コンピュータビジョンと
イメージメディア 111－8
（1998．5．27）

能動カメラによる運動物体追跡と実時間3次元形状復元

出口 光一郎　鏡 慎吾　嵯峨 智　本谷 秀堅　（東大工学部）

動いている物体に対し、カメラのパン、チルト角を能動的に制御して対象上のある点を注視し続け，得られた画像の系列から実時間で3次元形状を復元するシステムを構築した．物体上のある点を画像上のある一点に常に留めるようにカメラの首振りを制御する（注視制御と呼ぶ）と，物体の運動がその点を中心とした回転運動とみなせる画像が得られる．この画像を用いると，注視点以外の点のオプティカルフローから，その物体の3次元形状を復元できる．この原理に基づいて，並列画像処理ボード，および，首振りカメラを中心として運動物体追跡による実時間3次元形状復元システム構成した．一般にオプティカルフローの計算はコストが高く，実時間化への障壁となっている．これを克服するため，本研究では並列画像処理ボードを用いて処理を行う．そして，計算の効率化をはかることで，画像の入力から3次元形状の復元までの一連の処理を通常のビデオフレーム間隔(33ms)内で完了することができた．

Real-Time Object Tracking and 3D Reconstruction by Active Camera

Koichiro Deguchi　Shingo Kagami　Satoshi Saga　Hidekata Hontani
(University of Tokyo)

This paper reports a system to track a moving object and recover its 3D shape in real time. When a camera follows the object by fixing a point at the image center, the obtained images are just equivalent to those obtained when the object rotates around the point. Then, we can recover its 3D shape from the optical flows of the object points other than the fixed point. Based on this principle, we constructed a real time system by using a pan-tilt camera and a parallel processing unit. 256 PE's in the parallel processing unit overcome the computation costs of the optical flows by effectively rearranging the image data within them, and achieve the total computation from image input to 3D reconstruction within video-frame interval (33ms).

本研究は，日本学術振興会未来開拓学術研究推進事業，「分散協調視覚による動的3次元状況理解」研究の一部として行われた．

1 はじめに

動いている物体に対し，カメラのパン，チルト角を能動的に動かして対象上のある点を注視し続ける制御を行い，得られた画像の系列から実時間で対象の 3 次元形状を復元するシステムの構築について報告する．

運動している物体上のある点を画像上のある一点に常に留めるようにカメラのパン，チルトを制御する（注視制御と呼ぶ）と，物体の運動がその点を中心とした回転運動とみなせる画像が得られる（Fig.1,2,3）ことを用いると，注視点以外の点のオプティカルフローから，その物体の 3 次元形状を復元できる．

まず，要素技術として，時系列的に隣接する 2 画像から注視点の動きと，その周りの点に対するオプティカルフローを検出する必要がある．一般にオプティカルフローの計算はコストが高く，実時間化への障壁となっている．これを克服するため，本研究では並列プロセッサを持つ画像処理ボードを用いて並列処理を行う．そして，計算の効率化をはかることで，オプティカルフローを高速に計算する手法を開発，実装し，画像の入力から 3 次元形状の復元までの一連の処理を通常のビデオフレーム間隔 (33ms) 内で完了することができた．実験により，3 次元形状の復元が実時間で行われていることを確認した．

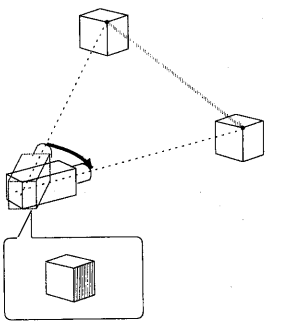


Fig.1: 動対象のカメラ追跡

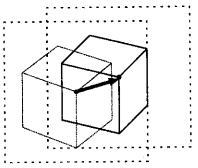


Fig.2: 注視点制御

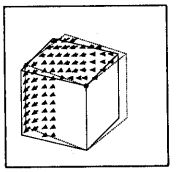


Fig.3: 注視点制御画像

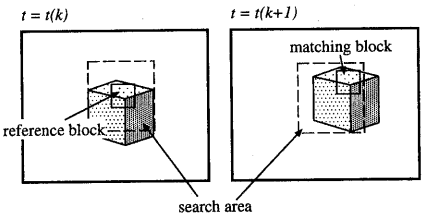


Fig.4: ブロックマッチング法

2 オプティカルフローの検出とカメラの注視点制御

まず，問題設定について述べる．パン・チルトの 2 自由度で動く首振りカメラによって対象となる運動物体を撮影する．このとき，対象のある点がつねに画像の中心に位置するようにカメラを注視制御する．

