分散式可視性裁切法應用於複雜顯像系統之設計

盧天麒 (Tainchi Lu) 張正和 (Chenghe Chang) 劉易凱 (Yikai Liu)

嘉義大學資訊工程所

E-mail : tclu@mail.ncyu.edu.tw

摘要

本論文是將大型複雜場景的即時顯像計算，建構在具有空間分割、空間分配、即時可視性以及可視性裁切等分散式資料切割技術上。整個顯像作業流程分為兩個階段，在前置處理時先利用二元空間分割樹 (BSP tree)的將整個場景劃分成許多空間細胞 (spatial cell)，再根據各個細胞的幾何複雜度去計算其複雜度權重值以及，並於每個空間細胞內部的幾何資料找出其物件遮擋者 (occluders)；在執行階段時，利用細胞分配機制動態賦予每台顯像資料計算機器適當權重值的處理個數，讓系統利用分散式負載平衡機制即時計算出顯像資料。因此本研究利用分散式系統資源的同步處理機制，將傳統利用單機處理的場景幾何計算工作分配至具有高速運算能力的分散式系統叢集中，並整合了以場景輸出感度 (output-sensitive)為主的可視性裁切技術，可針對大型複雜場景的瀏覽及遊走需求，提供了更快速的顯像速率及更平穩的顯像效果。

關鍵詞：可視性裁切、即時可視性、負載平衡、可適性策略、分散式計算

1.緒論

隨著電腦繪圖硬體技術的進步，描繪三維空間場景能力與過去相較之下具有大幅度的成長，但由於虛擬實境技術的應用內容逐漸趨於多元化，對於建構大型複雜場景的需求也相對的提高，並且在設計中還必須考量視覺品質、即時呈現及使用者互動等課題，一味地依賴硬體技術的支援並無法滿足這種種的需求，因此許多加快場景呈現計算速度的技術便開始發展起來，而其最主要的目的便是如何在使用者可接受的精細度、快速的顯像速率及平順的操縱性等因素下，將正確的場景動態呈現給使用者。

在許多加速場景描繪的技術中，最主要是減少描繪時的場景資料量，便是將觀測者在場景中無法看到的部分加以過濾剔除，只留下觀測者所能看見的範圍，之後再將計算完成的結果送至顯像程序進行顯像，利用這種作法將大幅減少顯像過程所必須花費的計算時間，而且不會影響所瀏覽之結果，我們稱之為可視性裁切法 (visibility culling)[5]。而除了可視性裁切法之外，多層精細度 (Level-of-Detail, LOD)演算法 [2][9]及影像式顯像技術 (Image-based Rendering) [1][10][12]也同樣可達到加速顯像速率的目的。上述方法都已成功地提昇了顯像速率，但是當場景趨於龐大且複雜時，系統要在執行期間即時將正確的場景呈現出來，使用這些傳統單一方法仍是略嫌不足，因此對於高複雜的大型場景而言，設計出一個能夠加速場景顯像速率的分散式演算法架構，便是本論文所提出來的方法。

以下是這篇論文的架構，於第二章介紹一些相關的研究方法；第三章將針對所提出的方法做詳細的介紹；第四章討論本論文的實驗結果；第五章則是本論文之結論與探討未來改進的方法。

2.相關研究

本論文的目的主要在利用分散式的系統叢集來加速計算三維場景顯像資料的速度，此章節針對相關的顯像技術及分散式計算進行介紹及說明。

2.1 可視性裁切法

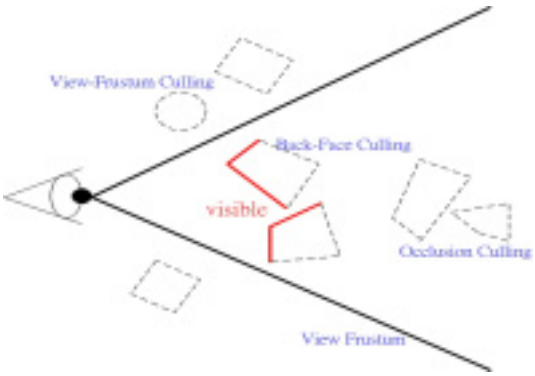
可視性裁切法 (visibility culling)是指將觀測者視角空間內無法看見的多邊形挑選出來，在顯像流程進行之前予以排除，以減少顯像時所需處理的多邊形數目，進而降低顯像流程的計算量以及處理時間。目前已有許多方法能夠達到此目的，以下就分別針對考慮單一視點和區域空間的可視性裁切法做介紹。

■ 考慮單一視點的可視性裁切法

考慮單一視點的可視性裁切法 (from-point visibility)是以觀測者在每個畫面(frame)顯像時所處的視點為考量，計算從此視點所看見的視角空間內有哪些多邊形是真正能被觀測者所見，並將觀測者無法看見的多邊形予以剔除，此類方法重視的是即時性的計

