分層場景中人物動畫之全域運動計畫

Planning Humanoid Gross Motions on a Layered Scene

Pei-Zhi Huang (黃培智) Tsai-Yen Li (李蔡彥)
Computer Science Department
National Chengchi University,
Taipei, Taiwan, R.O.C.
s8713@cs.nccu.edu.tw，li@nccu.edu.tw

摘要

以往有關 3D 場景中人物的路徑運動計劃，通常都只侷限於如平地這類單純的地形，對於如建築物這類具有較多限制條件的場景，尚未有較有效的處理方法。本系統主要是將現有運動計劃加以延伸，除了能處理更一般化的場景外，對於場景中障礙物的定義，會隨著虛擬人物的高度而有所不同，根據此定義在場景中所搜尋出來的路徑，更能符合虛擬人物的特性。我們透過前處理的方式，根據離地的高度(Offset)的不同，將 3D 場景轉為數個 2D 的 Layer，再利用一個全域路徑計畫器(Global Path Planner)，根據使用者的設定和虛擬人物的特性，在 Layer 與 Layer 之間找出一條符合條件的路徑。再利用一個動作產生器(Locomotion Generator)根據該路徑在 3D 場景中產生相對應的人物動作。實驗的結果顯示此路徑計畫器相當有效率，而且可以應用到虛擬平台這類 Real-Time 的系統上，提供了一個簡易的使用者操作介面。

1. 簡介

隨著電腦硬體、軟體以及網路等各項技術的進步，使得在 3D 場景中和虛擬人物即時互動的系統更加可行。目前有許多虛擬平台大都是靠滑鼠和鍵盤來操作，這類型的系統大多需要提供簡易的人機操作介面，使得虛擬人物能夠接受使用者的命令，並產生相對應的動作。近年來對於動作的產生、控制方面已有相當多的相關研究，但基本上全域路徑都當做是輸入的參數，並不是即時算出來的。所以在作動作模擬時只需考慮虛擬人物本身的動作，不用處理虛擬場景中地形高低起伏或是避開障礙物等問題。近年來，J. Kuffner[6]和 Z.Shiller[12]提出先用一個路徑計畫器在 3D 場景中先找出一條可行的路徑，再交給動作產生器來作進一步動作的模擬。但是這類研究對於場景的定義往往只侷限於平地和障礙物之間，對於樓梯這類具有高低起伏的地形，通常都不能處理。先前的研究[8]，對於處理高低起伏的地形，已有相當不錯的成果，但是對於如建築物內這類限制條件更多的場景，並未能有效的處理。本系統主要的目的便是將目前現有的人物路徑運動計劃加以延伸，使其能處理更一般化的場景。

在操作介面上，使用者只需指定目的地，在作運動計畫時會將虛擬人物的幾何特性考慮進去。使得所找出來的路徑，會隨著使用者和虛擬人物設定不同而有所改變。此路徑將交由動作產生器作進一步的處理，其中亦包含腳步的配置以及行走動作的模擬。但本文重點將放在全域運動計劃的計算上。在應用上，這類系統對於虛擬平台或是 3D 遊戲引擎都是相當方便的一項改進。

本論文接下來的架構如下：第二部分會對運動計劃及其他相關的研究部分作介紹。第三部分為問題描述，在此部份會對我們所要處理的問題作明確的定義。在第四部分會將全域運動計畫作進一步的說明，首先會介紹可通過區域的定義，再來是碰撞偵測，最後是路徑搜尋和後處理。第五部份為實驗結果、最後為結論與未來延伸。

