実時間自由視点映像生成のフレームレート安定化 形状復元の多重解像度処理

鍋嶋, 累
九州大学システム情報科学研究院知能システム学部門

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：画像の認識・理解シンポジウム, pp.642-647, 2006-07
バージョン：
権利関係：

実時間自由視点映像生成のフレームレート安定化

形状復元の多重解像度処理

鍋嶋　累† 上田　恵† 有田　大作† 谷口倫一郎†

† 九州大学大学院システム情報科学府　〒 816–8580 福岡県春日市春日公園 6–1
E-mail: †{nabeshima,ueda,arita,rin}@limu.is.kyushu-u.ac.jp

あらまし　本研究では, 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実時間で生成することを目指している. この自由視点映像生成システムの処理は, 視体積交差法を用いた形状復元 (ボクセル表現), ボクセル表現から三角パッチ表現への変換, 三角パッチの色付けの大きく 3 段階に分かれる. しかしこのシステムには, 対象物体の表面積が増加すると三角パッチの数が増加するため, 処理速度が低下してしまうという問題点があった. そこで, 本発表では形状復元時に空間解像度を対象にあわせて変化させ, 多重解像度処理を行うことで, フレームレートを安定化させる手法を提案する. 具体的には, 8 分木を利用した視体積交差法によって空間解像度を徐々に上げながら形状復元を行い, 一定の時間が経過すると処理を打ち切ることによりフレームレートの安定化を図る. さらに本発表では, これらの処理を PC クラスタ上に実装し, フレームレートが安定することを確かめた.
キーワード　自由視点映像, 実時間処理, フレームレート安定化, 多重解像度, 8 分木, PC クラスタ

Frame Rate Stabilization for Real-Time Free-viewpoint Video Generation

Multi-Resolution Shape Reconstruction

Rui NABESHIMA†, Megumu UEDA†, Daisaku ARITA†, and Rin-ichiro TANIGUCHI†

† Dept. of Intelligent Systems, Kyushu University　6-1, Kasuga-koen, Kasuga, Fukuoka, 816–8580, Japan
E-mail: †{nabeshima,ueda,arita,rin}@limu.is.kyushu-u.ac.jp

Abstract　Our research is aiming at real-time generation of a free-viewpoint video by reconstructing 3-D models from multiple camera images. The generating process is divided into three stages: 3-D shape reconstruction by the visual cone intersection method, conversion of 3-D shape representation from a voxel form into a triangular patch form, and coloring triangular patches. However, there is a problem that the frame rate decreases when the surface area of an object becomes larger since the processing time of the conversion and coloring depends on the number of triangular patches. To solve this problem, we propose a new method which flexibly varies the space resolution of the 3-D shape reconstruction stage. In this paper, we implement these processes on a PC-cluster and experimental results show that our method makes the frame rate more stable.
Key words　Free-viewpoint video, Real-time processing, Frame rate stabilization, Multi-resolution, Octree, PC–cluster

1. はじめに

近年, 計算機の性能向上, 大都市圏での地上デジタル放送の開始, ブロードバンド環境の充実などに対し, それらに対応するリッチコンテンツが徐々に普及し, さらに多くのリッチコンテンツが期待されている. その一つとして自由視点映像が挙げられる. 現実世界の対象の動作をそのまま記録し, 自由な視点から対象を表示する自由視点映像の研究が盛んに行われている [1] [2] [3]. その中で特に著者らは, 自由視点映像のオンライン生成について研究している [4] [5].

既存の自由視点映像のオンライン生成システムの処理の概略は以下の通りである.

形状復元　カメラ画像から対象物体の形状情報を獲得し, 視体積交差法を用いて形状復元を行い, 対象のボクセル表現を得る.

ボクセル表現から三角パッチ表現への変換　ボクセルデータを三角パッチ表現に変換する.

色付け　カメラと仮想視点の位置関係を基に三角パッチに色を塗る.

