

Sequential Point Clusters を用いた大規模モデルに対する 効率的なポイントベースレンダリングシステム

岡本 泰英[†] 山崎俊太郎 池内 克史

[†] 東京大学情報理工学系研究科 〒153-8505 東京都目黒区駒場 4-6-1

E-mail: †okamoto@cvl.iis.u-tokyo.ac.jp

あらまし モデリング技術の向上により、大規模な 3 次元モデルの生成が容易になってきた反面、生成された大規模モデルを従来のポリゴンを用いたレンダリング手法ではインタラクティブ性のある効率的描画が困難になっている。そこで本論文では、多重解像度の考えに基づき、Point Based Rendering 手法を用いた効率的な描画システムを提案する。本システムでは、Sequential Point Trees によるグラフィックスハードウェアのみを用いた多重解像度描画アルゴリズムに、データの幾何的な位置によるクラスタリングを加えることで、処理データ量を削減する Sequential Point Clusters への拡張を提案する。実際に、面数が数千万である大規模なモデルを用いて本手法の実用性を検証するとともに、最適なクラスタリングレベルを探り、その有効性を明らかにする。

キーワード ポイントベースレンダリング、多重解像度、グラフィックスプロセッサ (GPU)

Efficient Point-based Rendering Method for Huge 3D Models using Sequential Point Clusters

Yasuhide OKAMOTO[†], Shuntaro YAMAZAKI, and Katsushi IKEUCHI

[†] Information Science and Technology, University of Tokyo Komaba 4-6-1, Meguro-ku, Tokyo, 153-8505 Japan

E-mail: †okamoto@cvl.iis.u-tokyo.ac.jp

Abstract Advancement in modeling technologies has enabled us to obtain 3D models composed of an extremely large number of polygons. Unfortunately, however, the conventional polygon-based rendering method is not rapid enough for rendering such models interactively. In this paper, we propose a system for effectively rendering such models based on using the Point-Based Rendering technique that considers multi-resolution representations. Our system adopts the data structure, referred to as Sequential Point Trees, realizes the multi-resolution representations and the hardware accelerating rendering. In addition, we present the extension to Sequential Point Clusters, which is accomplished by using positional clustering, which can reduce the amount of data transfer. We have verified the efficiency of our proposed algorithms by rendering models with over ten million polygons. And we have searched the best level of clustering, and demonstrated the superiority of clustering over other methods.

Key words point-based rendering, multiresolution, graphics processing unit(GPU)

1. はじめに

実世界の物体を計測し 3 次元モデルに変換することはコンピュータビジョンの最も知られた応用例である。これにより物体のデジタルデータとしての保存やコンピュータグラフィ

クスによる再現などが可能となる。近年では計測やモデリング技術の発展により、非常に高精細な 3 次元モデルを生成することが可能になってきている。例えば実物体を計測する際に用いられるレーザーレンジファインダは、近距離では 1mm 以下の精度、また 50m ~ 100m といった遠距離からの計測に

においても 1cm 以下という精度を持つ。そのため非常に高精細な幾何形状の取得が可能となる。また、このような技術の発展に伴い、その対象となる物体も非常に広範囲に広がっている。我々の行うプロジェクトでは、その対象として文化遺産や美術品などを扱っている [7] が、その中には 10m を超える大仏や、100m 以上の幅を持つ遺跡といった非常に規模の大きなものも含まれる。このような物体の計測も技術の進歩からその効率が非常に高まり、可能となっている。

しかし生成される 3 次元モデルが高精細かつ大規模なものになる反面、そのデータ量もより大規模になりつつある。そのような大規模なデータをもつモデルの描画は従来の描画方法ではその処理コストが非常に高くなり、対話的な描画を行うことはほぼ不可能となっている。

