送到子動作圖（這裡指上半身動作圖）做搜尋，剛好在上半身的動作圖中有兩個具有該標籤的動作，最後經由比較整體動作相似度數值選則了 Walk_01 做為上半身的目標轉換動作。

而編號 3 為「WALK with Upper-Body TALK」，在這個例子中首先執行了「WALK」，在區域搜尋後選擇了最快且轉換最平順的 Walk_01 做為目標來做轉換，「WALK」轉換完成後便開始執行「Upper-Body TALK」的搜尋，在上半身的圖中有兩個具有該標籤的動作，最後經由比較整體動作相似度數值選則了 Talk_01 做為上半身的目標轉換動作。

最後編號 4 及編號 5 為一對照的組合，編號 4 所使用的 Script 為「TALK and IDLE」，而在全身動作圖上的 Talk_01 便是需要的結果，因此不需要更進一步的搜尋。而編號 5 的 Script 為「WALK and TALK」，因在各層搜尋後發現並沒有動作同時具有這兩個標籤，

因此將指令拆分為「WALK」及「TALK」在不同層上搜尋，最後找到了「WALK」在全身動作圖上（Full-Body Graph），而「TALK」在上半身圖上可以產生符合需求的解。這個測試驗證了我們在上一章中所提出的 Motion Script 在實際實用上的能力。

6.4.2 連續搜尋測試

在單獨的 Motion Script 測試後，我們設計了另一個測試來觀察 Motion Script 在連續使用上的效果。在這個測試中，沿用上一個例子中所使用了總共八個不同的動作，所組成的兩張分別為全身以及上半身動作的動作圖，並由這兩個動作圖組成一個兩層式的多層式動作圖。測試用的 Motion Script 如圖 6.15 所示，在這個測試中，我們輸入兩段除了開頭的動作外，之後指令完全一樣的 Motion Script，來同時觀察其產生的結果。

Script 1	Script2
Idle_01 WALK for 10 sec WALK with Sub TALK for 10 sec IDLE	Idle_02 WALK for 10 sec WALK with Sub TALK for 10 sec IDLE

圖 6.15、連續搜尋測試用的 Motion Script

使用這兩段 Motion Script 所產生的動作如圖 6.16 的動畫截圖所示。圖 6.16 上排的動畫截圖為 Script 1 所產生出來的結果，動作起始於一個手自然下擺的 IDLE 動作（Idle_01），接下來的 WALK 動作，在圖上搜尋時選擇了一個手自然擺動的 WALK 動作（Walk_01）。接下來，當想要將上半身轉換成 TALK 的動作時，根據事前計算的整體動作相似度值，選擇了手些微舉起的且動作較小的 TALK 動作（Talk_01），最後回到 IDLE

的動作時則選擇了與一開始相同的 IDLE 動作。而圖 6.16 下排的動畫截圖為 Script 2 所產生出來的結果，相對於 Script 1 所產生的結果，Script 2 由於起始在一個手抱頭的 IDLE 動作（Idle_02），因此接著所選擇的 WALK 以及上半身的 TALK 動作皆為手動作位置較高的動作（Walk_02、Talk_02）。

在這個測試中，我們使用標籤的方式來指定一類的動作，在只使用全身動作圖轉換動作時，根據當下的狀況使用路徑搜尋演算法來選擇動作，而在同時使用多張動作圖時，使用整體動作相似度所計算出的數值來選擇動作，可以讓我們產生轉換最小且符合需求限制的連續動作。這個測試中運用了本研究中所提到的所有要素，為本研究的整個流程做出了一個完整驗證。