算，顯像系統必須在畫面更新行進間快速的挑選出可見的多邊形並加以顯像，其優點是不需要耗費大量的前置處理作業且篩選出的可視多邊形數量較為精確，缺點則是由於它必須在每個畫面顯像之前完成即時性的可視計算，每一個畫面都必須根據目前的視點重新計算，顯像的畫面更新頻率便被此所侷限。

可視性裁切法是從考慮單一視點裁切法所演變發展而來，傳統的方法有三種，包含可視區域空間裁切法 (view-frustum culling)、背向面裁切法 (back-face culling)及遮擋裁切法 (occlusion culling)，其示意圖如圖一所示，以下便針對此三種方法進行介紹。



圖一、 可視性裁切技術的三種類型

可視區域空間裁切法 (View-Frustum Culling)：對於應用在一般三維場景中的裁切法而言，最常應用的裁切技術就是可視區域空間裁切法 [4]，所謂的可視區域裁切指的就是將觀測者可視區域空間 (view frustum)外的多邊形予以剔除，因為這些多邊形既然位於觀測者的視角之外，理所當然便無法被觀測者所見，系統只需要對視角空間內的多邊形進行顯像計算即可，以避免對不可見的多邊形做無謂的顯像計算，避免浪費系統資源及徒增運算時間。

背向面裁切法 (Back-Face Culling)：背向面裁切法 [7]則是根據多邊形與觀測者之間的角度來做計算，一般而言多邊形只有在正面角度的部分會被觀測者所見，其背面並無法被看見，因此若此多邊形經判斷之後被確認為背向觀測者的話，系統便可將此多邊形剔除，不對它進行顯像的動作，以減少顯像流程進行時的多邊形數目。

遮擋裁切法 (Occlusion Culling)：遮擋裁切技術 [11][13]考慮的則是場景中多邊形間的位置關係，當觀測者從其所在視點往外觀測，距觀測者較遠的多邊形有可能會被距觀測者較近的多邊形所遮擋，因此遮擋裁切法便是去計算場景中各個部分彼此的相互遮蔽關係，將被遮蔽的多邊形予以剔除。總體而言，此技術由於必須考慮整個場景中多邊形間的相互關係，所以其計算量遠較可視區域空間裁切法及背向面裁切法複雜及龐大。

■ 考慮區域空間的可視性裁切法

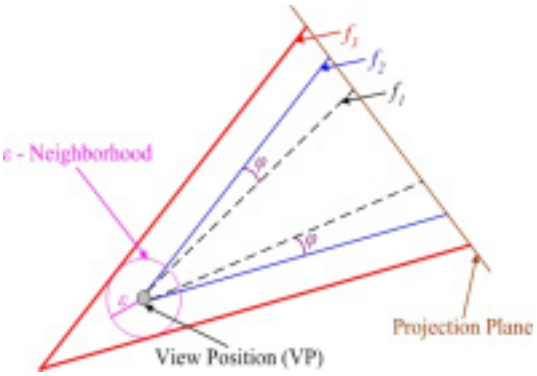
考慮區域空間的可視性裁切法 (from-region visibility)是以觀測者在一定的區域空間內所能位移的所有視點做考量，計算從此區域空間內所有視點所能看見最大可視空間內的可視多邊形，並將這些可視多邊形加入到一個稱為"潛藏可視資料集合" (potentially visibility set, PVS)的資料結構中，只要觀測者位於此區域空間內，系統便可從潛藏可視資料集合中挑選出所需的顯像資料進行顯像，而當觀測者離開此區域空間進入到另一個區域空間時，系統再以觀測者當時所處的區域空間加以重新計算顯像資料。此類方法是對觀測者在一定的位移區域空間內所能看見的最大顯像資料集合做顯像計算處理，所以它必須顯像的多邊形數目總和必定較實際上觀測者位於單一視點時為多，但其優點是每計算一次便可供數個畫面使用，無須在每個畫面顯像時都重新計算，而且在執行階段時只要經由少量的計算，便可快速得到正確的顯像結果。此類方法的缺點是必須耗費大量的前置處理時間去計算每個區域空間的最大顯像資料集合，以供執行階段時能夠直接取用，並且必須花費大量的儲存空間去儲存這些龐大的顯像資料。

