ハイパフォーマンス
コンピューティング 73－2
（1998．10．9）

データ並列ボリュームレンダリングのためのボリューム適応分割手法

佐野 健太郎　北島 宏之　小林 広明　中村 維男

東北大学大学院情報科学研究科

三次元の広がりを持つデータを可視化するボリュームレンダリングの高速処理手法として、汎用並列計算機による並列処理が注目を集めており、分散メモリ型汎用並列計算機に適した、画像合成法に基づくデータ並列ボリュームレンダリングアルゴリズムが提案されている。本アルゴリズムはリアルタイムレンダリングを可能とするが、計算要素が増加するにつれ並列処理不可能部分である画像合成処理時間が支配的となるため、全体の並列処理効率が低下する。本稿では、画像合成処理時間短縮を目的とした、ボリュームの適応分割手法を提案する。計算要素が多数である程、本手法を用いることにより画像合成時間の短縮が可能であることを、実験結果より確認した。

Adaptive Volume-Subdivision for Efficient Data-Parallel Volume Rendering

Kentaro SANO　Hiroyuki KITAJIMA　Hiroaki KOBAYASHI　Tadao NAKAMURA

Graduate School of Information Sciences, Tohoku University

Using parallel processing on a general-purpose parallel computer that is one of the promising strategies for fast volume rendering, we proposed a data-parallel volume rendering algorithm based on the image composition method. Although the algorithm achieves real-time rendering, a constant processing time of image composition lowers the efficiency of parallel processing as the number of processing elements increases. To solve this problem, this study proposes an adaptive subdividing method of volume data and discusses its performance through some experiments. The experimental results show that the method reduces the image-compositing time as the number of processing elements increases.

1　はじめに

物質の密度等に代表されるスカラー値の三次元分布データ（ボリューム）を可視化する手法であるボリュームレンダリングは、CT(Computed Tomography)やMR(Magnetic Resonance)技術と組み合わせることにより、外科手術シミュレーション等の医療分野への応用が期待されている[1]。外科手術シミュレーションを実現するには、毎秒10回以上レンダリングを行ない画像を更新するリアルタイムレンダリングが不可欠である。しかし、このような応用に求められる高品質な画像を生成するにはボリュームレンダリングは多くの計算時間を必要とし、リアルタイムでの処理を困難としている。このため、これまでにボリュームレンダリングの高速処理を目的とした様々な手法が提案されている。

ボリュームレンダリングの高速化手法は、大きく、1) レンダリングアルゴリズムの改良と 2) レンダリングハードウェアの改良に分類することができる。特に 2) としては、専用ハードウェアの開発もしくは汎用並列計算機を用いた並列処理の適用が考えられる。Lacroute

らは、シアワープ法を用いたレンダリングアルゴリズムの改良によりボリュームレンダリングの高速化を試み、毎秒1回のレンダリング速度を実現した[2]。しかし、ボリュームは今後さらに大きくなると予想され、シングルプロセッサの性能向上を加味してもアルゴリズムの改良のみではリアルタイムレンダリングの実現は困難であると考えられる。Kaufmanらは、毎秒100回以上のリアルタイムボリュームレンダリングが可能である専用のハードウェアを設計した[3]。専用ハードウェアは、非常に高速なレンダリングを可能とする反面、レンダリングアルゴリズムの変更等に関して柔軟性を欠くという短所を持つ。汎用並列計算機による並列処理の適用は、アルゴリズムの改良のみよりもレンダリングを高速化でき、かつ、レンダリングアルゴリズムの変更が比較的容易であるため、ボリュームレンダリングの高速化手法として注目を集めている[4][5][6][7]。

著者らは、並列計算機を用いたボリュームレンダリングの高速化を目的として、分散メモリ型並列計算機のためのデータ並列ボリュームレンダリングアルゴリズムを提案した[4]。本アルゴリズムは、以下の点でこれまでの汎用並列計算機のための並列アルゴリズムよりも優れている。

1. 高速なボリュームレンダリング手法であるシアワープ法[2]を並列化し、さらに高速なレンダリングを実現する。
2. 並列計算機として分散メモリ型並列計算機を使用することにより、共有メモリ型並列計算機を使用した他のボリュームレンダリングアルゴリズム[5]と比べて、共有資源における競合を原因とする台数効果の低下を抑えることが可能である。
3. 画像合成法[8]に基づいており、計算要素(Processing Element、以下PE)間では画像データが通信されるため、ボリュームデータが通信される他の並列ボリュームレンダリングアルゴリズム[6]よりもPE間通信に必要な時間を短縮可能である。