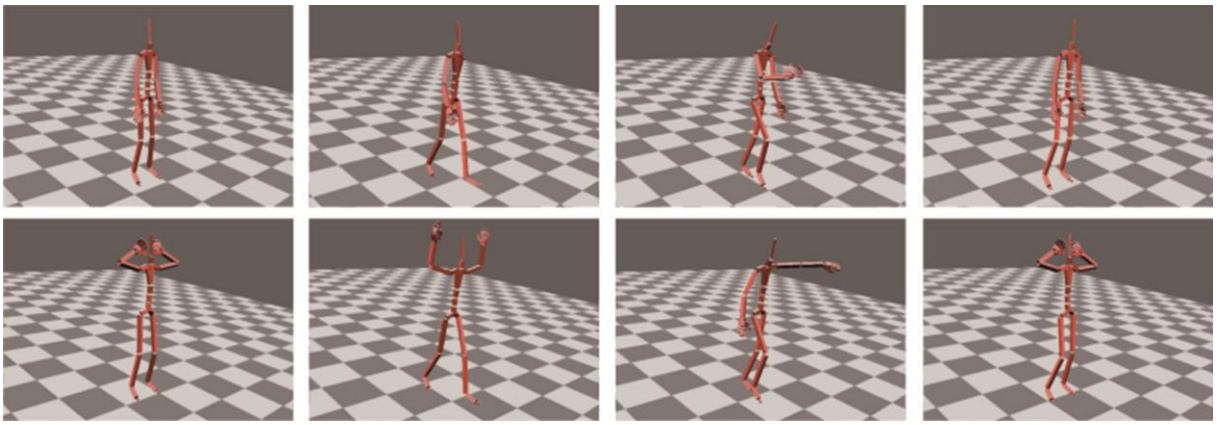


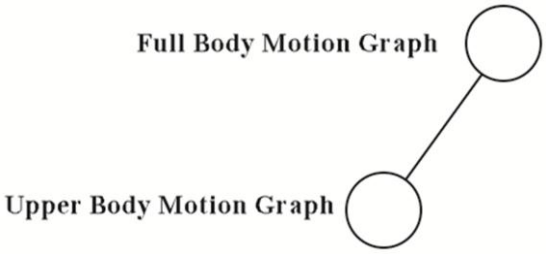
圖 6.16、Motion Script 連續搜尋測試結果截圖

第七章

結論與未來發展

在本論文中，我們嘗試設計了一個可以同時使用多張動作圖的階層式資料結構，同時提出了兩個在不同階層間動作圖運作的規則。依照這個規則，我們可以使用多層式動作圖來產生數個連續的複合式動作。此外我們更加入了動作標籤、Motion Script 以及動作相似度的計算來做為在動作選擇上的輔助，只要在起始的動作資料庫中，各類動作皆有足夠多的動作數量且都具有一定的差異性（在節奏或是在風格上等），就可以透過以量取質的概念，搭配事前計算的整體動作相似度的數值，在不同層之間找到相互最匹配且符合需求的動作組合；而在單一層動作圖的動作轉換上使用了 BFS 演算法，讓我們在不同類別動作轉換時，可以找到最快速的轉換。最後，我們透過實驗證實了整體動作相似度的正確性。而由於多層式動作圖轉換動作所需的時間與層數相關，為了找到可用的層數範圍，我們分別對一、二、三層的多層式動作圖進行了測量。我們發現當使用到第三層後，轉換所需的時間已使人有等待的感覺了，因此使用上建議以二層為主。此外，在實際的使用測試時，我們可以順利的使用我們所提出的 Motion Script 來控制多層式動作圖，產生出同時使用多張動作圖所結合的動作。

雖然本研究所提出的多層式動作圖，讓我們可以使用階層式的動作圖結構產生出合理且平順的動作，但這個階層式的結構也限制了多層式動作圖產生動作的能力。像在目前的結構下，以一張由全身以上半身兩動作圖來組成一個二層的多層式動作圖為例（組



Full-Body Graph Motions		Upper-Body Graph Motions	
動作名稱	動作標籤	動作名稱	動作標籤
Idle_01	IDLE	Idle_01	IDLE
Idle_02	IDLE	Walk_01	SWING_HAND
Walk_01	WALK	Walk_02	SWING_HAND
Walk_02	WALK	Sit_01	
Sit_01	SIT	Sit_02	
Sit_02	SIT	Talk_01	TALK
		Talk_02	TALK

圖 7.1、說明本研究限制用的多層式動作圖範例

成與設定如圖 5.1 所示），我們無法保留上半身的動作並且只轉換全身的動作，像是從 WALK with Upper-Body TALK 轉換到 SIT with Upper-Body TALK，依照目前的規則，整個過程會先將上半身由 Talk 轉換回 Walk，全身動作再由 Walk 轉換到 Sit ，最後再將 Sit 的上半身轉換為 Talk。雖然產生了較為複雜的轉換方式，但我們的規則仍是合理的，原因是原先與 Walk 所匹配的上半身動作，不一定適合後面選擇到的 Sit 動作。但在這方面仍有許多可以討論的空間，希望在未來能夠找到更好的規則組合，在降低多層式使用上的限制時，同時保有產生動作的合理性。

而在產生合理動作的方法上，目前是使用動作標籤來一次選擇一群動作後，再根據

整體動作相似度數值來從中選擇一個最好的。而這個方法最明顯的限制之一便是，當資料庫中不具有適合的動作時，那便有可能產生出視覺上不合理的動作組合。對於這點，在未來我們希望可以定義出兩動作間適不適合相互替換的判斷方式，在事前計算時就將會產生視覺上缺陷的組合給去除掉。而另一個較為間接的限制則是，每個動作的動作標籤的制定目前是以人工的方式在進行，如果標籤下的不嚴謹的話，在使用標籤來指令動作時便有可能會產生出不符合期待的動作結果。在這個問題上，我們希望參考動作分類（Motion Classification）方面的研究，以自動化的方法來輔助動作標籤的制定。

參考文獻

[1] Boulic, R., N. Magnenat-Thalmann, and D. Thalmann, *A global human walking model with real-time kinematic personification.* Vis. Comput., 1990. **6**(6): p. 344-358.