2.2 即時可視性方法

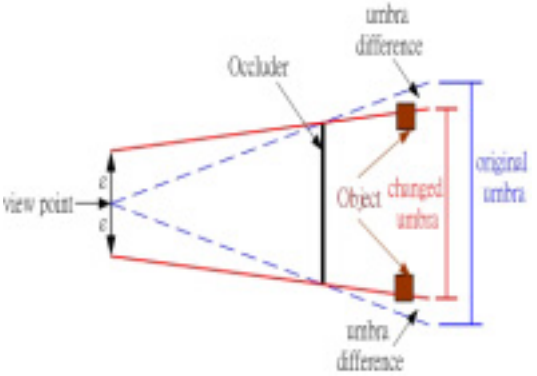
即時可視性 (instant visibility)方法[15]是以單一視點的裁切演算法為基本考量，並配合顯像資料的計算速率及觀測者的移動速度，利用兩種計算資源 (第一種資源為計算顯像資料使用，另一種資源則為顯像用)，計算出觀測者在一定時間 (假設為 $t\varepsilon$)內所可能遊走的最大面積，之後再求出在此最大遊走面積所可能觀視到的顯像資料。其示意圖如圖二所示，觀測者原本的可視區域為 f1，假設觀測者在 $t\varepsilon$ 的時間內，最大的轉動角度為 φ，而最大的移動距離為 ε，則 f2 即為轉動 φ 角度的最大可視區域，f3 即為轉動 φ 角度並移動 ε 距離的最大可視區域，因此位於 f3 範圍內的顯像資料即為觀測者在 $t\varepsilon$ 的時間內所可能觀視到的最大顯像資料集合。

計算出 $t\varepsilon$ 時間內的最大可視區域範圍之後，便可對此區域範圍內的物體做遮擋性裁切的測試，由於遮擋者相對於觀測者的遮擋面積會因觀測者所處位置的不同而有所改變，原本在遮擋範圍內的物體可能因為觀測者位置的

改變而變成可視物體，如圖三所示，位於遮擋面積差值範圍內的物體在觀測者移動 ε 的距離後已成為可視物體，若系統事先未針對此種情況做考量，仍以原本的遮擋面積做遮擋性裁切的測試，將可能造成顯像時的錯誤。

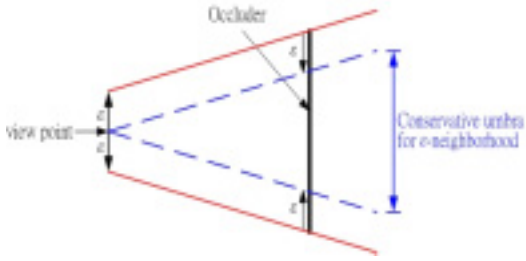


圖二、 在 t_ε 時間內的最大可視區域範圍



圖三、 遮擋面積差值示意圖

換言之，觀測者在 tε 時間內的最大移動距離為 ε，系統就必須保守性地計算在此 ε 範圍內的有效遮擋面積，並保證此遮擋面積在 tε 時間內相對於觀測者於 ε 範圍內的任一點均為有效，此種方法稱之為遮擋者收縮法 (occluder shrinking) [14]。此方法是將遮擋者本身的邊界往內收縮 ε，再針對收縮後的遮擋者執行遮擋性裁切測試，如圖四所示，此方法將遮擋者的邊界往內收縮 ε 之後，相較觀測者從觀測點所見的遮擋面積範圍 (虛線內部之範圍)和原始未收縮的遮擋面積，收縮後的遮擋面積包含於觀測者移動了 ε 距離後與原遮擋者的遮擋面積範圍 (實線內部之範圍)中，以收縮後的遮擋者進行遮擋性裁切所形成的遮蔽面積雖較原本為小，且所計算的顯像資料可能較實際為多，但卻可保證觀測者在移動了 ε 距離後，此遮蔽範圍仍為有效，可確保其正確性。



圖四、 遮擋者收縮示意圖

2.3 分散式計算技術

分散式計算主要是運用大量閒置工作站的計算資源，利用網路加以連結，將資料處理的工作配置於各工作站進行分散處理，而達到縮短處理時間的目的。其中包含許多研究議題，以下就針對分散式計算中之負載平衡進行介紹。

在分散式架構中，負載平衡 (load balancing)扮演著相當重要的角色，它必須保證系統中的每一部處理機器在任一時間點都有著幾近相同的工作負載量，在系統運作期間每部機器的工作可依據機器的負載情形相互移轉，高負載的機器可將其工作轉讓給低負載的機器執行，以求得負載平衡的效果，並減少系統中因相互等待計算結果而降低分散式計算的成效。

根據 Casavant 和 Kuhl 所提出的演算法分類架構中 [3]，靜態演算法的工作分配必須在前置處理時進行，且在執行期間無法更動其狀態；動態演算法的工作分配則可依據目前系統中各部機器的負載情形，動態調配工作量，因此相較之下靜態演算法較為缺乏彈性，且擴充性不佳，所以目前的分散式系統大多採用動態演算法為其工作分配的原則。

此外大部分移轉策略都是以門檻值 (threshold)為基礎考量點，判斷機器的負載量高低是視其負載量高於某一高負載門檻值或低於某一低負載門檻值而定，然而，若只使用單一固定門檻值將導致狀態擺盪 (state waggling)的情形發生，此情形指的是機器時常在高負載與低負載間來回跳動，無法長時間處於穩定的狀態，如此將產生許多不必要的工作移轉情形，大幅影響整個系統的效能。解決的方法為使用兩階層的門檻值 [8]或評斷兩機器間的負載量差值 [6]，以決定是否允許在兩機器之間產生工作移轉的行為。

