2005－CVIM－149（19）
2005／5／12

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

鍋　嶋　累†　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

‡九州大学大学院システム情報科学研究院

近年, 現実世界の対象を自由な視点から表示する自由視点映像のオンライン生成についての研究が盛んに行われている. 既存のシステムの処理は, 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. ここで, 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. そこで, 本研究では形状復元の空間解像度を変化させ三角パッチの数をほぼ一定にすることにより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, 空間解像度を可変にするために, 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行い, 一定の時間が経過すると処理を打ち切る. このシステムを実装しフレームレートが安定することを確かめた.

Stabilization of Frame Rate by Variable Resolution for On-line Free-viewpoint Video

Rui Nabeshima†, Daisaku Arita‡ and Rin-ichiro Taniguchi‡

†Department of Intelligent Systems, Kyushu University

‡Department of Intelligent Systems, Kyushu University

Recently, there are a lot of researches on on-line generation of a free-viewpoint video which shows objects in the real world from an arbitray viewpoint. The processing method is divided into three stages, reconstruction of a 3D model by the visual cone intersubsection method, conversion of 3D model representation from a voxel form to a triangular patch form, and coloring triangular patches. Here, if the surface area of objects becomes larger, the frame rare becomes lower since the processing time of the conversion and coloring depends on the number of triangular patches. Then, in this paper we propose a new method to stabilize the frame rate by changing the space resolution of 3D model reconstruction 3D for stabilizing the number of triangular patches. It is realized by raising the space resolution step by step and stopping the process when a time is over by using an octtree-based visual cone intersection method. And, experimental results show that our method makes the frame rate stabler.

1. はじめに

1.1 研究の背景

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てきた. リッチコンテンツとは映像や音声を利用した高度な品質を持つコンテンツのことを言う. そして, 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. 金出らが Virtualized Reality のコンセプトを提案[1] して以来, 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[2)3)4)5)]. 著者らは, スポーツ中継などを自由視点映像として生中継することを目指し, 自由視点映像のオンライン生成について研究している.

1.2 研究の目的

既存の自由視点映像のオンライン生成システム[6)] の処理の概略は以下の通りである. 詳細は次章で説明する.

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. ボクセルとはピクセルの概念を 3 次元に拡張したものである.

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. 三角パッチ表現とは物体表面を三角形の集合で表現することで, その集合の中の一つの三角形を三角パッチと呼ぶ.

色付け　カメラと仮想視点の位置関係を基に三角パッ

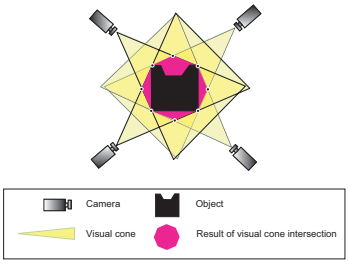


図 1　視体積交差法

チに色を塗る.

ここで, 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, 三角パッチに変換する処理以降は処理速度が低下してしまう. 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. そこで本研究では空間解像度を対象にあわせて変化させ, 三角パッチの数をほぼ一定にすることによりフレームレートを安定化させる手法を提案する. 空間解像度を可変にするために, 形状復元の処理を空間解像度を徐々に上げながら行い, 一定の時間が経過すると処理を打ち切るようにする. しかし, 既存のシステムでの視体積交差法では空間解像度を徐々に上げていくことは難しいため, 本研究では佐藤ら[7] によって提案された 8 分木を利用した視体積交差法を導入した.

2. オンライン自由視点映像生成システム

この章では既存のオンライン自由視点映像生成システムについて述べる. まず, 既存のシステムにおいて用いている諸手法に関して述べ, その後にシステムの概要を述べる.

2.1 視体積交差法

視体積交差法は 3 次元形状を復元する際によく用いられる手法である. 複数のカメラで得られた対象の 2 次元シルエット像を 3 次元実空間 (ボクセル空間) に逆投影し, 視点を頂点とする錐体 (視体積) を各カメラごとに得る. 視体積交差法は, その錐体の共通部分を求め対象の形状を復元する手法である (図 1).