コンピュータグラフィクスで 3 次元モデルを描画する際に最も広く用いられている手法は、モデルの表面をポリゴン(三角形)の集合として表現し描画の際にそのポリゴンを描画することによりモデルを表現する、ポリゴンレンダリングである。この方法は多くの物体を非常に忠実に再現しうる描画方法であり、スムーズシェーディングなどの手法を用いることで曲面などの形状もそれらしく描画を行うことが出来る。しかしポリゴンレンダリングはその描画計算に複雑な計算を要し、物体の持つポリゴン数が大きくなるにつれてそのパフォーマンスは急激に落ちる。

大規模モデルを効率的に描画する手法の一つに level-of-detail(LOD) の手法がある。これは投影サイズが非常に小さくなるポリゴンの描画処理を省いたり [6]、そのようなポリゴン同士を結合して大きなポリゴンを生成して描画する [1] ~ [3] といった手法である。これにより描画コストを抑えることが可能になるが、そのようなデータを生成する前処理の計算が非常に複雑で現実的な方法ではない。

一方で、描画単位としてポリゴンよりもその処理コストの小さいポイントを用いて描画するポイントベースレンダリング [4] も提案されている。これはポリゴンの投影サイズがピクセルの大きさを下回るような場合は、複雑な計算を必要としないポイントで描画する方が効率が良くなるという考えに基づく。さらにポイントベースレンダリングに LOD を適用することが可能であり、その前処理計算のコストはポリゴンのそれに比べ、接続情報が無いことなどから非常に小さくなっている。QSplat [5] や Surfel [8] では多重解像度を用いたポイントベースレンダリングシステムを実現している。QSplat は元のデータの点群を葉とする階層構造を生成し、描画の際にはそれを適当に探索することで動的に解像度を決定する多重解像度を実現している。

近年はグラフィクスプロセッサ (GPU) の性能も向上しており、これを利用した描画法の研究も盛んである。Sequential Point Trees(SPT) [9] は GPU の逐次処理の優位性を生かすた

め、点群の階層構造を 1 次元のリストに変換しこれを逐次アクセスするだけで必要な点群のみを描画するアルゴリズムとなっている。

我々の研究ではこれらの先行研究を踏まえ、SPT のアルゴリズムを基本とした大規模モデルの効率的な描画システムの提案を行う。そして SPT に対する効果的な拡張として、SPT の 1 次元リストを点群の幾何的な位置によってクラスター化することにより、SPT の探索処理を効率的にし、描画速度を向上する拡張を提案する。我々はこれを Sequential Point Clusters と呼ぶことにする。この論文ではこの新手法が従来の SPT やその他の手法に比べ効率的であるかを示し、かつベストなクラスターレベルを実験によって探ることにする。

2. QSplat [5]、Sequential Point Trees [9]

本手法では先行研究である QSplat の階層構造生成アルゴリズムと Sequential Point Trees の描画アルゴリズムを基礎としたシステムを構成している。そのアルゴリズムに幾何的な位置によるクラスタリングを加えることで SPT の描画処理における枝狩りの処理を効率化する。この章ではまず先行研究に基づく基本的なアルゴリズムを述べることにする。

まず多重解像度の実現のため元のデータから階層構造の生成を行う必要がある。入力としては元の 3 次元モデル (通常ではポリゴンデータを想定) を用いる。まずこの入力モデルよりそれぞれの点要素が位置・法線・半径などの情報を持つ包囲球となるような、点群データを生成する。この時得られる点群が階層構造の葉、つまり多重解像度における最も高い解像度のデータとなる。この点群データから次のような再帰的な手順によって階層構造の生成を行う。

