TR－H－004

45

2回逆投影法による複数画像からの3次元情報の抽出

Double Backprojection Method to Extract 3-D Information from Multiple Images

Shinjiro KAWATO
川戸 慎二郎

1993. 3. 22

ATR人間情報通信研究所

〒619-02 京都府相楽郡精華町光台2-2 ☎07749-5-1011

ATR Human Information Processing Research Laboratories

2-2, Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02 Japan

Telephone: +81-7749-5-1011
Facsimile: +81-7749-5-1008

2回逆投影法による複数画像からの3次元情報の抽出

川戸 慎二郎

Double Backprojection Method
to Extract 3-D Information from Multiple Images

Shinjiro KAWATO

1 まえがき

これまでにも2次元画像から3次元の情報を抽出するさまざまな方法が提案されている(1) (14)。

複数画像を利用する最もシンプルな方法は2眼立体視法であるが、画像間での対応付けの問題が非常に難しいことが指摘されている。誤対応を避けるために3眼立体視法が提案されているが(12)、本質的な問題はかわらない。

これに対し対応点の探索過程が不要になるとして、エピポーラ平面画像解析法と称される、カメラを直線移動させながら連続的に撮像した画像列（時空間画像）から3次元の情報を抽出する方法が提案されている(2) (3) (19)。この手法によれば対応点探索問題はより容易な直線抽出問題に置き換えられる。

一方、複数のシルエット画像を用いて、シルエットを空間に逆投影したときの共通領域を求める方法で3次元物体を再構成する手法も提案されている(9) (13) (15) (17) (18)。いづれもオクトツリーのデータ構造をもちいて、ボクセルが逆投影されたシルエットの境界の内部か外部かの判定を行い、境界と交わるボクセルは8分割して再び内外部および交わりを調べるということを繰り返して精度をあげていく手順をとっている。

浜野ら(6) (7) (8)は、エピポーラ平面画像解析法でカメラ運動が直線運動に限定されていることは著しい制約であり、かつ直線運動だけでは効率よく視差を大きくできない、またエピポーラ平面画像に現れる直線の抽出もオクルージョンやノイズによって頻繁に分断されているためそれほど容易でないとして、時空間画像から3次元のボクセル空間に重みつきボーティングをおこなうという方法を提案している。これによれば任意のカメラ運動が可能で、かつ画像間での対応点探索や特徴点追跡を全く必要としない。彼らはどのボクセルにボーティングするかをきめるのに、画像に現れた特徴点を空間に逆投影している。逆投影という点ではシルエット像からの3次元物体の再構成に似ているが、逆投影の対象が特徴点という点で、アルゴリズムや、結果として得られる3次元情報に大きな違いがある。

本研究は浜野らとは独立になされたものであるが、画像をボクセル空間に逆投影するという意味では類似の手法を提案するものである。ただし、浜野らが仮定している「画像の撮影頻度は、時空間画像中で連続な特徴点軌跡が形成されるのに十分なほど高い」(6)という条件は必要としない。また誤対応点抑制のために、浜野らが工夫した時空間トンネル画像(6)を逆投影する代わりに2回逆投影を行うことを提案し、結果的にボクセル空間から集積点を検出するための複雑な計算(8)を不要

とした。さらにボクセルサイズが大きくてもそこに特徴点が含まれるかどうかを評価できることが浜野らとの大きな違いで、そのためオクトツリーのデータ構造をもちいて、粗いボクセルから始めて順次特徴点の含まれるボクセルのみを再分割していく効率のよいアルゴリズムを組むことができる。

アルゴリズムを検証するため、今回は視点を移動するのではなく回転ステージに載せた立方体を２０°づつ回転しつつ斜め上方から撮像した１８枚の画像を用いて実験を行った。原画像にエッジ抽出オペレータをほどこし、得られたエッジ点を特徴点とする２値画像を本アルゴリズムへの入力画像とした。実験の結果、オクルージョンなどで「見え」の回数が低い点も抽出できることを確認した。

以下、次節で従来の逆投影法の問題点を指摘し、第３節でそれらの問題点を解決する２回逆投影法を提案する。第４節では２回逆投影法がボクセルサイズフリーになることを利用して、オクトツリーをもちいた計算効率のよい実行アルゴリズムを提案する。第５節で実験結果について述べ、最後にまとめと今後の課題について述べる。

2 画像の逆投影による３次元情報の抽出と問題点

カメラの位置と姿勢が既知ならば、画像にあらわれた特徴点とレンズ中心を結ぶ直線（以下、逆投影直線と呼ぶ）を３次元空間中に引くことができ、その直線は物体上の特徴点を通過する。複数の位置から同じ特徴点を観察すれば観察数に応じた逆投影直線が得られるが、それらは物体上の特徴点の位置で交差する（図１）。したがってこの交差点が抽出できれば特徴点の３次元位置が抽出できたことになる。　画像間での特徴点の対応関係が不明であっても、画像に現れたすべての特徴点についてこのような逆投影直線を引き、その交差回数を３次元空間中の各点で評価すれば各特徴点を抽出できることが予想される。現実には、観測誤差や計算精度などの問題により厳密には交差しないので、空間をボクセルに分割し、逆投影直線が同じボクセルを通過すればそのボクセル位置で交差したとみなす。

浜野ら[6]は観察空間を微小なボクセルに分割し、各ボクセルにおいて上に述べた逆投影直線が何回通過するかをカウント（ボーティング）し、そのカウント値が局所的なピークをなすボクセルを抽出するという「単純なボーティングアルゴリズム」（以下H1アルゴリズムと呼ぶ）をまず示し、これでは偽の局所ピークが生じる可能性があるとして、「時空間トンネル画像による重み付きボーティングアルゴリズム」（以下H2アルゴリズムと呼ぶ）なるものを提案している。これは画像に現れた特徴点の８近傍（時空間画像の場合26近傍）の画素で, かつ別の特徴点でないものについて、その画素が逆投影されるボクセルには-1のカウントを行うもので、これによって、偽の局所ピークの発生が抑制されるとしている。

