コンピュータビジョン46－5
(1987. 1. 16)

クリークを用いた重なりのある多面体シーンの認識

A Method for Recognition of Compound Polyhedra in a Scene Based on Clique Detection

金沢　靖*　　北橋　忠宏**

Yasushi Kanazawa*　　Tadahiro Kitahasi**

*豊橋技術科学大学　　**大阪大学産業科学研究所

*Toyohashi University of Technology　　**The Institute of Industrial and Scientific Research Osaka University

あらまし　重なりを含む2次元物体の認識法の一つである、Association Graphにおけるクリーク抽出による手法を用い、輻輳したシーンや稜線が見えにくいシーン内の多面体を認識する試みについてのべる。ここでは、シーンおよびモデルを線画に変換し、その頂点の形状記号と頂点間の連結関係を用いてAssociation Graphを生成する。シーンとモデル間の形状記号の一致した頂点のペアをノードとして登録し、あるノードのモデル側の頂点から、別のノードのモデル側の頂点までの探索グラフ上の経路の長さが、それらのノードのシーン側の頂点間の探索グラフ上の経路の長さが等しい場合にエッジを張ることにより、ＡＧを生成する。ＡＧ内のクリーク、すなわち極大な完全部分グラフは要素間対応の集合であり、それぞれが仮説に対応している。従って、多面体の一部の稜線が照明の加減で見えない場合、つまり線画としては不完全な場合や、多面体の一部が他の物体によって隠されている場合にも、他の部分の対応が取れれば、マッチングを行なうことができる。また、実験システムを試作し、比較的簡単な輻輳したシーンや線画が不完全なシーンにおいても良好な結果を得ることができた。

Abstract　This paper proposes an new method for recognizing compound polyhedra in a scene. Clique detection method is used for recognition of overlapped two-dimensinal objects. Its basic idea is detecting cliques in an association graph. We have extended the method to recognition of compound polyhedra in a scene. The association graph represents candidate of corresponding pairs of primitive elements between a real scene and models. Consequently, each detected clique expresses a set of consistent pairs, or a hypothesis.

In this paper, an association graph is generated from line drawings of a scene and models. Each node in the graph has a pair of vertex symbols, and each edge shows consistency between two nodes. It is shown that the proposed method is useful to recognize compound polyhedra in a scene and an incomplete line drawing of polyhedra.

1．はじめに

現在の単眼視によるロボット等の視覚では、スリット光などの特別な照明条件を用いている場合を除いて、対象物が孤立しているようなシーンから対象物を認識している場合が多く、対象物が互いに重なりあったり、接触したりしているシーンから、対象物を認識することはあまり行なわれていない。しかし、対象物が孤立している場合というのは非常に特殊な場合のみであり、実際の環境では、対象物が接触したり、他の対象物で隠されている場合の方が多い。従ってこのようなシーンから対象物を個々に認識することは非常に重要となる。

2次元物体における接触や重なり合いを含むシーンの認識法については、現在まで数多くの研究がなされている。その中の代表的な方法の一つに構造的形状マッチングの手法による、グラフ理論的なアプローチがある(1),(2)。これらは、対象物のモデルとシーンを各要素（幾何学的な特徴、たとえば輪郭線を折れ線近似したときの各頂点の位置や角度など）に分解し、シーンとモデルの要素間の可能な対応付けを表わす Association Graph（以下ＡＧ）を生成し、このＡＧから極大な完全部分グラフすなわちクリークを抽出することにより、マッチングを行なう方法である。このＡＧにおけるエッジが、対応の付

いた要素間の適合性を示していれば、このＡＧ内のクリークは、互いに適合性の良い要素間対応の集合に相当し、マッチングの有力な候補を得ることができる。従って、重なり合いや接触を含むシーンにおいても、このクリークを抽出することにより、マッチングの解を得ることができる。

これまでの研究は、２次元的な対象物を真上から見た場合のみを扱っており、要素となる特徴も幾何学的な特徴を用いている。このため見る角度によってそれらの特徴が変化するような物体、特に一般的な３次元物体にはそのままでは適用することはできない。

