平面間透視投影を用いた並列視体積交差法

ウ小軍 †, 和田俊和 †, 東海彰吾 †, 松山隆司 †

多視点映像から物体の 3 次元全周囲形状を復元する手法として，視体積交差法がよく用いられる．本論文では，**平面間投影**に基づく視体積交差法の高速化と，その並列化手法を提案する．本稿で提案する手法は，octree 表現やルックアップテーブルなど二値画像に特化した手法ではなく，入力画像から 3 次元空間への逆投影計算の高速化に主眼をおいている．この手法を，画像撮影と通信機能を有する計算機から成る PC クラスタ上に実装するための並列アルゴリズムの開発，画像撮影から形状復元までを高速に実行する試作システムの構成法，およびこのシステムを用いた実験結果について述べる．実験結果から，空間解像度が 3cm ボクセルの場合，自然な身体動作をビデオレートに近い速度で復元できることを示す．

Parallel Volume Intersection Based on Plane-to-Plane Projection

Silhouette volume intersection is one of the most popular methods for reconstructing the 3D volume of an object from multi-viewpoint silhouette images. This paper presents a novel parallel volume intersection method based on plane-to-plane projection *for real-time 3D volume reconstruction using active cameras. This paper mainly focuses on the acceleration of back-projection from silhouette images to 3D space without using any sophisticated software technique, such as octree volume representation, or look-up table based projection acceleration. Also this paper presents a parallel intersection method of projected silhouette images. From the preliminary experimental results we estimate near frame-rate volume reconstruction for a life-sized mannequin can be achieved at 3cm spatial resolution on our PC cluster system.*

1. はじめに

身体動作の 3 次元データはリアルな CG やアニメーション作成だけでなく，テレロボティックス，人間工学，生体力学，スポーツ動作解析など幅広い分野で用られている．通常，このような動きのデータは専用のモーションキャプチャー装置を用いて計測されることが多い．このうち，最も多く用いられているのは光学センサを用いたものである．

光学センサを用いたモーションキャプチャシステムは発光体や反射体を利用したマーカを関節などのキーポイントに取り付け，マーカからの光を周囲のカメラによって観測し，観測データからキーポイントの 3 次元位置を計算するものである．しかし，よりリアルな身体動作の計測を行うことを考えると，このシステムにも，次のような制限があると言える．

- **物体の形状情報が得られない**:
  従来のモーションキャプチャーシステムでは対象の関節などのキーポイントの 3 次元位置しか得られず，布などの変形物体の表面形状などは得られない．
- **計測可能な空間が狭い**:
  これは，カメラの画角が有限であり，且つ，カメラの視線方向が固定であることに起因している．

これらの制限は次のようにして解消できるものと考えられる．

(1) **3 次元形状の復元**: 対象表面の 3 次元形状が復元できれば，変形物体についても計測を行うことができ，よりリアルな身体動作の計測を行うことができる．

(2) **アクティブカメラコントロール**: 対象位置の移動に合わせてカメラの視線方向を変えることにより，空間解像度を落とさずに，観測可能な範囲を広げることができる．同様に，撮影条件に応じてカメラのズームパラメータを変化させることができれば，対象撮影時の空間解像度を高く保つことも可能である．

† 京都大学 大学院情報学研究科 知能情報学専攻
Department of Intelligent Science and Technology, Graduate School of Informatics, Kyoto University

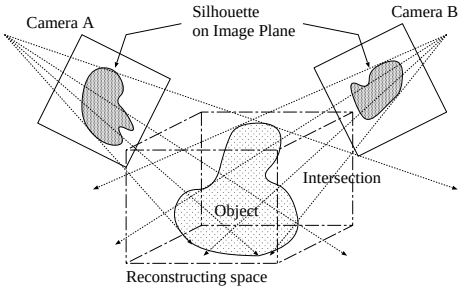
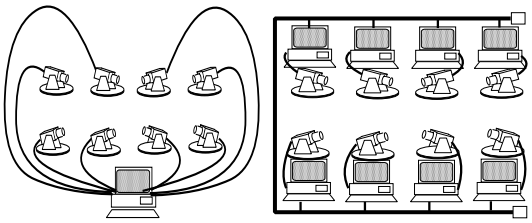


図 1 視体積交差法の概念図
Fig. 1 Silhouette Volume Intersection



(a) Single Processing (b) Parallel Processing

図 2 3 次元物体形状復元の計算方式
Fig. 2 Computation Schemes for volume Reconstruction

3 次元物体をその周囲から複数のカメラを用いて撮影し，物体全周の 3 次元形状を復元する問題では，ステレオ視よりも，視体積交差法[1)2)6)7)8)9)] がよく用いられる．この理由は，ステレオ視と比べて，

- 複数のカメラで共通に観測される部分のみが形状復元できるといった制約がない
- 基線長が長い場合に困難となる対応付け問題を扱う必要がない
- 物体全周の形状が自然に復元できる

等の利点があるためである．

この手法の基本的な考え方は**シルエット制約**[5)] に基づいている．これは,「対象は任意の視点から撮影して得られる物体の二次元シルエットを実空間に投影して得られる錐体（視体積）の中に含まれる」という制約条件である．この制約を多視点画像で得られたシルエットに拡張すれば,「複数の視点に対応する視体積の共通部分の内側に対象が存在する」ということになる．この共通部分は，*visual hull* と呼ばれ，その内部に対象が存在する．この visual hull は，より多くの視点からの視体積を用いるほど小さくなり，多方向からの視体積を用いた場合には，対象の概形を表しているものと見なすことができる．

視体積交差法による 3 次元形状復元アルゴリズムは，次の二つに大別される．

(1) **Space carving method(SCM)** 3 次元空間中の個々の点を，各画像平面に投影したとき，全てのシルエットの内部に投影される点は，対象の内部の点として残し，それ以外は対象の外部の点として除去する．

(2) **Volume intersection method(VIM)** 実際に複数のシルエットを元の 3 次元空間に逆投影し，それぞれの視体積の共通部分を求める