3.分散式顯像系統架構

3.1 系統規劃

整個系統運作流程區分為前置處理 (pre-

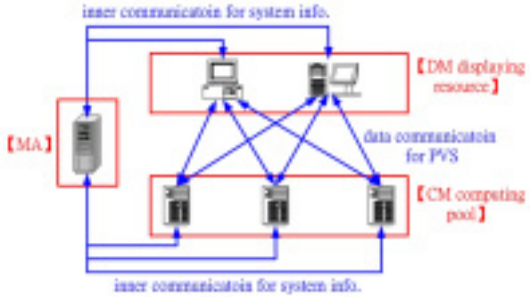
processing)與執行 (on-the-fly)兩階段，前置處理階段是為了加速執行階段的顯像速率所做的預備動作，將一些執行階段所可能用到的顯像資訊，預先在前置處理階段先行計算，在執行期間便可直接取用並加以處理，以此來縮短執行階段的計算時間。執行階段則計算觀測者目前視點所見的顯像資料 (potentially visibility set, or PVS)，並將最後的計算結果匯集至顯像流程，以便將場景呈現給觀測者瀏覽。本系統架構由以下三種不同類型的機器所構成：

(一)系統管理代理者 (Management Agent，簡稱 MA)：MA 為整個系統的管理者，負責記錄系統中各部機器的資訊，分配場景細胞給顯像資料計算機器叢集，並擔任新增或移除顯像資料計算機器及顯像機器時的訊息溝通橋樑，可讓整個系統經由此系統管理代理者得以串聯成一個完整的計算網絡。

（二）顯像資料計算機器 (Calculating Machine，簡稱 CM)：CM 為系統中負責計算顯像資料的機器叢集，此叢集中的機器根據系統管理代理者所分配的細胞，幫助顯像機器計算其顯像資料，讓顯像機器的計算負載可經由此叢集中各部機器的分擔計算而縮減，以加速顯像流程的進行。

（三）顯像機器 (Drawing Machine，簡稱 DM)：DM 為系統中負責顯像的機器叢集，此叢集中的機器依據觀測者目前所處的視點位置，利用顯像資料計算機器叢集輔助其計算顯像資料，而其本身只需負責場景的顯像即可，將可大幅縮減顯像前顯像資料的計算動作。

圖五為系統架構以及各叢集間的訊息傳遞示意圖，其中 MA 為整個系統的管理者，負責記錄整個系統的資訊，並控管整個系統的運作，DM 及 CM 都必須從 MA 獲取系統資訊，而 DM 及 CM 之間則相互傳遞著顯像資訊，DM 將目前觀測者的視點資訊傳送給 CM，CM 則將計算後的顯像資料回傳給 DM，供 DM 將最後的場景結果呈現給使用者。



圖五、 系統架構圖

一個複雜場景由於其幾何複雜度高，若執行期間也對整個場景的多邊形資料做計算，當場景愈趨於龐大，勢必無法即時進行平穩的場景瀏覽，所以在前置處理階段，我們必須對場景做預先處理，以利執行階段顯像的進行，而前置處理階段主要有以下四個執行步驟：

步驟一、以 BSP tree 切割場景 (spatial partition)：首先，以二元空間分割樹 (Binary Space Partitioning tree)演算法對場景做空間分割，將整個場景切割成一塊塊的細胞 (cells)，如此我們便可將所要考慮的場景空間局部化，只針對某些會影響最後顯像結果的細胞做計算即可。

步驟二、建置細胞鄰接樹 (adjacency tree construction)：細胞與細胞之間依據其場景位置，彼此間存在著相鄰關係，將彼此相鄰的細胞連接起來建置成樹狀結構，便稱此樹狀結構為細胞鄰接樹 (adjacency tree)。建置細胞鄰接樹主要是為了執行期間分配細胞時使用，細胞分配機制可從細胞鄰接樹獲知各個細胞間的相鄰關係，以做為分配細胞時之重要考量因素。

步驟三、計算細胞場景複雜度權重 (weighted value calculation)：當場景切割完成之後，便針對每個細胞計算其場景複雜度權重 (complexity weight)，此步驟之用意在於執行期間要將細胞分配給分散式系統叢集時，可依據場景複雜度權重的高低，平衡分散式系統叢集中每部機器的負載量，達到動態負載平衡 (dynamic load balancing)之目的。

步驟四、找出每個細胞的遮擋者 (occlusion culling)：對每個細胞而言，皆可找到相對於此細胞之遮蔽性較好的遮擋者 (occluder)，透過此遮擋者，便可剔除位於遮擋者之遮蔽範圍中的多邊形，以減少顯像時的顯像資料量。