本報告では、対象物を多面体に限定して、クリーク手法を３次元物体の認識に応用しようと試みた。ここでは、シーンの線画を用い、なるべく幾何学的な特徴（座標、距離、角度など）を用いずに、記号レベルでのマッチングを行なっている。そして、重なりを含むシーン内の個々の物体を認識すると同時に、３次元の多面体シーンの解釈[3]において取り扱えなかった不完全な線画の場合における物体の認識を行なった。

２．モデル表現

２次元物体認識におけるクリーク手法は、モデルとなる物体がシーン内でどのように見えるかについての詳細な情報、すなわち基本の要素となる各特徴間の幾何学的な構造がわかっていなければ、ＡＧを生成することができず、マッチングを行なうことができない。逆に言えば、このような詳細な構造が、わかっているからこそ、重なりや隠れを含むようなシーンにおいてもマッチングが行なえると言える。

一般の３次元多面体は、見ている位置およびどこを注目しているかによって、頂点の形状、位置、頂点間の幾何学的な関係、たとえば見掛け上の距離・方向などが変化する（図１ａとｂ）。従って、２次元におけるＡＧの生成方法は、そのままでは適用すできない。しかし、視点の移動が小さければ、頂点間の距離や方向は変化しても、頂点の形状を示す記号（Ｉ、Ｌ、Ｙ、Ｗ、Ｔ等に分類した場合）および頂点間の連結関係は変化しない（同図ａとｃ）。

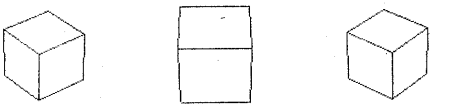
従って本報告では、頂点の形状記号（Ｉ、Ｌ、Ｗ、Ｙ、Ｔ等）を基本要素とし、頂点の形状記号とその連結関係を用いてモデルを定義する。

ＡＧ生成時において実際に使用されるモデルは、対象物体のある一つの見え方（インスタンス）であり、その多くの異なるインスタンスを一つのクラスとして統一的に扱えるように、図２に示すような記述を用いる。

図２においてinstanceは見え方の違いによるインスタンスを示し、その中のpriorityはそのインスタンスにおける頂点の優先順位を示す。この優先順位はそのインスタンスにおいて、より数の少ない種類の頂点をより高い順位とする。もし同じ種類の頂点が複数個存在する場合は、その頂点の画像平面上での座標が上にあるものを順位を高くする。たとえば、図１ａの場合、中央の‘Ｙ’の頂点を最も順位を高くし、次は左上の‘Ｗ’、その次は右上の‘Ｗ’とする。このような優先順位を設けるのは、マッチングを簡単にするため、およびマッチングの際に探索の順を決めるためである。また、同じ種類の頂点に対し、上にある頂点の順位を高くするのは、一般に上の頂点の方が他の物体に隠されることが少ないためである。また、vertexは、その頂点の番号、形状、そのインスタンスにおける画像平面上での位置、およびその頂点が隣接している他の頂点のリストである。

図３に図１の立方体に対する記述の例を示す。

モデルの各インスタンスのデータ構造はＡＧ生成時に図７で示されるような構造に変換される。



（ａ）　（ｂ）　（ｃ）

図１　立方体の見え方

```
(class  <<CLASS_NAME>>
   (instance  <<INSTANCE_NAME1>>
      (component
           (vertex  <FEATURE_LIST>)
                    .
                    .
                    .
      (priority  <VERTEX_LIST>))
   (instance  <<INSTANCE_NAME2>>
      (component
           (vertex  <FEATURE_LIST>)
                    .
                    .
                    .
      (priority  <VERTEX_LIST>)))
```

図２　モデル記述

３．処理

３．１．前処理

ここでは、実際にカメラから入力された画像からＡＧ生成に適したデータ表現への変換について述べる。

本システムは、特別な照明や機器（スリット光や照度差ステレオ）は用いず、通常の照明で処理しようとしているため、雑音、物体同士の反射、稜線におけるハイライト等により、細線化した場合、線画がきたなくなることが多く、比較的きれいな線画を得るために、たとえば図４のようなシーンに対し、次のような処理を行なっている。

1) 雑音の除去
2) Ｓｏｂｅｌオペレータによる微分
3) ２値化
4) 細線化（図５参照）
5) 線画のLINE_SEGMENTへの分割
6) データ構造の生成（図６参照）