SCM はシルエット制約に基づいて，物体の内点を残して，それ以外の点を削り落とす手法である．これに対し，VIM はシルエット制約を満足する点の集合を視体積の共通部分として求める手法である．VIM に比べ通常 SCM の方がより少ないメモリで実装可能なことから，実際に用いられる手法の多くが SCM に分類される．

3 次元形状復元を行うには，すべてのカメラが対象を画像枠にかからないように撮影できていることが前提条件となる．これは，カメラの共通視野を籠に例えれば，この籠の中に常に対象の全体が入っているように，対象の移動に合わせてカメラコントロールを行うことに相当する．これを実現するためには，対象の 3 次元位置・形状が不可欠であり，形状復元の速度がカメラコントロールにとって，非常に重要になる．このように，3 次元形状復元はできるだけ高速に行う必要がある．

この形状復元の速度とカメラ台数の関係は，次に示すように使用する計算機アーキテクチャによって規定されている．

図 2 (a) に示すように，一台の高速な計算機があり，ある台数のカメラで撮影を行う場合に，ビデオレートで形状復元が行えると仮定する．明らかに，カメラの台数が増えれば，計算速度は落ちる．つまり，単一の計算機で 3 次元形状復元を行うことは拡張性がないと言える．これに対し，1 台ずつカメラが接続された複数の計算機ネットワークシステム (図 2 (b)) では，形状復元の速度はカメラの台数に影響されにくい．これは，画像の撮影やシルエットの生成などの処理が各計算機で独立に行え，且つ，形状復元計算も並列化可能であるからである．したがって，多視点画像から視体積交差法による形状復元を行うには，単一の計算機よりも，メモリ分散型並列計算機，あるいは，PC クラスタが適していると言える．

本論文では，視体積交差法に基づく高速な並列形状復元手法を提案する．この後の章では，その並列手法の基本的な考え方，シルエットの逆投影の高速化，並列化の詳細および実験結果を述べる．

```
foreach v ∈ {voxel} begin
    v :=′ occupied′
    foreach c ∈ {camera} begin
        project v to image plane of c
        if projected point isn't in the silhouette
then
            v :=′ empty′
            goto ENDINNERLOOP
        endif
    end
ENDINNERLOOP:
end
```

図 3 SCM の基本アルゴリズム

Fig. 3 Basic Algorithm for SCM:{$voxel$}: set of all voxels, {$camera$}: set of all cameras, $'occupied'$: value of occupied voxel, $'empty'$: value of non-occupied voxel.

```
{voxel} := {′occupied′}
foreach c ∈ {camera} begin
    {F(c)} := {′empty′}
    foreach p ∈ {silhouette(c)} begin
        project p to 3D space and generate a ray
b(p, c)
        {F(c)} := {F(c)} ⋃ r(p, c)
    end
    {voxel} := {voxel} ⋂ {F(c)}
end
```

図 4 VIM の基本アルゴリズム

Fig. 4 Basic Algorithm for VIM
{$voxel$}: set of voxels, {$silhouette(c)$}: set of silhouette pixels observed by camera c, p: silhouette pixel, {$camera$}: set of all cameras, {$F(c)$}: frustum generated by back projection from silhouette observed by camera c, $b(p, c)$: a 3D ray passing both p and the opticalcenter of c.

2. 基本アルゴリズム

視体積交差法の高速化にあたって，まず基本アルゴリズムを選択しなければならない．

SCM と VIM の基本アルゴリズムを図 3 と図 4 にそれぞれ示す．SCM の場合，一番外側のループは *voxel* についてであるのに対し，VIM の場合は *camera* を一番外側にループしている．この場合，カメラは各計算機に相当する．従って，シルエットの逆投影は各計算機で完全に独立して処理でき，さらに視体積の交差を求める計算も各計算機で並列に実行できることから，VIM の方が並列処理に適していると言える．

一方，VIM の短所は，視体積の表現に膨大なメモリを要することである．これに関しては，次の 2 種類の解決法が挙げあられる．

(1) 視体積の octree 表現
(2) ひとつの視体積を複数の部分集合に分割し，個々の部分集合について計算する方法

前者はもっとも有効な方法のひとつである．octree 表現は，メモリ消費量を抑えるだけでなく，視体積の交差計算も高速化できるという利点があり，実際にこれを用いた高速化手法[9] が提案されている．

しかし，この手法はシルエットのような二値データに関して最適化されており，濃淡あるいはカラー画像[4] に拡張することは不可能である．しかも，骨格などのように隙間の多い物体に関しては octree 表現は必ずしも効率的ではない．以上の理由から，本論文では，後者の方法で視体積交差法を実現する．

3 次元ボクセル空間を面や箱などのより小さい部分集合に分割した場合，VIM のアルゴリズムは図 5 のようになる．この修正したアルゴリズムでは，一番外側のループが各部分集合になっているが，逆投影計算は各計算機で並列に処理できることに注意されたい．

このアルゴリズムに沿って考えると，VIM の基本操作は次の二つに分類できる．

(1) 逆投影計算
(2) 逆投影されたシルエットの交差計算

次章では，これらの操作の詳細および高速化について述べる

3. 逆投影計算の高速化

逆投影計算は膨大な数値計算を伴うため，VIM のうち最も計算コストの高い部分である．この部分の高速化には 3 つのアプローチが考えられる．

(1) ボクセル空間の各ボクセルと二次元画像の画素とを対応付けしたルックアップテーブルの利用
(2) 投影計算用ハードウェアの利用
(3) 透視投影の高速アルゴリズムの利用

我々はアクティブカメラを用いることを前提に議論してきた．視線方向などのカメラパラメータが変われば，ボクセルと画素間の対応関係も変わるので，ルックアップテーブルの利用は明らかに不可能である．ハードウェアの利用は高速化にとって有利ではあるが，その設計にはまずアルゴリズムを固めなければならない．従って，本論文では最後のアプローチをとることにした．