ここで、H1アルゴリズムの問題点をもう一度見直してみると、

(**Q1**) 一つの特徴点に対してボクセルのカウント値が広い範囲に分布し、近接する別の特徴点による分布と重なって偽の局所ピークが発生することがある。

(**Q2**) ボクセルサイズの選び方によっては、いくつかの近接する特徴点の逆投影直線が同じボクセルを通過することがあり、この場合どうカウントすべきか明確でない。

(**Q3**) オクルージョンなどによって「隠れ」が生じるとき、真の特徴点を含むボクセルであってもカウント値が低くなり、(**Q1**)の問題で生じる偽の局所ピークとの識別が難しい。

の３点があげられる。

(**Q1**)の問題点は図2で説明できる。図2(a)は

近接した2つの特徴点を十分離れた、角度が5度づつことなる11の位置から観測して得た画像から逆投影直線を引いたようすを示したものである（特徴点によって実線と破線の区別をしている）。小四角形はボクセルを表し、図2(b)は各ボクセルを通過する直線のカウント値を表している。この図からわかるように真の特徴点位置を代表するボクセルの周囲になだらかなすそ野を引くようにカウント値が広く分布する。そして近接する2つの特徴点が互いに干渉しあって偽の局所ピーク（カウント値8のボクセル）が発生するのがわかる。浜野らはこの問題について実験結果を示している[6]。

そこで浜野らは(Q1)の問題を解決するとしてH2アルゴリズムを提唱しているわけであるが、H2アルゴリズムが有効であるためには時空間画像において特徴点が連続な軌跡をなすように高頻度に撮像した画像が必要で[6]、処理すべき画像数が非常に多くなり、それだけ計算量も膨大になる。逆にいえば、離散的に撮像した画像に対しては適用できない。

また、浜野らは明示的に述べていないが、H2アルゴリズムの前提として画像分解能よりボクセル分解能のほうを相対的に高くしておく必要があり、このことによって(Q2)の問題を暗黙のうちに避けている。

実際、図2の直線に隣接するように両側に−1をボーティングする直線を引いてみると、すべてのボクセルにおいてカウント値がマイナスになってしまう。したがってH2アルゴリズムが有効に働くには、隣接する画素から空間に引かれる2本の逆投影直線が同じボクセルを通過することがないほどにボクセルのサイズが小さくなくてはならない。これは画像分解能以上に空間分解能を考えることを意味しており、むだがおおいと思われる。ボクセルサイズが不適切に小さいと画像上ではエッジのように連なった特徴点が3次元空間では途切れ途切れのボクセルとして抽出される可能性もある。あるいは、本来カメラキャリブレーションの誤差などを考えれば交差したと見なすべき2つの逆投影直線が共通なボクセルを通過しないため交差したとは見なされないということも生じる。浜野らが提唱している「すい体型視線」によるボーティング[7] [8]は、この問題に対応するために考え出されたものと思われるが、これはさらに計算負荷を重くしている。

H2アルゴリズムにおいて(Q3)の問題は、偽の局所ピークが発生しないためボーティング結果の簡単な2値化処理で、オクルージョンが生じた部分でもよい結果が得られている[6]。

次節では上記のような問題点を念頭において、それらを解決する新しい逆投影法のアルゴリズムを提案する。

3　2回逆投影法による3次元情報の抽出

まず(Q2)の問題を検討する。観察空間を微小なボクセルに分割して、各ボクセルに物体の特徴点が含まれるかどうかの証拠を複数の画像から拾い出すことを考える。H1アルゴリズムにおいて、画像にあらわれた特徴点の逆投影直線が通過するボクセルに+1のカウントをする意味は、その画像からそのボクセルに対して特徴点が含まれる可能性の一つの証拠があったことである。それでは同じ画像にあらわれた近くの特徴点から引かれた逆投影直線が同じボクセルを通過したとき、証拠が増えたといえるであろうか。そうではなくて、それは複数の特徴点がそのボクセルに含まれている可能性を示しているに過ぎない。したがって、ボクセルに特徴点が含まれているかどうかという意味では同じ画像からの逆投影直線が1本通過しようと、複数本通過しようと重みはかわらない。一方、別の画像からの逆投影直線がそのボクセルを通過した場合、別の観測によって証拠が増えたということができる。

そこでH1アルゴリズムをすこし言いかえて、すべての画像にあらわれたすべての特徴点の逆投影直線を引き、各ボクセルにおいては幾つの画像からの逆投影直線が交差するかをカウントすることにする。これによって得られた各ボクセルのカウント値を、1回目の逆投影で支持が得られた画像の数という意味で「第1支持数」と呼ぶ。このようなカウント方法をとることによってボクセル分解能は画像分解能と独立に考えることができ(Q2)の問題は解決する。

各ボクセルにおけるカウントの内容を、通過する逆投影直線の数から「第1支持数」に変更しても、(Q1)の偽の局所ピークの問題はまだ残っている。そこでもう一度画像にたちかえって、画像にあらわれた特徴点の由来を考えてみると、それはその画素の逆投影直線上に並んだボクセルのうちのどれか1つにある。どのボクセルかといえば、それは「第1支持数」が最大のボクセルであると考えるのが合理的であろう。

そこで、もういちど各画像にあらわれたすべての特徴点の逆投影直線を引き、その直線が通過するボクセルのなかで最大の「第1支持数」を持っているボクセルにその画像から第2の支持を与えることとし、各ボクセルにおいて幾つの画像から第2の支持が得られたかをカウントすることにする。このカウント値を「第2支持数」と呼ぶ。