一般に、PE間通信時間は並列処理のオーバーヘッドであり、これが短い程並列処理効率の点で有利である。

本アルゴリズムを32台のPEから構成される汎用並列計算機に実装したシステムでは、256^3の解像度のボリュームから、256^2の解像度の画像を毎秒15回生成することが可能である[4]。しかし、並列処理効率はPE数が増加するにつれて減少し、32PEを使用した時には6割よりも低くなる。この原因は、PE数の増加に伴い並列処理可能な部分の処理時間が短縮される反面、PE間通信に代表される並列処理不可能な部分の処理時間全体に占める割合が増加することにある。

本アルゴリズムでは、ボリュームを部分ボリュームに分割しこれをPEに割り当て、各PEが部分ボリュームの投影により生成した画像を通信を行ないながらPE間で合成する。PE間で通信される画像のデータ量は部分ボリュームの形状に依存する。従って、各部分ボリュームの大きさ及び形状を考慮し、負荷分散が悪化することなくPE間で通信および合成される画像データ量が少なくなるようにボリューム分割を最適化することにより、画像合成時間を短く抑えることが可能であると考えられる。本稿では、本アルゴリズムにおいて並列処理効率低下の原因である画像合成時間の短縮を目的とした、ボリュームの分割手法を提案する。

以下、第2節では、著者らが提案したデータ並列ボリュームレンダリングアルゴリズムについて説明する。次に、第3節では、本アルゴリズムにおける画像合成時間を短縮するためのボリューム分割条件について考察し、考察より得られた条件に近い分割を実現するボリューム適応分割手法を提案する。第4節では、提案する分割手法に関する実験について述べ、その結果に基づき本手法の有効性を評価する。

2 画像合成法によるデータ並列ボリュームレンダリングアルゴリズム

本節では、著者らが提案した、分散メモリ型並列計算機のための画像合成法に基づくデータ並列ボリュームレンダリングアルゴリズム(以下、データ並列画像合成VRアルゴリズム)[4]と、その並列処理効率低下の原因について説明する。

2.1 アルゴリズムの概要

一般にボリュームは三次元配列で表現される。この三次元配列は三次元の直交格子とみなすことができる。配列の各要素はボクセルと呼ばれ、対応する格子点において三次元の元関数を標本化して得た値と、この値から計算により求めた輝度と不透明度を持つ。ボリュームレンダリングは、不透明度による光の減衰を考慮しながらボクセルの輝度を投影面に投影し、画像を生成する手法である[9]。

図1に示すように、データ並列画像合成VRアルゴリズムでは、ボリュームを分割し各PEに割り当てることにより、レンダリングを並列に行なう。ボリューム分割は前処理として、一度のみ行なわれる。本アルゴリズムは、シアステージ、画像合成ステージ、ワー