$\{voxel\} := \{'empty'\}$
foreach $\{v\} \in \{\{v\}\}$ begin
  $\{v\} := \{'occupied'\}$
  foreach $c \in \{camera\}$ begin
    $\{F(c)\}_v := \{'empty'\}$
    foreach $p \in \{silhouette(c)\}$ begin
      compute projection $b_v(p,c)$ from p to $\{v\}$
      $\{F(c)\}_v := \{F(c)\}_v \bigcup b_v(p,c)$
    end
    $\{v\} := \{v\} \bigcap \{F_v(c)\}$
  end
  $\{voxel\} := \{voxel\} \bigcup \{v\}$
end

図 5 部分集合への分割を行う VIM アルゴリズム
Fig. 5 VIM Algorithm with Voxel-Space Decomposition $\{\{v\}\}$: disjoint decomposition of $\{voxel\}$, $\{v\}$: subset of voxels, $\{F(c)\}_v = \{F(c)\} \bigcap \{v\}$, $b_v(p,c) = b(p,c) \bigcap \{v\}$

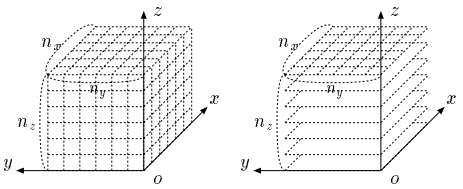


図 6 3 次元空間の分解法: (a) 3 次元ボクセル空間, (b) 平行平面群への分割.
Fig. 6 3D space representations: (a) 3D voxel space, (b) Disjoint decomposition into parallel planes.

具体的に，本論文では，図 6 に示すように，ボクセル空間を複数の互いに平行な平面集合に分割する．これによって，2 次元画像平面から 3 次元ボクセル空間への投影計算が，平面間の投影計算に分解することができる．

- 一般的に，3 次元の点 (X, Y, Z) から 2 次元の点 $(x_1/x_3, x_2/x_3)$ への透視投影は次式で表すことができる．

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} . \quad (1)$$

  この計算には 1 点につき **9** 回の加算と **9** 回の乗算と **2** 回の除算が必要である．
- 平面間投影の場合，つまり，3 次元の点がある 2 次元平面上に存在する場合には，投影計算は次のように簡単化される．

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} . \quad (2)$$

  この計算には 1 点につき，**6** 回の加算と **6** 回の乗算と **2** 回の除算が必要である．

以上のように，平面から平面への透視投影計算は一般的な点から点への投影に比べ計算量が少なく速いことが分かる．つぎの節では，この平面間投影をさらに高速化する手法について述べる．

3.1 非平行平面間投影の高速化

投影計算に要する演算回数を削減するために，次の性質を利用することができる．

性質 1 3 次元空間中に，2 つの平面 A と B と点 o を考える．このとき，点 o を通る直線のうち，次の性質を満足する直線 P が少なくとも 1 本は存在する：P を含む任意の平面 C と平面 A, B それぞれとの交線 $A \bigcap C$, $B \bigcap C$ が存在する場合，これらの交線は互いに平行である．

証明

- **平面 A と B が互いに平行でない場合**
  平面 A，B はそれぞれ交線 $A \bigcap B$ に平行な直線群 $\{L_A\}$ と $\{L_B\}$ に分解できる．一方，3 次元空間中の 1 点 o を通り，$A \bigcap B$ に平行な直線 P を仮定すると，P を含み，且つ，A と B のいずれとも平行でない平面 C が考えられる．$P \subset C$ ，且つ，$P \parallel A \bigcap B$ から，$C \parallel A \bigcap B$ が成り立つ．従って，C も $A \bigcap B$ に平行な直線群 $\{L_C\}$ に分解することができる．$\{L_A\}, \{L_B\}, \{L_C\}$ に注目すると，各直線集合の要素は全て互いに平行である．さらに，$A \bigcap C = \{L_A\} \bigcap \{L_C\}$ と $B \bigcap C = \{L_B\} \bigcap \{L_C\}$ から，交線 $A \bigcap C$ と $B \bigcap C$ はともに $\{L_C\}$ の要素であると言える．したがって，3 次元空間中の 1 点 o を通り $A \bigcap B$ に平行な直線を P とすれば，P を含み，且つ，A と B のいずれとも平行でない平面 C は，$A \bigcap C \parallel B \bigcap C$ という条件を満足する．
- **平面 A と B が互いに平行な場合**
  A または B に平行でない任意の平面 C について考えると，明らかに交線 $A \bigcap C$ と $B \bigcap C$ は平行である．従って，点 o を通る任意の直線 P について，問題の平面群 C は定義できる．

Q.E.D.

この性質から，A と B が平行でない場合は、図 7 に示すように，投影中心を点 o とし，点 o を通過し $A \bigcap B$ と平行な直線を P とすれば，P を含む平面 C

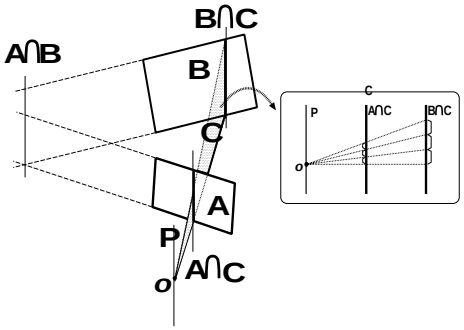


図 7 直線単位の平面間透視投影法
Fig. 7 Linear-wise Plane-to-Plane Perspective Projection

について，交線 $A \bigcap C$, $B \bigcap C$ は互いに平行になることが保証される．この場合，平面 A から B への平面間透視投影は A と B 上の互いに平行な直線間の投影計算に分解することができ，この平行直線間の透視投影は 1 次元のスケーリングに置き換えられる．したがって，平面 C を P を軸にして回転させれば，これら 2 平面間の透視投影が，1 次元のスケーリング計算に分解できることになる．

この 1 次元のスケーリング計算は，2 対の対応点 $\overrightarrow{x_0}$,$\overrightarrow{X_0}$ が与えられていれば，以下のように，2 本の線分上にあるほかの点は定数ベクトル $\overrightarrow{\delta}$ と $\overrightarrow{\Delta}$ をそれぞれの点に加算することによって実現できる．

$$\overrightarrow{x_{i+1}} = \overrightarrow{x_i} + \overrightarrow{\delta}, \quad \overrightarrow{X_{i+1}} = \overrightarrow{X_i} + \overrightarrow{\Delta}. \qquad (3)$$