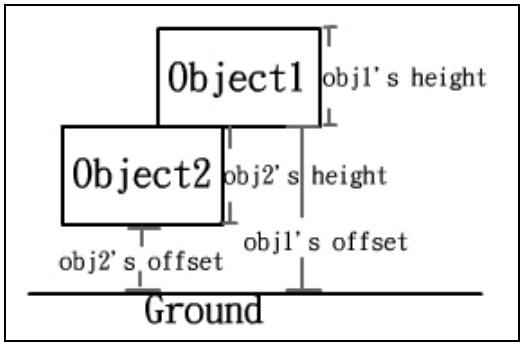
2. 相關研究

本系統所用的全域路徑計畫器(Global Path Planner)，是由機器人學(Robotics)中的運動計劃(Motion Planning)演算法所延伸而來；而此領域已有相當多而深入的研究成果[7][5][1]。運動計畫的目的是在某個有障礙物的環境中，透過碰撞偵測以及路徑搜尋，計畫可移動物體的移動路徑。根據[3]的定義，大多數運動計畫可分為兩個階段，首先是前處理階段(Preprocessing Phase)，將原本場景的幾何結構轉為抽象資料型態(如 Graph)來表示；在查詢階段(Query Phase)時便可從此抽象資料

型態找出路徑。在許多種運動計畫的方法中，位能場(Potential Field)是最常被使用的方法，利用在場景中建構虛擬位能場，來作為路徑搜尋時的依據[2]。在[6]中有提到，分成兩個部分來作人體的運動計劃，首先由路徑計畫器根據使用者所輸入的目的地，在場景中產生一條可行的路徑，再將該路徑利用動作產生器來產生相對應的動作，其中包括動作的模擬、如何使虛擬人物有效的跟隨著算出來的路徑等問題[4][9][13]。在[11]中則提到對於場景的定義的延伸，不再只是平地和障礙物兩種，增加了危險區域(Danger Zone)，所找出來的路徑要盡量避開危險區域。先前的研究[8]，已將場景的定義，根據虛擬人物腳的長度，延伸為三種，分別是平地、障礙物以及不穩定的區域(樓梯或平台邊緣)。在做路徑搜尋時，會考慮到使用者的偏好(水平距離較短或是較平坦的路徑)，找出來的適用路徑。

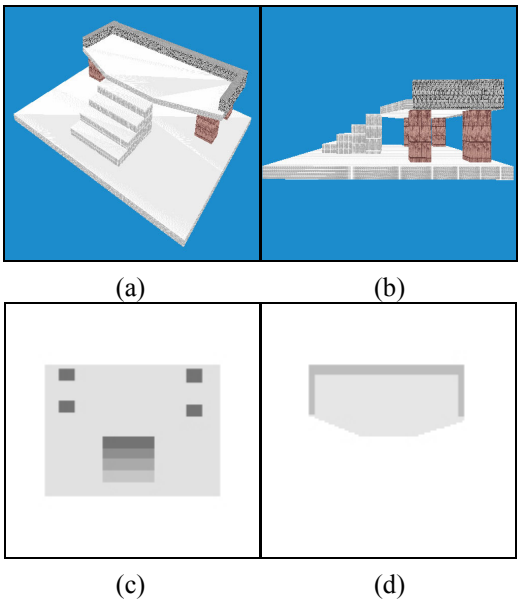
3. 問題描述

在本系統中，我們假設場景和虛擬人物的資訊(肩寬、身高)為已知，整個場景是由許多大小不等的多邊形所組成，每個多邊形有各自的高度Height和離地的高度(Offset)，如圖一所示。



圖一、物體的 Offset 與 Height 示意圖

對於場景中障礙物的定義，會因虛擬人物身高而有所不同。當虛擬人物踏不上去時，此物體即視為障礙物。在 3D 空間做路徑搜尋涉及到的自由度較高，而運動計畫問題的計算複雜度與自由度成等比級數成長[10]；因此為了能有效降低複雜度，我們將 3D 場景投影到 2D 平面再來作搜尋。這是簡化問題常見的做法，但是對於一般的場景而言，單一的 2D 平面未必能涵蓋整個場景的資訊。因為在投影時只能記錄該點最高的高度，當場景是由數個樓層所構成時，此做法便無法將原有場景資訊保留下來。於是本系統基於這點，提



圖二、(a)為 3D 場景俯視圖，(b)為側視圖，(c)(d)為依 Offset 值作分割後的 Height map，其中(c)為 Layer1，(d)為 Layer2。

