社団法人 情報処理学会　研究報告
IPSJ SIG Technical Report

2004－CVIM－144　(20)
2004／5／6

Sequential Point Clusters：
大規模モデルに対する効率的なポイントベーストレンダリングシステム

岡本　泰英 †, 池内　克史 ††

† 東京大学大学院　情報理工学系研究科　コンピューター科学専攻
†† 東京大学　情報学環

〒 153-8505　東京都目黒区駒場 4-6-1 東京大学生産技術研究所　第 3 部　池内研究室 03-5452-6242
okamoto@cvl.iis.u-tokyo.ac.jp

概要　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 ポイントベーストレンダリング、多重解像度、グラフィックスプロセッサ (GPU)

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

Yasuhide OKAMOTO†, and Katsushi IKEUCHI††

† Department of Computer Science, Graduate School of Information Science and Technology, The University of Tokyo
†† Graduate School of Interdisciplinary Information Studies, The University of Tokyo

Institute of Industrial Science, The University of Tokyo, 3rd Dept. Ikeuchi Laboratory 4-6-1 Komaba, Meguro-ku, Tokyo, 153-8505, JAPAN
+81-3-5452-6242
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.
Keywords　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となってきている。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。我々の今まで行ってきたプロジェクト [7] 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が、その中には 10m を超える大仏や、100m を超える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、センサー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。

こうして生成できる 3 次元モデルがより高精細かつ大規模になる反面、そのデータ量もより大規模になりつつある。しかしそのような大規模なデータからなるモデルの描画は、従来の描画方法ではその処理コストが非常に高くなり、現在のコンピュータの能力では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従来から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、描画の際はそのポリゴンを描画することにより 3 次元モデルを表現する、ポリゴンレンダリングである。この方法は多くの三次元物体を非常に忠実に再現しうる描画法であり、スムーズシェーディングなどの技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処理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描画速度が急激に落ちるという欠点を持つ。

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。これはスクリーン上での投影サイズが非常に小さくなるポリゴンについてはそ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、大きなポリゴンを生成して描画する [1, 3, 2] といった手法である。これによりその描画処理コストを抑えることが可能になる。しかし、ポリゴンレンダリングでは隣接するポリゴンとの関係も考慮に入れる必要があるためそのようなデータを生成する前処理計算は非常に複雑なものとなり、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トレンダリング [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トレンダリングに 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ト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 の逐次処理の長所を生かすため、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事を示し、かつ最適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とする。

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点群データを生成する。これは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データである。この点群が階層構造の葉にあたるデータとなる。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を定義し、そ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行い、子包囲球を定義する。(もし分割後の点群の点数が 1 点ならば、それに対応する包囲球を子包囲球とする)

3. 得られた子包囲球から元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る度に探索し、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。SPT ではこの判定処理を次のような方法によって再帰処理から逐次処理に置換することで、GPU を最大限生かした描画処理を可能にする。

ある包囲球の投影サイズはその半径 R とその中心から視点までの距離 r に依存する。投影サイズを R/r とするとその描画判定は $R/r < \epsilon$ が真であるか、という判定となる。それぞれの包囲球についてその半径と閾値の値は始めから決まっているので、この判定は r により決定されることが分かる。つまり $r > R/\epsilon$ が真であるか、という判定に置き換えられる。この判定はつまり、その包囲球が描画され得る r の下限が R/ϵ であることを示している。この値を r_{min} と置く。また再帰的な探索がその節に達する場合というのは、その親節ではこの判定が偽になる場合に限る。つまり $r \leq R_{parent}/\epsilon$ となる時である。これにより r の上限が決定でき、この値を r_{max} と置く。これはつまりそれぞれの包囲球に対してこの r_{min}, r_{max} をあらかじめ計算し