在執行階段，系統必須確保顯像機器叢集的每部顯像機器可以平順地進行即時顯像，且必須在動態加入或移除顯像資料計算機器或顯像機器後，系統仍可順利運作，不致因為機器叢集的變動而影響整個系統的運行，因此其運作行為模式可歸類成以下五種：

第一種、新增顯像資料計算機器 (add a new CM)：動態加入一部顯像資料計算機器，以增加整個系統的運算資源，隨著顯像資料計算機器數目的增加，將有更多的計算資源可供系統使用，計算顯像資料的時間也可以有條件的加以縮減。

第二種、新增顯像機器 (add a new DM)：動態加入一部顯像機器，每部顯像機器可提供一位觀測者進行獨立場景瀏覽，因此每部顯像機器各有各自的觀測點及觀測範圍，所顯像的場景也將隨著使用者觀測的位置及角度而變

化。

第三種、計算顯像資料並顯像 (calculating and rendering)：整個系統中最主要的動作便是顯像資料的計算及顯像，顯像機器將目前的觀測者視點位置告知顯像資料計算機器叢集，叢集中的機器便根據其所負責的細胞計算顯像資料，再將結果傳回至顯像機器，當顯像機器匯集完成所有顯像資料計算機器的顯像資料後，便可執行顯像流程，將最後的結果呈現給使用者進行觀測及遊走。

第四種、移除顯像資料計算機器 (remove an existing or crashed CM)：當某部顯像資料計算機器從系統中移除時，原本由其負責的細胞將暫時變成無任何機器負責的狀態 (orphan)，因此系統必須在顯像資料計算機器移除時提供一個因應機制，使得顯像流程的進行得以正確及平穩的進行，換句話說，系統在顯像計算機器移除時，必須進行容錯處理 (fault tolerance)的動作。

第五種、移除顯像機器 (remove an existing DM)：顯像機器移除時，對系統並無太大的影響，系統只需將與此顯像機器有關的動作停止，以及將此顯像機器的資訊從系統中移除即可，換句話說，此動作便如同系統所服務的用戶端已中斷與系統的連線，並非系統本身的服務提供者發生計算及顯像流程異常現象。

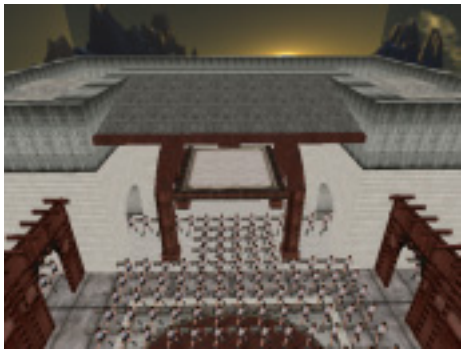
3.2 細胞分配機制

顯像資料計算速率的快慢端視細胞分配機制的優劣與否，因此本論文最主要的研究目的之一，便是細胞分配機制的探討，此機制可將場景中複雜的幾何資料適當地分派給顯像資料計算機器 CM 負責處理，以得到最短的回應時間，並提供顯像機器 DM 正確的顯像資料結果，以便進行成像處理，而其最主要的分配方式便是利用細胞內的多邊形個數以及顯像資料計算機器的效能來達到負載平衡的目的。

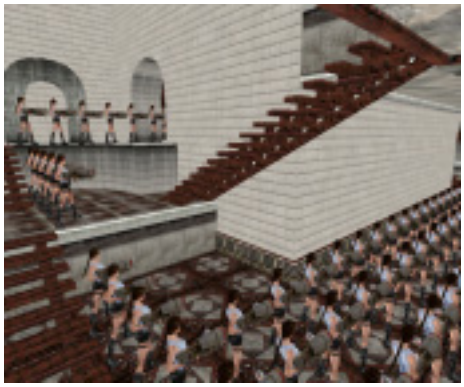
在前置處理階段，系統已將場景切割成一塊塊的細胞，也就是說，系統已將整個場景資料劃分為許多子叢集，在執行階段便可根據目前系統中 CM 的分佈情形，依據細胞分配機制將這些子叢集分別配送給適當的 CM 負責計算，DM 的顯像資料便可經由 CM 的計算取得。本論文所採用之系統模型是以集中式(centralized)的分散式架構建構而成，所有細胞的分配工作都由系統管理代理者 MA 進行，MA 即為整個系統的控管者，當系統中的計算叢集發生變動時，MA 便會根據細胞分配機制重新分配每個 CM 所負責計算的細胞，以求 DM 能在一定的時限內接收到所有 CM 的計算結果並予以顯像，所以說細胞分配機制的優劣與否，大大地影響了 DM 的顯像結果，效率較差的分配機制將導致計算叢集中每部 CM 計算速度上的嚴重落差，整個系統的效率也將會被計算速度較慢的 CM 所拖累，造成 DM 的顯像延遲及畫面失真，而這也是我們所要避免的現象。