以下に5)、6)の処理について説明する。

```
(class  CUBE
 (instance  CUBE1
  (component
     (vertex V1 (type L)  (position X11 Y11)
              (link (V6 V2)))
     (vertex V2 (type W)  (position X12 Y12)
              (link (V7 V3 V1)))
     (vertex V3 (type L)  (position X13 Y13)
              (link (V4 V2)))
     (vertex V4 (type W)  (position X14 Y14)
              (link (V3 V7 V5)))
     (vertex V5 (type L)  (position X15 Y15)
              (link (V4 V6)))
     (vertex V6 (type W)  (position X16 Y16)
              (link (V5 V7 V1)))
     (vertex V7 (type Y)  (position X17 Y17)
              (link (V4 V2 V6))))
  (priority (V7 V2 V4 V6)))
 (instance  CUBE2
  (component
     (vertex V1 (type L)  (position X21 Y21)
              (link (V6 V2)))
     (vertex V2 (type W)  (position X22 Y22)
              (link (V5 V3 V1)))
     (vertex V3 (type L)  (position X23 Y23)
              (link (V4 V2)))
     (vertex V4 (type L)  (position X24 Y24)
              (link (V3 V5)))
     (vertex V5 (type W)  (position X25 Y25)
              (link (V4 V2 V6)))
     (vertex V6 (type L)  (position X26 Y26)
              (link (V5 V1))))
  (priority (V2 V7 V1 V6))))
```

*)ここでX_{ij}はそのインスタンスに対応したその頂点の画像平面上の座標値である。

図３　モデル記述例（立方体）

・線画のLINE_SEGMENTへの分割

細線化後の画像に対し、次のような順序で線画をLINE_SEGMENTに分割する。

i)得られた線画から、分岐点（端点）から分岐点（端点）までを一つのセグメントとするLINE_SEGMENTに分割する。

ii)各セグメントに対し最小二乗法による折れ線近似を行ない、その折れ線の頂点でセグメントを分割する。

・データ構造の生成

5)で得られたLINE_SEGMENTから、各頂点の位置、形状、および他の頂点との連結関係を求め、図７のようなデータ構造を生成する。

またこのとき、２値化の際に切れてしまった線は延長してみて、その延長線上の近くに別の頂点があればその頂点と連結する。もし存在しない場合は、そのままにしておく。また、線の丸みによって生じる実際の頂点の近くの頂点などもここで除去する（図５および図６参照）。