ここで画像中心とは，カメラの投影中心から画像面に対して下ろした垂線の足である．これは，出力された画像の中心であるピクセルとは必ずしも一致しない．また，カメラのパン・チルトの回転中心は，カメラの投影中心と一致しているものとする．本システムで用いた首振りカメラも，ほぼ，この特性を持っている．

2.1 オプティカルフローの検出

画像上の対象の 2 次元の動きは，視覚センサに対する対象の 3 次元空間内での動きを，画像面へ投影したものである．この画像の 2 次元の速度の場をオプティカルフローと呼ぶ [2]．オプティカルフローを求める方法には，大きく分けて，画像の濃淡値の時間・空間勾配の関係を用いる方法と，画像の間の対応点を探す方法とがある．ここでは，画像間の対応点を検出することによって計算する．計算のコストは高いが，安定にオプティカルフローが得られるからである．

対応点を探すことによる計算は，時系列中の k 番目の画像上のある点 (x_k, y_k) に対して，その次の画像の中から対応する点 (x_{k+1}, y_{k+1}) を見出し，この点の移動ベクトル

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} x_{k+1} - x_k \\ y_{k+1} - y_k \end{pmatrix} \qquad (1)$$

とし，その場を求めるものである．

本研究では，ブロック間のマッチングを取る方法で，オプティカルフローを検出している (Fig.4)．すなわち，ある点 (x_k, y_k) の対応点を探すために，(x_k, y_k) を中心としたブロック (参照ブロック) を考え，そのブロックと最も「似ている」ブロックを次の画像におけるそのまわりの領域 (サーチ領域) から探す．そして，マッチしたブロックの中心点を対応点 (x_{k+1}, y_{k+1}) として採用する．

ブロックマッチングによる方法では，ある一定の大きさの領域どうしで対応を取るので，原理的にローカルな情報に基づく勾配法に比べて見え方の変化，ノイズや照明の変化，ぶれ等の影響を受けにくい [4]．

あるブロックとあるブロックがどのくらい「似ている」かの評価にはさまざまな方法が考えられる．本研

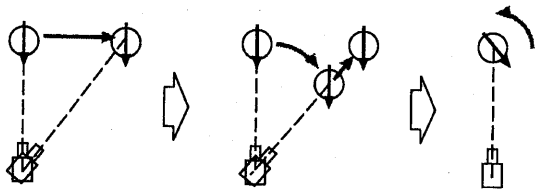


Fig.5: 対象の運動と注視点制御による画像

究では計算のコストを考慮して，単純に「対応する画素ごとの濃淡値の差の絶対値の和」を用いる．すなわち，(x_k, y_k) を中心とした参照ブロック $R(x_k, y_k)$ に対して，

$$D(\Delta x, \Delta y) = \sum_{(x,y)\in R(x_k,y_k)} |I(x, y, t_k) - I(x + \Delta x, y + \Delta y, t_{k+1})| \quad (2)$$

を最小にするような $(\Delta x, \Delta y)$ を $\Delta x_{min} \leq \Delta x \leq \Delta x_{max}$, $\Delta y_{min} \leq \Delta y \leq \Delta y_{max}$ の範囲内で求め，それを (x_k, y_k) におけるフローとする．ただし，$\Delta x_{min}, \Delta x_{max}, \Delta y_{min}, \Delta y_{max}$ はサーチ領域の大きさに合わせて定める．

2.2 カメラの注視制御

カメラを注視制御するためには，注視点が次の時刻においてどこに移動したかを求める必要がある．これは，オプティカルフローを検出する際に各点の移動を求めるのと全く同じことであり，ここではやはりブロックマッチング法を用いる．