2.2 マーチング・キューブ法

マーチング・キューブ法[8] は等値面生成の手法の一つである. 隣接した 8 個のボクセルを頂点とする立方体（キューブ）を考え, 各頂点におけるボクセルの有無の組み合わせで立方体にどのように面 (三角メッシュ) が構成されるか判断する. 立方体の頂点は 8 個なのでボクセルの有無の組み合わせは 256 通りあるが, 回転対称, 反転を考慮すると 15 通りになる. しかし, これにより生成された等値面が閉じていることを

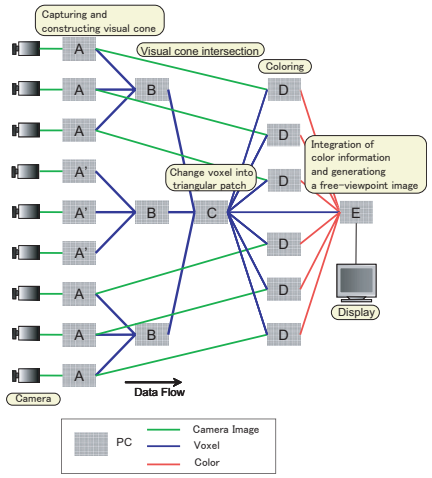


図 2　既存のシステム構成

マーチング・キューブ法は保証していない. そこで離散マーチング・キューブ法[9] が提案されている. 離散マーチング・キューブ法とは, マーチング・キューブ法を発展させた手法で点の連結性を考慮しており

- 等値面の位相的な問題が起こらない (閉曲面が得られる)
- 生成面パタンが 14 通りに減るため, 処理時間が短縮される.

といった利点がある. 既存のシステムでは離散マーチング・キューブ法を用いて対象物体表面を生成している.

2.3 システム構成

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)[10] を用いて, オンラインで並列画像処理を行なう. RPV とは, スイッチ型ギガビット LAN のひとつである Myrinet で接続された PC クラスタ上で動作し, 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.

図 2 に既存のシステム構成を示す. 以下に各 PC での処理を述べる.

ノード A:　カメラ画像を取得し, その中から対象物体を抽出し, 抽出した画像から視体積を構築する. 視体積をノード **B** に, 対象物体を抽出した RGB 画像をノード **D** に送る. 対象物体の抽出は, カメラ画像に対し背景差分を施した後に, クロージング処理を施すことで行う. 背景差分を施した際, 背景に前景と似たような色が存在した場合, 対象領域に穴が生じることがあるので, クロージング処理によりその穴を閉じる.

ノード **A'** は形状復元だけに使用しており, ノード **B** に視体積を送り, 画像は送信しない. 色付けに多くのカメラを使用しても, 精度向上にあまり寄与しないと考え, 処理時間の短縮のため色付けには使用しない. ノード **A**, ノード **A'** は, カメ

ラ位置に偏りが起こらないように決めている．

ノード B: 1 台の PC で全てのカメラ画像に対して視体積交差を行なうと処理時間がかかりすぎるためノード **B**, ノード **C** の多段に分けて視体積交差を行なう．ノード **A** およびノード **A'** から送られてきた各視体積の共通領域を求める．得られた視体積をノード **C** に送る．

ノード C: ノード **B** から送られてきた各視体積の共通領域を求める．ここで対象物体のボクセル表現が得られる．さらに, 得られたボクセル表現に対し離散マーチング・キューブ法を施すことにより三角パッチ表現へ変換する．そして, 三角パッチに変換されるボクセルとその対応する三角パッチパタンをノード **D** と **E** に送る．三角パッチとして送らないのは, 三角パッチは 1 つのボクセルに対し複数の三角パッチが張られるのでデータ量が大きく送受信に時間がかかるためである．

ノード D: ノード **C** から送られてきた, 三角パッチに変換されるボクセルとその対応するパタンより三角パッチを再構成する．得られた三角パッチに, 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 **D** と同様に三角パッチを再構成する．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．

ノード **A** からノード **C** へのボクセルの流れは 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．具体的にはノード **A** において, 1 フレーム分の視体積構築処理が完了する前に, 得られたボクセルから順に次のノードへ送信することにより遅延時間の削減を図る．

3. 8 分木を利用した視体積交差法

既存の視体積交差法では空間解像度を徐々に上げていくことは難しいため, 本研究では佐藤らによって提案された 8 分木を利用した視体積交差法を導入する．

3.1 8 分 木

8 分木とは子ノードを持つノードは必ず 8 個の子ノードを持つ木を指す．ボクセルを 8 等分に分割し, 分割されるボクセルを親, 分割してできたボクセルを子として, 親子関係を 8 分木構造で考える (図 3)．

