多視点動画像処理による3次元モデル復元に基づく自由視点画像生成のオンライン化 : PCクラスタを用いた実現法

上田, 恵
九州大学システム情報科学研究院知能システム学部門

有田, 大作
九州大学システム情報科学研究院知能システム学部門

谷口, 倫一郎
九州大学システム情報科学研究院知能システム学部門

出版情報：情報処理学会論文誌. 46 (11), pp.2768-2778, 2005-11. 情報処理学会
バージョン：

多視点動画像処理による3次元モデル復元に基づく自由視点画像生成のオンライン化 —PCクラスタを用いた実現法—

上田 恵[†] 有田大作[††] 谷口倫一郎[††]

複数台のカメラによって撮影される映像から，3次元モデルを復元することで自由な視点からの映像を生成する研究が近年さかんに行われている．自由視点画像生成は処理に時間がかかり，オンライン処理が難しいため，その研究の多くは，できる限り正確な3次元モデルの復元をオフラインで行う，あるいは，陽には3次元形状を復元しないがオンラインで自由視点画像を生成する，というアプローチをとっている．本稿では，3次元形状復元による自由視点画像生成をオンラインで行う手法について述べる．PCクラスタを利用して多視点カメラ画像からの視体積交差法によって3次元形状を復元し，三角パッチで表現された3次元表面に，仮想視点位置とカメラ画像を考慮した色を付けることで，写実性の高い自由視点画像を生成する．このとき，表面への色付けを高速に行うことが難しかったが，本論文ではこの処理を高速に行うためのZバッファを利用した手法を提案する．また，生成された自由視点画像を示し，提案手法を評価する．

On-line free-viewpoint video generation based on a 3D model reconstructed from multi-viewpoint videos -Implementation on a PC-cluster-

Megumu Ueda,[†] Daisaku Arita[††] and Rin-ichiro Taniguchi[††]

Recently, there are a lot of researches for generating free-viewpoint videos by reconstructing 3D models from multiple camera images. Since it is difficult to generate free-viewpoint videos on-line for the large amount of computation, most of these researches aim to generate free-viewpoint videos off-line, or generate free-viewpoint videos without 3D model reconstruction. In this paper, we will propose a method that generates free-viewpoint videos by reconstructing 3D models on-line. The method first reconstructs 3D models by visual cone intersection method using multiple cameras, second colors the surfaces of 3D models in terms of triangular patch representation, and displays the colored models on a screen on-line. In these procedures, it is difficuld to color the surfaces. Then, we propose a new method for coloring, which is based on the Z-buffer method. And we show generated free-viewpoint images to estimate the method.

1. はじめに

近年，計算機の高性能化にともない，情報メディアのさらなる発展を目的とした研究が盛んに行われている．その一つとして自由視点画像生成が挙げられる．これは，複数の視点からの画像を基に任意の仮想視点からの画像を生成することである．金出らがVirtualized Realityのコンセプトを提案[1]して以来，自由視点画像生成に関するさまざまな研究が行われてきた．

自由視点画像生成の方法は，対象の3次元形状を復元するかどうかにより二つに分類することができる．対象の3次元形状を復元しない方法としては，Light Field[2]を利用した手法が典型的である．これは空間中のすべての地点におけるすべての光線を求めることによって，任意の仮想視点からの画像を生成することが可能になるという手法である．この手法は高精度な自由視点画像を生成することが可能であるが，そのためには対象形状の複雑さに対して十分な数の実画像が与えられる必要があるため，その手間と計算量は膨大なものとなり，実時間性に欠けるという欠点がある．一方，対象の3次元形状を視体積交差法[3]などによって復元し，それに色付けすることによってコンピュー

† 九州大学大学院システム情報科学府
Department of Intelligent Systems, Kyushu University
†† 九州大学大学院システム情報科学研究院
Department of Intelligent Systems, Kyushu University

タグラフィックスにおける対象モデルを構築し，これにより自由視点画像を生成する手法も数多く提案されている[4)5)6)]．これら Light Field を使わない手法は少ない実画像から自由視点画像を生成することが可能であるが，画像数が少ないために色の再現性が劣るという欠点がある．

これらの点を踏まえ，本研究では自由視点画像のオンライン配信を目指し，自由視点画像を実時間オンライン生成できるよう，実時間処理が可能である 3 次元形状を復元する手法を採用した．

3 次形状を復元する手法においても，明示的に復元するか，復元しないかの 2 つに分類できる．明示的に形状を復元しない手法として，Matusik ら[7)] は仮想視点からの光線を実カメラの画像平面へ投影することにより，仮想視点から見えるべき映像を計算する手法を提案している．この手法は処理量が少ないので高速に計算できるだけでなく，生成画像の座標系において処理を行うため再量子化による画像精度の低下が少ないという利点がある．しかし，生成画像の座標系において処理を行うことから，異なる仮想視点からの画像を生成するためには仮想視点数と同じ数の自由視点画像生成システムが必要になってしまう．つまり，仮想視点を操作するユーザが増えるごとに計算量が増えてしまうので，それぞれのユーザが仮想視点の位置を制御できるような自由視点画像のオンライン配信が難しくなる．よって，本研究では自由視点画像のオンライン配信を目指しているので，各ユーザが好みの仮想視点を選択することが可能である明示的に 3 次元形状を復元する手法を採用した．また，明示的に形状を復元することにより，物理シミュレーション等を利用した様々な実時間の映像処理が可能になる．例えば，形状が既知であれば仮想空間内での操作も可能となるので，撮影状況と異なった仮想環境（異なった光源や他の仮想 3 次元物体）での視覚化も可能であり，単にそのまま観るだけでない，新しい映像の生成も可能になる．