この場合，1 点の計算は **4** 回の加算で行える．しかし，1 組の平行線分につき，最初の 2 組の対応点と定数ベクトル $\overrightarrow{\delta}$, $\overrightarrow{\Delta}$ を計算するオーバーヘッドが生じる．これらのオーバーヘッドの内訳は (1) 透視投影計算 2 回，(2) ベクトルの差分計算 2 回，(3) スカラー除算 2 回，である．投影先の平面が $n_x \times n_y$ 個の点を含み，これが $\sqrt{n_x n_y}$ 本の直線を含むとすると，1 点の投影にかかるオーバーヘッドは次のようになる☆．$16/\sqrt{n_x n_y}$ 回の加算，$12/\sqrt{n_x n_y}$ 回の乗算，$8/\sqrt{n_x n_y}$ 回の除算である．

3.2 平行平面間の投影の高速化

前節では，平行でない平面間の投影計算を平行な 2 直線間の線形なスケーリング計算に置き換えて高速化できることを示した．ここでは，平行平面間の投影計算の場合には，さらに高速に計算できる[3] ことを示す．

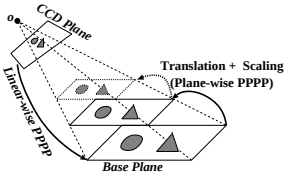


図 8 面単位の平面間透視投影
Fig. 8 Plane-wise Plane-to-Plane Perspective Projection

図 8 に示すように，平行平面間の透視投影は 2 次元の拡大縮小と平行移動として簡単化できる．

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = s \begin{bmatrix} X \\ Y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}, \qquad (4)$$

s と t_x, t_y はそれぞれスケーリング係数と平行移動ベクトルをあらわす．式 (4) に示すように，この変換は **2** 回の加算と **2** 回の乗算だけで行える．更に，画像のラスタスキャンと前節で述べた定数ベクトルの加算手法を併用すれば数値計算の回数はさらに減少できる．即ち，この場合には，両平面において，任意の平行線分割が平行線間の投影に適用でき，ラスタスキャンの場合には次のようになる．

$$\overrightarrow{x_{i+1}} = \overrightarrow{x_i} + (s_x, 0), \quad \overrightarrow{X_{i+1}} = \overrightarrow{X_i} + (S_X, 0). \qquad (5)$$

この計算には 1 点当り **2** 回の加算と，以下のオーバーヘッドしか要しない．加算：$2/\sqrt{n_x \times n_y}$，乗算：$2/\sqrt{n_x \times n_y}$.

この変換は 2 次元の幾何的変換であるので，2 次元の画像処理プロセッサを用いれば，容易に高速化することが可能である．

これまでに述べた，直線および平面単位の平面間投影を以下のように組み合わせることにより，効率の良い逆投影計算が行える．(1) 入力画像面から対象のシルエット画像をボクセル空間中の 1 つの平面（基準面）へ逆投影する．その際，前述の直線単位の平面間投影を適用する．(2) 基準面からその他の平面への投影は面単位の平面間投影を適用する．

3.3 比　　較

3 次元ボクセル空間は $n_V = n_x \times n_y \times n_z$ 個のボクセルを含むとすると，全体の算術演算回数は次のようになる．☆☆

☆ この場合，1 点当りの直線の本数は $\sqrt{n_x n_y}/(n_x n_y) = 1/\sqrt{n_x n_y}$ となる．1 本の直線につき，2 点の平面間透視投影，定数ベクトル $\overrightarrow{\delta}$ と $\overrightarrow{\Delta}$ を求めるための 2 次元ベクトルの減算と除算がそれぞれ 2 回必要である．したがって，加減算は $(2 \times 6 + 2 \times 2)/\sqrt{n_x n_y}$ 回，乗算は $(2 \times 6)/\sqrt{n_x n_y}$ 回，除算は $(2 \times 2 + 2 \times 2)/\sqrt{n_x n_y}$ 回，となる．

☆☆ 視体積は錐体の形をしているため，平行平面群のうち高い位置の平面について，投影計算の対象領域は低い位置より小さくなり投影計算は更に速いと考えられるが，以下の評価でこれを無視している．それについての厳密な評価はカメラ配置も考慮せねばならないからである．

表 1 演算回数の比較（$n_v = n^3$ の場合）
Table 1 Number of arithmetic operations in the case of $n_v = n^3$.

	Perspective Projection			Plane-to-Plane Perspective Projection (PPPP)			Linear-wise & Plane-wise PPPP		
n	add	mul	div	add	mul	div	add	mul	div
100	9.0E6	9.0E6	2.0E6	6.0E6	6.0E6	2.0E6	2.0E6	2.1E4	800
1000	9.0E9	9.0E9	2.0E9	6.0E9	6.0E9	2.0E9	2.0E9	2.0E6	8000

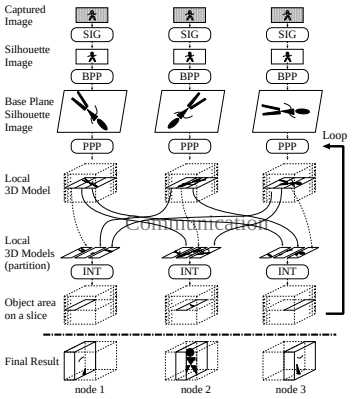


図 9 ローカルシルエット分割法の処理の流れ
Fig. 9 Process flow of the local silhouette division method:SIG: silhouette image generation, BPP: base plane projection, PPP:parallel plane projection, INT: intersection.