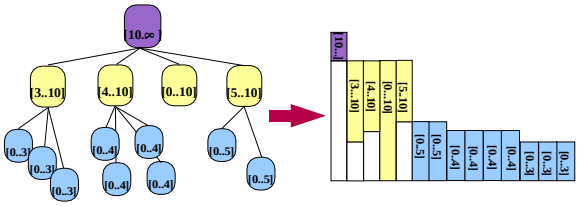


図 1: 階層構造から 1 次元リストへの変換

て保持させることで、その描画判定は

$$r_{min} < r \leq r_{max}$$

の真偽を階層構造中の包囲球に対して行うことで置換できることになる。これにより判定操作が単純化し、GPU 上でこの判定を行うことが可能となった。

SPT では、このようにして得られた 1 次元の点群リストを並び替えることで、この処理をより効率化する手法も提案している。点群を r_{max} の大きい順にソートし、その r_{max} が取り得る r の値よりも小さい点以降のデータについては、GPU への転送および描画判定を行わない。これにより、階層構造の葉に近い側の要素で明らかに描画されないものを枝狩りすることができる。この枝狩りによりモデルと視点が遠いシーンについては、多くの点群の描画処理を省くことが出来るため非常に効率性が高まる。ただ、SPT の方法では階層構造を 1 次元リストへ変換したために背面およびスクリーン外の点群要素の効率的な枝狩りが不可能となっている。そこで SPT では背面カリングのために [10] に基づく法線クラスタリングを採用している。

3 Sequential Point Clusters

前章で挙げたように、SPT のアルゴリズムは背面要素の除去は実現しているが、スクリーン外要素の除去が出来ていないという欠点がある。また、加えて SPT の枝狩りは効果的ではあるがまだ欠点がある。枝狩りの際に用いる閾値は、描画の可能性がわずかでもあるものを取り除かない為に、取り得る r の値の中で最小の値 (つまりモデル中で視点から最も近い点要素までの距離) にする必要がある。このことは、モデルに奥行きがあるようなシーンにおいては、視点から遠い方の点群要素からしてみればその閾値が実際の視点までの距離よりも非常に小さく取られる為、描画される可能

性が無い点が含まれる割合が多くなり、結果として視点から離れるほど枝狩りの効果が半減するという欠点を生み出す。提案手法はこれらの SPT の欠点を克服し、描画判定処理にかける点群要素をより少なくすることで描画速度を向上させることを目的とする。

3.1 位置情報によるクラスタリング

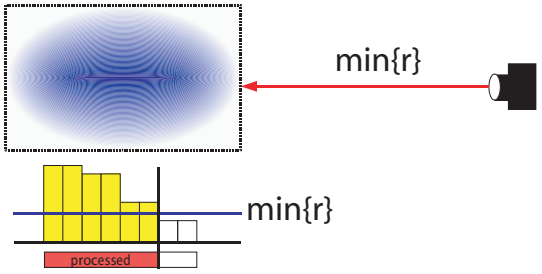
我々は点群をその幾何的な位置によりクラスタリングする手法を提案する。これはモデルを複数の直方体により領域分割を行い、点群をその直方体ごとにクラスターとすることによりそれを行う。

このクラスタリングにより、スクリーン外要素の大まかな除去が可能となる。クラスター化することによりそのクラスターに相当する直方体がスクリーン内に無い場合はその描画判定処理を行う必要が無い。これにより視点がモデルに非常に近い場合などは、一般にスクリーン内におけるクラスターの数は少ないといえるので描画判定処理を大幅に減らすことができる。

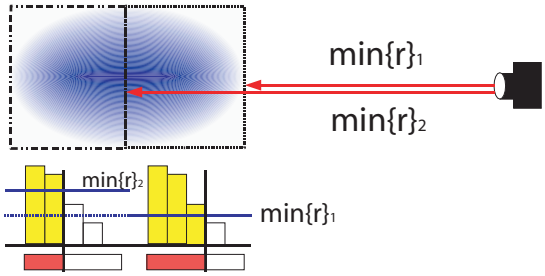
また、このクラスタリングは奥行きのあるモデルに対する枝狩りの効率性を高めるものでもある。枝狩りに用いる閾値は、それぞれのクラスターに相当する直方体への視点からの最小距離とする。これにより、閾値と点要素の実際の視点までの距離との差が縮まり、描画される可能性の無い点が多く除去できるという効果がある。