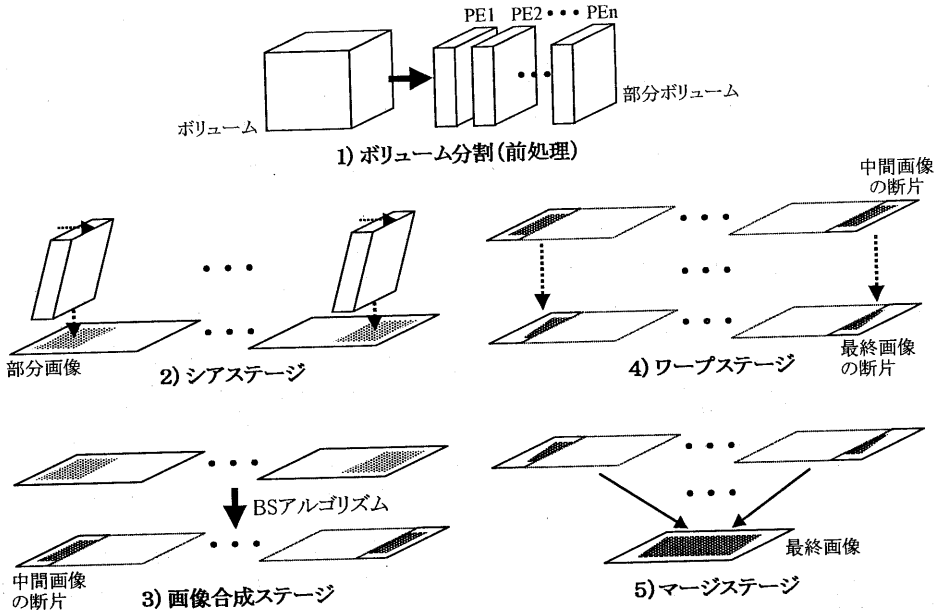

図 1: 画像合成法に基づくデータ並列ボリュームレンダリングアルゴリズム

プステージ、マージステージの 4 ステージからなる。シアステージでは、シアワープボリュームレンダリング法におけるシア処理により、各 PE に分散された部分ボリュームが並列にせん断及び投影される。この結果各 PE において生成される画像を、部分画像と呼ぶ。合成ステージでは、部分画像がバイナリスワップ画像合成アルゴリズム（以下、BS アルゴリズム）[7] により合成される。BS アルゴリズムによる画像合成処理の後には、各 PE は合成済み画像である中間画像の断片を分散して持つ。ワープステージでは、シアワープボリュームレンダリング法におけるワープ処理により、中間画像の断片における歪みを取り除き、最終画像の断片を並列に生成する。マージステージでは、最終画像の断片を 1PE に集め、1 枚の最終画像を生成する。シアステージの処理時間は、部分ボリュームに含まれる不透明ボクセル数に比例する傾向を持つことから、本ステージの負荷分散を保つために、ボリュームの分割は不透明ボクセル数がほぼ均一となるように行なわれる。

2.2 BS アルゴリズム

BS アルゴリズムは、各 PE の持つ部分画像を階層的に合成するアルゴリズムであり、部分画像の送受信のために PE 間通信が必要となる。BS アルゴリズムでは合成処理全般に渡り合成演算の負荷が全 PE に分散され、効率良く処理が行なわれる。

並列計算機の持つ PE 数を $N_p(=2^n)$ とすると、BS アルゴリズムでは、n ステップで PE 間の画像合成が完了する。図 2 に、PE 数が 4 の場合の画像合成手順を示す。BS アルゴリズムの第 k ステップでは、PE 番号の第 k ビットのみが異なる 2 つの PE 間で部分画像を交換する。PE 間では、通信量及び各 PE における合成処理の負荷が均等となるように、互いに合成すべき画像領域の半分を交換し、相手から送られてきた画像領域を自分の持つ領域と合成する。従って、n ステップの画像合成の後には、各 PE は中間画像の $\frac{1}{2^n}(=\frac{1}{N_p})$ の大きさの断片を持つことになる。

BS アルゴリズムによる、合成すべき面積（画素数）が A_{pix} である 画像領域の合成時間 T_{comp} を考える。BS アルゴリズムにおける第 k ステップでは、元の大きさの $\frac{1}{2^k}$ の画像が通信及び合成されるため、T_{comp} は式 (1) のように求められる。

$$T_{comp} = A_{pix}\sum_{k=1}^{n}\frac{1}{2^k} = A_{pix}(1-\frac{1}{N_p}) \qquad (1)$$

BS アルゴリズムによる画像合成時間は、合成する画像領域の面積に比例する。また、PE 数が増加した場合にも発散せず、ある一定値に漸近する。実際には、画像中において必要最小限の領域のみを転送するため、BS アルゴリズムに基づく画像合成時間は多くの場合