3.2 ボクセル占有判別

ボクセルを以下の三つに判別する．

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)

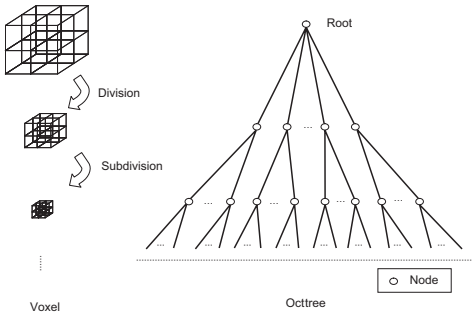


図 3 ボクセルと 8 分木

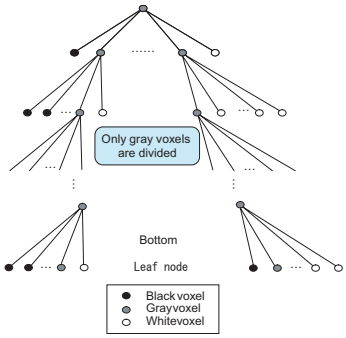


図 4 パスアルゴリズム

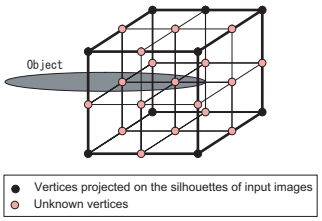


図 5 失敗ボクセル

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, 図 4 のように灰色ボクセルのみ 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, 分割を繰り返す．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．

3.3 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を行わない．このようなボクセルを失敗ボクセルと呼ぶ (図 5)．

これを避けるために図 6 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

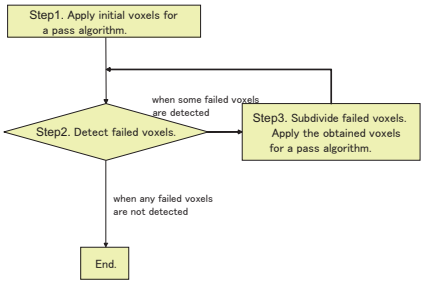


図 6 マルチパス再分割アルゴリズム

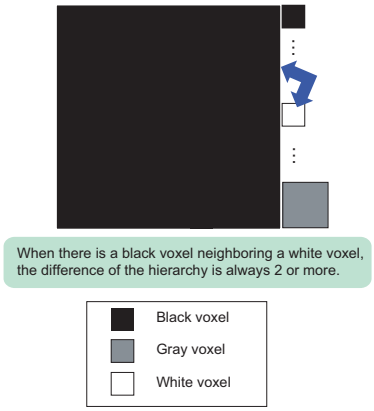


図 7 白ボクセルと黒ボクセルが隣接する例 (2 次元)

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図 4 において葉ノードを指す. ここで, もし白ボクセルと黒ボクセルが 6 近傍で隣接していれば, 得られた表面は閉じていないことになるので, これらを失敗ボクセルとする. また, 失敗ボクセルは分割可能なボクセルのみとする. さらに, 白ボクセルと黒ボクセルが隣接する時は必ず 2 つ以上木の階層が異なる (図 7).

失敗ボクセルを 8 分割し, 得られたボクセルにパスアルゴリズムを施す. そして, 再び失敗ボクセルが見つかる限り Step2 と Step3 を繰り返す. 最終的に灰色ボクセルのみで閉じた表面を得ることができる. しかし, 以下の場合は失敗ボクセルを見つけることができない.

- 初期ボクセルが対象物体より大きく, 対象物体がそのボクセルの中に完全に含まれる場合
- 最小の灰色ボクセルが密集し, 灰色ボクセルの本来の連結が失われ, 別の連結が起こった時 (図 8)

失敗ボクセルの探索方法の詳細は次章で説明し, この問題は実験で評価する.

4. 可変解像度処理によるフレームレート安定化

4.1 形 状 復 元

形状復元を行うための視体積交差法に 8 分木を利用した. 木を使うことにより, ある一定の時間が経過した時に木の葉まで達していなくても, 先祖のノードにおける占有情報により空間解像度は低いが形状を復元することができる. そのため形状復元の処理を途中で打ち切り, フレームレート安定化を図ることができる.