4.實驗結果

本實驗之場景資料是採用 id Software 遊戲公司所開發之 Quake III 18 的場景資料格式，是為 1,230 x 720 x 1,780 平方單位的地圖測試場景，場景畫面如圖六所示，此場景依 bsp 檔案格式的定義共被分割成 2,233 個細胞，我們再於其中加入 500 個物件，共計有 980,401 個多邊形，並在每個細胞找出投影面積大於 30 平方單位的多邊形做為該細胞的遮擋者。開發語言為 C 語言，Visual C++ 6.0 為開發工具，搭配 OpenGL 繪圖函式庫以及採用 Win32 視窗程式設計，並在 Windows 2000 的作業平台上開發完成。本章之中的相關實驗數據是以 Intel Pentium 2.4 GHz CPU、512 MB DDR SDRAM 及顯示晶片為 GeForce4 Ti 4200 Series 64 MB DDR SDRAM 的 PC 為顯像機器，Intel Pentium 2.0 GHz CPU 及 512 MB DDR

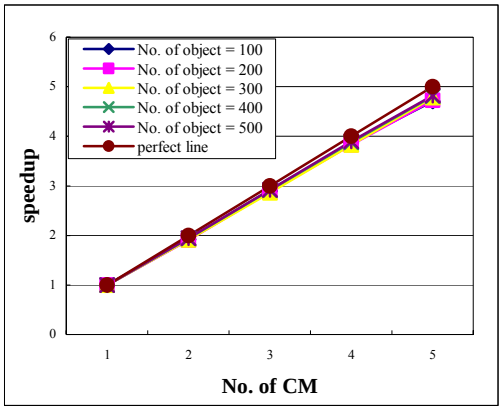


(a) 場景鳥瞰圖



(b) 場景內部畫面

圖六、 測試場景取樣圖



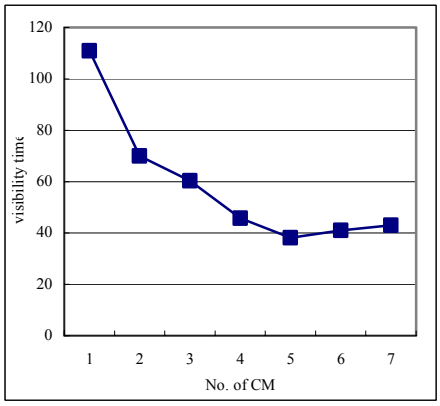
圖七、 分散式叢集計算顯像資料折線圖

SDRAM 的 PC 叢集為顯像資料計算機器，在 Windows 2000 的作業環境下測試而得。

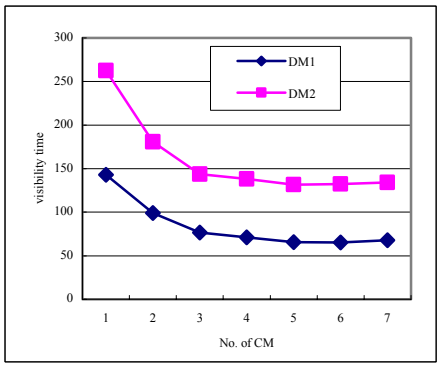
在前置在前置處理時間，本實驗分別比較以一至五部計算資源進行運算所需的時間，由實驗數據可以發現當使用的計算資源數量愈多時，效果愈好，如圖七所示。

圖八則為 DM 數為 1 以及 2 時之顯像資料計算時間折線圖，由圖中可明顯看出，當 CM 數增加為 6 時，其顯像資料計算時間往上回升的情形，主要是由於每增加一部 CM，DM 便須新增一個執行緒與之溝通，而當 CM 數愈來愈多時，DM 的工作排程必須在顯像執行緒以及 6 個與 CM 溝通之執行緒間交替切換，所以花費在執行緒切換的時間卻反而影響了整個顯像資料的計算時間。

表一是使用我們所提出的細胞分配機制和平均分配所做的一個比較，由表中我們可以很清楚的看出，採用細胞分配機制的效果有顯著的提升，最主要是由於使用細胞分配機制可依據 CM 的計算效能，讓效能較佳的 CM 負責較多的細胞，而效能較差的 CM 則負責較少量的細胞；而平均分配方式只是將細胞等份切割再派送給所有 CM，無法依據 CM 的工作效能做出調配。



(a) DM 數為 1



(b) DM 數為 2

圖八、 不同 DM 數之計算時間折線圖

圖九是統計新 CM 加入或移除時，顯像資料計算時間的轉折情形，我們可以發現在折線圖中，有部份顯像資料計算時間明顯地攀升，因為每當有新 CM 加入系統或從系統中移除時，MA 就必須重新分配各部 CM 的細胞，並將分配後的結果傳送給各部 CM，此時 CM 就必須更新自己所負責的細胞，而這個動作需要一段時間方可完成，而這也就造成了新 CM 加入時或移除舊 CM 時，顯像資料計算時間的延遲。