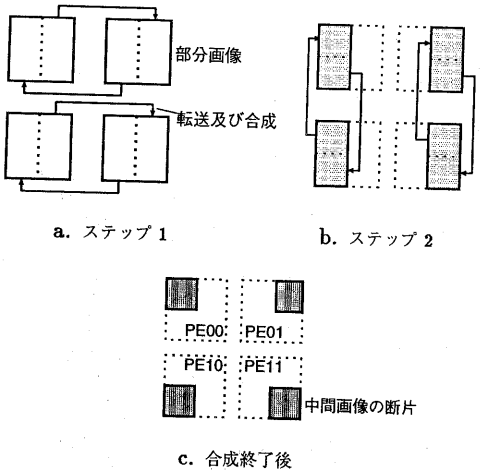

図 2: PE 数が 4 の場合の BS 法に基づく画像合成

T_{comp} 以下となる。

2.3 並列処理効率低下の原因

並列性を持つシアステージとワープステージの処理時間は PE 数の増加に伴い減少する。一方、画像合成ステージとマージステージは並列性を持たず、PE 数が増加しても処理時間は減少しない。2.2 節で述べたように、画像合成ステージの処理時間は PE が増えるにつれて増加し、ある一定値に漸近する。また、マージステージでは、全画像分のデータを 1PE に転送するため、PE 数に関わらずある一定の処理時間が必要となる。従って、PE が多数の場合、非並列処理部分の処理時間が全処理時間に占める割合は大きくなり、並列処理効率が低下する [4]。

並列処理効率を改善するためには、並列処理部分の負荷分散を悪化させずに非並列処理部分の処理時間を短縮する必要がある。マージステージはその性質上処理時間短縮が困難であるが、画像合成ステージでは、部分画像中において合成が必要な領域を削減することにより、画像合成時間の短縮が可能であると考えられる。

3 PE 間画像合成時間短縮のためのボリューム適応分割

本節では、BS アルゴリズムに基づく画像合成時間を短縮するためのボリューム分割条件について考察する。次に、この分割条件に近い分割を実現するボリューム分割方法を提案する。

3.1 最適分割条件

データ並列画像合成 VR アルゴリズムでは、シアステージにおける負荷分散のために、部分ボリュームに含まれる不透明ボクセル数が均等になるようにボリューム分割が行なわれる。しかし、BS アルゴリズムに基づく画像合成時間をさらに短縮するためには、画像合成処理における中間画像の通信及び合成時間の短縮を目的とした分割を行なう必要がある。

画像合成時間を短縮するためには、BS アルゴリズムの各ステップにおいて通信および合成される画像領域の面積を小さくすれば良い。部分画像中、部分ボリュームの投影領域以外は合成の必要が無く、通信は行なわれない。このため、各部分ボリュームの投影領域が小さくなるようなボリューム分割を行なうことにより、BS アルゴリズムに基づく画像合成時間を短縮できると考えられる。ただし、ボリューム分割は前処理として静的に行なわれるため、任意の視線の向きに対して投影領域の面積が均一である必要がある。以上より、シアステージの負荷分散を保ち、画像合成時間を短縮するボリューム最適分割条件は、次のようにまとめられる。

1. 各部分ボリュームに含まれる不透明ボクセル数が均等
2. 各部分ボリュームの投影領域の面積が、任意の視線の向きに対して一様に小さい

本稿では、この条件を「最適分割条件」と呼ぶ。

3.2 適応分割手法

任意の視線の向きに対して投影領域が均一であるような形状は球であるが、球を隙間なく敷き詰めることは不可能である。ボリュームが三次元の直交格子であり、任意の視線の向きに対して部分ボリュームの奥行きに関する前後関係を容易に決定可能である必要性を考慮すると、最適分割条件を可能な限り満たすボリューム分割は、部分ボリューム全てが同じ大きさの立方体となるような分割である。しかし、ボリューム中の不透明ボクセル分布が一様でない場合、不透明ボクセル数が均等でかつ部分ボリューム形状が同じ大きさの立方体であるような分割方法は不可能であり、このため、任意のボリュームに対し、最適分割条件を完全に満たす分割方法は存在しないと考えられる。そこで、任意のボリューム対し可能な限り最適分割条件に近い分割、

すなわち、各部分ボリューム中の不透明ボクセル数が均等になり、部分ボリュームが立方体に近くかつ可能な限り同じ大きの直方体となるような分割手法を提案する。立方体に近い直方体とは、各辺の長さの分散が0に近い直方体であると定義する。

BSアルゴリズムの制約によりPE数は2^nであり、部分ボリュームの数が2^nとなるように分割を行なう必要がある。このため、提案する分割手法では、ボリューム座標系の主軸に垂直な平面、すなわちxy-、yz-、zx-平面のいずれかに平行な分割面による2分割をn回行なう。このようにして生成される直方体の形状は、2分割に使用する分割面の選択により変化する。そこで、分割の結果が最適分割条件に近付くように、各2分割における分割面を以下の手順に従い決定する。