3 次元形状を復元する手法による自由視点画像生成を実現するには，対象物の形状情報と色情報の獲得および仮想視点からの画像の生成が必要であるが，本研究ではこの処理をオンラインで実現する．これにより，自由視点映像のライブ配信が可能となり，またユーザは対話的に仮想視点を制御することが可能となる．

従来研究でも形状情報の獲得まではオンラインで可能であった[8)9)10)]．しかし，メッシュ表現の形状に色を付けるまでオンラインで行うのは難しかった．その原因の一つとして，色付けを行うためには実カメラに対して形状の各頂点の可視判定，つまりカメラからの光線を遮蔽する物体があるかどうかを計算する必要があり，この計算量が膨大になるという点が挙げられる．M. Gross ら[11)] や O. Grau ら[12)] は配信まで目指した大規模なシステムを構築し，カメラ画像の取得から自由視点画像配信までの一連の処理をオンラインで実現しているが，色付けの際に各頂点の可視判定を行なっていない．そのため，仮想視点位置によっては誤った色付けが行なわれている．また，岩舘ら[13)] らは，形状頂点をカメラに向かって光学直線上に変位させ，各カメラ画像に逆投影し，全てのカメラ画像のシルエット内部に投影されればその頂点は不可視と判定する，という処理を繰り返し行なうことにより可視判定する手法を提案している．しかし，この手法ではどのくらい変位させ，どのくらい繰り返すのかが経験に基づいており，また，繰り返し計算を行なうので計算時間がかかってしまう．

そこで本研究では，高速な対象物表面の可視判定手法を提案することにより，従来は難しかったオンラインでの可視判定を実現させ，さらに PC クラスタによる並列処理を行うことで高速化を図る．これにより，従来は難しかったオンラインでの色情報獲得，及び獲得した 3 次元モデルの表示を実現した．本稿では，PC クラスタを用いた並列システムの概要および具体的な自由視点画像生成手法について説明する．さらに実験により，オンラインで自由視点画像生成が可能なことを示す．

2. オンライン自由視点画像生成システムの概要

本章では提案するシステムの概要を述べる．2.1 節では自由視点画像を生成する処理の流れを説明し，2.2 節では PC クラスタを用いた並列処理について説明し，2.3 節では提案するシステムの構成を説明する．

2.1 処理概要

自由視点画像の概要は以下の通りである．

(1) カメラ画像の取得
(2) カメラ画像からの対象物体抽出
(3) 各カメラ画像から抽出した対象物体領域を用いた 3 次元形状の復元
(4) カメラ画像から 3 次元形状の色情報を取得
(5) 各カメラ画像から取得した色情報の統合
(6) 自由視点画像の表示

以上の処理を PC クラスタを用いて並列に行なう．

2.2 PC クラスタを用いた並列処理

実時間での自由視点画像生成には上述のように多くの処理を必要とする．そこで分散並列計算機の一

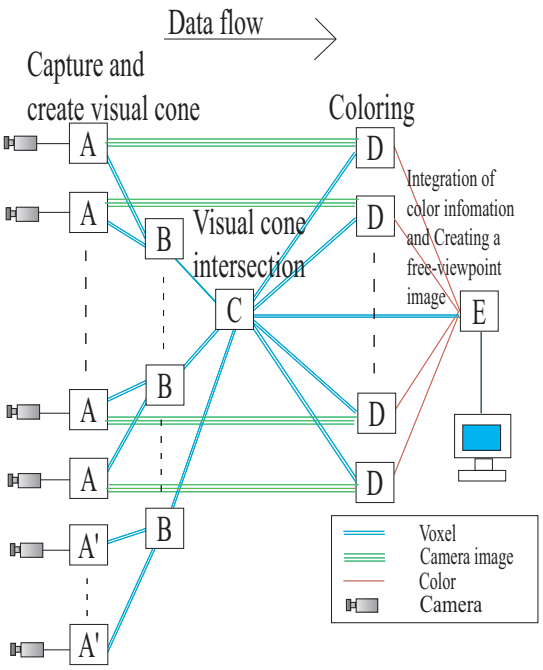


図 1 本研究のシステム構成
Fig. 1 System configuration

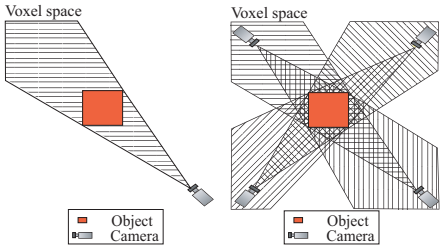
種である PC クラスタ，および PC クラスタをプラットフォームとする実時間並列画像処理システムのためのプログラミングツール RPV （Real-time Parallel Vision）[14] を用いて，オンラインで並列画像処理を行う．

ここで採用した並列処理の基本的な手法は，3 次元空間の分割処理ではなく，カメラ毎の並列処理である．すなわち，前段では各ノードはそれぞれ別のカメラについての処理を行ない，後段で各カメラから得られた情報の統合を行う．これは，空間を分割して並列処理を行なうと各 PC での処理量に偏りが生じやすいためである．カメラ間の並列処理であれば，視体積交差の処理量は同一であり，また，カメラから見える対象の表面積は基本的にはあまり大きな差がないので，色情報取得の処理量のばらつきも少ないという利点がある．

2.3 システム構成

図 1 に示すように本システムは 5 段階のノード PC からなり，処理の順番に，カメラ画像取得と対象抽出（ノード **A**，**A'**），2 段階に別けた形状復元の 1 段目（ノード **B**），形状復元の完了と形状表面の生成（ノード **C**），形状表面の色情報の取得（ノード **D**），色情報統合と仮想視点位置からの映像生成（ノード **E**）となっている．詳細は以下の通りである．

カメラ画像取得と対象物体抽出（ノード A，A'）：



(a) 視体積の構築 (b) 視体積の交差

図 2 視体積交差法
Fig. 2 Visual cone intersection: (a) Visual cone construction, (b) Visual cone intersection.