- **透視投影:** 加算 $9n_v$ 回，乗算 $9n_v$ 回 および除算 $2n_v$ 回.
- **平面間投影:** 加算 $6n_v$ 回，乗算 $6n_v$ 回 および除算 $2n_v$ 回.
- **直線および平面単位の透視投影:**
  直線単位の平面間投影: 加算 $4n_xn_y + 16\sqrt{n_xn_y}$ 回，乗算 $12\sqrt{n_xn_y}$ 回，除算 $8\sqrt{n_xn_y}$ 回
  面単位の平面間投影:加算 $(2n_xn_y + 2\sqrt{n_xn_y}) \times (n_z - 1)$ 回，乗算 $2\sqrt{n_xn_y}(n_z - 1)$ 回
  合計すると，加算 $2n_v + 2n_xn_y + \sqrt{n_xn_y}(2(n_z - 1) + 16)$ 回，乗算 $\sqrt{n_xn_y}(2(n_z - 1) + 12)$ 回，除算 $8\sqrt{n_xn_y}$ 回 となる．

表 1 に幾つかの数値例を挙げる．この表から，平行線および平行平面に基づき平面間投影によって，逆投影計算の高速化が行えると言える．

4. 視体積交差の並列化

多視点で得られた対象のシルエットは，以上の手法を用いて，平行平面群に逆投影することができる．ここでは，平行平面上に逆投影されたシルエットの交差計算の並列化について述べる．

図 5 に示した改良された VIM アルゴリズムには以下のような問題点がある．

(1) 各計算機で計算されたシルエットの交差計算を行い，その計算結果を他の計算機に転送する場合，データの到着待ち状態の計算機が発生し，システム全体の CPU パワーが十分引き出せない．

(2) 全ての平行平面上でシルエット交差計算ができて初めて 1 つのボリュームが復元されたことになるので，各計算機で行う逆投影計算はそれぞれの平面に関して同期を取る必要がある．しかし，実際には逆投影に要する計算時間はカメラ配置によってばらつきがあるため，この同期によって同様に待ち状態の計算機が発生してしまう．

(1) の問題については，計算すべきシルエットをさらに小さい部分に分割して，分割された各部分を計算機間で交換し合うことによって，シルエットの交差計算の並列度を向上させることが可能である．これを**ローカルシルエット分割法**と呼び，図 9 にその処理の流れを示す．

しかし，ローカルシルエット分割法では (2) の問題を解決することができない．この問題を解決するために，次に述べる並列化手法を提案する．

この手法は各計算機によって基準面に逆投影されたシルエット（基準シルエット）を通信によってコピーし，全ての計算機で全基準シルエットを持つ方法である．各 PC が全ての基準シルエットを持てば，任意の計算機で，任意の平面上での交差計算が他の計算機とは独立に計算することができる．したがって，シルエットの交差計算を行う平面集合を事前に各計算機に割り当てておけば，基準シルエットをコピーした後の交差計算には，通信は一切必要ない．この手法を基準シルエット多重化法と呼ぶ．その処理の流れを図 10 に示す．この方法では，基準シルエットのコピーを行う際の同期を除けば，他に同期を取る部分は必要なく，システム全体の CPU を有効に活用することができる．

5. 実　　験

この章では，平面間投影に基づく視体積交差法を実

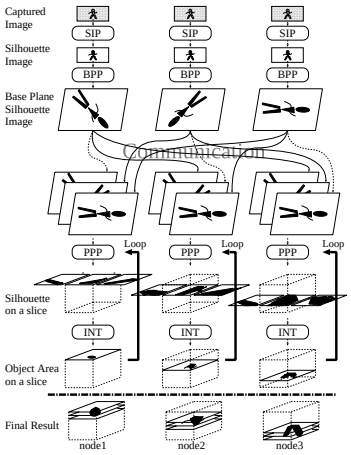


図 10 基準シルエット多重化法の処理の流れ
Fig. 10 Process flow of the base-plane duplication method

装した試作システムとそれを用いた実験結果について述べる．

5.1 PC クラスタシステム

試作システムで用いた PC クラスタの仕様は次の通りである．

- クラスタは 10 台の PC から構成される．各 PC には 2 つの PentiumIII 600MHz の CPU と 512MB のメモリが搭載されている．そのうち 9 台の PC にカメラ (SONY EVI-G20) が接続されており，残り 1 台は撮影時刻の同期をとるために使用する．これら PC の OS は LINUX である．
- すべての PC は Myrinet(Myricom 社製) と呼ばれる超高速ネットワークによって相互接続されている．このネットワークは全 2 重 1.28Gbps の通信帯域幅を持ち，2 台の PC 間の通信速度は実測値で約 100[MBytes/sec] である．

実験に使用したカメラは，RS-232C によってパン・チルト・ズームが制御できるカメラである．このカメラは，投影中心と回転中心がほぼ一致した視点固定カメラ (FVC)[10] とみなすことができる．この特性を利用して，内部パラメータのキャリブレーションが行える．また，複数の異なる視線方向で撮影された入力画像をひとつの仮想平面に投影すれば，1 枚のシームレスな超広角画像が得られる．この広角画像は，後述する床面を用いたカメラの外部パラメータのキャリブレーションにおいて，広範囲のデータを利用することを可能にし，精度の良いキャリブレーション結果を得るために不可欠なものである．

カメラ配置とキャリブレーション

9 台のアクティブカメラは図 11 に示すように，部屋の天井に設置してある．この配置で前述の広角画像を合成すると，床面はすべてのカメラの広角画像内に収まるようになる．つまり，床面は各カメラにとって共通視野に含まれる平面となる．これを利用すれば，以下のように，カメラの位置関係を計算することができる．2 つのカメラについて考える．平面である床面からそれぞれのカメラの撮影面への投影は平面間投影であり，これを行列 H_1, H_2 で表すとすると，Homography 行列 $M = H_2 H_1^{-1}$ を求めることができる．実際には，2 つのカメラによって撮影される 2 枚の画像上の同一平面内の 4 対の対応点が与えられれば，M を直接計算することができ，M を分解することによって，2 台のカメラ間の相対位置を表す回転行列 R と並進ベクトル $\vec{T}$ 及び床面の法線ベクトル $\vec{n}$ が求められる．☆すなわち，カメラの外部パラメータ・キャリブレーションも平面間投影のみで行える．