この効果を図 2 に示す簡単なモデルにおいて検証する。この点群データは直方体中に一様に分布しており、その r_{max} の値も一様な分布を示しているモデルを考える。ここでクラスタリングを行わなかった場合 (図 2 (a)) に、枝狩り後に残ったリストの枝狩り前のリストに対する長さの割合を p とする。次に図 2(b) のようなクラスタリングを行った場合について考える。このときの枝狩りで残ったリストの長さの割合を p' とし、また視点に対して手前のクラスター c_1 中のその割合を p'_1、奥のクラスター c_2 中の割合を p'_2 とする。クラスタリングを行わなかった場合の閾値 $\min\{r\}$ と c_1 の閾値 $\min\{r\}_1$ は、r の最小値が同じであることからその閾値も同じになる。よって一様なモデルにおいては枝狩りされる割合は同じであり、$p'_1 = p$ がいえる。また c_2 の閾値 $\min\{r\}_2$ については、r の最小値は c_1 に比べて明らかに大きくなることがいえる。つまり、$p'_2 > p = p'_1$ がいえる。これらのことから次のような式が成り立つ。

$$p' = \frac{1}{2}p'_1 + \frac{1}{2}p'_2 < \frac{1}{2}p + \frac{1}{2}p = p$$



(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。

これにより点群をクラスター化することで、特に奥行きのある方向から見た場合はその枝狩り効率を向上させることができることが示された。

SPT のアルゴリズムによる描画処理において最も多くの時間を占める操作は、GPU 上での描画判定処理および点の投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

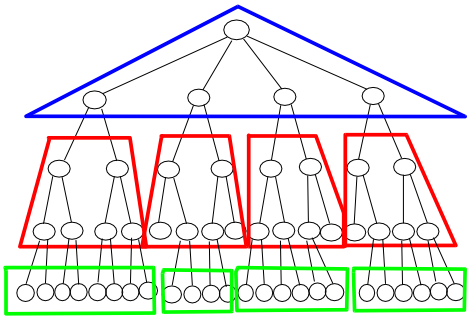


図 3: 階層的なクラスターの分割

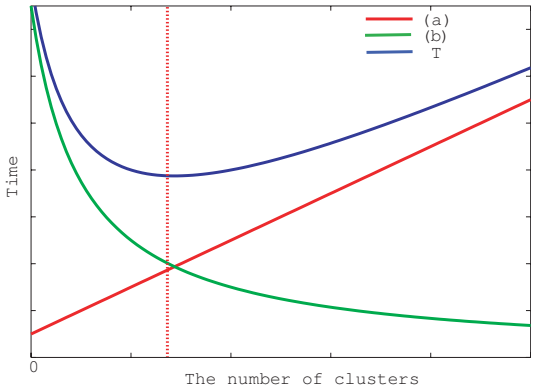


図 4: クラスタリングによる効率化とオーバーヘッドの関係。

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グ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

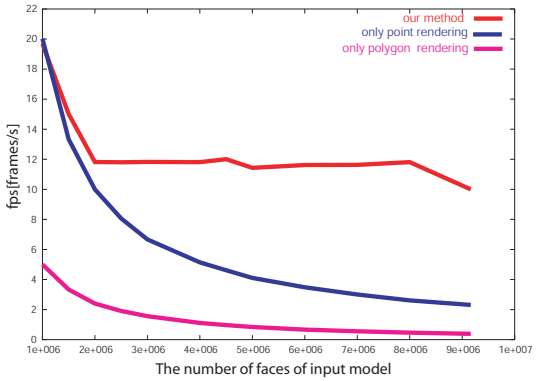


図 6: 入力モデルのサイズに対する描画速度の変化 (提案手法、点のみによる描画、ポリゴンのみによる描画)