カメラ画像からの対象物体抽出は背景差分を用いて行う．ノード **A** と **A'** の違いは，カメラ画像を色付けに使用するかどうかである．色付けに多くのカメラを使いすぎても，あまり精度に寄与しないと考え，処理の高速化のために使用するカメラ画像を制限した．

形状復元（ノード A，A'，B，C）： 対象物体のシルエットを 3 次元空間に投影することによって，視体積を獲得する（**図 2**(a)）．視体積とは，カメラ視点を頂点，シルエットを断面とする錐体のことであり，対象物体は必ずこの内部に存在する．本研究ではボクセルによって視体積を表現する．☆

ノード **A**，**A'** で各カメラからの視体積を構築し，その結果を用いてノード **B**，**C** の二段階で視体積交差を行なう（図 2(b)）．ノード **B**，**C** の二段階に分割する理由は，一回で全ての視体積の交差を求めるためには一度に全てのカメラからの視体積を受信しなければならないため，受信するデータサイズが大きくなり，受信に時間がかかるためである．

復元した形状に対してノード **C** で離散マーチング・キューブ法[15] を施し三角パッチ表現へ変換する．ここで，単純にそのまま三角パッチ表現のままで送信すると，三角パッチ表現のデータ量が多いために送受信の時間が大きくなってしまう．そこで，ノード **C** は三角パッチに変換されるボクセルの座標と，対応する三角パッチの生成パタンのみを送信し，そのデータを受信するノード **D**

☆ 視体積のボクセル表現とは，3 次元空間を立方体で標本化し，1 ビット 1 ボクセルとして，立方体が視体積に含まれるか含まれないかの 0/1 のデータ表現である．

（色情報取得ノード），ノード **E**（色情報統合と映像生成ノード）で三角パッチを再構築する．

色情報取得（ノード D）： ノード **A** から送られてきたカメラ画像と，ノード **C** から送られてきた復元した形状情報を基に，対象形状の色情報を提案手法により取得する．色情報取得についての詳細は 3.2 節で述べる．

色情報の統合と自由視点映像生成（ノード E）： ノード **C** から送られてきた形状情報とノード **D** から送られてきた各カメラからの色情報を基に，仮想視点位置に応じて色情報を統合し，自由視点画像を生成する．色情報の統合についての詳細は 3.3 節で述べる．また，自由視点画像の表示には OpenGL を用いる．OpenGL は，ハードウェアや OS には依存しない 3 次元グラフィックスのためのプログラミングインタフェースである．

また，本システムでは RPV が提供するストリーミング処理を利用して遅延の削減を図っている[14]．ストリーミング処理とは，1 フレームのデータをいくつかに分割し，分割単位毎に処理が終わったらデータを送信し，受信側は受け取ったデータから処理を行うような処理のことである．このようにすることにより，1 つのフレームに対する処理を異なるノードである程度オーバーラップして実行できるので，システム全体での遅延時間が削減できる．しかし，ノード **D** における色情報獲得の処理の際，全ての三角パッチが揃わないと三角パッチの可視判定ができないため，ノード **C** 以降のデータの流れは，フレーム単位で処理を行なっている．

3. 色 付 け

本章ではノード **D**，**E** における，三角パッチデータへの色付けについて述べる．ノード **D** では形状の色情報を受信したカメラ画像と三角パッチから取得し，ノード **E** では各三角パッチについて各カメラからの色情報を統合する．

3.1 色付けの方針

対象物体形状への色付けの従来手法は，

(1) カメラ位置と対象物体表面方向の関係を利用する手法

(2) カメラ位置と仮想視点位置の関係を利用する手法

の二つに分類できる．文献 4) で実際に上記 2 つの方法を比較しており，後者の手法の方が写実性が増すという結果が得られているので，本研究では後者の手法を採用した．具体的には，まず，三角パッチの頂点に仮想視点位置に応じた色を付け，表示時には各三角パッチにその三つの頂点の色を補間した色を塗る．これによって，三角パッチの境界に色のギャップが生じない色付けを行うことができる．ただし，この手法では，原画像の画素に対して三角パッチが大き過ぎると，三角パッチの色解像度が足りず，生成画像の色の精度が低くなってしまう．これを防ぐためにボクセルの解像度を上げると処理時間が劇的に増加してしまう．そこで**図 3** に示すように，1 個の三角パッチを 6 個の三角パッチに分割する[4]．これにより，ボクセル解像度を上げることなく三角パッチを小さくすることができ，生成画像の色の精度を上げることができる．また，視体積交差法で復元した形状の中にはカメラから見えない頂点も存在するが（例えば股下や足の裏など），そのような頂点については，隣接する三角パッチの頂点全ての平均の色を付ける．こうすることで，どのカメラからも見えない頂点にもある程度自然な色を付けることが可能となる．

以下，3.2 節でノード D における色情報取得処理，3.3 節でノード E における色情報統合処理について述べる．

3.2 単一画像からの色情報の取得

三角パッチの各頂点に対して，それを画像に投影した位置の画素の色を付ける．この処理は，頂点と画素の対応に関するテーブルを事前に作成しておくことによって高速に実現できる．ただしこのとき，その視点から頂点が見えていないときは色を付けてはいけないので，頂点が可視かどうかを判定する必要がある．一般的には，各頂点に対して，すべての三角パッチについて遮蔽されていないかどうかを調べなければならない．この処理をすべての頂点について行わなければならないので，その計算量は，頂点の数を n とすると，三角パッチの数は n に比例すると考えることができるので $O(n^2)$ となってしまい，高速に実行することは不可能である．