すなわち，注視すべき点を中心としたブロックを参照ブロックとし，マッチするブロックを次々と探していくことで，時系列中のあらゆる画像における注視点の位置を検出する．マッチングが完全であるとすれば，この点が常に画像中心にくるようにカメラに制御量を与えることで，注視制御が実現できる．

3 オプティカルフローからの形状復元

3.1 対象の運動の成分分解

対象物体の移動は，カメラを中心とする視線方向の直線移動と，カメラを中心とした回転移動の組合せで実現できる (Fig.5)．

運動が回転成分のみであるときは，得られる画像は，いまカメラは常に対象の一点を注視していることを考えると，カメラを固定して物体を注視点まわりに自転させた場合の画像と等価であることがわかる (同図)．また，この場合のパン・チルト回転角は，カメラのパン・チルト回転角と大きさが同じで符号が逆向きになる．

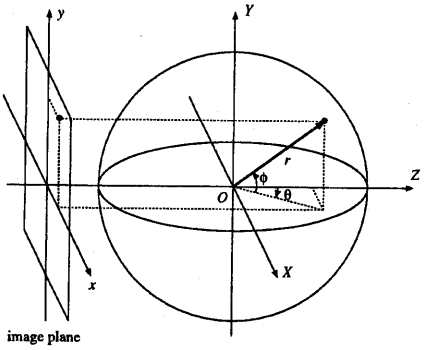


Fig.6: 注視点まわりの回転から形状を復元する

一方，直線移動の成分は，画像の拡大・縮小を引き起こす．画像面から距離 Z の位置にある点が，Z 軸方向に D だけ遠ざかると，画像面での座標は $Z/(Z+D)$ 倍になる．ここで対象が，カメラからの距離に比べて十分に小さいと仮定すれば，画像面から注視点までの距離を Z_0 として，対象の画像は倍率

$$\frac{1}{k} = \frac{Z_0}{Z_0 + D} \quad (3)$$

で一様に拡大・縮小するとしてよい．よってこの倍率を用いて，遠ざかっているときには拡大，近づいているときには縮小してやることで，視線方向の動きを打ち消すことができる．実際には，画像が実時間で連続して得られるとすると，連続する画像間では視線方向の運動成分は微少で無視できると考えられる．

以下，まず，視線方向の移動がないものとして議論をする．この場合，問題は注視点まわりに既知の回転速度でパン・チルト回転する対象の 3 次元形状を復元することである．

3.2 注視点まわりの回転からの 3 次元形状復元

注視点から距離 r，パン角 θ，チルト角 ϕ の位置にある点の直交座標系での位置 (X, Y, Z) は，Fig.6 より，

$$X = r\cos\phi\sin\theta \quad (4)$$

$$Y = r\sin\phi \quad (5)$$

$$Z = r\cos\phi\cos\theta \quad (6)$$

となる．ただし，原点は注視点とする．式 (4), (5) を微分して，

$$\Delta X = r\cos\phi\cos\theta\Delta\theta - r\sin\phi\sin\theta\Delta\phi \quad (7)$$

$$\Delta Y = r\cos\phi\Delta\phi \quad (8)$$

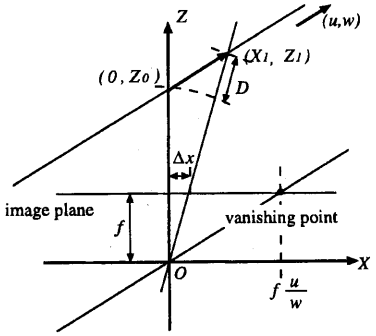


Fig.7: 光軸方向の移動距離 D を求める

を得る．これらより，奥行き方向の距離 Z は，

$$Z = \frac{\Delta X}{\Delta\theta} + XY\frac{\Delta\phi^2}{\Delta\theta\Delta Y} \quad (9)$$

として与えられる．

画像への投影は正射影であるとすると，画像面上の像の位置 x, y は X, Y と等しく，$\Delta X, \Delta Y$ はオプティカルフロー $\Delta x, \Delta y$ と等しい．$\Delta\theta, \Delta\phi$ はカメラのパン・チルト角変化の符合を逆にしたものとして既知である．よって，これらよりこの点の距離情報 Z を得ることができる．