H1アルゴリズムによって偽の局所ピークが発生した図2の例で説明すると、「第1支持数」は図2(b)と同じになる。しかし「第2支持数」は「第1支持数」が11のボクセルのみ「第2支持数」も11として残り、ほかのボクセルの「第2支持数」はゼロになる。「第1支持数」は逆投影直線が通過するすべてのボクセルに与えられたが、「第2支持数」はこのように限定的に与えられるため、近接する別の特徴点による分布と重なって偽の局所ピークが発生するということはなくなり、(Q1)の問題は解決される。

そのカウント方法からわかるように、「第2支持数」は「第1支持数」を越えることはない。理想的な場合、すなわち観察空間の特徴点以外の点がノイズなどにより画像特徴点として現れることがなく、オクルージョンは生じても写るべき角度からは必ず画像に特徴点が現れるような場合、「第2支持数」は「第1支持数」に一致するか、せいぜい視線方向に特徴点が重なった回数だけ「第1支持数」より低い値となるだけである。

ボクセルを「第2支持数」で評価するとしても、オクルージョンなどによって「隠れ」が生じる場合にその値が低くなる(Q3)の問題はまだ残されている。筆者らは当初この問題にたいして、「第2支持数」を「第1支持数」で正規化したボクセル評価値

$$p = (\text{第2支持数})/(\text{第1支持数})$$

で各ボクセルを評価すれば、「見え」の少ない点に対しても平等な評価をすることになり、(Q3)の問題は解決されると考えていた[10] [11]。しかしこの評価方法は、画像ノイズなどの影響で生じる、「第2支持数」は低い値であるが「第1支持数」も非常に低いボクセルをも抽出することになり、あまりよくない。

ここでは「第2支持数」にたいする絶対的なしきい値θをきめて、「第2支持数」がθ以上のボクセルのみを抽出するようにするのがよい。ここでしきい値θの値の意味は次のように明確である。すなわち、N個の視点からの画像を処理したとすると、N回観測したうちθ回以上見えた点を抽出するということを意味する。「第2支持数」においては「第1支持数」のような偽の局所ピークは発生しないのでθの値は低く設定することができる。

以上をまとめて、

Step1: 全画像の特徴点からその逆投影直線を引く

Step2: 各ボクセルの「第1支持数」を計算する

Step3: 各ボクセルの「第2支持数」を計算する

Step4: 「第2支持数」$\geq \theta$ のボクセルを抽出する

によって、3次元情報が抽出できる。これを画像特徴点を2回逆投影することにちなんで2回逆投影法（ＤＢＰ法：Double Backprojection Method）と呼ぶことにする。

4　オクトツリーを用いたＤＢＰ法

オクトツリーのデータ構造は3次元の情報を表現する方法としてよく知られている[4][16]。複数のシルエット画像を空間に逆投影する研究[9] [13] [15] [17] [18]では3次元のボリュームを表現するのにオクトツリーが用いられた。ここでは特徴点の3次元位置を表現するのに利用する。

オクトツリーで表現する空間は$2^n \times 2^n \times 2^n$の単位キューブからなる立方領域でモデル化できる[4]。ここでnは分解能パラメータで2^nはその空間の大きさである。各単位キューブは特徴点を含むか含まないかによって1か0の値をもつ。オクトツリー表現は立法領域を再帰的にオクタントに8分割（XYZの各次元を2分割）していくことによって得られる。$2^d \times 2^d \times 2^d, 1 \leq d \leq n$ のオクタントが特徴点を含むならそれはより小さい$2^{d-1} \times 2^{d-1} \times 2^{d-1}$の8個のオクタントに分割される。このプロセスをオクタントが単位キューブになるまで繰り返す。オクトツリーの名称はこの再帰的分割プロセスの結果が、各ノードが「葉」か8本の「枝」を持つ「木」であらわされることに由来している。

ＤＢＰ法にオクトツリーを用いる利点は2つある。

第1は、オクトツリーは立法領域の再帰的分割によって得られるので、ボクセルサイズをあらかじめ決めておく必要がない。まず対象領域を含む大きなボクセルをひとつ設定して、再帰的分割を繰り返し、必要な分解能に達した段階で計算プロセスを止めればよい。またその後の結果の利用段階においてもそこに至るまでの粗いデータを利用した、いわば多重解像度処理が可能である。

第2はメモリーと計算量の節約である。ボクセルを8分割するとき分割すべきは「第2支持数」が1以上のボクセルのみである。「第2支持数」がゼロのボクセルの領域は以後の計算で無視してよい。このことによって、最初から必要なボクセル分解能で計算する場合に比べてメモリーと計算量が格段に節約される。図3はそれを例示したものである。

図3は「第2支持数」≥ 1 の1個のボクセルが8分割され、そのうちの4個が「第2支持数」≥ 1 となったためにそれぞれが8分割され、その段階でさらに19個が「第2支持数」≥ 1 となってやはりそれぞれが8分割されるという状況をしめしている。次の段階で「第2支持数」が計算されるべきボクセルの数は$19 \times 8 = 152$ 個である。ここで「第2支持数」を計算するボクセルの数を比較してみると、最初からこの最終段階の大きさのボクセルで計算する場合は$2^4 \times 2^4 \times 2^4 = 512$ となるが、この図のように順次分割していくようにすると$1 + 1 \times 8 + 4 \times 8 + 19 \times 8 = 193$ で、その比は$193/512 = 0.38$ となる。