そこで，本研究では新たな手法を提案する．この手法は Z バッファ法を利用したものである．通常の Z バッファ法は，画像に対して最も手前にある対象物体表面上の点（ここでは頂点）を選ぶことにより可視の頂点の色を画素に付ける手法である．つまり，各画素に対して一つの頂点を対応付けることになる．一方，頂点への色付けは，可視の各頂点に対して一つの画素を対応付けなければならない．提案手法では，通常は画素から頂点への対応を求めるために利用する Z バッファ法を，逆に頂点から画素への対応を求める問題で利用する．しかし，一つの画素に対して複数の頂点が

対応する可能性があるため，通常の Z バッファ法によって頂点と画素の対応をとると，一つの画素に対して一つの頂点しか対応しないため，それ以外の頂点にはたとえその頂点が可視であっても色が付かないことになってしまう．そこで本研究では，まず Z バッファを利用した手法により可視の頂点であるための条件を求め，次にその条件を満たす頂点に対して対応する画素の色を付ける．ここでは，不可視な頂点の次のような特徴を利用する．

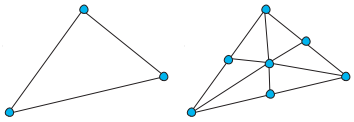
特徴 1: カメラの方向を向いていない三角パッチは不可視である．

特徴 2: 上述のような三角パッチによって遮断される三角パッチの頂点も不可視である．

この遮断の有無を Z バッファ法を利用して求める．手法の処理の流れは**図 4** のようになる．具体的な手法は以下のようになる．

- **前準備:** Z バッファには無限大と見なせる数値を入れておく．また，可視三角パッチリストは空にする．
- **処理 1:** 処理 1 では前述の特徴 1 に当てはまる三角パッチを求め，そのような三角パッチを Z バッファに保存する（**図 5**(a)）．具体的には，まず三角パッチの法線方向☆を求める．そして，三角パッチの法線とカメラの視線方向のベクトルとの内積を求める．内積が正となる三角パッチ（つまり，カメラの方向を向いていない三角パッチ）はカメラから見えないと判断し，頂点には色を付けずに，カメラからの距離値を Z バッファに保存する．内積が負となる三角パッチ（つまり，カメラの方向を向いている面）は可視三角パッチリストに保存する．
- **処理 2:** 処理 2 では前述の特徴 2 に当てはまらない頂点を求め，そのような頂点を可視と判定する（図 5(b)）．具体的には，処理 1 によって可視三角パッチリストに保存された三角パッチについて，Z バッファの値よりもカメラからの距離が近い場合，その頂点に対応する画素の色を付ける．ここで，Z バッファには処理 1 により，カメラの方向を向いていない三角パッチの内でその画素に投影される最も近い三角パッチの距離値が保存されているので，Z バッファの値よりも距離値が遠い頂点は前述の特徴 2 に当てはまる頂点ということになる．

☆ 離散マーチング・キューブ法で得られる三角パッチは面の向きが考慮されており，面の向きが物体外部になるようになっている．三角パッチの頂点データを物体外部（又は内部）から見て右回りか左回りに統一しておけば，同一三角パッチの辺同士で外積を求めることにより三角パッチの法線が求まる．



(a) 通常の三角パッチ　(b) 高精度な色付けのため分割された三角パッチ

図 3 三角パッチの分割
Fig. 3 Dividing triangular patch: (a) Original triangular patch, (b) Divided triangular patch.

この処理を行うことで，カメラ視点から可視である頂点の色情報を取得することができる．この処理はすべての頂点を多くとも 2 回走査するだけなので，計算量は $O(n)$ となり，大幅に高速化される．

3.3 複数画像からの色情報の統合

各カメラ画像から取得した色情報について重み付き平均を求めることにより統合し，頂点に色を付ける．重みは仮想視点と実カメラの角度ではなく，その方向のみに応じて決める．ある頂点が N 個のカメラから可視である場合，仮想視点から頂点へのベクトルとカメラ $n(1 \leq n \leq N)$ から頂点へのベクトルとがなす角度を θ_n とする（**図 6**）．その中のカメラ n から得られる色の重み W_n は，θ_n のみに依存すると考え，

$$W_n = \frac{(cos\theta_n + 1)^\alpha}{\sum_{k=0}^{N}(cos\theta_k + 1)^\alpha}$$

として求める．括弧内を $cos\theta$ のみにすると色の重みが負の値をとることがあるので $cos\theta + 1$ としている．このようにすることで仮想視点と方向が近いカメラからの色が優先され，遠いカメラの色ほど優先度が下がることになる．また，α の値の分だけ指数倍しているので，仮想視点とカメラの方向が離れているときの色付け精度はほぼ変わらないが，仮想視点とあるカメラの方向がほぼ同じときには，あるカメラの重みが他のカメラに比べて十分に大きくなり，その結果，よりカメラ画像と近い色となり，色付け精度が上がると考えられる．