出以離地高度作為該將 3D 場景分為幾個 2D 平面(Grid Layer)的衡量標準。具有同樣 Offset 值的多邊形便屬於同一個 Layer。以此為基準來做投影，所得到的結果，如圖二所示。根據 Offset 值的不同，可將原有場景切成兩個 Grid Layer，利用灰階來表示高度，其中 cell 顏色越深，代表相對於該 Layer 的高度越高。

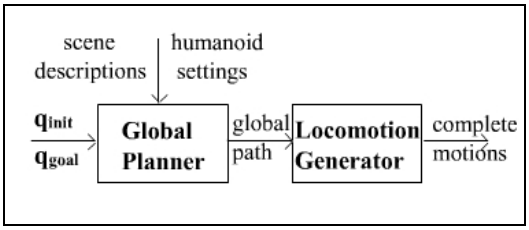
至於虛擬人物部分，因為人在走路時大多面向前方，根據這個特性，我們利用圓柱體來模擬。其中圓柱體的半徑(R)為虛擬人物的肩寬，圓柱體的高度(H)為虛擬人物的身高。由於圓形在幾何上不具有方向性，所以在作計算時便可暫時忽略旋轉這個自由度。當路徑計畫器找出路徑之後，再利用後處理的方式求出旋轉的角度。

人在作爬樓梯等動作時，並不會在樓梯邊緣停留太久，因為此時為不穩定的狀態。基於這個觀察，我們假設虛擬人物在平台邊緣逗留的時間不能超過 M 個 cell。根據這個假設所找出來的路徑比較合乎一般人的走法，因為當你走近樓梯或平台邊緣時，通常都是要作爬上或爬下的動作，並不是要貼著邊緣走。所以限制在 M 之內作完爬上或爬下的動作，除了對爬樓梯的路徑有較佳的模擬外，亦可以避免虛擬人物沿著平台的邊緣走。

4. 運動計畫

本系統透過一個簡易的操作介面，使用者只

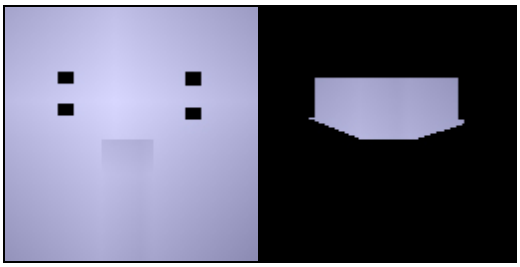
需輸入起點(q_{init})、終點(q_{goal})，結合場景及虛擬人物的資訊，全域路徑計畫器便能自動的計算出最適當的路徑，再將此路徑交由動作產生器(Locomotion Generator)產生相對應的動作。系統架構如圖三所示。以下便對運動計畫的每一步驟作進一步的說明。



圖三、系統架構圖

4.1. 可通過的區域

由於將 3D 場景根據 Offset 值不同，分為數個 Layer，使得原本可通過的地形(如樓層與樓層之間的樓梯)，可能會被拆成數個獨立的 Layer。為了維持場景原有的可通過性(Reachabilty)，在建構 Layer 時，我們做了兩方面的考量。首先利用相鄰兩個組態間的高度差值與虛擬人物可爬高度(Step-on Height)，來作為可到達與否的標準。若小於可爬高度，表示虛擬人物可由這個組態到達下一個組態，反之則否；其次是利用 upper Layer 的 Offset 值和目前高度的差值，當這個值大於虛擬人物的身高時，表示在垂直的方向上有足夠的高度能讓虛擬人物通過。當滿足上述兩個條件，則我們稱該組態為可通過(Reachable)。再利用 NF1[2]的演算法在可通過的區域建立虛擬位能場(如圖四)，作為路徑搜尋時的依據。

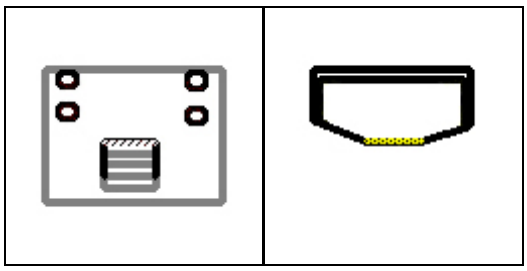


(a) (b)