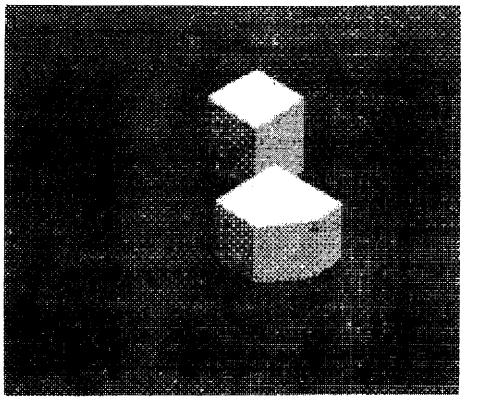


図4　シーン

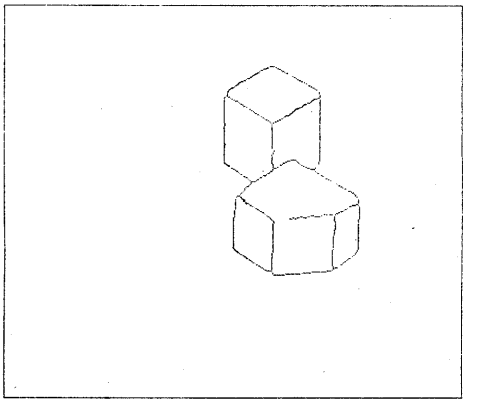


図5　シーンの線画

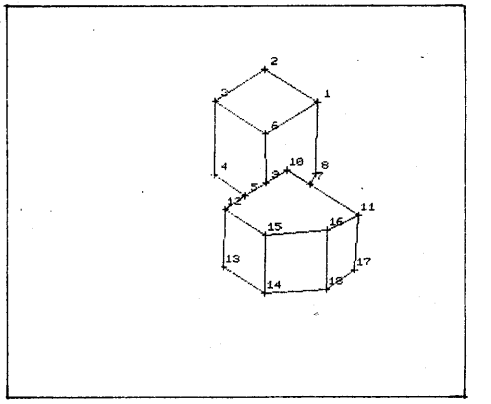


図6　各頂点の検出

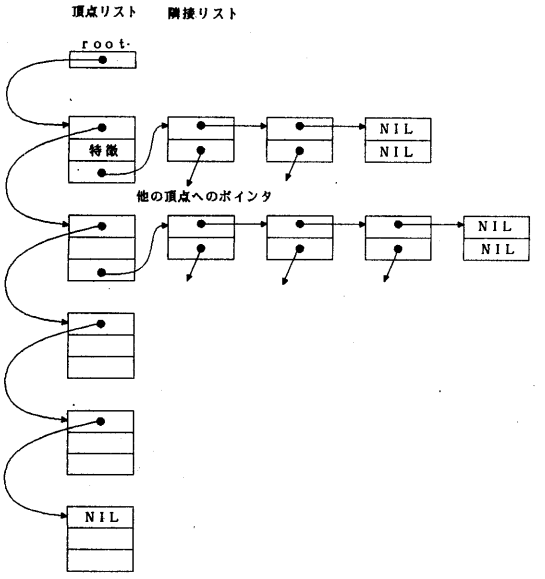


図7　データ構造

３．２．ＡＧの生成

マッチングのための基本要素となる特徴は、線画における頂点の形状記号である。そして、シーンとモデルは、共に線画に変換され、その頂点の形状記号と他の頂点との連結関係で示されるような構造となるため、グラフ的な構造となる（図８および図６参照）。このような２つのグラフから、ＡＧを生成すること、すなわち２つのグラフのノード間の可能な対応付けを表わすグラフ（２つのグラフの部分同型）を見つけることは、本質的にＮＰ完全の問題である。ここでは、この問題は頂点の形状による拘束を用いて、すなわち同じ３本のエッジが出ているノードでも、‘Ｙ’と‘Ｗ’とは異なるノードであるという前提を用いて簡単にしている。

ＡＧを生成する際、ノードを登録する基本的な条件は、頂点の形状記号が一致することであり、エッジを張る条件としては、２つのノードのモデル側の頂点間の探索グラフにおける経路の長さとそれらのノードのシーン側の頂点間の探索グラフにおける経路の長さが等しいことである。

ここでは、次のような条件を付けた場合と取除いた場合のそれぞれについて、ＡＧの生成アルゴリズムを示す。

（条件ａ）どの対象物体も水平面上に置かれており、その置かれている水平面とカメラの水平面とは一致している。

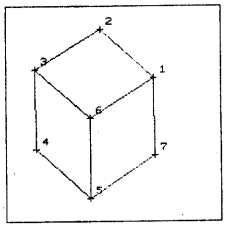


図８　立方体のインスタンス

図８と図６における探索グラフの例を図９および図１０に示す。図９および図１０において、ノードの上の数字は探索の順序を示している。

実際のＡＧはこれらの探索グラフのノードをそのままノードとし、それらの間に適当にエッジを張ったものである。

・条件ａを付加した場合

このような条件を付加すると、シーンとモデルの拘束条件が強くなるため、アルゴリズムは簡単になり、生成されるＡＧも小さくなる。

1)　最も高い優先順位を持つモデルの要素と、同じ形状記号を持ち、隣接している要素への大まかな方向（ここでは垂直方向および各象限）が一致しているシーン内の要素を探し、もし存在すればその対をＡＧのノードとして登録する。また、もしシーン内にそのような要素が存在しなければ、次に高い順位を持つ要素に対し、同様の事を行なう。