4.1.1 処理の概要

形状復元処理を以下に示す (図 9). Step2 から Step4 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, Step2, Step3 が終了した際に処理を打ち切る. 打ち切りを行う空間解像度を打ち切り解像度と呼ぶこととする.

***Step*1** ある一定の空間解像度までマルチパス再分割アルゴリズムを施す

***Step*2** 葉ノードの情報を送る

***Step*3** 前回の空間解像度で分割できなかった灰色ボクセル, 失敗ボクセルを空間解像度を上げて分割する

***Step*4** Step3 の空間解像度でマルチパス再分割アルゴリズムを用いて失敗ボクセルをなくす

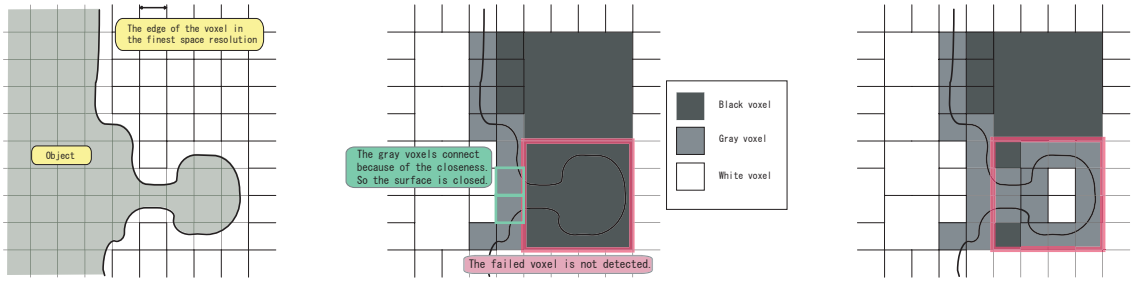
Step1 において, あるノードを黒ボクセルと判定した時は, そのノードの子孫の葉ノードを黒とする. 白ボクセルと判定した時も同様である. 一方灰色ボクセルは, その空間解像度において分割できないノードまで分割し終えた時のみ, 子孫の葉ノードを灰色ボクセルとする.

Step2 において, 次の Step3 でマルチパス再分割アルゴリズムの空間解像度を上げることにより分割できる灰色ボクセルは白ボクセルとして送り, 空間解像度が最大となり分割不可能となった灰色ボクセルは頂点の占有情報を送る. さらに, ストリーミング転送を利用して送信することにより遅延の削減を行う.

Step1, Step4 において, 現在の空間解像度においては分割できないが, その後の Step3 において空間解像度を上げることにより分割できるような失敗ボクセルが見つかった場合, これは他の失敗ボクセルとは別に保持しておき, その後の Step3 において分割を行う.

4.1.2 失敗ボクセル探索

マルチパス再分割アルゴリズムにおいて一度目に失敗ボクセルを探す時は, 白ボクセルを全走査する. 白ボクセルの表面と隣接する最小ボクセルを全走査し, 黒ボクセルがないか探す. 黒ボクセルが発見された場合, そのボクセルの親ノードへのリンクを辿り分割されていないノードまで辿る. 分割されていないノードがあればそれを失敗ボクセルとして分割する. さら



(a) 対象物体　(b) マルチパス再分割アルゴリズムを施した結果　(c) 正しい結果

図 8　失敗ボクセルが見つからない例 (2 次元)