圖四、根據圖二的場景所建 Reachability map，(a)為 Layer1，(b)為 Layer2。圖中黑色的部分，表示在該 Layer 中不可到達的區域，淺色的區域為可到達，並且每個 cell 都有一個虛擬位能場的位能值。

4.2. 碰撞偵測

當虛擬人物與障礙物有所接觸時，此時我們稱為有碰撞產生。在路徑搜尋時要避開此種狀況。另外，由於我們要限制爬樓梯的動作在 M 之內作完，所以必須將樓梯的邊緣找出來，以便在路徑搜尋時，能夠針對這些區域加以限制。因此本系統對碰撞偵測分為五種情況，如圖五所示。首先是 Stable，表示我們的虛擬人物正站在平地上，處於穩定的狀態。其次是 Forbidden，在此情況下，表示我們所用來模擬虛擬人物的圓柱體和障礙物有所接觸，當該點在半徑 R 的範圍內，有高度差大於腳長時，即視為 Forbidden。第三為 Unstable，這種通常是發生在屬於同一 Layer 的樓梯或是平台邊緣，亦是在路徑搜尋時要加以限制的部分。當該點在同一 Layer 半徑 R 的範圍內，



(a) (b)

圖五、根據圖二的場景所建 Collision map。(a)為 Layer1，(b)為 Layer2。圖中白色部分為 Stable，黑色為 Forbidden，灰色為 Unstable，條紋為 Up-stair，斑點為 Down-stair。

有高度差產生並且小於腳長時，則視為 Unstable。

第四為 Up-stair，發生於 Layer 與 Layer 相接處。當該點在同一 Layer 為 Forbidden，且與上層 Layer 半徑 R 的範圍內，有高度差產生並且小於腳長時，稱為 Up-stair。這時表示虛擬人物能向上層 Layer 前進，不一定要留在同一層。最後為 Down-stair，表示虛擬人物只能向下層 Layer 前進。做法則類似 Up-stair，只是比較時要和下層 Layer 作比較。其中 Forbidden 為和障礙物有所接觸，在路徑搜尋時要避開這種區域。Unstable、Up-stair 和 Down-stair 這三個區域都為樓梯或平台的邊緣，在路徑搜尋時經過這些區域，都要限制在 M 的時間內完成，以便於對爬樓梯的路徑有較合常理的模擬。

4.3. 路徑搜尋

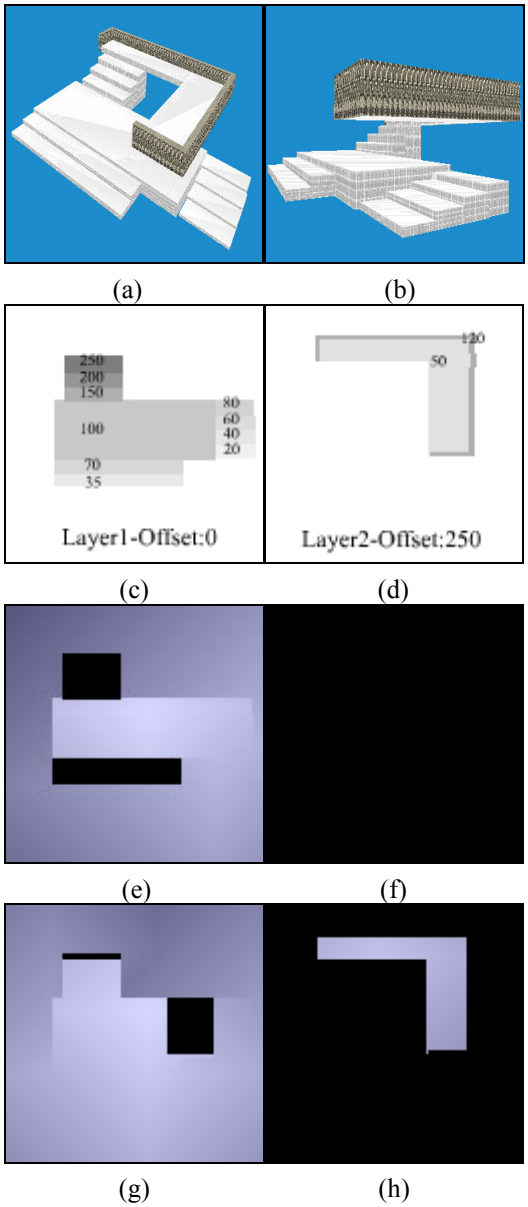
```
STABLE_BFP()
1   install q_i in T;
2   INSERT(q_i, OPEN); mark q_i visited;
3   SUCCESS ← false;
4   while ┐EMPTY(OPEN) and ┐SUCCESS do
5     q←FIRST(OPEN);
6     for every neighbor q' of q in the grid do
7       if LEGAL(q') then
8         if q' is unstable then
9           q'.cnt=q.cnt+1;
10        else
11          mark q' visited;
12        install q' in T with a pointer to q;
13        INSERT(q', OPEN);
14        if q' = q_g then SUCCESS ← true;
15  if SUCCESS then
16    return the backtracked feasible path
17  else return failure;
```