このシステムの問題点として, 対象物体の表面積が増加すると三角パッチの数が増加するため, 処理速度が低下してしまうことが挙げられる. 自由視点映像のオンライン配信を考えると, 対象の動きを自然に表示させるには処理速度の安定が必要不可欠である. そこで, 本研究では形状復元時に空間解像度を対象にあわせて変化させ, 多重解像度処理を行うことで, フレームレートを安定化させる手法を提案する. 具体的には, 佐藤ら [6] によって提案された 8 分木を利用した視体積交差法によって空間解像度を徐々に上げながら形状復元を行い, 一定の時間が経過すると処理を打ち切ることにより空間解像度を可変にする. さらに, 多重解像度でのボクセル表現から三角パッチ表現への変換を行うときに必要な, 閉じた形状を得るために三角パッチの穴をふさぐ処理を行う.

2. オンライン自由視点映像生成システム

この節では既存のオンライン自由視点映像生成システム [4] について述べる.

自由視点映像生成には多くの処理時間を必要とする. そこで RPV(Real-time Parallel Vision) [7] を用いて, オンラインで並列画像処理を行なっている. RPV とは, 高速ネットワークで接続された PC クラスタ上で動作する実時間多視点動画像処理アプリケーションを構築するためのプログラミングツールである. さらに, 複数台の IEEE1394 ディジタルカメラ [8] が PC に接続されており, 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.

処理の手順を以下に示す.

（1） カメラ画像を取得し, その中から背景差分により対象物体を抽出し, 抽出した画像から視体積を構築する. この視体積はボクセルを使って表現する.

（2） これらの視体積を集めて共通領域を求めることにより形状を復元する.

（3） 得られたボクセル表現に対し離散マーチング・キューブ (DMC) 法 [9] を施すことにより三角パッチ表現へ変換する.

（4） 得られた三角パッチに, 画像を基に色を付けている.

（5） 色データとユーザから入力された仮想視点位置から重み付き色付き対象形状を生成, すなわち対象物体の自由視点映像を生成している.

3. 8 分木を利用した視体積交差法

前節で述べた手法には, 得られる対象物体の三角パッチの数により処理時間が変化するため, フレームレートが安定しないという問題があった. この問題を解決するために, 視体積交差による形状復元の空間解像度を徐々に上げていき, 決められた時間が経つとそこで処理を打ち切ることによりフレームレートの安定化を図る. 空間解像度を徐々に上げていくために, 佐藤らによって提案された 8 分木を用いた視体積交差法を導入した. 本節ではこの佐藤らの手法について説明する.

3.1 パスアルゴリズム

ボクセルを 8 分木構造に従って分割していく過程において, 各ボクセルを以下の三つに判別する.

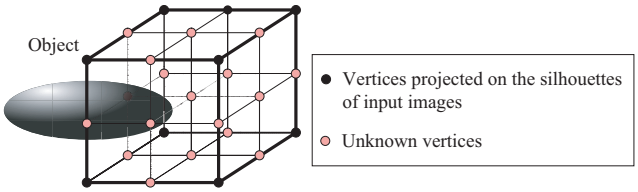


図 1 失敗ボクセル

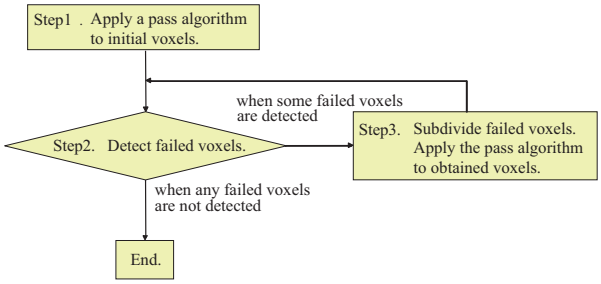


図 2 マルチパス再分割アルゴリズム

白ボクセル 8 頂点がすべて入力画像のシルエット部分に投影されるもの

黒ボクセル 8 頂点がすべて入力画像の背景部分に投影されるもの

灰色ボクセル その他 (シルエットの境界部分)

ある特定の大きさのボクセル (以下, 初期ボクセルと呼ぶこととする) の 8 頂点を調べて上記の 3 種類のボクセルに判別する. さらに, 灰色ボクセルのみを 8 分割する. そして分割されたものに対して繰り返しボクセルを判別し, 灰色ボクセルを分割する. この処理を決められた解像度になるまで再帰的に繰り返す. ここで粗大なボクセルから最も細かいボクセルへの分割のことをパスと呼び, このアルゴリズムをパスアルゴリズムと呼ぶこととする.