3.3 直線運動成分に対する補正

視線方向の移動がある場合，それに合わせて画像の拡大・縮小を行う必要がある．この拡大・縮小率は，カメラの焦点距離 f(既知とする) とオプティカルフローの消失点より求めることができる．

対象の各点の 3 次元空間内での移動が平行移動であると，画像上の各点におけるオプティカルフロー線はすべて同一の消失点をもつ．この消失点 (p_x, p_y) は，3 次元空間内での移動ベクトル (u, v, w) で，

$$\begin{pmatrix} p_x \\ p_y \end{pmatrix} = \begin{pmatrix} f\frac{u}{w} \\ f\frac{u}{w} \end{pmatrix} \quad (10)$$

と表される [1].

Fig.7 において，注視点 $(0, 0, Z_0)$ の移動ベクトルを (u, v, w) とおくと，注視点のこの瞬間の移動軌跡を延長した直線は，

$$\frac{X}{u} = \frac{Y}{v} = \frac{Z - Z_0}{w} \quad (11)$$

と書ける．また，注視点の移動ベクトルを画像面への投影したベクトル Δx(既知) の始点を原点としたときの終点とカメラ中心を結んだ直線は，

$$\frac{X}{\Delta x} = \frac{Y}{\Delta y} = \frac{Z}{f} \quad (12)$$

と書ける．移動後の注視点の位置 (X_1, Y_1, Z_1) は，この 2 直線の交点となるので，

$$Z_1 = \frac{f\frac{u}{w}}{f\frac{u}{w} - \Delta x} Z_0 \equiv \frac{f\frac{v}{w}}{f\frac{v}{w} - \Delta y} Z_0 \quad (13)$$

$$X_1 = \frac{\Delta x}{f} Z_1 \quad (14)$$

$$Y_1 = \frac{\Delta x}{f} Z_1 \quad (15)$$

として与えられる．

これらより，対象の Z 方向の移動距離 D は，

$$D = \sqrt{{X_1}^2 + {Y_1}^2 + {Z_1}^2} - Z_0 \quad (16)$$

と得られる．ここに fu/w は消失点として観測できることに注意すれば，D の計算に必要なものはすべて既知であることがわかる．

この D を式 (3) に代入することで拡大・縮小率 k を得る．ここで，D は Z_0 の倍数なので，k は Z_0 に依存しない．この k を用いて，隣接 2 フレームの後の方の画像だけを倍率 k で拡大・縮小すれば，光軸方向の移動を打ち消すことができる．実際には画像の拡大・縮小を行う必要はなく，通常通り求めたオプティカルフローを (u, v) を用いて，

$$\begin{pmatrix} u' \\ v' \end{pmatrix} = \begin{pmatrix} k(x+u) - x \\ k(y+v) - y \end{pmatrix} \quad (17)$$

を計算すればよい．ただし，(x, y) は画像面での位置とする．

4 システムの実装

4.1 並列処理によるオプティカルフローの計算

画像処理は，局所的な演算を画像上のいたるところで一斉に行うことが多く，SIMD(Single Instruction and Multiple Data stream) 型の並列計算機構を用いて自然に，効率良く実装できるものが多い．

このような SIMD 型計算機構をもつハードウェアとして，本研究では NEC 製の IMAP-VISION LSI を搭載した，IMAP-VISION ボード RVS-10G (Fig.8) を用いた．IMAP-VISION は，Linear Processor Array(LPA) 型と呼ばれるアーキテクチャ (Fig.9) を採用している．これは多数のプロセッ