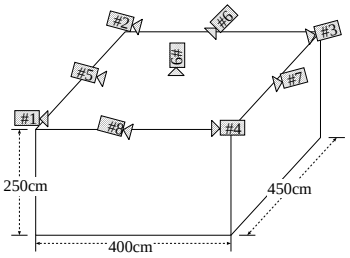


図 11 カメラの配置
Fig. 11 The camera settings

5.2 システムの実装

試作システムは，基準シルエット多重化法に基づいて設計され，シルエットは背景差分によって計算している．さらに，このシステムには，高速化のために以下のような機能を追加している．

(1) 各画像上で求められたシルエットに外接する二次元の矩形を求め，その範囲内だけで基準平面への逆投影計算を行う．
(2) 平行平面間の投影および交差計算を行う範囲は，上述の矩形を逆投影して得られる視体積の共通部分の内部に限定する．

これらの機能によって，計算範囲が限定され，通信量と投影計算の回数などが削減できる．

実装に当たっては，各 PC 上でも処理の並列化を行った．これは，実験に使用したシステムでは 1 台の PC に 2 つの CPU が搭載されているため，処理の並列化が可能なためである．並列化の方法としては，

☆ 一般にこの分解によって 2 通りの結果が得られるが，3 台のカメラを用いることにより，正しい分解を決定することができる．

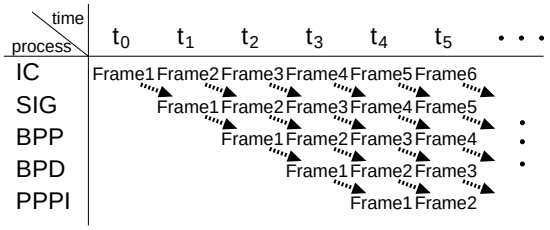


図 12 各ノード PC でのパイプライン処理
Fig. 12 Pipeline processing on node PC
IC: Image capture, SIG: silhouette image generation, BPP: base-plane projection, BPD: base-plane duplication, PPPI: parallel-plane projection and intersection

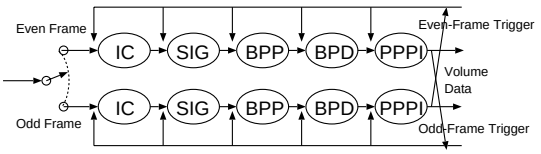


図 13 複数スレッドによるパイプライン処理の実装
Fig. 13 Multi-thread implementation of pipeline processing. IC: Image capture, SIG: silhouette image generation, BPP: base-plane projection, BPD: base-plane duplication, PPPI: parallel-plane projection and intersection

1 フレーム分の視体積交差計算を幾つかの処理ステージに分け，各ステージで異なる時刻のデータを並列に処理するパイプライン処理を採用した．実際の処理ステージは次の 5 つである．画像キャプチャ(IC), シルエット画像生成 (SIG), 基準平面への逆投影 (BPP), 基準シルエットの多重化 (BPD), 基準平面から平行平面への投影と交差計算 (PPPI)．各ステージは並列実行可能なスレッドを用いて実装しており，これらは図 12 に示すように，時間的に異なるデータを並列に処理するように設計している．つまり，IC ステージでフレーム i の入力画像をキャプチャしているとき，SIG ステージではフレーム $i-1$ における対象シルエット計算，BPP ステージではフレーム $i-2$ の基準シルエット生成，BPD ステージではフレーム $i-3$ の基準シルエットの通信，PPPI ステージではフレーム $i-4$ のボリュームデータ計算，が行われている．

これらの各スレッドは入出力バッファ上でのデータの上書きが起きないように，奇数フレーム用と偶数フレーム用に分けて 2 系統のスレッドを図 13 に示すように実装してあり，各系統のスレッドは奇偶のトリガーによって切り替わり，交互に実行される．

当然，各 PC には 5 つの CPU は搭載されていないので，実際に同時に実行できるスレッドは厳密には 2 つしかない．しかし，OS のスケジューリング機能によって，CPU 資源を効率良く使うようにスレッドが実行され，全体のスループットが向上するものと期待できる．

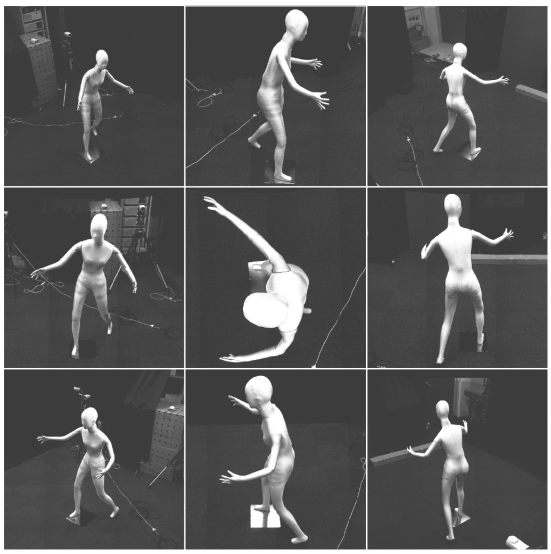


図 14 9 台のカメラで撮影した画像
Fig. 14 Examples of captured images by nine node PCs

5.3 実験結果

最初の実験は，形状復元精度に関するものである．入力画像サイズは 640×480 ピクセルとし，対象は，等身大のマネキンである．この対象は，$70\times70\times180\text{cm}^3$ の直方体領域内に存在することを実測により確認している．まず，9 台のカメラで背景を撮影し，次に対象を置き，撮影された画像と背景画像との間での背景差分によって，対象のシルエット画像を計算した．図 14 に実際の入力画像を示す．計算されたシルエット画像を基準平面である床面に投影して得られる基準シルエットを図 15 に示す．空間解像を 1cm^3 とした場合☆の復元結果は図 16 のようになる．この図において欠けている部分は，背景差分の処理で対象検出に失敗した領域である．復元結果を見ると，指などの細部まで復元されていることが分かる．これはカメラパラメータのキャリブレーション精度が高いことを示唆している．

2 つ目の実験は計算時間に関するものである．異なるボクセルサイズについて，各ステージの処理時間（図 17），およびその合計時間とボリューム復元のスループット（表 2）を計測した．