3.2 マルチパス再分割アルゴリズム

上記の判別方法は大変単純なものであるので, 実際には誤りがしばしば起こる. 例えば, 頂点を避けるような細長い先端部分は実際には灰色ボクセルであり分割するべきであるが, 黒ボクセルと判定してしまい分割が行われない (図 1). このような分割されるべきであるのに分割されないボクセルのことを失敗ボクセルと呼ぶ.

これを避けるために図 2 のマルチパス再分割アルゴリズムを用いる. Step1 では初期ボクセルにパスアルゴリズムを施す. Step2 において失敗ボクセルを探索し, 発見された場合は Step3 において失敗ボクセルを 8 分割し, 再び Step2 に戻る. 失敗ボクセルが発見されない場合は, そこで処理を終える. Step1 あるいは Step3 で得られる隣接した最小の灰色ボクセルをつなぐと, 復元した形状の表面となるはずである. 最小のボクセルとは, 最大解像度におけるボクセルのことであり, 木構造における葉ノードを指す. 本来灰色ボクセルは物体表面を表すので, 灰色ボクセルをつなぐと閉じた面が得られるはずである.

4. 失敗ボクセル探索

前節で説明したマルチパス再分割アルゴリズムによって対象

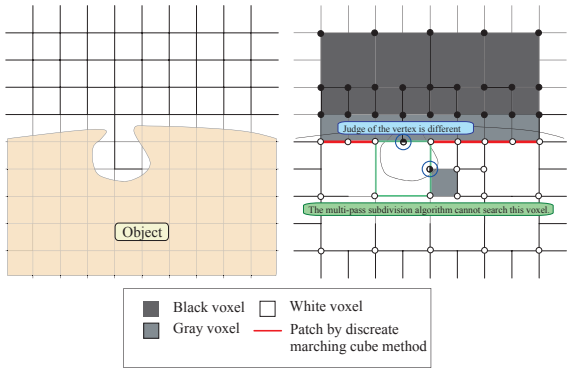


図 3　クラックが起こる例 (2 次元)

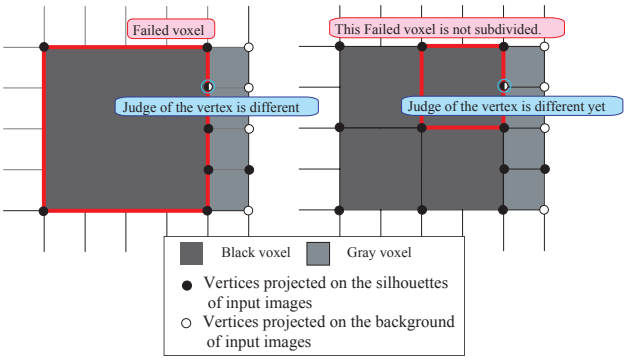


図 4　クラックが起こる例　(2 次元)

形状を得ることができる．しかし, 最小の灰色ボクセルをつないだ形状表面が閉じていても, DMC 法により三角パッチに変換すると表面が閉じないことがありうる. 図 3 にその例を示す. 左側に対象物体, 右側にマルチパス再分割アルゴリズムを施した結果を示している. 最小の灰色ボクセルをつないだ形状表面は閉じているが, 失敗ボクセルを完全には探しきれておらず, その結果穴があいてしまっている．すなわち, 白ボクセルと黒ボクセルが隣接している場合にそれらのボクセルを失敗ボクセルとするだけでは不十分である.