対象の性質によりその比が一般にどうなるかはいちがいに言えないが、例えば1個のボクセルからはじめて、毎回8分割したうちのα個が「第2支持数」≥ 1 として残るとし、n 回分割を繰り返したとすると、「第2支持数」を計算すべきボクセルの数の比は、

$$\frac{1 + 8(1 + \alpha + \alpha^2 + \ldots + \alpha^{n-1})}{(2^n)^3} \simeq (\frac{\alpha}{8})^{n-1} \quad (1)$$

となって指数関数的に小さくなる。

以上の議論から次のような計算ステップが導き出される。

Step1: 異なる視点で得られた全画像の特徴点を抽出する

Step2: 対象空間を含む大きなボクセルをひとつイニシャルボクセルとして設定する

Step3: 全特徴点からその逆投影直線を引いて、各ボクセルの「第1支持数」を計算する

Step4: 同様にして各ボクセルの「第2支持数」を計算する

Step5: ボクセルサイズが必要な解像度に達していれば **Step6:** へ進む。さもなければ、「第2支持数」≥ 1 のボクセルを8分割して新しいボクセルとし **Step3:** へ飛ぶ。

Step6: 「第2支持数」$\geq \theta$ のボクセルを抽出する

姚らは浜野らの研究を引き継いで、ボーティングの計算時間を削減するため画像とボクセル空間の両方の多重解像度化によるアルゴリズムを提案しているが[20]、オクトツリーの考え方に基づくものではない。

5 実験

実画像を用いてＤＢＰ法の実験をおこなった。

カメラが移動する代わりに回転テーブルを用い、カメラは適当にテーブルを見おろす角度で設置し（キャリブレーション結果でふ角26.5度と判明）、ズームレンズを用いて物体が適当な大きさに写るように調節した。物体は立方体とし回転テーブルのほぼ中央において、20度づつ角度の異なる18枚の画像を撮像した。一枚の画像は 640×480 の画像データとなったが、メモリ節約のためその中央部分 400×300 を切り出して処理した。切り出された画像に対してエッジ抽出オペレータを施し、抽出されたエッジ点を特徴点とした。このようにして得られた画像が図4である。側面が暗い色だったので、縦の稜線で輪郭になる部分は抽出されているが、両側の面が見えている中央の稜線はエッジとして抽出できていない。また左後方には立方体の影によるエッジが1本出ているものがある。

ＤＢＰ法の効果を見るため、図4の画像から1頂点でつながっている互いに直交している3つの稜線以外の部分を人為的に消した画像（図5）を逆投影処理して「第1支持数」「第2支持数」の分布をみた（図6）。ボクセルは図示されている大きな立方領域を $64 \times 64 \times 64$ に分割したものである。

図6(A-1)は「第1支持数」が17以上のボクセルを抽出したもので、上面に属する2つの稜線は抽出されているが、縦の稜線はまだあらわれていない。これは当然のことで、前者の2つの稜線は18枚のすべての画像で見えているが、後者の稜線は図5の1、2、3、8、9、10、11、18の8枚の画像でしか見えていないからである。

図6(A-2)は縦の稜線まで抽出されるようにしきい値を下げて、「第1支持数」が6以上のボクセルを抽出したものである。しきい値を下げたことによって他の2つの稜線は逆にその形状が見えなくなっている。

図6(A-3)は「第1支持数」が2以上のボクセルを抽出したもので、少なくとも2枚の画像からの逆投影直線が通過したボクセルがすべて抽出されている。

これからわかるように、「第1支持数」つまり1回の逆投影だけでは2節に述べた(Q3)の問題のため、目的とするボクセルを一度にうまく抽出することは難しい。

これに対し図6(B)はＤＢＰ法を適用して「第2支持数」が6以上のボクセルを抽出したものである。固定のしきい値で3つの稜線が抽出されていることがわかる。ここでしきい値6は6枚以上の画像で観測された点を抽出するということを意味している。逆にいえば6未満の画像でしか観測されなかった点はノイズとみなして捨ててしまう

ことでもある。

図7は図4の18枚の画像に対してオクトツリーを用いてＤＢＰ法を適用した結果で、(a)(b)(c)(d)の順に順次特徴点位置が詳細化される様子を示しており、それぞれ対象立法領域を8^3,16^3,32^3,64^3に分割したボクセルサイズとなっている。ただし、表示にあたっては「第２支持数」に対するしきい値６を適用した。ちなみに対象立方領域の各コーナにある半分のサイズのキューブは領域を理解しやすくするために配置したものである。各図の右上の数字はその図を得るまでに計算したボクセルの累計数を示している。

この図からわかるように立方体の形状が良く抽出されている。ただし、底面に属する稜線は元の画像にも良くあらわれていないこともあって抽出できていない。上面の４つの稜線に相当するボクセルはほぼすべて下から43段目にあり、２個のボクセルだけが44段目で抽出されている。またその各辺のボクセルを長さ方向に数えてみるといずれも30個である。ボクセルの並びと立方体の辺が約45°傾いているのでその長さは
$30\sqrt{2} = 42.4$ となってほぼ抽出された高さに等しく、立方体が立方体として抽出されているといえる。ちなみに用いた立方体は１辺が80mmであるのでボクセルサイズは約2 mm立方である。

図7(d)を出すのに計算が必要だったボクセルの数は14201個で、このサイズでは全ボクセル数は$64^3 =$
262144であるから、約1/18で済んだと言える。

6　むすび

多数の観察点からの画像を空間に２回逆投影することにより３次元情報を抽出するＤＢＰ法を提案し、実画像を用いた実験例をしめした。

画像にあらわれた特徴点とレンズ中心を結ぶ直線（逆投影直線）を空間に引き、何本の直線がボクセルを通過するかをカウントする単純なアルゴリズムには、(Q1)近接点が干渉しあう問題、(Q2)ボクセルサイズと画像分解能の関係の問題、(Q3)「隠れ」によるカウント低下の問題があることをまず指摘した。そして新提案のＤＢＰ法では何本の逆投影直線がボクセルを通過するかをカウントするのではなく、何枚の画像がボクセルを支持するかをカウントすることにより(Q2)の問題を解決し、２回目の逆投影を行うことにより(Q1)の問題を解決し、その結果しきい値が低くできることにより(Q3)の問題を解決した。このときしきい値θの意味は、θ回以上観測された点を抽出するという意味がある。