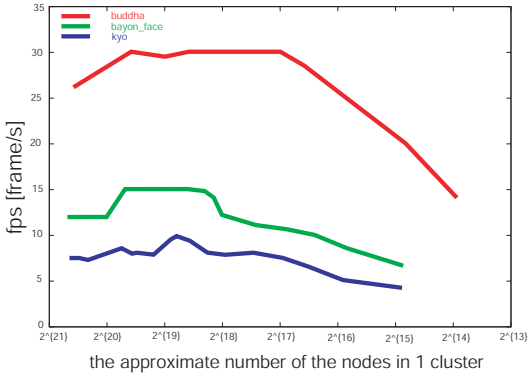


図 7: クラスタリングに対する描画速度の変化:入力モデルのサイズは上から 1M、6M、9M ポリゴン。グラフの右ほど細かいクラスタリングをしている。

来の描画法が入力データのサイズと共に速度が一様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トレンダリングと 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、適度なクラスター化を行う必要性がある。

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。

参考文献

[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.

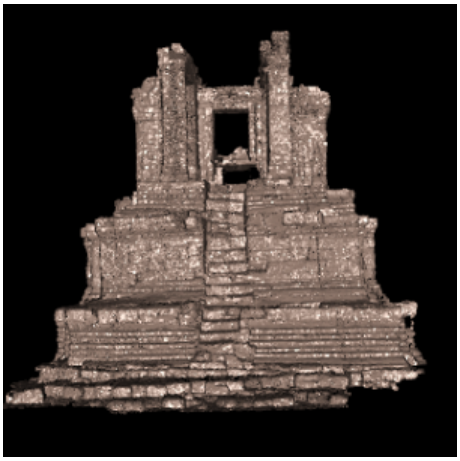
[6] D. Luebke, M. Reddy, J. Cohen, A. Varshney, B. Watson, and R. Huebner. *Level of Detail for 3D Graphics*. Morgan Kaufmann, 2002.

[7] D. Miyazaki, T. Ooishi, T. Nishikawa, R. Sagawa, K. Nishino, T. Tomomatsu, Y. Takase, and K. Ikeuchi. The great buddha project:modelling cultural heritage through observation. In *Proc. International Conference on Virtual Systems and Multimedia(VSMM)*, pages 138–145, 2000.

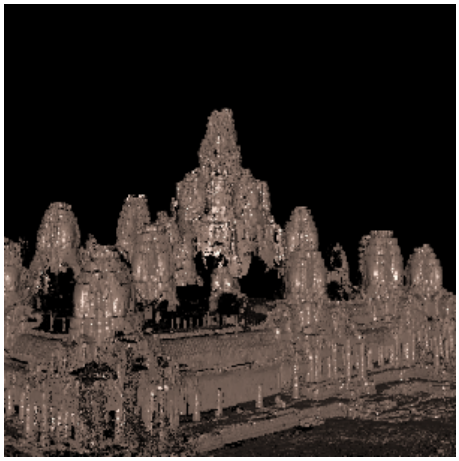
[8] H. Pfister, M. Zwicker, J. van Baar, and M. Gross. Surfels: Surface elements as rendering primitives. In *Proceedings of ACM SIGGRAPH 2000, Computer Graphics Proceedings*, pages 335–342, 2000.

[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.



model	happy buddha	kyo
Input polygons	1,087,716	9,162,909
Preprocessing time	42.8s	201.0s
Rendering time	19.7fps	14.5fps



towers	3towers
13,939,070	18,132,893
377.2s	471.5s
15.0fps	10.0fps

図 5: 描画結果およびパフォーマンス