そこで本研究では, マルチパス再分割アルゴリズムにおける失敗ボクセル探索を行った後, さらに発見できなかった失敗ボクセルを別の手法で探索する手法を提案する. 具体的には，灰色ボクセルが生成されるたびに, 隣接する黒および白ボクセルを探索し, 共有する頂点の判定が隣接ボクセルと異なる場合は, 失敗ボクセルとする．このとき, 白ボクセルの表面上の頂点は白, 黒ボクセルの表面上の頂点は黒とする.

しかし, 灰色ボクセルが階層が二つ以上異なる黒あるいは白ボクセルと隣接しているときには注意が必要である. それは発見した失敗ボクセルを分割しても, 頂点の判定がまだ異なることがありうるためである. 図 4 にその例を示す. 左側の図で失敗ボクセルを探索し分割すると, 右側の図となる. この時, 隣接ボクセルと共有する頂点の判定が異なるような頂点がまだ存在するが, 新たに灰色ボクセルが生成されないため, 頂点を調べないことになる. そこで, 階層が二つ以上異なるときには, パスアルゴリズムを施した後, 再度灰色ボクセルとそれに隣接する頂点に矛盾がないか調べる必要がある.

5.　多重解像度処理での視体積交差法

8 分木を利用した視体積交差において, 初めから最大解像度までマルチパス再分割アルゴリズムを適用するのではなく, それよりも低い解像度 (この解像度を打ち切り解像度と呼ぶ) まで適用し, その処理が終了したら打ち切り解像度を高くする処理を最大解像度まで繰り返す．そして, ある一定の時間が経過した時点で処理を打ち切り, その時点での復元形状を出力とする．これにより処理時間の変動が小さくなり, フレームレートを安定させることができる. 処理が途中で打ち切られた場合は解像度の低い復元形状となるが, 計測空間全体に対して形状復元を行うことができる.

5.1　視体積構築

視体積構築処理を以下に示す (図 5). Step2 から Step7 は最大解像度となるまで繰り返し行う．一定の時間が経過すると, Step4, Step5 の途中で処理を打ち切り, Step6, Step2 の処理を最後に行う.

Step1　打ち切り空間解像度までマルチパス再分割アルゴリズムを施す

Step2　処理結果を出力する

Step3　打ち切り解像度を上げる

Step4　新たに分割できるようになった失敗ボクセルを分割する (図中の A のボクセル)

Step5　新たに分割できるようになった灰色ボクセルを分割する (図中の B のボクセル)

Step6　新たに発生した失敗ボクセルを探索し，それを分割する (図中の C のボクセル)

Step7　Step3 から Step6 で得られた処理結果を出力する

Step8　Step3 に戻る

Step7 での処理結果の出力は, 前回の出力との差分情報を出力する．差分情報とは, Step4, 5 での失敗ボクセル, 灰色ボクセルの子ノードと Step6 での失敗ボクセルを根とする部分木である．また送信データには, 黒ボクセルの子ノードや白ボクセルの子ノードのような調べていないノードを含まないため，データ転送量を削減することができる.

Step1, Step6 において, 最小のボクセルにおいても失敗ボクセルが見つかった場合は, その打ち切り解像度ではもう失敗ボクセルを分割することはできず, 次の Step4 で打ち切り解像度を上げることによりまた分割できるようになるので, その情報を保持しておく.

5.2　多重解像度での視体積交差

視体積交差では, 視体積構築によって求められた複数の対象形状の積演算を行うことによって, 最終的な対象形状を求める. これは, PC のデータ入力能力の制限のため, 1 台の PC ではすべてのカメラ画像を獲得することはできないためである. そのため, 多段処理で視体積交差を行う. また, これにより PC の台数を増やしさえすればカメラ台数をいくらでも増やすことができるという利点がある.