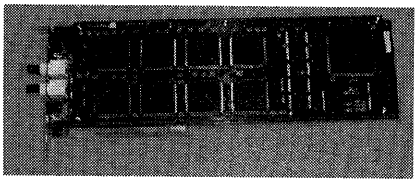


Fig.8: IMAP-VISION の概観

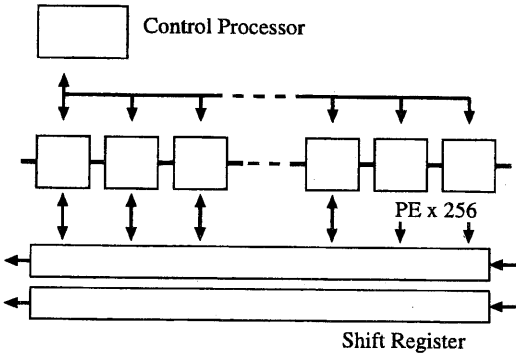


Fig.9: IMAP-VISION の概念図

サ (PE, Processing Element) を 1 次元に結合したもので，画像の水平方向の 1 列に対して一斉に演算を行うことができる．IMAP-VISION のプロセッサアレイは，1 チップ内で 32 個の PE が 1 次元的に接続されており，これを 8 チップ接続することで PE の総数は 256 個となる．実際に画像を処理する際には，典型的には縦の 1 列を 1 つの PE の中の配列として扱う．また，PE 間でデータのやりとりも行うことができるので，それ以外の柔軟な使い方も可能である [6].

IMAP-VISION には，このプロセッサアレイとは別にコントロールプロセッサがあり，ホストコンピュータとのインタフェース，プロセッサアレイの制御，割り込み処理等の機能を持つ．インストラクションの流れは単一なので，このコントロールプロセッサがプログラムカウンタを持ち，コントロールプロセッサ自信とプロセッサアレイに分配する．画像の入出力は，ビデオインタフェースと接続されたシフトレジスタを通して行われる．

ビデオインタフェース部で 30Hz でサンプリングされた，256 × 240 画素の 8bit 濃淡画像系列について，以下の処理を行った．

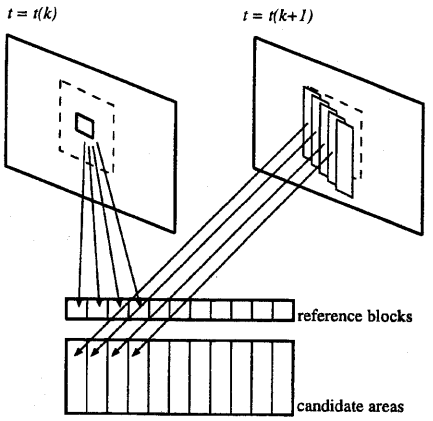


Fig.10: 参照ブロックとサーチ領域を再配置する

4.2 ブロックマッチングによるオプティカルフローの計算

ブロックマッチング法によりオプティカルフローを計算する際には，ある 1 点における移動ベクトルを求めるために，

(i) 参照ブロックと，ある候補ブロックの間で，対応する画素どうしの差をとりそれらの総和を求める．

(ii) 候補ブロックを，サーチ領域の中で動かしながら (i) を繰り返す．

必要がある．さらにこれを，画像中で移動ベクトルを求めたいすべての点に対して行わなければならない．

これを並列処理で行う場合，アルゴリズムとしては

- 各 PE に並列に各点の移動ベクトルを計算させる

ものと，

- ある点の移動ベクトルをすべての PE を使って計算し，各点について繰り返す

ものとが考えられる．このうち後者は，計算点を自由に選ぶことができる (規則的に位置していなくてもよいし，あとから追加・変更が容易にできる) という利点がある．本研究ではこちらの方法をとった．

実際にマッチングをとる際には，Fig.10 のように参照ブロックと，サーチ領域をプロセッサアレイ上に再配置する．参照ブロックは単純に同じパターンを r ピクセルごとに繰り返す，ただし r は参照ブロックの横幅のピクセル数とする．サーチ領域は，まず左端から r ピクセル分だけ切りだし，次に 1 ピクセルだけず