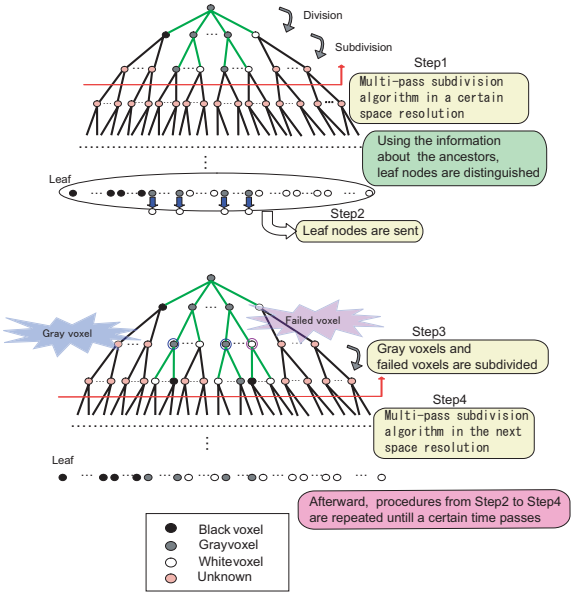


図 9　形状復元の流れ

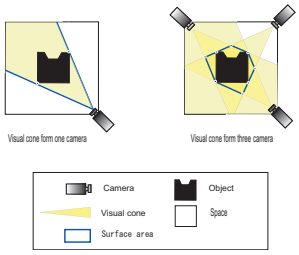


図 10　視体積交差法

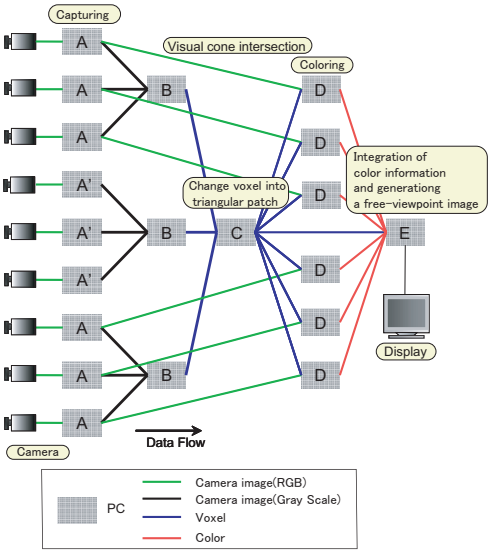


図 11　本研究のシステム構成

に, 元の白ボクセルが分割されていなければ白ボクセルも失敗ボクセルとして分割する. 二度目以降の失敗ボクセル探索は, その直前に失敗ボクセルを分割して得られた白ボクセル, 黒ボクセルを全走査する. これは, 失敗ボクセルとなる部分がその直前のパスで分割して発生したのであれば, 新たに発生したボクセルのみ走査すれば良いためである.

4.2　システムへの組み込み

既存のシステムのように 1 視点ごとに視体積を作り共通部分を求めるのではなく, 3 視点ごとに視体積を構築して共通部分を求める. 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する (図 10). それにより分割せずに済むボクセルが増えるため, 参照せずに済む頂点が減り処理時間が短くなると考えられる.

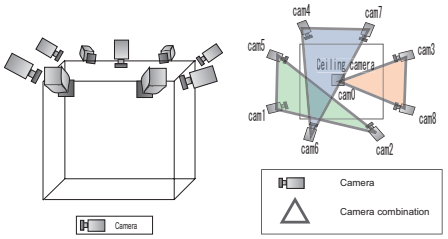
以下に提案するシステム構成を示す (図 11).

ノード A:　カメラ画像を取得し, その中から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** には, RGB 画像を色付けのために送る.

ノード B:　ノード **A** およびノード **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,

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 12 実験のカメラ配置と組み合わせ

すべて占有の時のみその頂点を占有とする．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．ただし，空間解像度が違う場合は最も高いものにあわせる．さらに，得られたボクセルに対し空間解像度に合わせた離散マーチング・キューブ法を施すことにより三角パッチ表現へ変換する．そして，三角パッチに変換されるボクセルとその対応する三角パッチパタンをノード **D** と **E** に送る．

ノード D, E: 既存のシステムと同じである．

5. 実験と考察

5.1 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，遅延時間，8 分木を利用した視体積交差法の精度を計測した．本実験では合計 20 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 9 台の IEEE1394 ディジタルカメラ[11] 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 12 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 320×240 で，最大空間解像度は $128 \times 128 \times 128$，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 $8 \times 8 \times 8$ として実験を行った．また，打ち切り解像度は $64 \times 64 \times 64$，$128 \times 128 \times 128$ の 2 段階とした．つまり，8 分木の深さ 4 の空間解像度の $64 \times 64 \times 64$ までマルチパスアルゴリズムを行い，葉ノードを送信した後，深さ 5 の解像度の $128 \times 128 \times 128$ にする．