しかし, 積演算の際に解像度が異なる場合があるので, これに対応した視体積交差手法が必要となる．また, 視体積構築にお

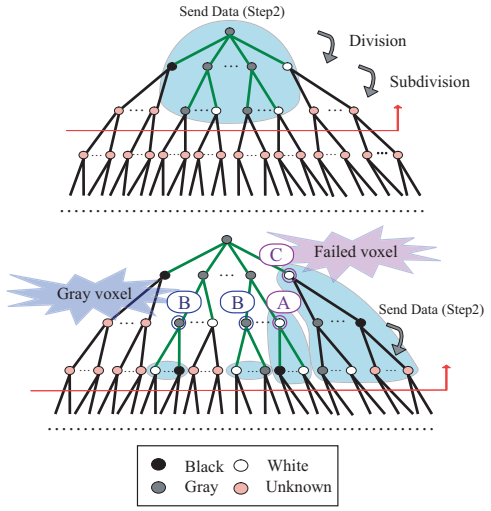


図 5　視体積構築の流れ

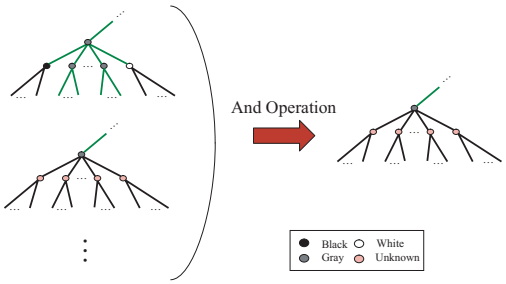
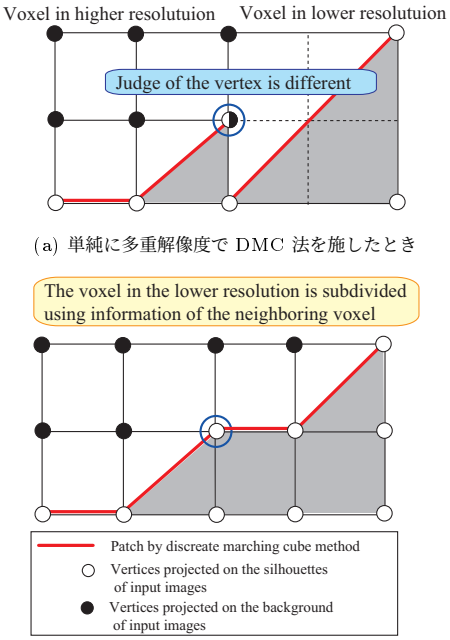


図 6　積　演　算

いて空間解像度を上げるたびに各視体積の処理結果を出力するので, これを用いて各視体積が最終的な処理結果を出す前から積演算を行う. そして更新データが得られると, それを用いて必要なノードだけに対して積演算を行う. このように処理の各段階をオーバーラップさせて実行することで, 遅延時間の削減を図ることができる.

まず最初に, 前節での Step2 で処理結果が出力されたときについて説明する. すべての視体積の 8 分木を根から幅優先探索で走査をしていく. 8 分木中の同じ位置のノードは同じ空間位置のボクセルを指すので, ノードどうしの積演算を行うことで視体積交差を行うことができる. まず, 木のノードの中で少なくとも一つ黒があれば, 黒と判定する. 黒ボクセルがなく, 失敗ボクセルが含まれている場合は, 木を掘り下げる. すべての木のノードが白であれば, 白と判定する. それら以外に関しては, 灰色と判定し木を掘り下げてる. また同時に, 8 頂点の情報はビット論理積を用いて求める.

次に, Step7 で処理結果が出力されたときについて説明する. Step4, Step5 で得られたノードに関しては基本的には先と同じ方法で判定を行う. しかしながら, 処理の打ち切りにより, あるボクセルにおいて分割されている視体積と分割されていない視体積が混在する場合がある. この場合は, 低解像度のボクセルにあわせて, それ以上木を掘り下げない（図 6）. Step6 で得られたノード情報は, 失敗ボクセルが根である部分木として出力されるが, この失敗ボクセルのノードから再び積演算を行い, 木を掘り下げていく. しかし, 視体積交差を行っている木においてこの失敗ボクセルのノードの親や先祖のノードが黒と判定されている場合はこの部分木の情報は棄却する. これは, 親や先祖のノードに黒と判定されているノードがあるときは, 視体積交差法の性質上必ず子や子孫のノードは黒であるためである.