1. 与えられた(部分)ボリュームを、不透明ボクセル数がほぼ均等になるように、xy-、yz-、zx-平面それぞれに平行な3つの分割面により3通りに分割する。
2. 3通りの分割に対し、得られた2つの部分ボリュームのうち立方体からより遠い形状である部分ボリュームのバウンディングボックスを求める。
3. 求めた3つのバウンディングボックスを比較し、立方体に最も近いバウンディングボックスを生成する分割面を選択する。

ここでバウンディングボックスとは、与えられたボリューム中の全ての不透明ボクセルを内含する最小の直方体である。図3に、二次元における分割面選択の例を示す。提案する分割手法では、上下に分割する分割面が選択される。図3から、選択した分割面による分割結果である部分ボリュームの方が、最適分割条件に近いことが分かる。

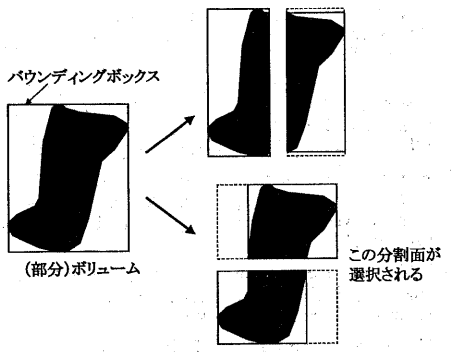

図3: 分割面の選択

4 性能評価

本節では、提案するボリューム分割手法に関する実験とその結果について述べる。次に、実験結果に基づき提案する手法の性能を評価する。

4.1 ボリューム分割実験

本研究では、提案する分割手法による画像合成時間短縮を評価するために、データ並列画像合成VRアルゴリズムを汎用並列計算機SP2へ実装し、実行時間を計測した。実験に使用したSP2は、66MHzで動作するIBM POWER2プロセッサを搭載した32台のPEがHPS(HighPerformance Switch)により接続された、分散メモリ型並列計算機である。HPSは双方向の多段クロスバ網であり、片方向最大40MB/secの転送能力を持つとともに、PEの増加に伴う転送能力の低下がないという特徴を有する。実装したBSアルゴリズムでは、部分画像の投影像を囲み各辺が画像の縦横に平行である最小の矩形領域内部についてのみ転送および合成が行なわれる。以後、この矩形領域を投影矩形領域と呼ぶ。

実験には、人体頭部のCT断層画像から生成した、解像度が256^3のボリュームを使用した。以後、このボリュームをCTheadと呼ぶ。この断層画像は、ノースキャロライナ記念病院により、Chapel Hillボリュームレンダリングテストデータセットとして提供されている。CTheadを、分割数=$2^1, 2^2, 2^3, 2^4, 2^5$の5通りに分割し、それぞれに対して、View1(1,0,0)、View2(1,1,0)、View3 (1,1,1)の3つの視線の向きによる並列レンダリングを行ない画像合成時間を計測した。View1はボリュームを正面から見る視線の向きであり、部分ボリュームの投影矩形領域は最小となる。一方、View3はボリュームを左斜め上から見下ろす視線の向きであり、部分ボリュームの投影矩形領域は最大となる。分割方法の違いによる画像合成時間の差異をみるために、分割方法として、提案する分割手法の他にスラブ分割手法を使用した。スラブ分割手法は、xy-、yz-、zx-分割面のうち、画像平面に最も平行に近い面のみを用いて2分割を行なう方法である。

4.2 実験結果および評価

図4に、スラブ分割手法および適応分割手法による画像合成時間を示す。スラブ分割を用いた場合、画像合成時間は、View1,2,3ともにPE数の増加に伴い増加し、ある一定値に漸近する傾向を持つ。この傾向は、A_{pix}がPE数に対して一定である場合の式(1)の傾向