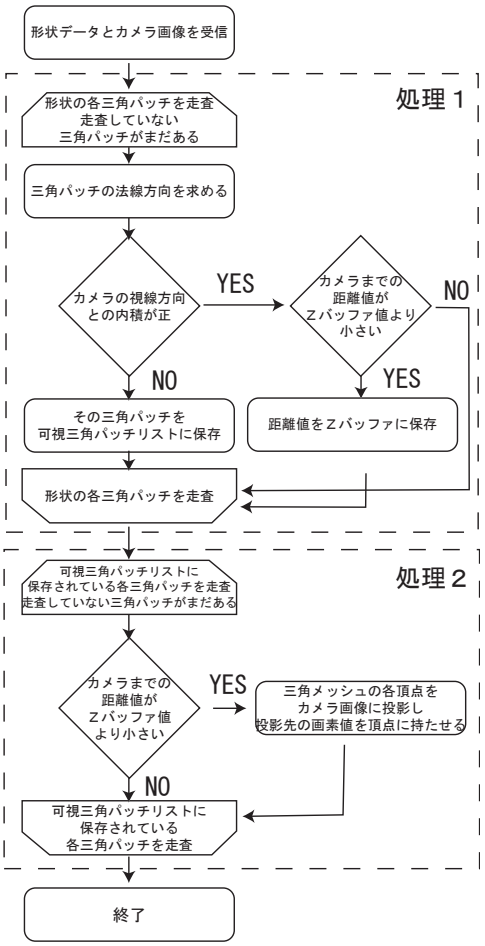
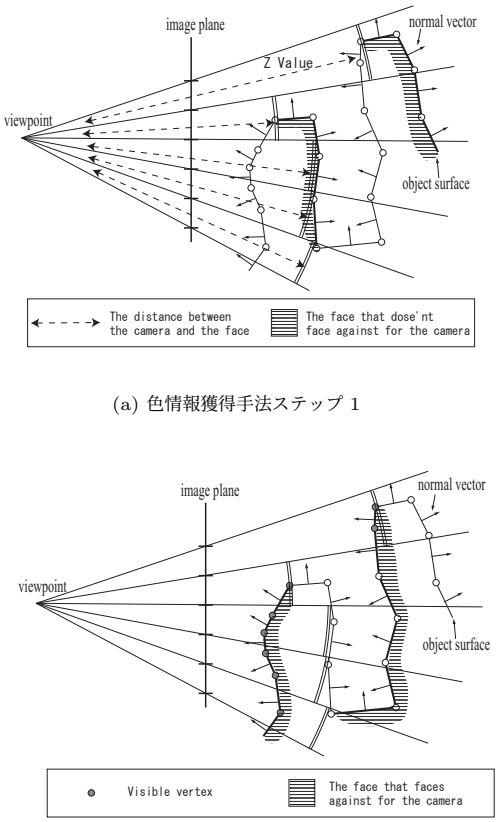


図 4 色情報獲得手法の処理の流れ
Fig. 4 Flowchart of vertex coloring



(a) 色情報獲得手法ステップ 1

(b) 色情報獲得手法ステップ 2

図 5 色情報獲得手法
Fig. 5 Vertex coloring: (a) Step1, (b) Step2.

4. 実験と考察

4.1 実　　験

本手法を用いてオンラインで自由視点画像を生成し，その処理時間，遅延時間，通信量，色付け誤差を計測した．

本実験では合計 17 台の PC を利用した．具体的には，ノード **A** が 6 台，ノード **A'** が 1 台，ノード **B** が 2 台，ノード **C** が 1 台，ノード **D** が 6 台，ノード **E** が 1 台とした．各 PC はスイッチ型ギガビット LAN の一つである Myrinet によって相互に結合されており，1Gb/sec の通信が可能である．さらに 7 台の IEEE1394 ディジタルカメラ[16] が接続されており，全てのカメラは同期信号発生装置により同期がとられている．**図 7** にカメラ配置を示す．

ノード **A'** の一つには天井カメラを用いた．全てのカメラが上方から見下ろしているので，天井カメラは色付けにあまり寄与しないと考えたからである．また，カメラは予めキャリブレーションしておいたものを使用した．カメラキャリブレーション手法としては，レンズ歪みを考慮した Tsai の手法[17] を利用した．カメラ画像の解像度は 320×240 で，空間解像度は $128 \times 128 \times 128$，ボクセルの一辺を $2cm$ として実験を行った．実験で使用した PC の性能を**表 1** に示す．

表 1 PC の性能
Table 1 Performance of PC

OS	Red Hat Linux9
CPU	Intel Pentium4 3GHz
メモリ	1GB
コンパイラ	gcc-3.2.2

実験には文字の書いてある服装と，縞模様のある服装を使用した．縞模様の間隔はボクセル表現の量子化間隔よりも広い．また，色付けの誤差は縞模様の服装

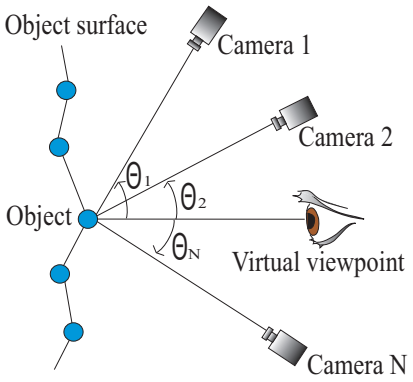


図 6 色情報の重みの付け方
Fig. 6 Angle between camera and virtual viewpoint

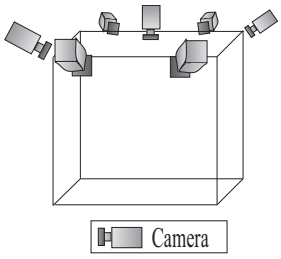


図 7 本実験のカメラ配置
Fig. 7 Camera arrangement

において計測した．オンラインで処理を行うことは可能であるが，評価実験においては先に述べた測定を同じカメラ画像に対して行うために，保存しておいたカメラ画像を入力としてオフラインで実験を行った☆．このとき保存しておいた画像に対するオフライン処理は全く施していない．したがって，計算処理および画像データは実際にオンラインで処理したものと同等と考えてよい．

また，3.3 節の色情報の重みにおいて述べた α の値は，処理速度を優先するため実験では $\alpha = 1$ とした．

4.2 自由視点画像についての考察

図 8，**図 9** に原画像と同じ視点から生成された自由視点画像を，**図 10** に実際にはカメラのない視点からの自由視点画像を示す．図 10 の周りの小さな立方体はカメラ位置を表している．

図 8，図 9，図 10 を見ると，服の模様が再現されていることがわかる．また，体の一部が欠けている生成画像があるが，これは周りに暗幕があり，床にも黒い布を敷いているため，服の影の部分が背景と誤認識され背景差分に失敗し，対象領域を抽出できなかったためである．

原画像（図 8(a)，(c)，図 9(上)）と比べても，原画像と同じ視点から生成された自由視点画像（図 8(b)，(d)，図 9(下)）はほぼ遜色がないように見える．これらの結果から主観的に評価すると