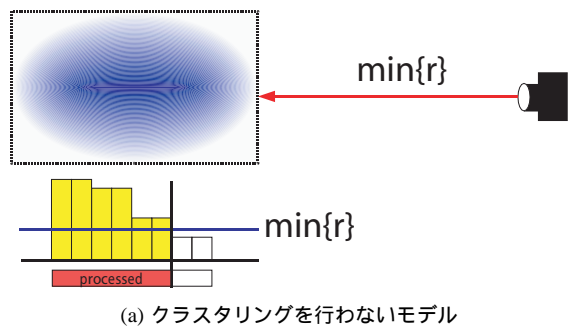
(1) 点群を包含する直方体の一番長い軸に関して、点群データを 2 等分する。

(2) それぞれの点群データに対してこの処理を再帰的にを行い、子包囲球を定義する。

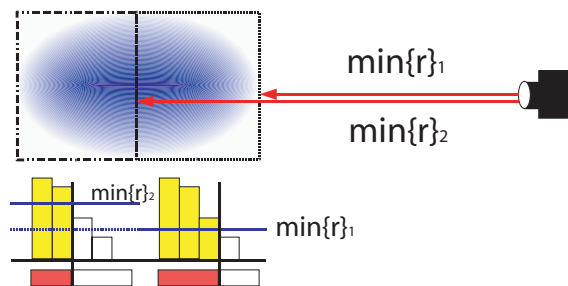
(3) 得られた子包囲球から現在の点群データの包囲球を定義する。

これにより、各節が包囲球となるような階層構造が生成される。QSplat ではこうして得られた階層構造を視点が切り替わった際に探索し、探索中の節の包囲球の描画サイズが要求された解像度を実現できるだけの細かさであれば、その包囲球を描画しそれ以下の木は探索しない。また、包囲球の投影位置がスクリーン中に無い、もしくは背面を向いているようなものであれば枝狩りすることが出来る。

QSplat の描画時の探索である包囲球を描画するか再帰処理をつづけるかの判定は、描画した時の投影サイズが閾値 ϵ より小さいかどうかで決定する。投影サイズはその包囲球の半径 R と視点までの距離 r に依存する。投影サイズを R/r とするとその判定は $R/r < \epsilon$ により決定する。それぞれの包



(a) クラスタリングを行わないモデル



(b) 2 個のクラスターに分割したモデル

図 2 位置によるクラスタリングによる枝狩りの効果。モデルの下の棒の列が r_{max} でソートした点要素を表し、黄色い棒が枝狩りで残った点要素を表す。

Fig. 2 The pruning of positional clustering. The row of bars means the list of points, and yellow bars is rest points after pruning.

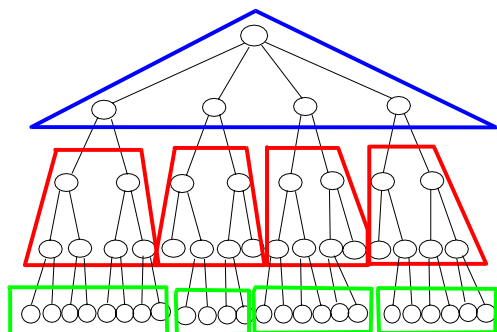


図 3 階層的なクラスターの分割

Fig. 3 Hierarchical splitting.

影座標色の計算である。GPU に送られた点群リストはこのうち少なくとも描画判定処理は行われる。よって枝狩りの効果を高めることで GPU に送られる点群要素の数を減らすことは、描画速度を上げる面で非常に意味がある。

3.2 クラスタリング方法

幾何的な位置情報によるクラスタリングの具体的な方法として図 3 のように示すような階層的な分割を行う。階層構造を生成する過程でそれぞれのクラスターがほぼ同じ数の点要素を持つように、それぞれの節 (に相当する点要素) をクラスターに分配する。階層構造生成の方法から、このように階層構造で近い節同士をクラスターにすることで同時に位置によ

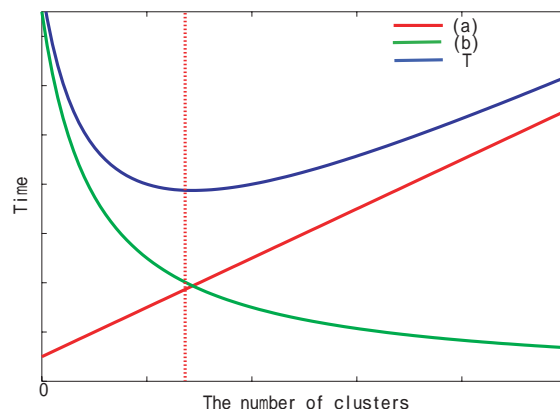


図 4 クラスタリングによる効率化とオーバーヘッドの関係。
Fig. 4 The relation between efficiency and overhead by clustering.