圖六: The STABLE_BFP algorithm

如圖六所示，本系統在搜尋路徑時使用的是類似 Best-First Search 的演算法，首先將 q_i(起點)放入 OPEN 中(用來儲存候選組態)，利用 FIRST 這個函數從 OPEN 中取出目前最好的組態(由虛擬位能場的值做衡量的標準)，再透過 LEGAL 這個函數檢查該組態是否符合下述三種條件條件。首先是沒有碰撞產生(不為 Forbidden)。第二是該組態尚未被拜訪過(unvisited)。最後為該組態目前處於暫時穩定狀態或穩定狀態，當一個組態是在 Unstable、Up-stair 或是 Down-stair 的區域時，而且所待的時間 q.cnt 小於 M，此時為暫時穩定狀態。當該組態通過 LEGAL 的檢驗後，將四周未拜訪過的合法組態(亦包含相鄰 Layer 間的組態)放入 OPEN 中，直到該組態與 q_g(終點)相同為止。但是對於在 Unstable、Up-stair、Down-Stair 這些區域裡的組態的處理跟其他組態有所不同。因為在 counter(q.cnt)這個值小於 M 時，該組態亦可能再次被拜訪，所以必須紀錄被拜訪的方向，直到所有的可能都出現過了，才可以將此組態設定為被拜訪過，以確保搜尋的完整性。

4.4. 後處理

在找出路徑後必須再作路徑的平滑化和恢復旋轉維度這兩個動作。首先在平滑化部分，利用直線來對局部的路徑做取代，以獲得較平滑的路徑。其次在計算路徑時我們假設虛擬人物永遠是面向前方，而將旋轉這個維度忽略。因此在計算出路徑後，我們進一步計算切線方向，找出旋轉角度。最後在將處理過後的路徑，交由動作產生器作腳步的配置及行走動作的模擬。



圖七、(a)為 3D 場景的俯視圖，(b)為側視圖。根據離地高度的不同，將場景分為(c)(d)兩個 Layer。 (e)(f)為根據身高 130 公分，可爬高度 30 公分的虛擬人物所建的 Reachability map。(g)(h)為根據身高 180 公分、可爬高度 60 公分的虛擬人物所建的 Reachability map。其中(c)(e)(g)為 Layer1，(d)(f)(h)為 Layer2。