ＤＢＰ法を実際のアルゴリズムに組むにあたってはオクトツリーを用いると、無駄なボクセルに対する計算を大幅に減らして、計算効率がよくなり、また結果の利用段階においても小さいボクセルサイズに至るまでの粗いデータを利用した、いわば多重解像度処理が可能になる。

立方体を対象に20°づつ観察方向のことなる18枚の画像を用いて実験をおこない、「隠れ」が生じている点もよく抽出出来ることを確認した。また、オクトツリーによって順次形状が詳細化されるようすも示した。

ボクセル空間がハフ変換のパラメータ空間に対応すると考えると、「第１支持数」をカウントしていくプロセスはハフ変換に類似している。ハフ変換においてもパラメータ空間に投票したあとその中のどの点を抽出するかの判断は難しいものがあり、その点で「第２支持数」カウントに相当する手順を追加することが考えられる。

Gerig はハフ変換のあと、Backmapping という処理を追加して、２次元形状の抽出・認識に非常に有効だという例を示している[5]。Gerig のいう Backmapping はハフ変換のあと、実空間の各特徴点に対してパラメータ空間に引かれる曲線上で最大投票値を持つ点を関係づけて、パラメータ空間から実空間へのマッピングデータを持つとい

うもので、そのプロセスはＤＢＰ法の２回目の逆投影処理と類似している。ただ
Gerig のように２次元データを対象とする限り、ＤＢＰ法で考慮しているオクルージョンなどによる「隠れ」は発生することがなく、また２節で述べた(**Q2**)のような問題も発生しない。その点でＤＢＰ法は Gerg の Backmapping の考え方を少し拡張したものになっている。

今後、さらに複雑な形状を対象に実験をおこない、問題点を検討する予定である。また、今回は全周からの観察画像を入力としたが、観察方向が広く取れない場合の問題も検討する予定である。予測される問題として、カメラの移動方向に平行に並ぶ特徴点は位置の曖昧さが大きく残り、その解決のためには異なったふ角で臨む２台のカメラが必要になると考えられるが、別々に調整されたカメラの画像の融合という新たな課題がある。

謝辞

本研究をまとめるにあたり、助言をいただいた当研究室の赤松茂室長に感謝します。Gerig の論文を紹介していただいたATR通信システム研究所の田中弘美氏に感謝します。またボクセルの３次元表示の環境作りに援助いただいた、元ATR視聴覚機構研究所の客員研究員Philippe Quinio氏に感謝します。

文　献

(1) J.K.Aggarwal and C.H.Chien: 3-D structure from 2-D images, in "Advances in Machine Vision" edited by J.Sanz, pp.64-121, Springer- Verlag, 1989.

(2) H.H.Baker and R.C.Bolles: Generalizing epipolar-plane image analysis on the spatiotemporal surface, Int. J. of Computer Vision, vol.3, no.1, pp.33-49, 1989.

(3) R.C.Bolles, H.H.Baker and D.H.Marimont: Epipolar plane image analysis: An approach to determining structure from motion, Int. J. of Computer Vision, vol.1, no.1, pp.7-55, Jun. 1987.

(4) H.H.Chen and T.S.Huang: A survey of construction and manipulation of octrees, Comput. Vision, Graphics & Image Process., vol.43, pp.409-431, 1988.

(5) G.Gerig: Linking image-space and accumulator-space: A new approach for object-recognition, Proc. of ICCV'87, pp.112-117, Jun. 1987.

(6) 浜野輝夫、安野貴之、石井健一郎：空間への Voting による３次元環境情報抽出手法、信学論、vol.J75-D-II, no.2, pp.342-350, 1992.

(7) T.Hamano, T.Yasuno and K.Ishii: Direct Estimation of Structure from Non-linear Motion by Voting Algorithm without Tracking, Proc. of ICPR'92, vol.I, pp.505-508, 1992.

(8) 浜野輝夫、安野貴之、石井健一郎：すい体型視線を用いた空間への Voting による３次元環境情報抽出手法、信学論、vol.J76-D-II, no.1, pp.50-58, 1993.

(9) T.Hong and M.Shneier: Describing a robot's workspace using a sequence of views from a moving camera, IEEE Trans. on Pattern Anal. & Mach. Intell., vol.7, no.6, pp.721-726, 1985.

(10) 川戸慎二郎：２回逆投影法による複数画像からの３次元情報の抽出、情処研報、92-CV-79, pp.43-50, 1992-09.

(11) S.Kawato: 3D shape recovery by octree voting technique, Proc. of SPIE, vol.1820, 1992.

(12) 北村喜文、谷内田正彦：三眼視による三次元情報の計測、ロボット誌、 vol.5, no.2, pp.47-54, 1987.

(13) 登尾、福田、有本: 複数枚の画像を用いて3次元の物体を近似したオクトツリーを生成する一手法、情処論、 vol.29, no.2, pp.178-189, 1988.

(14) 大田友一：画像を用いた三次元計測技術の動向、電学論C、 vol.107, no.7, pp.608-612, 1987.

(15) M.Potmesil: Generating octree models of 3D objects from their silhouettes in a sequence of images, Comput. Vision, Graphics & Image Process., vol.40, no.1, pp.1-29, 1987.

(16) H.Samet: The Design and Analysis of Spatioal Data Structures, pp.493, Addison Wesley, 1989.