るクラスタリングが行える。このような階層的な分割を行うことにより、枝狩りの可能性はさらに高まる。

SPT では階層構造の上部 (根に近い部分) の枝狩りは全く行わない。これは階層構造を再帰的な探索で行う QSplat も同様である。そのため視点がモデルに近づく程、階層構造の上部の描画判定処理は無駄なものとなる。そこで階層構造を階層的なクラスターに分割し、それぞれのクラスターにそれに属する点要素の r_{min} の最小値 $\min\{r_{min}\}$ の情報を保持させる。もし、そのクラスターが $\min\{r_{min}\} > \max\{r\}$ という条件を満たしていれば、そのクラスター中の点要素は描画される可能性が全く無いためこのクラスターについても枝狩りが可能である。こうして階層構造上部の枝狩りも可能になった。

ただし階層構造の根側の節数は葉側のそれに比べ非常に少ないために、根側の枝狩りは葉側ほど効果的ではない。しかしモデルが非常に大きくなりクラスターの階層の深さが大きくなれば、このような方法によるクラスタリングの効果も高くなる。

3.3 最適なクラスタリング

3.1 の議論を再帰的に行うことにより、枝狩り効率は細かくクラスター化するほど上がることがいえる。実際に 1 つ 1 つの点要素ごとに枝狩り判定を行うことで最も枝狩り率が高くなる。しかし、クラスタリングにより点群リストをより細かいリストに分けることで、GPU への転送準備コスト・転送コスト・GPU 上での描画処理の準備コストなどのオーバーヘッドが生じる。

そこで、単純なモデルを考える。クラスタリングをした場合のオーバーヘッドを (a)、GPU 上での描画判定および描画時間を (b) とする。(a) はクラスタリングをする程増加する値であり、(b) はクラスタリングをする程減少する値である。これらの値と描画時間 $T = (a) + (b)$ との関係は図 4 のようになる。このため、クラスタリングによる枝狩り効率と、それによるオーバーヘッドとのバランスを満たしたクラスタリン

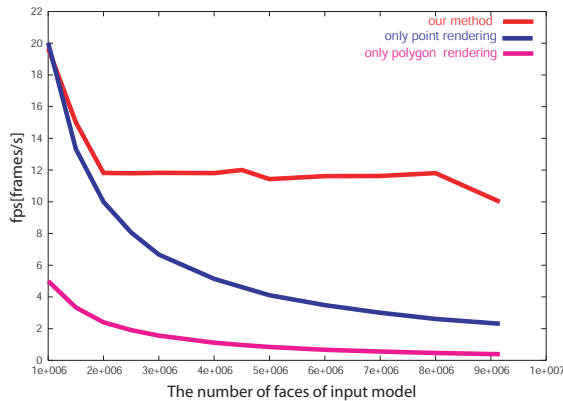


図6 入力モデルのサイズに対する描画速度の変化 (提案手法、点のみによる描画、ポリゴンのみによる描画)
Fig. 6 The performance to the size of the input model (proposed method, point rendering, polygon rendering).

レベルでこれを行う必要がある。この決定にはハードウェア性能やモデルの形状などの要因も関わってくると考えられるが、本論文ではクラスターの要素数について実験の章でこれを考察することとする。

4. 実験結果

我々はこの手法を実装およびそのパフォーマンスの測定において、次のような環境下で行った。CPUとしてIntel Pentium4の3.2GHzおよび2GバイトRAM、そしてグラフィクス環境としてGPUにATI RADEON9800XTを用い、プログラミングにはDirectXを使用した。