5. 實驗結果

本系統實作上都是使用 Java 語言，實驗的平

台為 AMD TB 1.2G，512MB RAM 的一般個人電腦。所有 map 的解析度皆為 128*128。

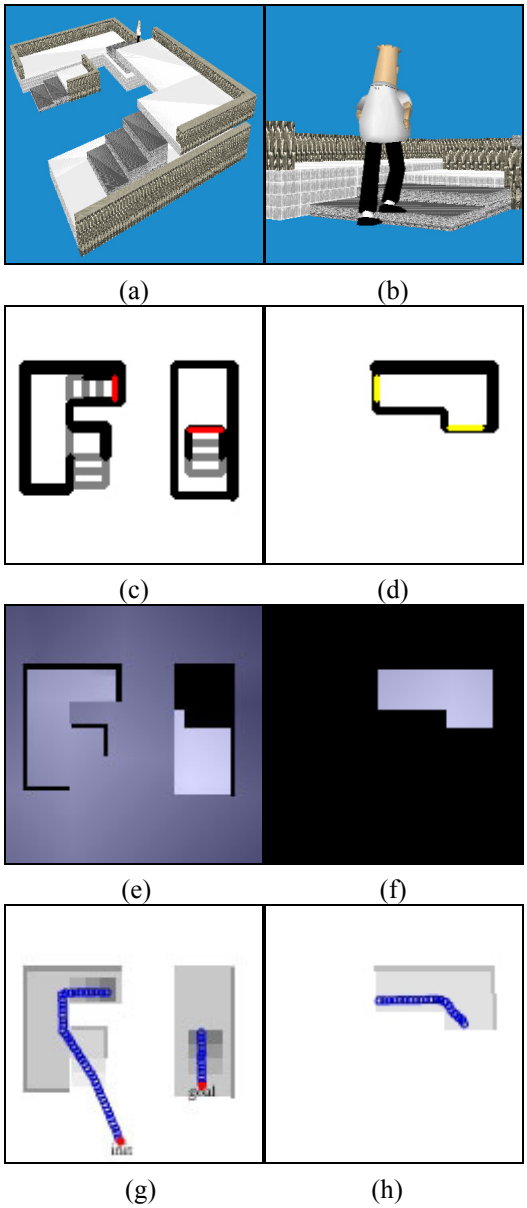
5.1. 虛擬人物高度的影響

虛擬人物的高度以及可爬高度，可以決定一個物體是否為障礙物，並且可以決定某個組態是否為可通過。如圖七所示，(e)(f)為依據身高 130 公分，可爬高度 30 公分的虛擬人物(相當於一般的小孩)，所建構出的 Reachability map。根據這兩張 map，我們可以發現，左側兩個樓梯，階層之間的高度差皆大於 30 公分，以至於虛擬人物無法爬上去，所以在(e)(f)中的黑色區域所表達的意義為無法爬上去。(g)(h)為依據身高 180 公分、可爬高度 60 公分的虛擬人物(相當於一般的成人)，所建構出的 Reachability map，(g)圖中右側黑色的區域，所表達的意義為無法通過。因為 Layer2 的離地高度為 250 公分，而該區域 Layer1 的高度為 100 公分，所以兩層之間的空間只有 150 公分高，並不能讓一個 180 公分的虛擬人物通過。依據不同的虛擬人物，而會有不同的 Reachability map，再根據此圖建立虛擬位能場，作為搜尋時的依據，使得所找出來的路徑，會隨著虛擬人物的幾何特性而有所改變。

5.2. 結合動作產生器

如圖八所示，根據使用者所指定的起點、終點，配合虛擬人物的特性，可以建構 Reachabiltiy map(e)(f)和 Collision map(c)(d)，利用這些資訊配合路徑計劃器，在場景中找出一條符合條件的路徑(g)(h)。再利用動作產生器對虛擬人物作進一步的控制(b)。

根據圖八的場景，路徑計劃器各個步驟所需花費的時間如圖九所示。將 3D 場景分為數個 2D Grid Layer 所需花費時間為 10ms。建構可通過的區域，需花費 40ms。用來作碰撞偵測的部分需 170ms，也是整個運動計劃花費時間最久的一個步驟。因為在作碰撞偵測時，我們以模擬虛擬人物的圓柱體，當作判斷的基準。因此在作檢查時，必須考慮的範圍較大，所以相對的所需時間也較久。路徑搜尋只需 20ms，由於碰撞偵測已事先處理好，所以搜尋時的速度能大幅提昇。



圖八、(a)為場景的俯視圖，(b)為動作產生器根據找出來的路徑，所做的動作模擬，(c)(d)為 Collision map，(e)(f)為 Reachabiltiy map，(g)(h)為所搜尋出來的路徑；其中(c)(e)(g)為 Layer1，(d)(f)(h)為 Layer2