(a) 単純に多重解像度で DMC 法を施したとき

(b) 提案手法

図 7　多重解像度での表面パッチ構成

6.　多重解像度での表面パッチ構成

異なる空間解像度のボクセルを DMC 法を用いてパッチに変換すると, パッチの裂け目 (クラック) が生じることがある (図 7(a)). クラックが生じると, 見た目が大変不自然になる. そのため, クラックをなくす処理が必要となる.

図 7(a) のクラックは, 中央の印のついた頂点において, 左側の高解像度のボクセルと右側の低解像度のボクセルにおいて異なる判断が生じたことにより発生している. このように, クラックは高解像度のボクセルと低解像度のボクセルが隣接する際に, 共有するボクセルの判定が異なる場合に発生する.

そこで, このような頂点を探索し, 頂点の判定を高解像度ボクセル側に合わせて低解像度のボクセルを分割する (図 7(b)). なお, 問題となっている頂点以外の頂点に関しては, DMC 法の性質から, 再び画像と照らし合わせる必要はなく, 低解像度の頂点情報より判定することができる. 具体的には, 決定しようとしている頂点をはさむ頂点の組が一組でも黒と黒の組であれば黒, そうでなければ白となる. つまり簡単な論理演算によって実装可能であるため高速に処理を行うことができる.

7.　提案システム

既存のシステム [4] では, 1 視点ごとに視体積を作り共通部分を求めていたが, 本研究では 3 視点ごとに視体積を構築して共通部分を求める. これは, 3 視点ごとに視体積を構築することにより表面積が小さくなり, 灰色ボクセルは減少し, それにより分

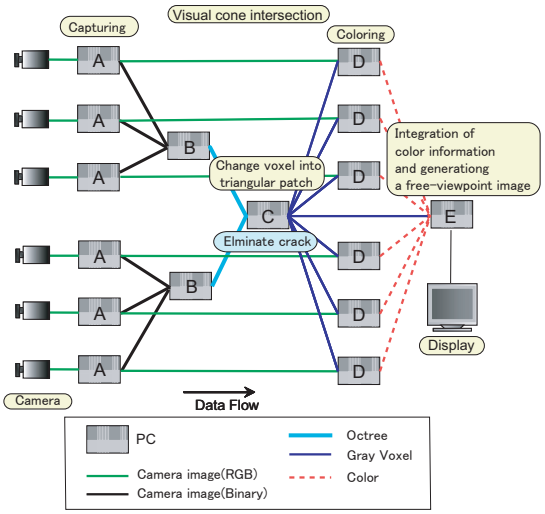
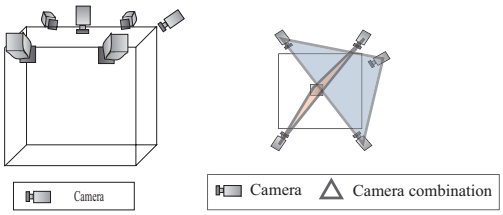


図 8　本研究のシステム構成

表 1　PC の性能

OS	RedHatLinux9
CPU	IntelPentiumIV 3GHz
メモリ	1GB
コンパイラ	gcc-3.2.2



(a) 斜め上から見た図　　(b) 上から見た図

図 9　実験のカメラ配置と組み合わせ

割せずに済むボクセルが増えるため, 処理時間が短くなることが期待できるからである.

以下に提案するシステム構成を示す (図 8). なお, ノード D とノード E における処理は, 既存のシステムと同じである.

ノード A: カメラ画像を取得し, その中から背景差分によって対象物体を抽出する. 対象物体を抽出した画像をノード **B**, ノード **D** に送る. ただしノード **B** は視体積構築のためだけに用いるので, データ量を少なくするために白黒画像を送る. 一方ノード **D** では色付けを行うためカラー画像を送る.