(17) S.Srivastava and N.Ahuja: Octree generation from object silhouettes in perspective views, Comput. Vision, Graphics & Image Process., vol.49, pp.68-84, 1990.

(18) R.Szeliski: Real-time octree generation from rotating objects, Tech.Rep. 90/12, DEC Cambridge Research Lab., p.30, Dec. 1990.

(19) 山本正信：連続ステレオ画像からの3次元情報の抽出、信学論、 vol.J69-D, no.11, pp.1631-1638, 1986.

(20) W.Yao, T.Horikoshi, T.Yasuno and S.Suzuki: Structure from Motion using Coarse to Fine 3D Voting, Proc. of MVA'92 (IAPR Workshop on Machine Vision Applications), pp.313-316, 1992.

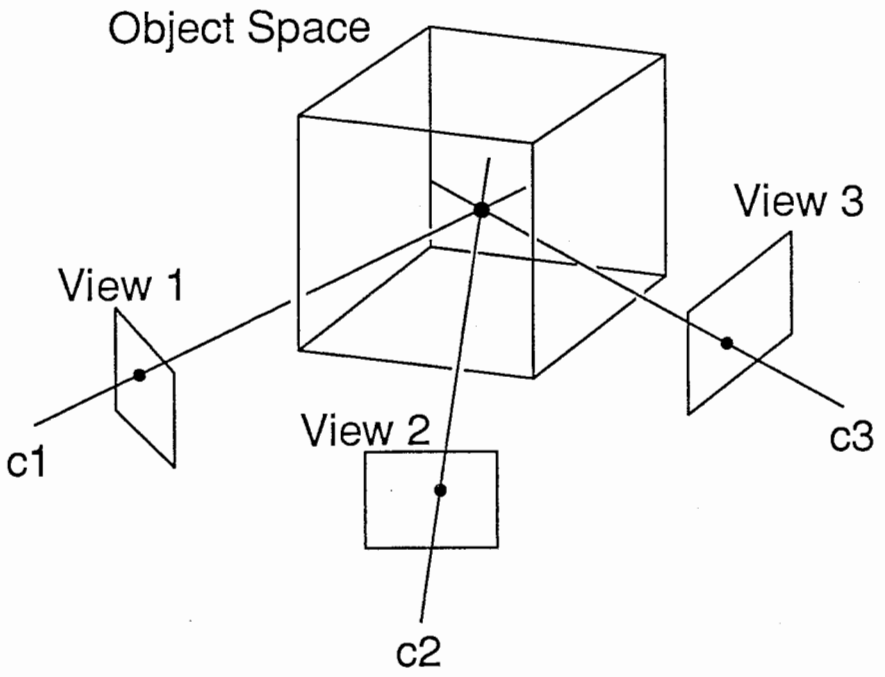
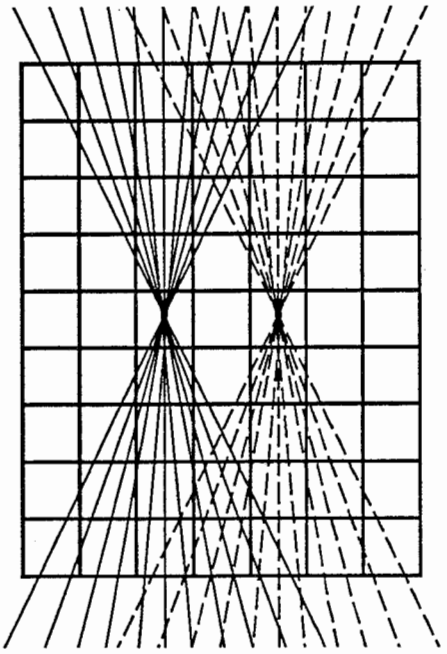


図 1 複数視点からの特徴点の観測

Fig. 1 Observation of a feature point from multiple view points.



2	3	5	6	5	3	2
1	4	6	⑧	6	4	1
	3	7	6	7	3	
	2	11	4	11	2	
		⑪		⑪		
	2	11	4	11	2	
	3	7	6	7	3	
1	4	6	⑧	6	4	1
2	3	5	6	5	3	2

(a) (b)

図 2 H1 アルゴリズムと偽ピーク

Fig. 2 H1 Algorithm and false peak.

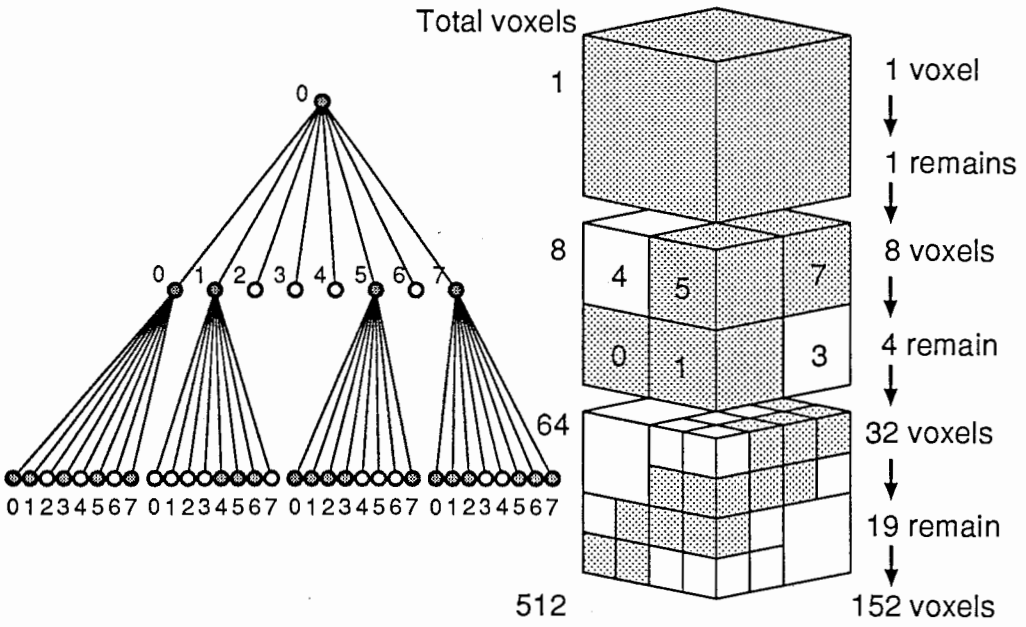


図 3 オクトツリーによる計算の効率化

Fig. 3 Octree improves calculation efficiency.