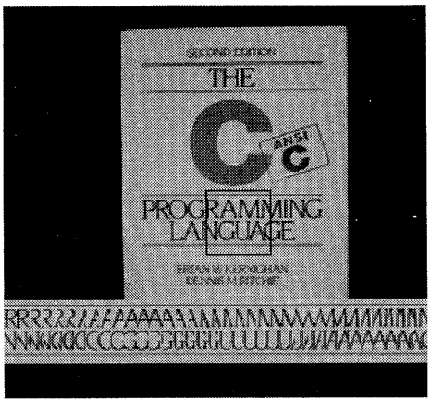


Fig.11: 再配置されたサーチ領域（中央の四角内をサーチ領域とすると，それを下部に示すような縦の1ラインずつを各1PEへ配置する）

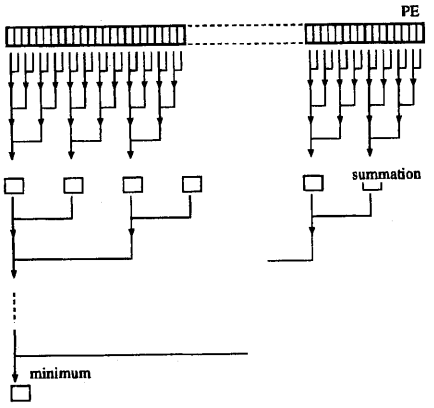


Fig.12: 横方向の集計はピラミッド状に行う

らしてやはり r ピクセル切りだし…というように，それぞれ隣と1ピクセルだけずれたものが順番に並べられる (Fig.11).

一度このように配置してしまえば，対応する画素ごとの差，およびその縦方向の総和はすべて，それぞれのPE内でローカルに計算できる．その後，自分の右となり r ピクセルの和を計算することで各ブロックにおける総和が (PE番号) $\bmod r = 0$ であるPEに得られる．(PE番号は左から順に0から数えるとする．) これらの中から最小値を選ぶことで，移動ベクトルが決定できる．横方向の集計は，Fig.12 のようにピラミッド状に行なうことで効率良く行うことができる．

$r = 8$ である場合には，サーチ領域を横幅32ピクセルとすることで，256個のPEをフルに活用することができる．サーチ領域を横幅16ピクセルとするとPEの半分は遊んでしまうが，この場合はサーチ領域を上下に分割して，余った半分のPEに割り当ててやればよい．

しかし，サーチ領域をあまり小さくする（この1/4以下）と，再配置にかかるオーバヘッドが相対的に大きくなるため，あまり効果的ではない．このような場合にはむしろ，移動ベクトルの計算点を4ヶ所並行して計算する方が効率がよい．

ブロックマッチング法によるオプティカルフローを計算する場合，参照ブロックの濃淡値が一様な場合には正確な移動ベクトルを求めることができない [4]. そこで，参照ブロックの各画素の濃淡値の分散

$$\sigma^2 = \frac{1}{N}\sum_{i=0}^{N-1}(I_i - \mu)^2 \qquad (18)$$

を，信頼性の評価のために用いることができる．ただし，$I_0, \ldots I_{N-1}$ は，参照ブロック内の各画素の濃淡値，μ はその平均である．これにより，あるしきい値より σ^2 が小さい点における移動ベクトルは，信頼性がないものとして3次元形状の復元には用いない．

また，移動ベクトルの決定時に用いた，式(3)の最小の値があるしきい値を越えている場合も，正しいマッチングがとれていないと考えられるので，やはり形状復元には用いない．

4.3 3次元形状の計算

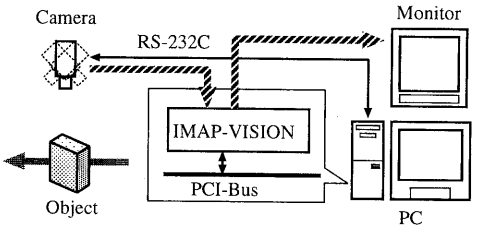
カメラを中心とした座標系における視線（奥行き）方向の対象の動きは，画像の拡大・縮小を引き起こし，この拡大・縮小率はオプティカルフローの消失点の座標から計算できることはすでに述べた．しかし，隣接するビデオフレーム間では対象の視線方向の動きは小さく，ほとんど無視できる．このときは，オプティカルフローが求められてしまえば，各点の3次元空間内での位置は，式(9)を用いて計算することができる．

本システムでは，オプティカルフローまでをIMAP-VISIONで並列計算を行い，それ以降の前節の原理に従った3次元形状復元の計算はホストコンピュータに結果を転送してから行った．