まず，図 17 から，次のことが確認できる．

- IC ステージと SIG ステージの処理時間はボクセ

☆ すなわち，平行平面間の間隔を 1cm とし，各平面を 1cm^2 の単位で量子化している．

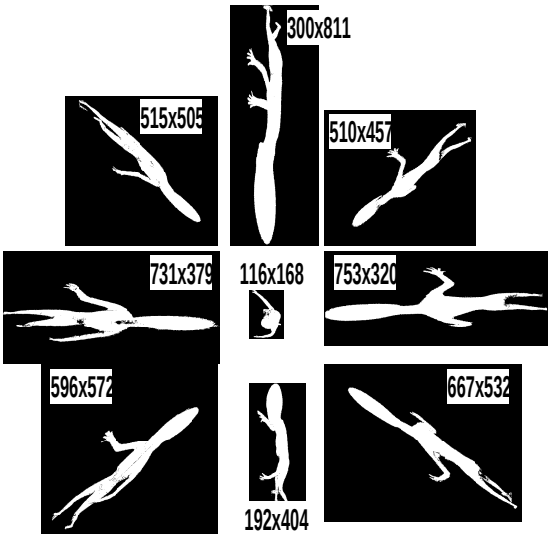


図 15 基準平面に投影された画像
Fig. 15 Examples of base-plane images

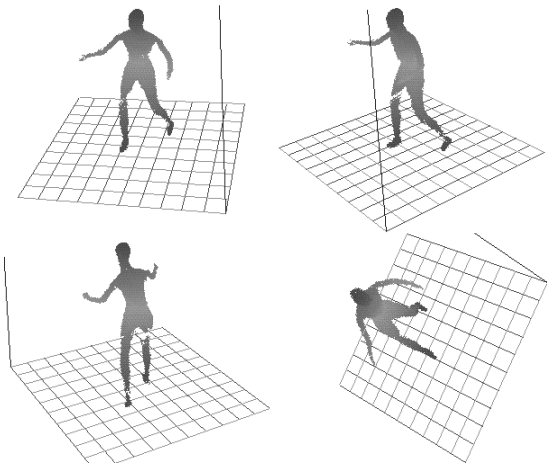


図 16 再構成された物体形状
Fig. 16 Reconstructed volume of the object

ルサイズに依存しない．

- BPP, BPD と PPPI ステージの処理時間はボクセルサイズに依存する．

特に，PPPI，BPP の各ステージの処理時間はボクセルサイズのほぼ 3 乗と 2 乗に反比例して減少しており，理論通りの結果になっている．BPD ステージの処理時間はボクセルサイズの 2 乗に反比例していないが，ボクセルサイズの増大に伴って減少している．

表 2 には 1 フレームの処理につき，各ステージで要する時間の総和と実際のボリューム復元間隔が示してある．試作システムでは，パイプライン処理を行っ

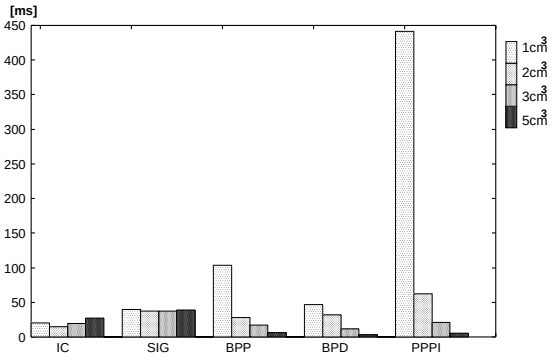


図 17 各ステージでの処理時間（50 フレームの平均処理時間）
Fig. 17 Average elapsed time in processing stages for 50 frames
IC: Image capture, SIG: silhouette image generation, BPP: base-plane projection, PPPI: parallel-plane projection and intersection, BPD: base-plane duplication

ているので，ボリューム復元間隔の方が，総処理時間よりも短くなっている．この効果を表では”Pipeline Factor”として示した．1 秒当りに復元できるボリューム数も”Volume/sec”の欄に示した．ボリューム復元の速度はボクセルサイズが大きくなるにつれて向上するものの，ある一定の値に収束している．これは，IC と SIG ステージの処理がボクセルサイズに依存せず，ボクセルサイズが大きくなると，これらの処理時間が全体の処理時間に占める割合が高くなるためであると考えられる．

技術的には，SIG ステージの処理は MMX 命令の利用によって高速化することが可能である．また，ボクセルサイズが大きい場合には入力画像サイズも小さくすることができる．これらの改善を施し，IC と SIG ステージでの処理時間が無視できる程度に小さくできると仮定すると，ボリューム復元速度は，ボクセルサイズが $1\text{cm}^3, 2\text{cm}^3, 3\text{cm}^3, 5\text{cm}^3$ であるとき，それぞれ 1.9, 13.6, 26.6, 87.6 [Volume/sec] となる．

3 つ目の実験では，カメラの視線方向を固定にして、対象の身体動作を多視点で連続撮影して得られた複数のビデオシーケンスから，オフライン処理によって身体の表面形状を復元し，テクスチャマッピングを行った．図 18 に復元された身体動作を仮想的なカメラワークによって再映像化した画像の一部を示す．この例からも分かるように，本システムを用いて，任意の視点から動きのある対象の観測を行うことができる．

6. ま と め

本論文では，平面間投影に基づく並列 3 次元物体

表 2 処理時間

Table 2 Processing speed
Total Elapsed Time: sum of the elapsed time in all subprocesses, Throughput Time: time interval between the outputs, Volume/sec.: number of volumes reconstructed per 1 second, Pipeline Factor=(Total Elapsed Time)/(Throughput Time)

Voxel Size	Total Elapsed Time	Throughput Time	Volume /sec.	Pipeline Factor
$1cm^3$	651.019ms	564.413ms	1.77	1.15
$2cm^3$	145.127ms	114.065ms	8.77	1.27
$3cm^3$	107.031ms	80.873ms	12.37	1.32
$5cm^3$	80.484ms	64.013ms	15.62	1.25