- ほぼ忠実に動きが再現されている
- 形状が多少不正確であるが 320 × 240 の解像度のカメラ画像とほぼ遜色ないように見える

という評価ができた．

以上より，自由視点画像の生成はほぼ実現されているが，より精細な自由視点画像を生成するためには形状復元の高精度化，背景差分の強化が必要であるということがわかった．

4.3 色付け誤差についての考察

測定した色付け誤差を，**図 11**(a) に，その平均と分散を**表 2** に示す．図 11(a) において，cam7 は天井カメラであり，cam8，cam9 は今回の実験では使用していないカメラである．実験に使用したカメラと使用していないカメラとで色付け誤差に差があるか比べるために，cam8 と cam9 もグラフに載せた．

色付け誤差の測定は，色情報を統合した結果の三角パッチ頂点を各カメラ画像に投影し，RGB 値の 2 乗平均平方根誤差を各画素について求め，その平均値を各カメラ画像ごとに求めた．その際，投影された三角パッチ頂点間には線形補間した RGB 値を用い，その誤差を求めた．

色付け誤差の原因は，以下の五つが考えられる．

(1) 視体積交差法により復元した形状の誤差のため
(2) キャリブレーションの誤差のため
(3) 細かい模様をスムージングによって表現しようとするため
(4) 重み付き色情報の統合による誤差のため
(5) カメラの個体差による色の違い

(3) については，三角パッチよりも小さい模様は原理的に再現不可能であり，そのような模様は描画時に OpenGL の機能で三角パッチ頂点間を滑らかに補間するために誤差が生じる．また，色付けに使用したカメラとの誤差と比べて色付けに使用していないカメラとの誤差が 2 倍程度になっていることがわかる．このことから，提案手法の効果により，カメラ視点と仮想視点の方向が近いときには生成画像の精度が高くなることが確かめられた．

4.4 処理時間についての考察

処理時間の測定には，送信完了待ち，受信待ちの時間を除いた，各ノード特有の処理にかかった時間を計

☆ 実験をオフラインで行なった理由は，利用したカメラは同期させると 15fps 以上で撮影することができないからである．

(a) 原画像 1　(b) 生成画像 1

(c) 原画像 2　(d) 生成画像 2

図 8　原画像と生成された自由視点画像
Fig. 8　Camera iamges(left) and generated images(right)

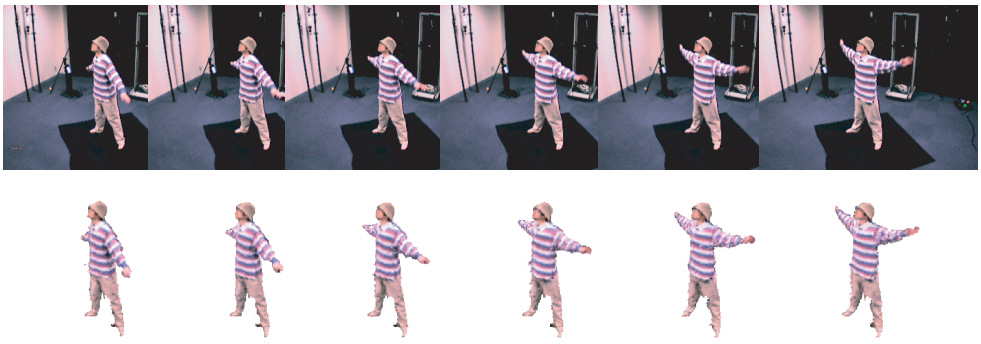


図 9　(上) 原画像と (下) 生成された自由視点画像
Fig. 9　Camera images(upper) and generated images(lower)

測した．

図 11(b)(c) に処理時間を示す．平均では 20fps 程度の速度での処理が可能であった．これは実用十分な速度が得られていると言える．しかし図 11(b) から，処理速度は安定していないことがわかる．これは対象物体の大きさや形によって三角パッチの数が変わるからである．

4.5　遅延時間についての考察

遅延時間は，全 PC の内部時計は ntp を利用することにより一致していると仮定して，カメラ画像が入力

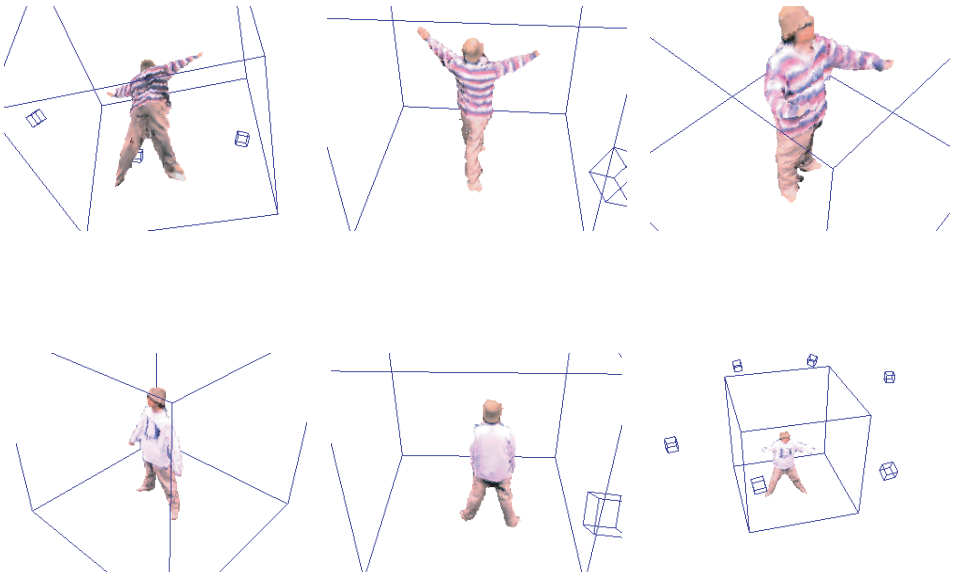


図 10 仮想視点からの生成画像
Fig. 10 Generated virtual-viewpoint images

表 2 色付けの 2 乗平均平方根誤差の平均と分散
Table 2 Average and variance of r.m.s. errors of coloring