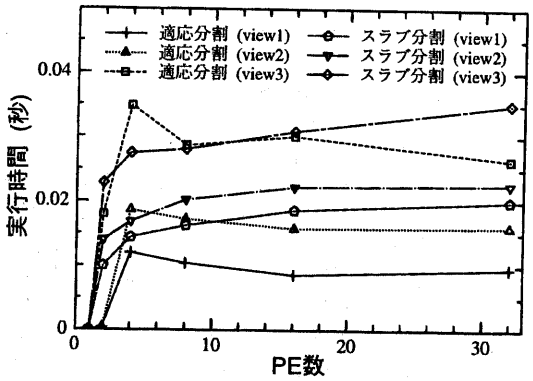

図 4: 画像合成時間

と同様であり、これは、スラブ分割の場合、分割数が増えても各部分ボリュームの投影矩形領域の面積が殆んど減少しないためであると考えられる。

適応分割を用いた場合、画像合成時間は View1,2,3 ともに PE 数の増加に伴い一旦増加した後に減少する。これは、PE が少ない場合には分割による投影矩形領域の面積減少が顕著ではなく式 (1) の示す画像合成時間増加が支配的であり、一方、PE が多数の場合には PE の増加に伴い投影矩形領域が小さくなるため、BS アルゴリズムの初期に PE 間で通信される画像面積が削減され画像合成時間が減少することが原因であると考えられる。このため、View1,2,3 ともに、ある PE 数を越えると適応分割を行なった場合の画像合成時間の方が、スラブ分割を行なった場合よりも短くなっている。例えば、View3 の場合には、4PE のときに適応分割による画像合成時間が最大となった後、PE 数が 8 を越えると適応分割による画像合成時間がスラブ分割による画像合成時間を下回り、その差は PE が増えるにつれ増大する。

視線の向きにより、投影矩形領域が変化するため、BS アルゴリズムにおける各ステップでの通信および合成時間も変化する。このため、図 4 に示されるように、PE 数が小数でボリューム分割による投影矩形領域の面積減少が顕著でない場合、視線の向きによってはスラブ分割を行なった方が適応分割よりも画像合成時間が短くなることもある。しかし、PE 数が多くなるにつれ、適応分割を行なった場合にはスラブ分割を行なうよりも部分ボリュームが遥かに小さくなり、任意の視線の向きに対して画像合成時間は短くなる。従って、提案する分割手法は、多数の PE を用いて高速にレンダリングを行なう際に有効であり、PE 数が多い程その有効性は増大するといえる。より多くの PE が必要となるようなより大きなボリュームを実時間でレンダリングする場合には、本手法により並列レンダリングのオーバーヘッドである PE 間画像合成時間を短く抑え、高い並列処理効率を達成可能であると考えられる。

5 まとめ

本稿では、部分画像における投影矩形領域を小さくし PE 間画像合成時間を短縮する、ボリューム適応分割手法を提案した。提案する手法を用いることにより、PE 数が多数の時に、スラブ分割を用いた場合よりも画像合成時間を短縮可能である。

今後の課題としては、本手法を使用した場合の並列レンダリング全体の性能評価が挙げられる。

参考文献

[1] A.Kaufman, "Volume Visualization," *IEEE* Computer Society Press, Los Alamitos, California, 1991.

[2] P.Lacroute and M.Levoy, "Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transformation," Comput. Graph. Annual Conference Series, pp.451-458, 1994.

[3] H.Pfister and A.Kaufman, "Cube-4 – A Scalable Architecture for Real-Time Volume Rendering," Proc. of the 1996 Symposium on Volume Visualization, pp.47-54, 1996.

[4] K.Sano, H.Kitajima, H.Kobayashi, and T.Nakamura, "Parallel Processing of the Shear-Warp Factorization with the Binary-Swap Method on a Distributed-Memory Multiprocessor System," Proc. of Parallel Rendering Symposium'97, pp.87-94, 1997.

[5] P.Lacroute and M.Levoy, "Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transformation," Proc. of SIGGRAPH'94, pp.451-5.558, 1994.

[6] M.B.Amin, A.Grama, and V.Singh, "Fast Volume Rendering Using an Efficient, Scalable Parallel Formulation of the Shear-Warp Algorithm," Proc. of Parallel Rendering Symposium'95, pp.7-14, 1995.

[7] K.Ma, J.S.Painter, C.D.Hansen, and M.F.Krogh, "A Data Distributed, Parallel Algorithm for Ray-Traced Volume Rendering," Proc. of Parallel Rendering Symposium'93, pp.15-22, 1993.

[8] S.Molnar, J.Eyles, and J.Poulton, "PixelFlow: high-speed rendering using image composition," Comput. Graph., 26:pp.231-240, 1992.

[9] M.Levoy, "Volume Rendering: Display of surface from volume data," *IEEE* Trans. Comput. Graph. and Appl., 8:pp.29-37, 1988.