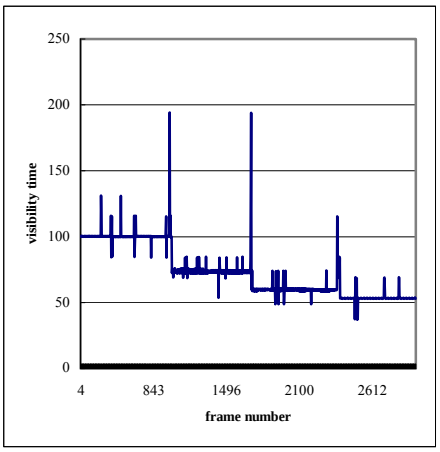
表一、 顯像資料計算時間比較表

(a) CM 數為 2

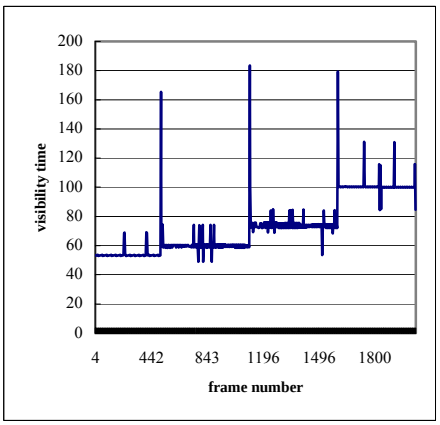
採用的方法	平均值	最小值	最大值
細胞分配機制	66.895 ms	15.585 ms	156.335 ms
平均分配機制	92.311 ms	31.034 ms	172.022 ms

(b) CM 數為 3

採用的方法	平均值	最小值	最大值
細胞分配機制	47.446 ms	0.116 ms	172.615 ms
平均分配機制	67.927 ms	15.962 ms	156.963 ms

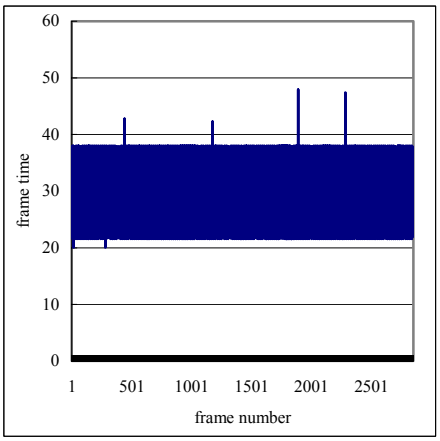


(a) 新增顯像資料計算機器

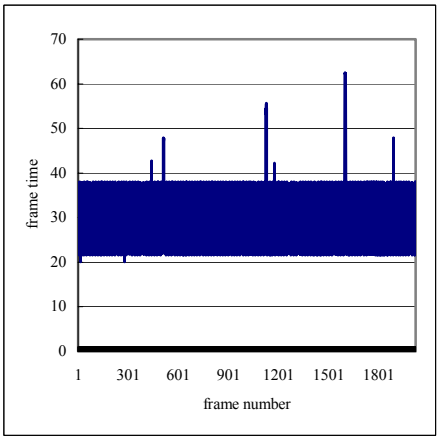


(b) 移除顯像資料計算機器

圖九、 新增與移除顯像資料計算機器之顯像資料計算時間折線圖



(a) 新增顯像資料計算機器



(b) 移除顯像資料計算機器

圖十、 新增與移除顯像資料計算機器之畫面顯像時間折線圖

圖十為新增與移除顯像資料計算機器的畫面顯像時間折線圖，我們分別在執行期間，自系統中逐次新增與逐次移除 CM，由圖中之折線可以發現，每當有 CM 新增或移除的情況發生時，顯像時間便會發生短暫的延遲情形，其原因是由於當 CM 從系統中新增或移除之後，必須重新去分配每台顯像資料計算機器所需處理的細胞，其計算負載量也就隨之提高，間接地影響了顯像效能，但是對 DM 的影響不大。

5.結論與未來發展

本論文之顯像環境技術提供了一個縮短前置處理時間的方法，系統只需要挑選出每個空間細胞的遮擋者集合，在執行階段便可流暢的進行顯像動作，且當場景過於複雜以及分割而得的空間細胞過多時，透過分散式系統叢集也可大幅降低前置處理的計算時間。在執行階段，藉由新增顯像資料計算機器，將使得即時遮擋性裁切測試的複雜比對計算工作，得以分散給新增的顯像資料計算機器協助處理，進而提高顯像機器的畫面顯像頻率 (frame rate)。而經由顯像機器的加入功能，可提供多位使用者利用不同的顯像機器同時於場景中瀏覽遊走，並可同時獲得顯像資料計算機器的計算支援。

本論文後續研究建議可往以下幾點發展及探討：

將集中式 (centralized)架構調整為非集中式 (decentralized)架構：本論文所使用的分散式系統負載平衡機制是以集中式架構進行，所有分配工作的動作都是由系統管理代理者(MA)負責處理，若當 MA 發生非預期性的錯誤時，整個系統也將無法順利運作。因此未來的發展方向可往非集中式架構的方向進行。