[2] Bruderlin, A. and T. Calvert, *Knowledge-driven, interactive animation of human running,* in *Proceedings of the conference on Graphics interface '96*1996, Canadian Information Processing Society: Toronto, Ontario, Canada. p. 213-221.

[3] Bruderlin, A. and T.W. Calvert, *Goal-directed, dynamic animation of human walking.* SIGGRAPH Comput. Graph., 1989. **23**(3): p. 233-242.

[4] Bruderlin, A. and L. Williams, *Motion signal processing,* in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*1995, ACM. p. 97-104.

[5] Chen, K.-Y., *Multi-graph Motion Synthesis,* Master Thesis, 2005.

[6] Heck, R. and M. Gleicher, *Parametric motion graphs,* in *Proceedings of the 2007 symposium on Interactive 3D graphics and games*2007, ACM: Seattle, Washington. p. 129-136.

[7] Hecker, C., B. Raabe, R.W. Enslow, J. DeWeese, J. Maynard, and K.v. Prooijen, *Real-time motion retargeting to highly varied user-created morphologies.* ACM Trans. Graph., 2008. **27**(3): p. 1-11.

[8] Hsu, E., M.d. Silva, and J. Popovic, *Guided time warping for motion editing,* in *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer*

*animation*2007, Eurographics Association: San Diego, California. p. 45-52.

[9] Jang, W.S., W.K. Lee, I.K. Lee, and J. Lee, *Enriching a motion database by analogous combination of partial human motions.* The Visual Computer, 2008. **24**(4): p. 271-280.

[10] Kovar, L., M. Gleicher, and F. Pighin, *Motion graphs.* ACM Trans. Graph., 2002. **21**(3): p. 473-482.

[11] Lee, J., J. Chai, P.S.A. Reitsma, J.K. Hodgins, and N.S. Pollard, *Interactive control of avatars animated with human motion data.* ACM Trans. Graph., 2002. **21**(3): p. 491-500.

[12] Perlin, K., *An image synthesizer.* SIGGRAPH Comput. Graph., 1985. **19**(3): p. 287-296.

[13] Perlin, K., *Real Time Responsive Animation with Personality.* IEEE Transactions on Visualization and Computer Graphics, 1995. **1**(1): p. 5-15.

[14] Rahim, R.A., N.M. Suaib, and A. Bade, *Motion Graph for Character Animation: Design Considerations*, in *Proceedings of the 2009 International Conference on Computer Technology and Development - Volume 02*2009, IEEE Computer Society. p. 435-439.

[15] Rose, C., B. Guenter, B. Bodenheimer, and M.F. Cohen, *Efficient generation of motion transitions using spacetime constraints*, in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*1996, ACM. p. 147-154.

[16] Safonova, A. and J.K. Hodgins, *Construction and optimal search of interpolated motion graphs.* ACM Trans. Graph., 2007. **26**(3): p. 106.

[17] Tarjan, R., *Depth-first search and linear graph algorithms.* SIAM J. Comput., 1972. **1**:

p. 146-160.

[18] Thorne, M., D. Burke, and M.v.d. Panne, *Motion doodles: an interface for sketching character motion.* ACM Trans. Graph., 2004. **23**(3): p. 424-431.

[19] Tomovic, R. and R. McGhee, *A finite state approach to the synthesis of bioengineering control systems.* IEEE Transactions on Human Factors in Electronics, 1966: p. 65-69.

[20] Unuma, M., K. Anjyo, and R. Takeuchi, *Fourier principles for emotion-based human figure animation*, in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*1995, ACM. p. 91-96.

[21] Witkin, A. and Z. Popovic, *Motion warping*, in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*1995, ACM. p. 105-108.

[22] Zeltzer, D., *Motor Control Techniques for Figure Animation.* IEEE Comput. Graph. Appl., 1982. **2**(9): p. 53-59.

[23] Zhao, L. and A. Safonova, *Achieving good connectivity in motion graphs.* Graph. Models, 2009. **71**(4): p. 139-152.

[24] Intel Threading Building Blocks for Open Source: http://threadingbuildingblocks.org

[25] OGRE: http://www.ogre3d.org

[26] Qt: http://qt.nokia.com/products