5 実験と結果

5.1 実験システム

前章まで述べた方法を用いて，実際に物体の3次元形状を復元する実験を行った．実験に用いたシステムをFig.13に示す．



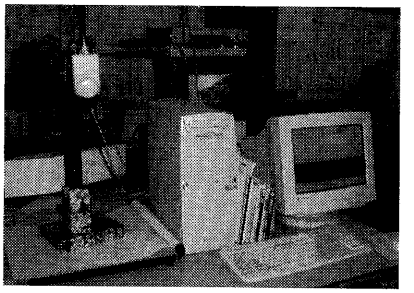


Fig.13: 実験に用いたシステムの構成と外観

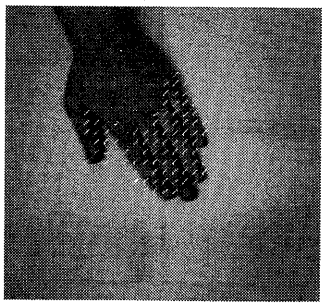


Fig.14: オプティカルフローの検出結果

ビデオカメラには，SONY 製の首振りカメラ EVI-G20，並列処理ボードには，前章で述べた IMAP-VISION ボードを用いた．ホストコンピュータは Gateway2000 製の PC/AT 互換機 G6/266 (CPU: PeutiumII/266MHz) 上で OS として，FreeBSD 2.2.1-RELEASE を用い，IMAP-VISION は PCI バス上に接続した．

5.2 オプティカルフローの検出

まず，固定した状態のカメラの前で手を動かし，そのときのオプティカルフローを検出した結果を Fig.14 に示す．

オプティカルフローは画像の中央 128 × 128 ピクセルに対して 8 ピクセルおきに 256 点について求め，

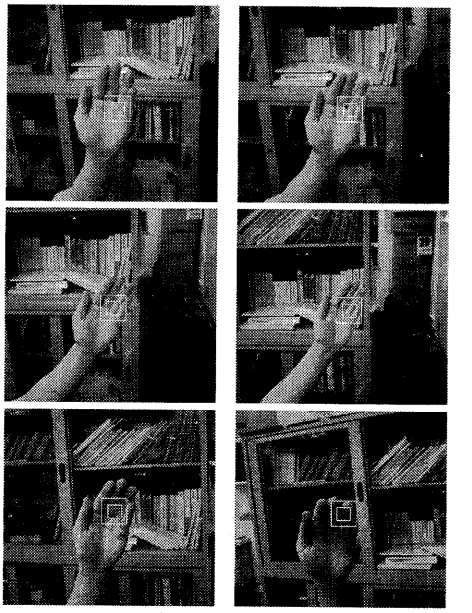


Fig.15: 運動物体の追跡結果 (左上 → 右下)

すべての点に対する計算は 23ms (1 点あたり 90 μs) であった．参照ブロック，サーチ領域の大きさはともに 8 × 8 ピクセルとした．

5.3 運動物体の追跡と注視制御

次に，カメラの前で動かした手を注視して追跡した結果を Fig.15 に示す．各図の中央近くの 2 つの白い四角形のうち，内側の小さい方が参照ブロックであり，外側の大きい方がサーチ領域である．

参照ブロックの大きさは 16 × 16 ピクセル，サーチ領域はそのまわり 16 × 16 ピクセル分として対応点を追い続けた．カメラの制御は，単純な PID 制御，すなわち，注視点の変位とその微分，積分をカメラの速度にフィードバックして，注視点が常に画像中心にくるように行った．この制御方法を，例えばカルマンフィルタ等を用いて注視制御の精度を上げれば，より正確な復元が可能であると考えられる．

追跡の全過程を通してつねに同じ点を追い続けることができているわけではない．少しずつ誤差が蓄積するため注視点の位置にずれが生じている．後に 3 次元形状を復元する際は，注視制御は完全に行われていることが前提となるので厳密にはこれは問題となるが，復元は隣接する 2 フレーム (あるいは次節で述べるように連続した数十フレーム分の統合) という短い時間内で行われるため，この影響は無視し得る．