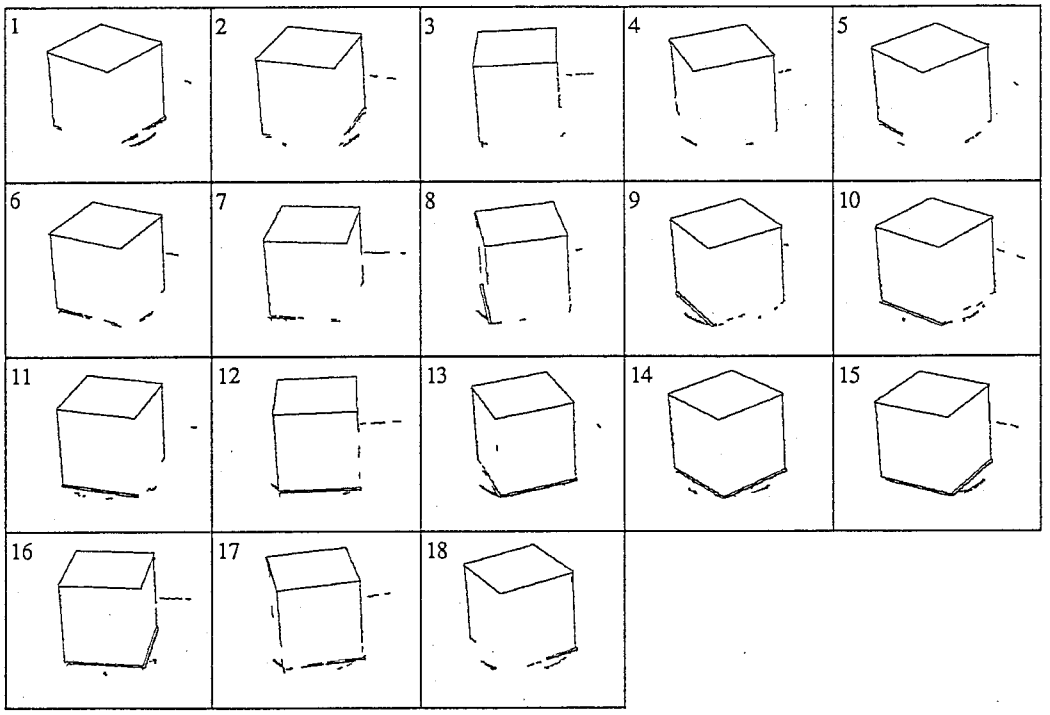


図 4 入力画像

Fig. 4 Input images.

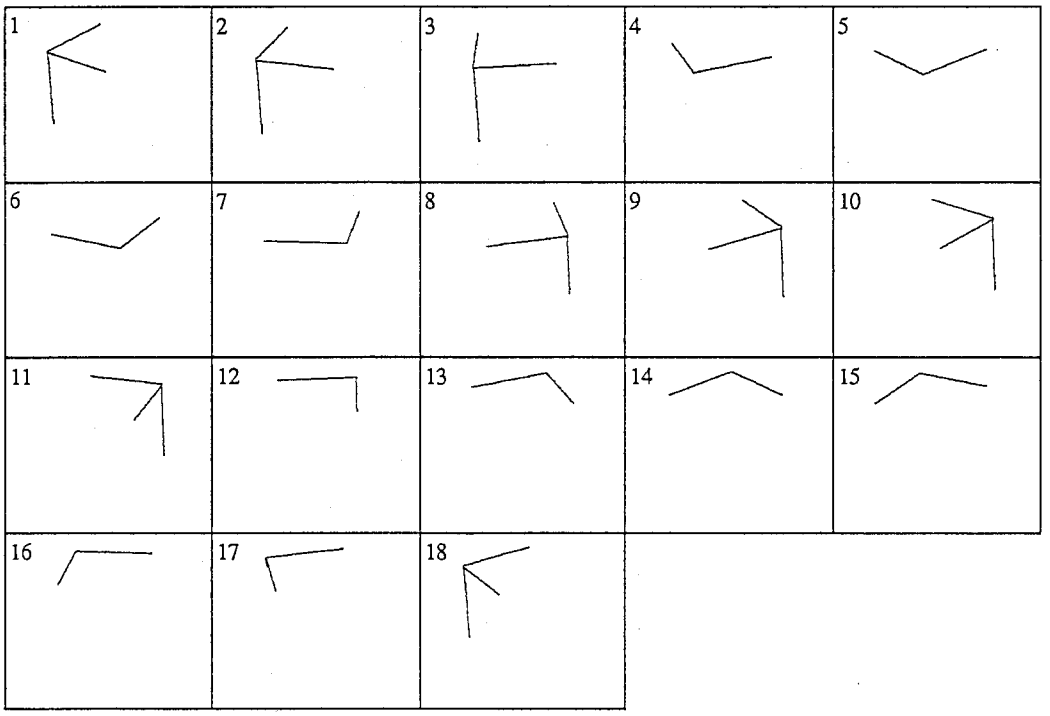
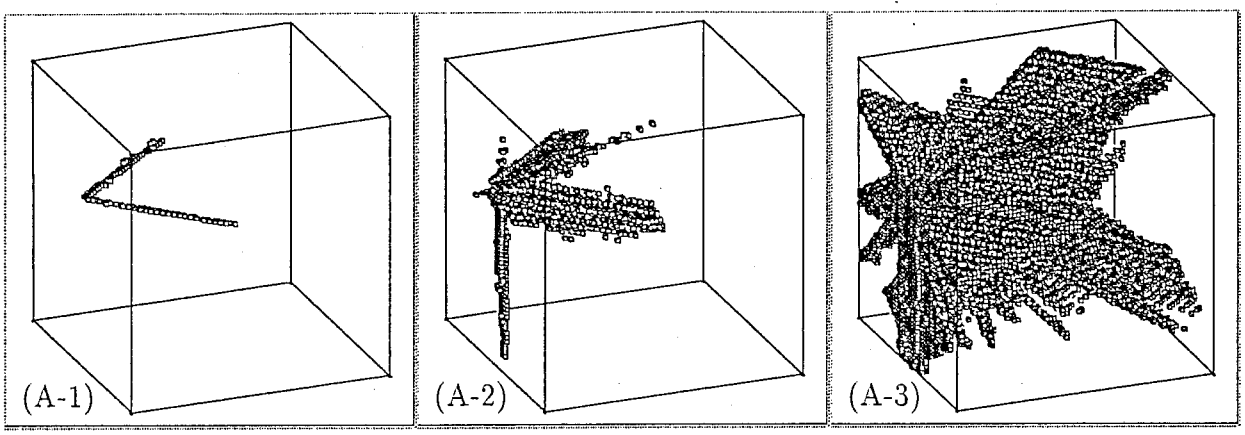