カメラ	色付けに使用	色付けに未使用
平均	30.43	57.00
分散	85.29	294.8

表 3 各ノードの 1 フレーム辺りの平均送信量
Table 3 Amount of data transfer of each node

ノード	平均 (Kbyte)
A (Image)	93.5 (variable)
A and B (voxel)	256.0 (constant)
C	28.6 (variable)
D	92.0 (variable)

された時刻と自由視点画像を生成した時刻との差を計算することによって求めた．

図 11(d) に遅延時間を示す．主観的にはまだ遅延を感じるので，本システムを人同士のインタラクションなどに使う場合には，さらなる改善が必要である．ノード **C** 以降ではストリーミング処理を行なっていないので，ストリーミング処理化を導入できれば遅延時間を削減できると思われる．

4.6 通信量についての考察

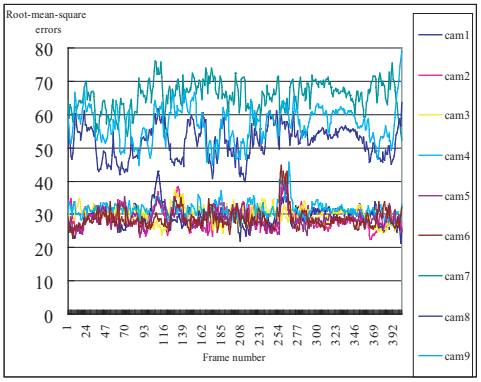
表 3 に各ノードからの 1 フレームあたりのデータ送信量を示す．

表 3 から，今回の実験ではノード **B**，**C** の受信するデータ量が全ノード中で最大であり，受信するデータ量の合計は，1 フレーム平均で 768 キロバイト必要であることがわかる．4.1 節で述べた Myrinet の性能から通信時間を計算すると，ノード **B**，**C** が 1 フレーム分のデータを受信するには 6.1 ミリ秒程度必要である．この時間がシステムのスループットに影響を与えていると考えられる．実際に，各ノードの平均処理時間 (図 11(b)(c)) からスループットを計算すると理論的には 20fps 以上の処理速度がでるはずである．つまり，スループットが理論値より遅くなっている原因の一つとして，送受信にかかる時間が考えられる．高速化や自由視点画像のライブ配信などといったシステムの応用を考えるとデータを圧縮する必要があると考えられる．

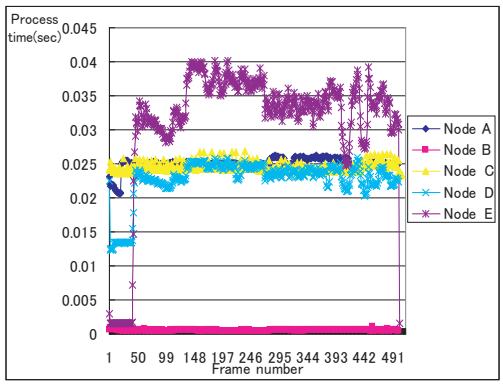
5. おわりに

本論文では，オンラインで実現可能な対象形状表面の可視判定手法を提案した．さらに PC クラスタを利用することにより，多視点画像からのオンライン自由視点画像生成の手法を提案した．また，実験では 20fps 程度の処理速度で自由視点画像を生成することができ，オンラインでの自由視点画像生成が可能なことが確かめられた．今後の課題としては以下が挙げられる．

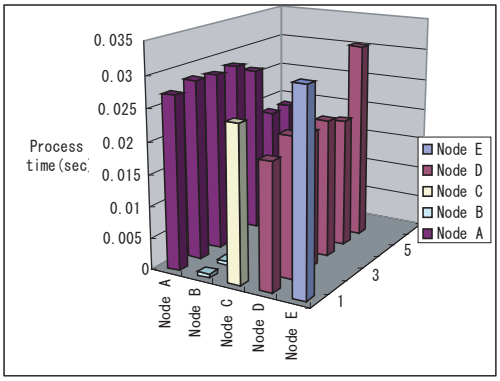
- **処理速度の安定化:**



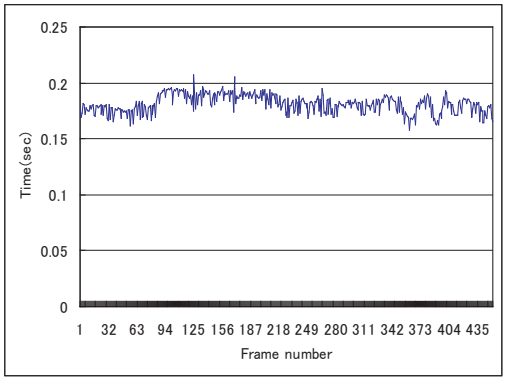
(a) カメラ画像との色の誤差



(b) 各パイプライン毎の平均処理時間



(c) 各ノードの平均処理時間



(d) 遅延時間

図 11 計測結果
Fig. 11 Measurement result: (a) Coloring error, (b) Processing time from each pipeline , (c) Processing time from each node, (d) Latency.