Routine Name	Time
Construct Layered Height map	10ms
Construct Reachability map	40ms
Construct Collision map	170ms
Search for a Global Path	20ms

圖九、各個步驟所需的時間表

6. 結論及未來延伸

在本系統中，我們提出利用離地的高度來將 3D 場景分為數個 Grid Layer。由於 Layer 的資訊是各自獨立的，我們藉由虛擬人物的身高及可爬高度，可得到 Layer 與 Layer 間可通過的區域。再

根據這些資訊，在 Layer 中建立虛擬位能場，作為搜尋時的依據。路徑計畫器則以運動計畫演算法來產生一條適合虛擬人物的路徑。再將此路徑交由動作產生器作配置腳步及動作的模擬。對於場景中障礙物的定義會因為虛擬人物的高度而有所不同，具有更大的彈性，而且可適用於更一般化的場景。

在目前的假設下，場景中的物體都是靜態的。所以在運動計劃時，可將碰撞偵測部分事先處理好，以加快路徑搜尋時的速度。但是實際上環境中的物體位置，可能會有所變動，如行走中的人或是移動中的車子。以至於在搜尋時亦要作動態的碰撞偵測，以確保搜尋出來的路徑，在動作模擬時能夠有效的避開障礙物。這部分是本系統目前尚未能處理的部分，也是將來繼續研究的方向。

參考文獻

[1] N.M. Amato, O.B. Bayazit, L.K. Dale, C. Jones, and D. Vallejo, “OBPRM: An Obstacle-Based PRM for 3D Workspaces,” *Robotics: The Algorithmic Perspective*, pp.630-637, 1998.

[2] J. Barraquand, L. Kavraki, J.C. Latombe, “Robot Motion Planning: A Distributed Representation Approach,” *Intl J. of Robotics Research*, 10:628-649,1991.

[3] J. Barraquand, L. Kavraki, J.C. Latombe, T.Y. Li, and P. Raghavan, “A Random Sampling Scheme for Path Planning,” *Intl. J. of Robotics Research*, 16(6), pp.759-774, Dec. 1997.

[4] S.-K. Chung, and J. K. Hahn, “Animation of Human Walking in Virtual Environments,” *Proc. of Computer Animation '99*, 1999.

[5] Y.K. Hwang and N. Ahuja, “Gross Motion Planning – a Survey,” *ACM Comp. Surveys*, 24(3):219-291, 1992.

[6] J. Kuffner, “Goal-Directed Navigation for Animated Characters Using Real-time Path Planning and Control” *Proc. Of CAPTECH’98 Workshop on Modeling and Motion capture Techniques for Virtual Environments*, Springer-Verlag, 1998.

[7] J.C. Latombe, *Robot Motion Planning*, Kluwer, Boston, MA, 1991.

[8] T.Y. Li and P.Z. Huang, “Motion Planning for a Humanoid Walking in a 3D Space,” *Proc. of the 2001 National Computer Symposium*, Taipei, Taiwan, 2001.

[9] H. Miura and I. Shimoyama, “Dynamic Walk of a Biped,” *Intl. J. of Robotics Research*, pp.60-74, 1984.

[10] J.H. Reif, “Complexity of the Mover's Problem and Generalizations,” *Proc. of the 20th IEEE Symp. on Foundations of Computer Science*, pp. 421-427, 1979.

[11] D. Sent and M.H. Overmars, ”Motion Planning in Environments with Dangerzones” *Proc. Of 2001 IEEE Intl.Conf. on Robotics and Automation*, pp.1488-1493, May 2001.

[12] Z. Shiller, K. Yamane, Y. Nakamura, “Planning Motion Patterns of Human Figures Using a Multi-Layered Grid and the Dynamics Filter” *Proc. of 2001 IEEE Intl. Conf. on Robotics and Automation*, pp.1-8, May 2001.

[13] K. Yamane and Y. Nakamura, “Dynamics Filter – Concept and Implementation of On-Line Motion Generator for Human Figures,” *Proc. of IEEE Intl. Conf. on Robotics and Automation*, pp.688-695, April 2000.