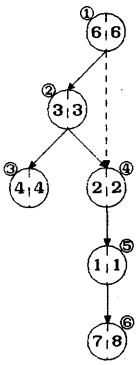


図９　条件ａがある場合の探索グラフの一部

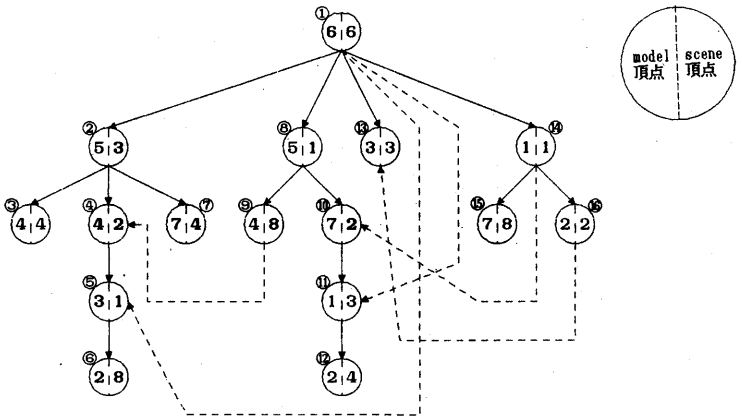


図１０　条件ａを取除いた場合の探索グラフの一部

ここで、“隣接する”とは、次のように定義する。

i)２つの要素が直接接続しており、かつその形状が双方ともに‘Ｔ’でなければ（例：図６の頂点４と頂点３）、それらの要素は隣接しているとする。

あるいは、そのうちひとつが‘Ｔ’の足であるとき（同図頂点４と頂点５）、それらの要素は隣接しているとする。

ii)また、直接接続していなくとも、その経路が‘Ｔ’のBARのみである場合（同図頂点12および頂点10）、隣接しているとする。

2) 直前に登録されたＡＧ内のノードのシーンとモデルの要素にそれぞれ隣接している要素を調べ、その形状記号およびその要素から隣接している他の要素への方向が一致すれば、その要素対もＡＧのノードとして登録する。そして、以前に探索した全てのノードから、登録したノードに対し、次のような条件が成立したとき、エッジを張る。

（条件ｂ）ＡＧ内の２つのノードのシーン側の要素が異なり、かつモデル側の要素も異なっている。

たとえば、新しく登録したノードが図１０の④であるとき、①、②に対し、エッジを張るが、③に対してはエッジを張らない。

そして次はその登録されたノードを中心に探索を行なう。

また、その対が、以前にＡＧに登録した対と同じ対である場合、それ以前に登録したノードから同様にエッジを張るが、そのノードから先への探索は行なわない（図９の①と④）。

3) 2)を拡張できなくなるまで繰り返す。

4) 1)～3)をシーン内の別の要素に対し、行なう。

・条件ａを除いた場合

上述の条件を取除くことによって、カメラの位置、つまり視点が任意となり、傾いた物体でも認識可能となるが、同時に生成されるＡＧも大きくなる。

1) 最も高い優先順位を持つモデルの要素と、同じ形状記号を持つシーン内の要素を探し、もし存在すればその対をＡＧのノードとして登録する。また、もしシーン内にそのような要素が存在しなければ、次に高い順位を持つ要素に対し、同様の事を行なう。

2) 1)で登録されたＡＧのノードの要素対に隣接しているそれぞれの要素を調べ、その要素の形状記号が一致していれば、その要素対も新たにＡＧのノードとして登録し、1)で登録されたノードとそのノード間にエッジを張る。

3) 直前に登録されたＡＧ内のシーンとモデルの要素に隣接している要素の形状記号が一致し、かつその要素から隣接している要素への方向が、直前に登録されたノードからの線を基準とした大まかな方向（ここでは左右のみとしている）が一致していれば、その要素対を新たにノードとして登録する。そして、以前に探索した全てのすべてのノードから、登録したノードに対し、条件ｂが成立したときエッジを張り、次はその登録されたノードを中心に探索を行なう。