この章では、まず複数の大規模モデルについてその描画速度の測定を行い、次にクラスター化による描画速度の変化を調べ考察することとする。

4.1 描画速度

図5に3つのモデルのデータ生成コストと、同一の解像度で描画した場合の描画速度を示す。入力モデルのポリゴン数が10Mを超える場合においてもその描画フレームレートは毎秒10を達成できていることが分かる。この描画速度はシステムの対話性を維持する上で十分な速度を達成している。また、我々の提案手法と従来行われてきたような単純な描画法(多重解像度を用いないポイント・ポリゴンレンダリング)の、入力モデルに対する描画速度の変化は図6のようになる。これは従来の描画法が入力データのサイズと共に速度が一緒に減少していることに対し、多重解像度を用いた我々の手法ではその速度の減少がフレームレート毎秒10付近で非常に緩やかになっていることが分かる。これは多重解像度の採用により、不要に微細な点の描画を省いた結果である。

4.2 クラスタリングによる効率性の検証

前述した通り枝狩りにより点群要素を減らすことは有効であるが、過度のクラスタリングは転送・描画命令発行の上昇

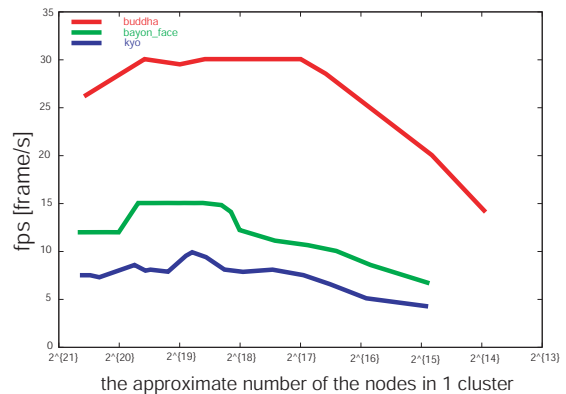


図7 クラスタリングに対する描画速度の変化:入力モデルのサイズは上から1M、6M、9Mポリゴン。グラフの右ほど細かいクラスタリングをしている。
Fig. 7 The performance to the level of clustering:the numbers of polygon are 1M, 6M, 9M from upper to bottom. The go to the right, the finer clustering is done.

からオーバーヘッドが高まる。このためこの章では、これらの要因の釣り合いを取った適切なクラスタリングを探る。

図7にクラスター中の点要素の数に対する描画速度の変化の様子を3つのモデルについて表す。このグラフからそれぞれのモデルの場合においても、その描画速度はピーク部分が存在することが分かる。そしてそのピーク部分の範囲はモデルのサイズが大きくなるに従い、狭くなっていることが分かる。これはモデルのサイズが大きいほど、クラスターサイズを変えた際の総クラスター数の変化は大きくなる。このクラスター数の変化により、枝狩り処理やGPUへの転送処理などが増えCPU負荷が高まることが考えられる。そのためモデルのサイズが大きいほど、描画速度は変化しやすいと考えられる。

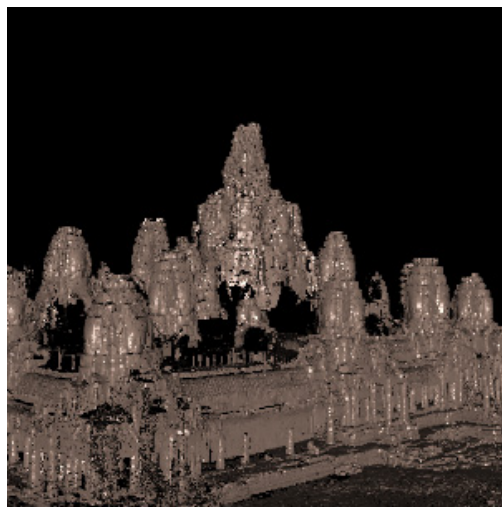
そしてこのピーク部分は、1クラスターあたりの点要素数がおよそ500kの場合、どのモデルにおいても最も描画速度が高くなっているといえる。1クラスター内ではさらに法線クラスタリングが行われているため、およそ数k~10kのオーダーの点要素が1度の転送・描画処理にかけられていることになる。1つの結論として、クラスターのサイズがこのピーク部分に来るようなクラスタリングを行うことで、クラスタリングを行わない場合(SPT)、図7ではグラフの一番左にあたる部分よりも、その描画速度が向上することが分かった。グラフからはその速度は少なくとも20%程度上昇していることが分かった。