ノード B: ノード **A** から送られてきた抽出画像を用いて視体積を求める. 頂点を参照する際は 3 視点からの画像をそれぞれ参照し, すべて占有の時のみその頂点を占有とする. 視体積を, 8 分木としてノード **C** に送信する. ノード **B** はノード **C** に対し視体積と空間解像度情報を 1 回以上送り, 処理の打ち切りの際に前回送った視体積と空間解像度情報が最終的な結果であることをノード **C** に伝える.

ノード C: ノード **B** からぞれぞれ得られた視体積の共通部分を視体積が送られてくる毎に求める. そして, 得られた灰色ボクセルのみに対して, DMC 法を施すことにより三角パッチ表現へ変換する. この灰色ボクセルは空間解像度が異なるものを含むため, 穴が生じるここで, クラックパッチングにより, 穴をふさぐ. そして, ノード **D** と **E** に送る.

ノード D: ノード **C** から送られてきた, 三角パッチに, ノード **A** から送られてきた画像を基に色を付ける. 得られた色情報をノード **E** に送る.

ノード E: ノード C からの三角パッチデータとノード **D** からの色データとユーザから入力された仮想視点位置から, 重み付き色付き対象形状を生成, すなわち対象物体の自由視点映像を生成する.

8. 実　　験

8.1 実　　験

本手法を用いてオンラインで自由視点映像を生成し, その処理時間, 生成された三角パッチの数を計測した. 本実験では合計 16 台の PC(表 1) を利用した. 各 PC は Myrinet によって相互に結合されており, 100MB/s 以上で通信が可能である. さらに 6 台の IEEE1394 ディジタルカメラ [8] 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 9 にカメラ配置とノード **B** において視体積交差をする際のカメラの組み合わせを示す.

カメラ画像の解像度は 640 × 480 で, 最大空間解像度は 128 × 128 × 128, 最小ボクセルの一辺を $2cm$, 木の深さは 5, 初期ボクセルの空間解像度は 8 × 8 × 8 として実験を行った. また, 打ち切り解像度は 64 × 64 × 64, 128 × 128 × 128 の 2 段階とした. つまり, 8 分木の深さ 4 の空間解像度の 64 × 64 × 64 までマルチパス再分割アルゴリズムを行った後, 深さ 5 の解像度の 128 × 128 × 128 にする.

8.2 自由視点映像についての考察

図 10 に, 形状復元処理が固定解像度の時と多重解像度の時の生成画像を示す. 固定解像度の時, 空間解像度は 128 × 128 × 128 となっている. 多重解像度の時, 対象が二人であるため形状復元時に打ち切りが起こっており, 空間解像度は 64 × 64 × 64 と 128 × 128 × 128 が混在し, 空間解像度が低いために形状が荒くなっている.

8.3 処理時間についての考察

自由視点映像のオンライン配信の際には, フレームレートの平均だけでなく, 分散も重要な要素であると考えられる. これは, 分散が大きい場合, オンライン配信における通信用バッファを大きくとらなければならず, それにより遅延が大きくなってしまうためである.

図 11 に, 形状復元処理の打ち切りを起こしたときと起こしていないときにおいて対象人数が変化したときのフレームレートを示す. 対象が一人から二人へ増加したときに, 処理の打ち切りを起こしていないときに比べて処理の打ち切りを起こしたときはフレームレートの低下が小さいことがわかる. しかし, 処理の打ち切りを起こしたときにおいてもフレームレートが未だ多少低下している. この理由に関しては次節で述べる.