また、その対が、以前にＡＧに登録した対と同じ対である場合、それ以前に登録したノードから同様にエッジを張るが、そのノードから先への探索は行なわない。

4) 3)を拡張できなくなるまで繰り返す。

5) 2)において別の要素が一致する場合、3)～4)を繰り返す。

6) 1)～5)をシーン内の別の要素に対し、行なう。

３．３．クリークの抽出

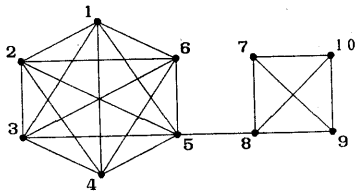
３．２で生成されたＡＧ内のクリークを抽出する。

クリークとは極大な完全部分グラフであるため、一番小さなクリークは図１１ｂである。従って、クリーク抽出は次に示すようなアルゴリズムにより、行なう。

1) １つのエッジの両端のノードをクリークのリストに登録する。

2) そのリスト内の全てのノードが、リストに含まれないある一つのノードと隣接している（そのノードに直接連結しているエッジを持つ）とき、そのノードもリストに加える。

3) 1)～2)を全てのエッジに対して行ない、全てのクリークを抽出する。



a：(1, 2, 3, 4, 5, 6)
b：(5, 8)
c：(7, 8, 9, 10)

図11　クリーク

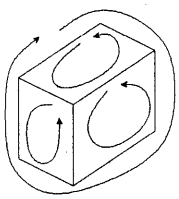


図12　面の回転方向

4．クリークの検証

3．で生成されたクリークは、互いに適合性のよい要素間対応の集合であるため、一つ一つが仮説に相当する。条件aを取除いた場合のAGは、冗長なエッジが多く、よって冗長なクリークも抽出される。従って何らかの評価により、これらのクリークを検証することが重要となる。

ここでは、そのクリークをラベルを付けることにより分類し、その仮説の信頼性を求める。

4．1．クリークのラベル付け

抽出されたクリークは次のような基準により、ラベルを付け、分類する。

1) モデル内の要素が全てそのクリークのシーン内の要素に対応し、そのシーンの要素に隣接している要素が全てそのクリークに含まれるとき、"Complete"とする。

2) クリーク内のシーンの要素に隣接している要素が、そのクリークに含まれる場合、あるいはそうでないとき'T'の足である場合、"Hidden"とする。

3) 上記以外の場合、"Non-complete"とする。

4．2．信頼性

クリークの信頼性は次式で計算する。

$$\text{Confidence} = \frac{(n-m)(p-q)}{V \cdot F} \times 100$$

ここで　V：モデルの全要素数
n：対応のとれたシーン内の要素数
m：対応のない要素と隣接しているシーンの要素数
F：モデルの全面数
p：モデルの面と順序および回転方向が一致するシーン内の面数
q：モデルの面と順序あるいは回転方向が一致しないシーン内の面数

ここで、面の回転方向および順序は図12のように定義する。

"Complete"あるいは"Hidden"のラベルの付けられたクリークはID（Identifier）リストと呼ぶシーン内から同定されたクリークのリストに加える。また、線画が不完全である場合や折れ線近似によって生じてしまう新たな頂点による影響を考慮して、"Non-complete"のラベルの付いたある程度信頼性の高いクリークは、CD（Candidate）リストと呼ぶ候補リストに加える。そして、IDリスト内に含まれるシーン内の要素は、次の別のモデルとのマッチングからは除外される。また別のモデルとのマッチングにより抽出された"Complete"や"Hidden"のラベルの付いたクリークが、CDリスト内の要素と同じ要素を含めば、それらのクリークはIDリストに加えられると同時に、CDリスト内のそのクリークはCDリストから除かれる。

５．処理例

処理例を以下に示す。

図６のシーンに対し、図８の一つのインスタンスを適用した場合の結果を図１３に示す。

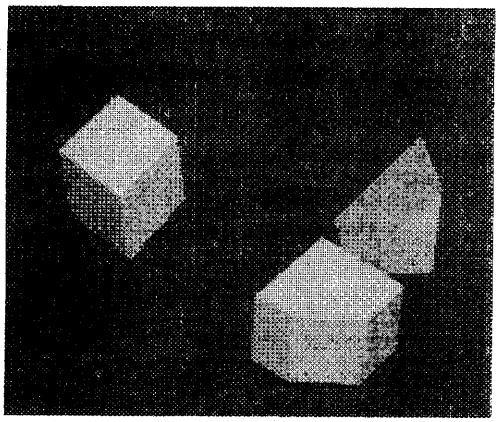
このときのＡＧのノード数は条件ａを付加した場合、ノード数は１５個で、エッジ数は４１本である。また、条件ａを取除いた場合、ノード数は５２個でエッジ数は３３７本である。