5. まとめと考察

本論文では大規模なモデルを対話的な速度で描画するためのシステムを提案した。その具体的手法として先行研究であるSequential Point Treesのアルゴリズムに、データをクラス



Model name	happy buddha	kyo
Number of polygons	1,087,716	9,162,909
Preprocessing time (sec)	42.8	201.0
Rendering speed (fps)	19.7	14.5



Model name	towers	3towers
Number of polygons	13939070	18132893
Preprocessing time (sec)	377.2	471.5
Rendering speed (fps)	15.0	10.0

図5 描画結果およびパフォーマンス
Fig.5 Rendering result and performance

タリングすることによりその枝狩りを効率化した Sequential Point Clusters を提案した。この手法は従来の多重解像度によるポイントベースレンダリングと SPT の GPU 性能を生かした逐次的な描画探索アルゴリズムに、枝狩りのより高い効率性を加えたものとなっている。また、SPT では行われていなかったスクリーン外要素の除去や、階層構造の根側の除去などのさらなる効率化が可能になっている。ただし、過度のクラスタリングはオーバーヘッドの増大を招くため、適度なクラスタ化を行う必要がある。

これにより実際、クラスタリングを用いなかった場合に比べて、ある程度のクラスタリングを行った場合の方がその描画速度が約 20% 高くなる結果が実験により得られた。

しかし、本手法のデータ構造では従来手法の QSplat で実現しているような階層構造を利用した圧縮法の実現が不可能になっている。大規模データを扱うという目的から考えるとデータ圧縮の必要性は高く新たな圧縮手法の考案が必要であるが、これは将来の課題とする。

文 献

- [1] H HOPPE. Progressive meshes. In *Proceedings of SIGGRAPH 96, Computer Graphics Proceedings*, pages 99–108, 1996.
- [2] H HOPPE. View-dependent refinement of progressive meshes. In *Proceedings of SIGGRAPH 97, Computer Graphics Proceedings*, pages 189–198, 1997.
- [3] G. KLEIN, R. LIEBICH and W STRASSER. Mesh reduction with error control. In *IEEE Visualization '96*, pages 311–318, 1996.
- [4] M. Levoy and T. Whitted. The use of points as a display primitive. Technical report, University of North Carolina at Chapel Hill Technical Report TR 85-022, 1985.
- [5] Szymon Rusinkiewicz. Marc Levoy. Qsplat: a multiresolution point rendering system for large meshes. In *Proceedings of ACM SIGGRAPH 2000, Computer Graphics Proceedings*, pages 343–352, 2000.
- [6] D. Luebke, M. Reddy, J. Cohen, A. Varshney, B. Watson, and R. Huebner. *Level of Detail for 3D Graphics*. Morgan Kaufmann, 2002.
- [7] D. Miyazaki, T. Ooishi, T. Nishikawa, R. Sagawa, K. Nishino, T. Tomomatsu, Y. Takase, and K. Ikeuchi. The great budha project: modelling cultural heritage through observation. In *Proc. International Conference on Virtual Systems and Multimedia (VSMM)*, pages 138–145, 2000.
- [8] H. Pfister, M. Zwicker, J. van Baar, and M. Gross. Surfels: Surface elements as rendering primitives. In *Proceedings of ACM SIGGRAPH 2000, Computer Graphics Proceedings*, pages 335–342, 2000.
- [9] Carsten Dachsbacher. Christian Vogelgsang. Marc Stamminger. Sequential point trees. In *Proceedings of ACM SIGGRAPH 2003, Computer Graphics Proceedings*, pages 657–662, 2003.
- [10] H. Zhang and K.E.H. III. Fast backface culling using normal masks. In *Symposium on Interactive 3D Graphics*, pages 103–106, 1997.