5.2 自由視点映像についての考察

図 13 に，入力画像とそれと同じ視点からで形状復元処理が固定解像度の時と可変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 $128 \times 128 \times 128$ となっている．可変解像度の時，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 $64 \times 64 \times 64$ となっており，空間解像度が低いために形状が荒い．また，対象の表面にある灰色ボクセルを白ボクセルとしたために，空間解像度が高い時よりも形状は多少大きくなっている．そのため，対象の境界部分など色付けが正しく行われていない部分が現れてしまう．しかし，空間解像度が低くても姿勢や動きはほぼ認識できたので，空間解像度が $64 \times 64 \times 64$ の形状情報も十分利用できると思われる．

さらに，図 14 に可変解像度で，実際にはカメラのない視点からの生成画像を示す．小さな立方体は，カメラ位置を表す．先ほどと同様に打ち切りが起こっており，空間解像度は $64 \times 64 \times 64$ となっている．

最後に，図 15 に可変解像度で，対象人数が変化する時の生成画像を示す．対象が二人の時は形状復元時に打ち切りが起こっており，空間解像度は $64 \times 64 \times 64$ となっているが，対象が一人の時は処理時間が比較的短いため，空間解像度は $128 \times 128 \times 128$ となっている．対象に合わせて空間解像度が変化していることがわかる．

5.3 形状の精度についての考察

失敗ボクセル探索は完全でないため，既存の方法と完全に一致した形状は得られない．そのため，どの程度形状が一致するのか測定した．空間解像度が $128 \times 128 \times 128$ において，すべてのボクセルを画像と照らし合わせる既存の視体積交差法と 8 分木を利用した視体積交差法により得られた対象のボクセル表現を比較した．ただし，8 分木を利用した視体積交差法は打ち切りは行わず，大きなボクセルは最小ボクセルまで分割して比較する．

$$\text{再現率} = \frac{n(A \cap B)}{n(A)} \times 100 \qquad (1)$$

n(A): A のボクセル数

A: 既存のシステムでの視体積交差法で占有と判断されたボクセル

B: 8 分木を利用した視体積交差法で占有と判断されたボクセル

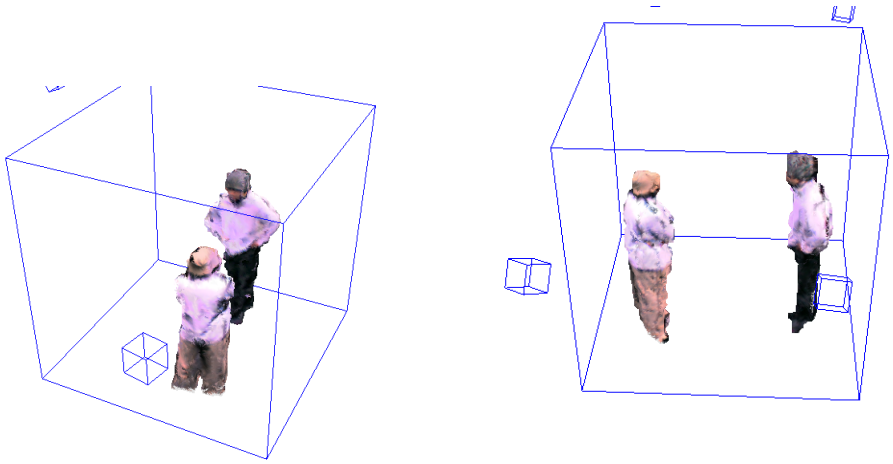
対象が人 1 人で計測すると，再現率は平均 99.6%となった．よって，既存のシステムの手法と比べ，遜色がないと言える．

5.4 処理時間についての考察

図 16 に，本手法と既存のシステムの手法において

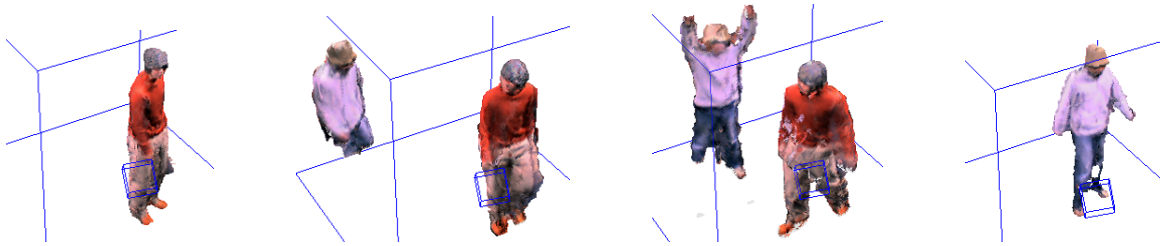
(a) 原画像　(b) 生成画像 (可変解像度)　(c) 生成画像 (固定解像度)