図 5 3 稜線のデータ

Fig. 5 Three-edge data from Fig.4.



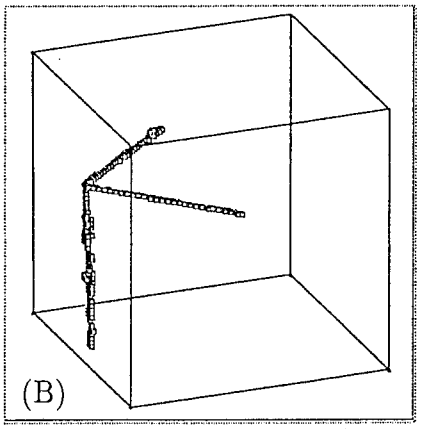
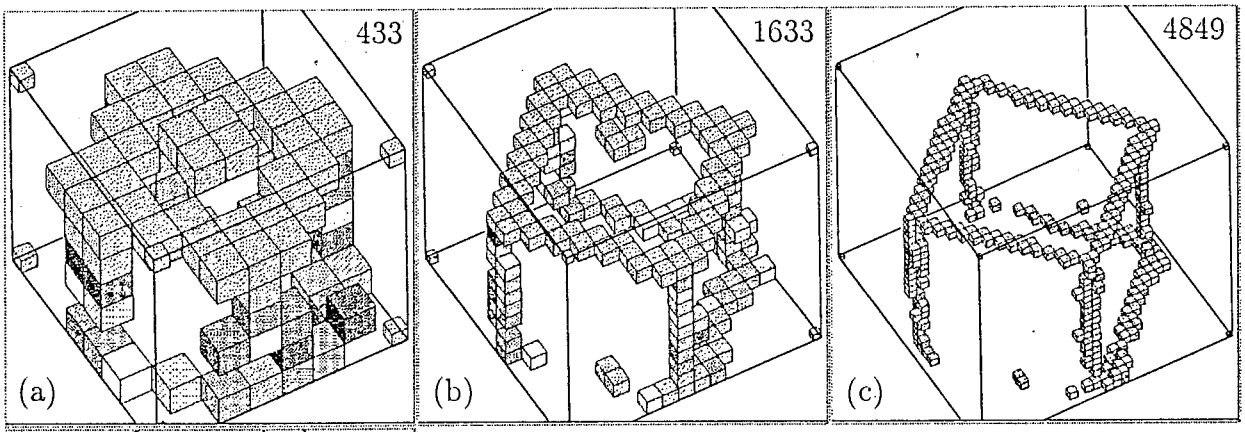


図 6 3稜線のデータの逆投影カウント結果

Fig. 6 Results for the three-edge data in Fig.5.
Initial cube is divided into $64 \times 64 \times 64$ voxels.
(A-1) Voxels with first support number ≥ 17.
(A-2) Voxels with first support number ≥ 6.
(A-3) Voxels with first support number ≥ 2.
(B) Voxels with second support number ≥ 6.



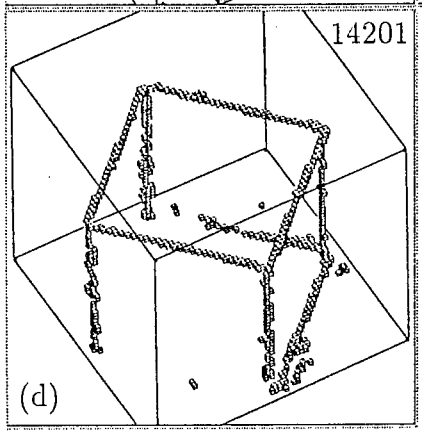


図 7 オクトツリーを用いた DBP の結果例

Fig. 7 Result of DBP with octree.
Voxels with second support number ≥ 6 are shown.
Voxel divisions are (a) 8^3, (b) 16^3, (c) 32^3, (d) 64^3.
Numbers are of total voxels to be calculated