$\rightarrow time$

図 18 人物動作の連続的再構成例
Fig. 18 Example of continuous human body reconstruction

形状復元法について述べた．我々は，カメラの視線方向をコントロールすることにより，形状復元が行える範囲を拡大することを目指しているため，カメラパラメータの変更に適さないテーブルルックアップによる投影計算の高速化は採用しなかった．また，濃淡やカラー等の情報に着目した形状復元手法[4] への拡張の可能性も考慮して，2 値シルエット画像に特化した物体形状の octree 表現なども利用しなかった．これらの高速化に代る手法として，3 次元ボクセル空間を平行平面群に分割し，平面間投影計算の高速化により，高速な物体形状復元が行えることを実証した．

本論文の内容を要約すると，以下のようになる．

- 平面間投影が視体積交差法の高速化にとって有効であることを示した．
- 平面間投影の 2 つの高速化法を示した：非平行平面間の高速投影手法として直線単位の平面間透視投影法を提案し，平行平面間の投影に関しては，Space Sweep[3] などで用いられている面単位の平面間透視投影が有効であることを示した．
- 計算機間の同期を必要としない視体積交差計算のために，基準シルエット多重化法という視体積交差法の並列アルゴリズムを提案した．
- より効率の良い並列アルゴリズムの実装法としてパイプライン処理が有効であることを示した．

本論文に示した方法に基づいて，3 次元運動物体の幾何学的な情報だけでなく，色やテクスチャなどの物体表面の光学的情報も復元することが可能である．復元された情報は，図 18 に示すように任意の視点・視線・ズームで観測することが可能である．言い換えると，冒頭で述べたモーションキャプチャシステムの拡張が意味することは，物体の「見え」に関わる，幾何学的，光学的情報の全てを復元することである．

我々は，このような目的にしたがって，3 次元運動物体の「見え」に関わる全ての情報を復元・圧縮・伝送・再生する「3 次元ビデオシステム」の構築を行っている．このシステムを構築するためには，数多くの課題を解決する必要があるが，今後は，カメラアクションの導入と，ビデオレートでの物体形状復元システムの実現を行う予定である．

謝　　辞

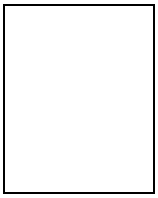
本研究を行うにあたり、日本学術振興会未来開拓学術研究推進事業 (JSPS-RTTF 96P00501) の補助を受けた。

参 考 文 献

1) H. Baker. Three-dimensional modelling. In *Fifth International Joint Conference on Artificial Intelligenc*, pages 649–655, 1977.
2) B.G. Baumgart. Geometric modeling for computer vision. Technical Report AIM-249, Artificial Intelligence Laboratory, Stanford University, October 1974.

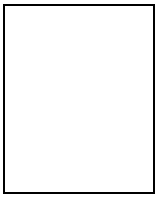
3) R. T. Collins. A space-sweep approach to true multi-image matching. In *IEEE Computer Vision and Pattern Recognition*, pages 358–363, 1996.
4) K. N. Kutulakos and S. M. Seitz. A theory of shape by space carving. In *IEEE International Conference on Computer Vision*, pages 307–314, 1999.
5) A. Laurentini. How far 3d shapes can be understood from 2d silhouettes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):188–195, 1995.
6) W. N. Martin and J. K. Aggarwal. Volumetric description of objects from multiple views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):150–158, 1987.
7) M. Potmesil. Generating octree models of 3d objects from their silhouettes in a sequence of images. *Computer Vision,Graphics, and Image Processing*, 40:1–29, 1987.
8) P. Srivasan, P. Liang, and S. Hackwood. Computational geometric methods in volumetric intersections for 3d reconstruction. *Pattern Recognition*, 23(8):843–857, 1990.
9) R. Szeliski. Rapid octree construction from image sequences. *CVGIP: Image Understanding*, 58(1):23–32, 1993.
10) 和田, 浮田, 松山,"視点固定型パン・チルト・ズームカメラとその応用", 信学論 D-II, Vol. J81-D-II, No.6, pp. 1182-1193, 1998, 6 月

(平成 12 年 2 月 4 日受付)
(平成 12 年 5 月 11 日採録)



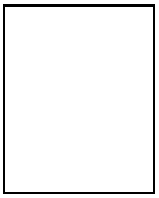
ウ小軍 (う しょうぐん)

1998 年京都大学工学部電気工学第二学科卒．2000 年同大学院情報学研究科知能情報学専攻修士課程修了．現在同大学院同専攻博士後期課程在籍．実時間能動的 3 次元形状復元の研究に従事．



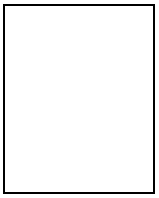
和田俊和 (わだ としかず)（正会員）

平成 2 年東工大大学院博士課程修了．同年岡山大学工学部助手．平成 9 年京都大学大学院工学研究科助教授．工学博士．画像理解，パターン認識の研究に従事．平成 7 年 David Marr 賞．平成 9 年 情報処理学会山下記念研究賞．平成 11 年 電子情報通信学会論文賞各受賞．



東海彰吾 (とうかい しょうご)（正会員）

1991 名古屋大学工学部情報工学科卒．1996 同大学院工学研究科情報工学専攻博士課程修了．同年京都大学工学部助手．2000 福井大学工学部講師となり現在に至る．コンピュータグラフィックス，コンピュータビジョンの研究に従事．博士（工学）



松山隆司（まつやま たかし）（正会員）

1976 年京大大学院修士課程修了。京大助手、東北大助教授、岡山大教授を経て 1995 年より京大大学院電子通信工学専攻教授。現在同大学院情報学研究科知能情報学専攻教授。工博。画像理解、人工知能、分散協調視覚の研究に従事。1980 年情報処理学会創立 20 周年記念論文賞、1990 年人工知能学会論文賞、1993 年情報処理学会論文賞、1994 年電子情報通信学会論文賞、1995 年第 5 回国際コンピュータビジョン会議 Marr Prize、1996 年国際パターン認識連合 Fellow、1999 年電子情報通信学会論文賞、2000 年画像センシングシンポジウム優秀論文賞。人工知能学会評議員、情報処理学会理事。