現在のシステムでは処理速度が安定していないので安定させなければならない．例えば対象物体の大きさによってボクセル解像度を可変にすることにより安定したスループットで自由視点画像を生成することが考えられる．

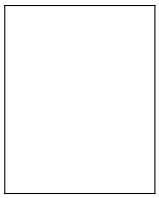
- **形状復元の高精度化:**
  形状復元を高精度化することにより，より精細な映像を生成する
- **ノード A' の動的選択：**
  色付けには使用しないノード **A'** を動的に選択する．現在はノード **A'** は最初に決定している．それは仮想視点位置で動的に選択する方法も考えられるが，自由視点映像の配信を考えた場合，仮想視点位置が複数存在することも考えられ，選択的に決めることが難しくなってくるからである．動的に選択することにより，より効率良く色付けが行われると考えられる．

参 考 文 献

1) Kanade, T., Rander, P. W. and Narayanan, P. J.: Concepts and early results, *IEEE Workshop on the Representation of Visual Scenes*, pp. 69–76 (1995).
2) Levoy, M. and Hanrahan, P.: Light field rendering, *Proc. of SIGGRAPH*, pp. 31–32 (1996).
3) Martin, W. N. and Aggarwal, J. K.: Volumetric Description of Objects from Multiple Views, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 5, No. 2, pp. 150–158 (1983).

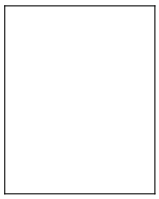
4) 高井勇志, 松山隆司:3 次元ビデオ映像の高精細表示アルゴリズムと編集システム, 映像情報メディア学会誌, Vol. 56, No. 4, pp. 593–602 (2002).
5) 斉藤英雄, 木村 誠, 矢口悟志, 稲木奈穂:射影幾何に基づく多視点カメラの中間視点映像生成, 情報処理学会論文誌, Vol. 43, No. SIG 11(CVIM 5), pp. 21–32 (2002).
6) Goldlucke, B. and Magnor, M.: Real-Time Microfacet Billboarding for Free-Viewpoint Video Rendering, *Proc. IEEE International Conference on Image Processing (ICIP '03)*, Vol. 3, pp. 713–716 (2003).
7) Matusik, W., Buehler, C., R.Raskar, Gortler, S. and McMilla, L.: Image-Based Visual Hulls, *Proc. of SIGGRAPH*, pp. 369–374 (2000).
8) ウ将軍, 和田俊和, 東海彰吾, 松山隆司:平面投資投影を用いた並列視体積交差法, 情報処理学会 CVIM 研究会論文誌, Vol. 42, No. SIG 6(CVIM 2), pp. 33–43 (2001).
9) 浜崎省吾, 吉田裕之, 重永信一:多視点シルエット画像からの高速な 3 次元形状復元手法, 第 7 回画像センシングシンポジウム講演論文集, pp. 59–64 (2001).
10) Cheung, G. K. M., Kanade, T., Bouguet, J. Y. and Holler, M.: A real time system for robust 3D voxel reconstruction of human motions, *Proc. CVPR2000*, Vol. 2 (2000).
11) Gross, M., Wurmlin, S., Naef, M., Lamboray, E., C.Spagno, A.Kunz, Koller-Meier, E., Gool, T. S. L. V., Lang, S., Strehlke, K., Moere, A. V. and O, S.: blue-c: A Spatially Immersive Display and 3D Video Portal for Telepresence, *Proceedings of ACM SIGGRAPH2003*, pp.819–827 (2003).
12) Grau, O., Pullen, T. and Thomas, G. A.: A Combined Studio Production System for 3D Capturing of Live Action and Immersive Actor Feedback, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 14, No. 3, pp. 370–380 (2004).
13) 富山仁博, 折原 豊, 岩舘祐一:多視点画像を用いた高精細 3 次元モデルの高速生成手法, 画像の理解・認識シンポジウム (MIRU2004), pp. 85–90 (2004).
14) 有田大作, 花田武彦, 谷口倫一郎:分散並列計算機による実時間ビジョン, 情報処理学会論文誌, Vol. 143, No. No. SIG 11(CVIM5), pp. 1–10 (2002).
15) 剣持雪子, 小谷一孔, 井宮 淳:点の連結性を考慮したマーチング・キューブ法, 電子情報通信学会技術研究報告, pp. 197–204 (1999).
16) 吉本廣雅, 有田大作, 谷口倫一郎:1394 カメラを利用した多視点動画像獲得環境, 第 6 回画像センシングシンポジウム講演論文集, pp. 285–290 (2000).
17) Tsai, R. Y.: A Versatile Camera Calibration Technique forHigh-Accuracy 3D Machine Vision Metrology UsingOff-the-Shelf TV Cameras and Lenses, *IEEE Trans. on Robotics and Automation*, Vol. 3, No. 4, pp. 323–344 (1987).

(平成 16 年 9 月 13 日受付)
(平成 17 年 9 月 5 日採録)



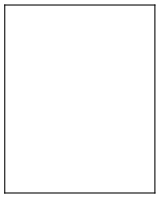
上田 恵

平成 16 年九州大学工学部電気情報工学科卒業. 同年九州大学大学院システム情報科学府知能システム工学専攻修士課程入学, 現在に至る. 主にコンピュータビジョンに関する研究に従事.



有田 大作(正会員)

平成 4 年京都大学工学部情報工学科卒業. 平成 10 年九州大学大学院システム情報科学研究科博士後期課程修了. 同年, 同(現大学院システム情報科学研究院)助手. 博士(工学). 文書画像処理, 画像処理における知識獲得, 実時間並列画像処理, 会話情報学の研究に従事. 電子情報通信学会, 映像情報メディア学会各会員.



谷口倫一郎(正会員)

昭和 53 年九州大学工学部情報工学科卒業. 昭和 55 年九州大学大学院工学研究科修士課程修了. 同年九州大学大学院総合理工学研究科助手. 平成元年同助教授. 平成 8 年九州大学大学院システム情報科学研究科(現大学院システム情報科学研究院)教授. 工学博士. 画像処理, コンピュータビジョン, ヒューマンインタフェース, 並列処理等に関する研究に従事. 本会論文賞(1993), 同坂井記念特別賞(1995)を受賞.