図 13　原画像とそれと同じ視点における生成画像



(a) 生成画像 1　(b) 生成画像 2

図 14　実際にはカメラのない視点からの画像 (可変解像度)



(a) 生成画像 1　(b) 生成画像 2　(c) 生成画像 3　(d) 生成画像 4

図 15　対象人数が変化する時の生成画像 (可変解像度)

対象が 1 人から 2 人に変わった時のフレームレートを示す．およそ 150 フレーム目から 160 フレーム目で対象が 1 人から 2 人へと増加する．既存のシステムの手法は人が 1 人から 2 人になるとフレームレートが著しく減少しているが，本手法ではおよそ 20fps を保っている．このことより，既存のシステムの手法に比べ本手法はフレームレートが一定の数値以上で安定しているといえる．ただし，本手法は既存のシステムの手法に比べてフレームレートのぶれが大きい．これは，3 章での Step2, Step3 を終えてから処理を打

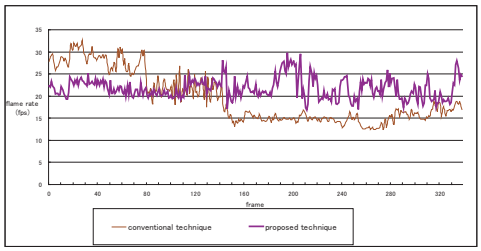


図 16　フレームレート

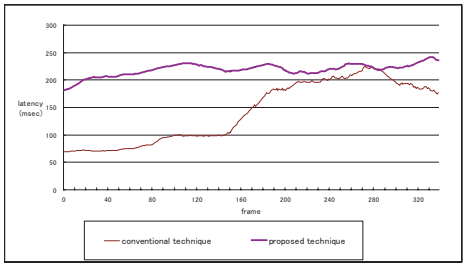


図 17　遅延時間

ち切るかどうか判断するため, 処理の打ち切りが一定の時間で起きるとは限らないためである.

5.5　遅延時間についての考察

図 17 に, 本手法と既存のシステムの手法において対象が 1 人から 2 人に変わった時の遅延時間を示す. 遅延時間は, 全 PC の内部時計は一致しているという前提で, カメラ画像が入力された時刻と自由視点映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. また, データは前節と同じものを使っている. 既存のシステムの手法は人が 1 人から 2 人になると遅延時間が著しく増加しているが, 本手法では変化があまり見られない. これは, 本方法では空間解像度が低いことにより三角パッチに変換して以降のノードの処理時間が短縮されたためであると考えられる.

6. おわりに

本稿では, 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,

- フレームレートのぶれを小さくする
- ボクセルではなく木を送ることにより, 送信量を圧縮する
- 異なる空間解像度が混在するボクセルを三角パッチに変換する
- 色情報を圧縮する
- 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「北田奨学会記念財団」の補助を受けて行った.

参 考 文 献

1) T. Kanade, P. W. Rander, P. J. Narayanan:“ Concepts and early results”, IEEE Workshop on the Representation of Visual Scenes, pp.69–76, Jane 1995.
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) 北原 格, 大田 友一:“ 大規模空間に適した 3 次元形状表現手法による自由視点映像の実時間生成”, 信学技法, PRMU2003, pp. 61–66, 2003.
4) 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.
5) Shohei Nobuhara, Takashi Matsuyama:“ Heterogeneous Deformation Model for 3D Shape and Motion Recovery from Multi-Viewpoint Images”, Proceedings of the 2nd International Symposium on 3D Data Processing, Visualization, and Transmission, pp. 566–573, Thessaloniki Greece, September 2004.
6) 上田 恵, 有田 大作, 谷口 倫一郎:“ 3 次元ビデオ映像のオンライン生成”, 画像の認識・理解シンポジウム 2004(CD-ROM), Vol. 2, pp. 283-288, (2004.07).
7) 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.
8) William E. Lorensen, Harvey E. Cline:“ Marching Cubes: A High Resolution 3D Surface Construction Algorithm”, Computer Graphics, Vol. 21, No. 4, pp. 163–169, 1987.
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.
10) 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.
11) 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.