8.4 三角パッチ数についての考察

処理の打ち切りを起こしていないとき, 処理の打ち切りを

図 10　生成画像 (動画)　左:固定解像度　右:多重解像度)

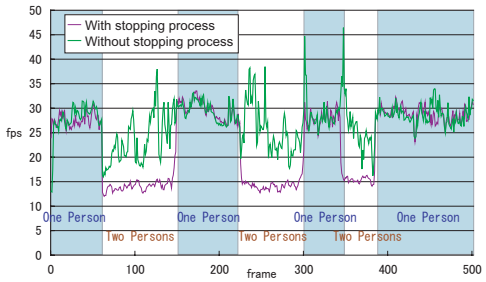


図 11　フレームレート

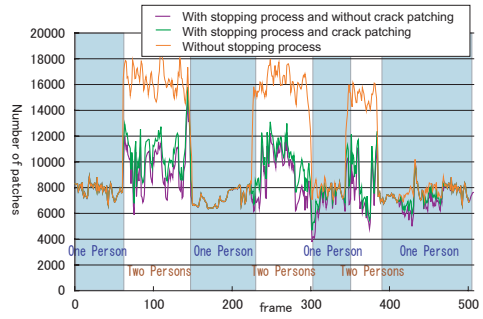


図 12　三角パッチ数

起こしクラックパッチングを行った時と行っていない時の三角パッチ数を図 12 に示す. 対象が一人の時は, 空間解像度はほぼ $128 \times 128 \times 128$ であり, クラックパッチングの必要がないためパッチ数はほぼ等しい. 対象が二人のときは, 形状復元処理の打ち切りが起き空間解像度が異なるボクセルが混在するため, クラックパッチングを行うためパッチ数は増加している. しかし, クラックパッチングによるパッチ数の増加はさほど大きくない. よって, 提案しているクラックパッチングは有用であることがわかる.

また対象が一人から二人へ変化したとき, 処理の打ち切りを起こしていないときの著しいパッチ数の増加に比べると処理の打ち切りを起こしたときのパッチ数の増加は小さい. しかしながら, 処理の打ち切りを起こしたときにおいてもパッチ数が増加している原因は, 対象が一人の時は最大解像度まで形状復元が行われ, 処理の打ち切りが起こっていないためである. つまり, 最大解像度を上げれば対象が一人から二人へ変化してもパッチ数は増加しないと考えられる. またこのパッチ数の増加のため, 対象が二人のときはフレームレートが多少低下している.

9. おわりに

本稿では, 実時間自由視点映像生成の多重解像度処理によるフレームレート安定化を提案した. さらに, 実験により対象に依らずフレームレートが安定したことを示した. 今後の課題としては,

- 木のノードの統合によるパッチ数の安定化
- 色付け等における処理の打ち切り
- 自由視点映像のオンライン配信の実現

などが挙げられる.

謝辞　本研究の一部は, 財団法人大川情報通信基金平成 17 年度研究助成「多視点映像からの自由視点映像の実時間生成」, および北田奨学会記念財団研究開発助成「自由視点映像生成システムに関する研究」の補助を受けた.

文　献

[1] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan:“ Image-Based Visual Hulls”, Proc. of SIGGRAPH, pp.369–374, 2000.

[2] 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.

[3] Yasuhiro Mukaigawa, Daisuke Genda, Ryou Yamene, Takeshi Shakunage:“ Color Blending based on Viewpoint and Surface Normal for Generating Images from Any Viewpont using Multiple Cameras”, Proc. IEEE MFI2003, pp. 95–100, July 2003.

[4] 上田 恵, 有田 大作, 谷口 倫一郎:“ 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”, 情報処理学会論文誌, vol.46, no.11, pp.2768-2778 (2005.11).

[5] 鍋嶋 累, 上田 恵, 有田 大作, 谷口 倫一郎:“ オンライン自由視点映像生成の可変解像度処理によるフレームレート安定化”, 画像の認識・理解シンポジウム, pp. 455–462, September 2005.

[6] Hidenori Sato, Hiroto Matsuoka, Akira Onozawa, Hitoshi Kitazawa:“ Image-Based Photorealistic 3D Reconstruction Using Hexagonal Representation”, 情報処理学会論文誌, Vol. 46, No. 2, pp. 639–648, 2005.

[7] 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. 43, No. SIG 11(CVIM5), pp. 1–10, 2002.

[8] 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.

[9] 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”, 信学技報, PRMU98-218, pp. 197–204, 1999.