執行期間動態偵測系統負載效能：在執行

階段時，顯像資料計算機器 (CM)的效能可能因為某些因素而降低，若此情況發生時系統仍以舊有的細胞分配方式進行計算，整個系統的顯像效能必將因為此 CM 之個別因素而降低，大幅地影響整體顯像流程的進行。因此，未來的發展工作也可針對整個系統的負載效能進行動態偵測，並提出一個評估方法來即時調整細胞分配的狀態。

動態場景的規劃及遮擋性裁切技術：對動態場景而言，場景中的物件並非是固定不動的，它們可能隨著時間的不同而有著不一樣的位置，此時若想對動態物件進行遮擋性裁切，勢必要事先追蹤出物件的位置資訊，因此在動態場景中的場景物件資料結構必須隨著物件的移動而隨時更動其狀態，而這個更新動作若是由單機進行處理，且伴隨著顯像流程的進行，必定會使顯像速率大受影響。因此若能利用分散式系統叢集的眾多計算資源來進行動態物件的處理，並對動態物件進行遮擋性裁切的測試，系統將會有良好的顯像頻率。

參考文獻

[1] D. Aliaga, J. Cohen, A. Wilson, E. Baker, H. Zhang, C. Erikson, K. Hoff, T. Hudson, W. Stuerzlinger, R. Bastos, M. Whitton, F. Brooks, and D. Manocha, “MMR: An Interactive Massive Model Rendering System Using Geometric and Image-Based Acceleration,” *Proc. ACM Symp. Interactive 3D Graphics, pp. 199-206, 1999.*

[2] Jonathan Blow, “Terrain Rendering at High Levels of Detail,” *In Proceedings of the 2000 Game Developers’ Conference, March 2000.*

[3] Thomas L. Casavant and Jon G. Kuhl, “A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems,” *IEEE Transactions on Software Engineering, Vol. 14, No. 2, pp. 141-154, 1988.*

[4] James H. Clark, “Hierarchical Geometric Models for Visible Surface Algorithms,” *Communications of the ACM, Vol. 19, No.10, pp. 547-554, October 1976.*

[5] Daniel Cohen-Or, Yiorgos L. Chrysanthou, Claudio T. Silva, and Fredo Durand, 2000, “A Survey of Visibility for Walkthrough Applications,” *IEEE Transactions on Visualization and computer graphics, Vol. 9, No. 3, pp. 412-431, 2003.*

[6] Orly Kremien and Jeff Kramer, “Methodical Analysis of Adaptive Load Sharing Algorithms,” *IEEE Transaction on Parallel Distributed Systems, Vol. 3, No. 6, pp. 747-760, November 1992.*

[7] Subodh Kumar, Dinesh Manocha, William Garrett, and Ming Lin, “Hierarchical Back-Face Computation,” *Computers and Graphics, vol. 23, No. 5, pp. 681-692, October 1999.*

[8] Chin Lu and Sau-Ming Lau, “An Adaptive Load Balancing Algorithm for Heterogeneous Distributed Systems with Multiple Task Classes,” *In Proceedings of 16th International Conference Distributed Computing Systems, pp. 629-636, May 1996.*

[9] Jarek Rossignac and Paul Borrel, “Multi-resolution 3D Approximations for Rendering Complex Scenes,” *Conference on Geometric Modeling in Computer Graphics, pp. 455-465, 1993.*

[10] Gernot Schaufler and Wolfgang Stürzlinger, “Three Dimensional Image Cache for Virtual Reality,” *Computer Graphics Forum, pp. 227-236, 1996.*

[11] Gernot Schaufler, Julie Dorsey, Xavier Decoret, and Francois X. Sillion, “Conservative Volumetric Visibility with Occluder Fusion,” *In Proceedings of SIGGRAPH’00 on Computer Graphics, pp. 229-238, 2000.*

[12] Jonathan Shade, Steven Gortler, Li-wei He, and Richard Szeliski, “Layered Depth Images,” *In Proceedings of SIGGRAPH’98 on Computer Graphics, pp. 231-242, July 1998.*

[13] Peter Wonka and Dieter Schmalstieg, “Occluder Shadows for Fast Walkthroughs of Urban Environments,” *Computer Graphics Forum, Vol. 18, No. 3, pp. 51-60, September 1999.*

[14] Peter Wonka, Michael Wimmer, and Dieter Schmalstieg, “Visibility Preprocessing with Occluder Fusion for Urban Walkthroughs,” *Rendering Techniques 2000: 11th Eurographics Workshop on Rendering, pp. 71-82, June 2000.*

[15] Peter Wonka, Michael Wimmer, and Francois X. Sillion, “Instant Visibility,” *Computer Graphics Forum, Vol. 20, No. 3, pp. 411-421, 2001.*