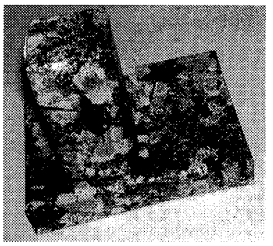


Fig.16: 3 次元形状の復元実験に用いた対象

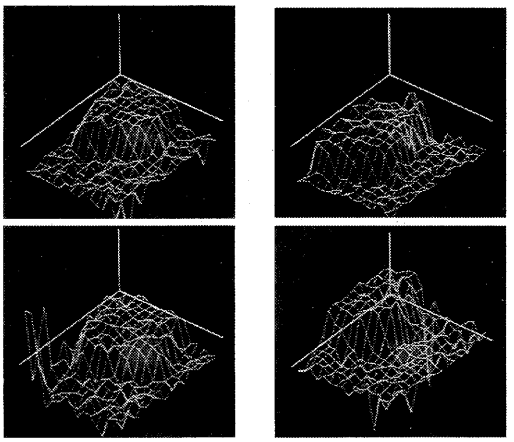


Fig.17: 3 次元形状の復元結果

5.4 3 次元形状の復元

対象として最も単純な直方体を 2 つ重ねたもの (Fig.16) を用いたときの実験結果を，以下に示す．

追跡と同時にオプティカルフローも計算される．求められたオプティカルフローは直ちにホストコンピュータに送られ，3 次元的位置が計算されて，X Window System の画面上にリアルタイムで表示される．

Fig.17 は，33ms ごとに得られる復元形状の例である．高さの異なる 2 つの面が検出されている．

本研究で用いた方法では，3 次元形状は隣接する 2 フレームから復元され，1 秒間に 30 組ずつ，復元された 3 次元形状のデータが出力される．これらの一つ一つは，対象の微小な動きから推定されたものであり，復元の原理が本質的には三角測量であることを考えると，あまりよい精度は期待できない．復元の精度をあげるためには，これらのデータを複数統合して，より大きな動きを元にした復元を行うことが必要となる．実際，連続する 30 枚を足し合わせ移動平均をとる実験を行ったところ，大きなノイズを低減する効果を得た．

実際にはこの時系列データ 30 組分を得る時間のうちに，対象の位置・傾きは変化する．しかし，その変化はカメラのパン・チルト速度より求めることができる．よって，これらを用いてより正確な統合をすることで，さらに復元結果が改善されることが期待できる．

6 まとめ

本研究では，動物体に対し注視制御を行い，その画像をもとに対象の 3 次元形状を復元するシステムの構築を行った．注視制御を行うことで対象の運動を回転運動とみなすことができることを用いて，3 次元形状をオプティカルフローとカメラの運動パラメータより決定できることを示した．

オプティカルフローの計算にはブロックマッチングを用い，参照ブロックとサーチ領域をプロセッサアレイ上に再配置して並列処理を行うことで，高速な計算を実現した．また，実験によって，3 次元形状の復元が実時間で行われていることを確認した．

今後の課題としては，カメラの制御方法の改善，得られた 3 次元形状の時系列の適切な統合，等が挙げられる．

参考文献

[1] 出口光一郎: “画像と空間,” 昭晃堂 (1991).

[2] Beauchemin, S. S. and Barron, J. L.: “The Computation of Optical Flow,” *ACM Computing Surveys*, Vol. 27, No. 3, pp. 433–465 (1995).

[3] Horn, B. K. P. and Schunck, B. G.: “Determining Optical Flow,” *Artificial Intelligence*, Vol. 17, pp. 185–203 (1981).

[4] 松山隆司, 弓場竜: “カメラ運動を利用した移動対象の検出と運動情報の獲得,” 分散強調視覚による動的 3 次元状況理解, 日本学術振興会未来開拓学術研究推進事業 知能情報・高度情報処理研究分野 1996 年度報告書, pp. 79–103 (1997).

[5] 梅尾博司: “SIMD 上の並列アルゴリズム,” 情報処理, Vol. 33, No. 9, pp. 1042–1055 (1992).

[6] 許昭倫, 佐藤完, 岡崎信一郎, 藤田善弘: “一次元プロセッサアレイに基づくリアルタイム画像処理システムの開発環境,” 情報処理学会研究報告 コンピュータビジョンとイメージメディア, Vol. 97, No. 70, pp. 195–202 (1997).