また、図１４のようなシーン、すなわち物体が傾いていたり、線画に変換したときに不完全な場合に対し、図１５ａ、図８、図１５ｂの順で、マッチングを行なった結果を図１６に示す。

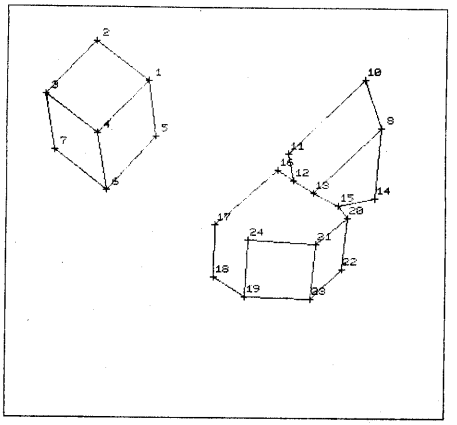
機器はＮＥＣのＰＣ－９８０１ＥとＡＤＳの画像処理装置ＩＰ－４を使用し、使用している言語はローレベルな処理に一部アセンブラを用いているのを除き、すべてOptimizing C86を用いている。

```
<< identifier list >>
[1] Hidden < CUBE :  85.7 % >
     model(   6) -- scene(   6)
     model(   3) -- scene(   3)
     model(   4) -- scene(   4)
     model(   2) -- scene(   2)
     model(   1) -- scene(   1)
     model(   7) -- scene(   8)
<< end >>


<< candidate list >>
[1] Non-complete < CUBE :  17.9 % >
    model(   6) -- scene(  15)
    model(   5) -- scene(  14)
    model(   4) -- scene(  13)
    model(   3) -- scene(  12)
    model(   2) -- scene(  10)
    model(   1) -- scene(  11)
    model(   7) -- scene(  17)

[2] Non-complete < CUBE :  17.9 % >
    model(   6) -- scene(  16)
    model(   5) -- scene(  18)
    model(   7) -- scene(  17)
    model(   1) -- scene(  11)
    model(   2) -- scene(  10)
    model(   3) -- scene(  12)
    model(   4) -- scene(  13)
<< end >>
```

図１３　図６と図８とのマッチングの結果

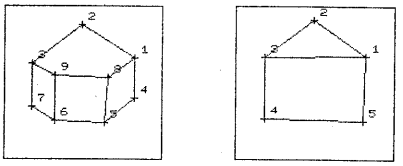


（ａ）　シーン



（ｂ）　線画

図１４　シーン



（ａ）　（ｂ）

図１５　モデルのインスタンス例

```
<< identifier list >>
<< end >>
```

```
<< candidate list >>
[1] Non-complete < PENTA :  17.9 % >
    model(   8) -- scene(  21)
    model(   5) -- scene(  23)
    model(   6) -- scene(  19)
    model(   7) -- scene(  18)
    model(   4) -- scene(  22)
    model(   1) -- scene(  20)
    model(   2) -- scene(  16)

[2] Non-complete < PENTA :  15.6 % >
    model(   8) -- scene(   4)
    model(   5) -- scene(   6)
    model(   4) -- scene(   5)
    model(   1) -- scene(   1)
    model(   2) -- scene(   2)
    model(   3) -- scene(   3)
    model(   7) -- scene(   7)
<< end >>
```

(a)　図14と図15aとのマッチングの結果

```
<<  identifier list >>
[1] Complete < CUBE : 100.0 % >
    model(   6) -- scene(   4)
    model(   5) -- scene(   6)
    model(   4) -- scene(   7)
    model(   3) -- scene(   3)
    model(   2) -- scene(   2)
    model(   1) -- scene(   1)
    model(   7) -- scene(   5)
<< end >>
```

```
<< candidate list >>
[1] Non-complete < Penta :  17.9 % >
    mocel(   8) -- scene(  21)
    mocel(   5) -- scene(  23)
    mocel(   6) -- scene(  19)
    mocel(   7) -- scene(  18)
    mocel(   4) -- scene(  22)
    mocel(   1) -- scene(  20)
    mocel(   2) -- scene(  16)

[2] Non-complete < CUBE :   7.1 % >
    model(   6) -- scene(  21)
    model(   5) -- scene(  23)
    model(   7) -- scene(  22)
    model(   1) -- scene(  20)
    model(   2) -- scene(  16)
<< end >>
```

(b)　図14と図8とのマッチングの結果

```
<< identifier list >>
[1] Complete < CUBE : 100.0 % >
    model(   6) -- scene(   4)
    model(   5) -- scene(   6)
    model(   4) -- scene(   7)
    model(   3) -- scene(   3)
    model(   2) -- scene(   2)
    model(   1) -- scene(   1)
    model(   7) -- scene(   5)

[2] Hidden < WEDGE :  80.0 % >
    model(   4) -- scene(  10)
    model(   3) -- scene(   8)
    model(   5) -- scene(  11)
    model(   2) -- scene(  14)
<< end >>
```

```
<< candidate list >>
[1] Non-complete < Penta :  17.9 % >
    mocel(   8) -- scene(  21)
    mocel(   5) -- scene(  23)
    mocel(   6) -- scene(  19)
    mocel(   7) -- scene(  18)
    mocel(   4) -- scene(  22)
    mocel(   1) -- scene(  20)
    mocel(   2) -- scene(  16)

[2] Non-complete < CUBE :   7.1 % >
    model(   6) -- scene(  21)
    model(   5) -- scene(  23)
    model(   7) -- scene(  22)
    model(   1) -- scene(  20)
    model(   2) -- scene(  16)
<< end >>
```

(c)　図14と図15bとのマッチングの結果

図16　マッチング例

6．まとめ

対象物体が水平面上に置かれており、その水平面とカメラの水平面とが一致しているという条件を付加した場合と、取除いた場合の２つについて、実験を行なった。前者では、ＡＧは小さくなり、クリーク抽出も時間がかからないが、少しでも物体が傾くと対応できなくなり、またインスタンスの構造が同じでも水平面と接触している面が異なれば、インスタンスを変えなければならなくなり、モデルのインスタンス数が増えてしまう。また、後者では、冗長なエッジやノードが増え、ＡＧは急激に大きくなり、クリーク抽出にも時間がかかるようになるが、物体が傾いていても対応でき、モデルのインスタンス数も減らすことができる。このＡＧの大きさについては生成のアルゴリズムを改良することにより、多少は小さくできるのではないかと思われる。

本手法における利点としては、

1） 処理例で示したように、不完全な線画や隠れを含むような線画においても認識（そのモデルであるという信頼性（可能性）を求めること）が可能である。

2） 他の物体の上に中途半端に乗っているような傾いた物体や、逆さになった物体でも認識可能である（条件を取除いた場合）。

3） 物体単体の認識と、それらが組み合わさってできた物体（ａｒｃｈ等）とが、同一の認識処理で行なえる。

等が挙げられる。

また問題点としては、

1） 現時点において、モデル記述内の各インスタンスはカメラから入力しており、それをまとめて一つのクラスとしている。従って、これらのインスタンスは、そのクラスにのみ関係しており、インスタンス同士の横の関係は存在しない。すなわち、あるインスタンスにおける頂点が、別のインスタンスにおいてどの頂点に対応するかわからない。

2） モデルとして線画の頂点形状記号とその隣接関係のみを用いているため、それらの構造の同じ物体、たとえば立方体と直方体、あるいは一般的な六面体など、あるインスタンスにおいて同じ構造を持っているような物体間の区別はつかない。

3） 対象物体が複雑になると、インスタンスの数が増えると共にグラフの大きさが増大してしまう。

等が挙げられる。このうち、1)および2)については、３次元モデルを用い、インスタンスの自動生成などを行なえるようにすれば、解決できると思われる。

最後に、本手法の応用としては、

1） ステレオ画像の対応点問題への応用

2） ビンピッキング問題への応用

等が考えられる。

参考文献

1) R.C.Bolles and R.A.Cain, "Recognizing and Locating Partially Visible Object : The Local-Feature-Focus Method",The International Journal of Robotics Research, Vol.1, No.3, pp.57-82, 1982

2) 坂根，"グラフのクリーク抽出に基づく部分形状マッチング", PRL82-84, pp.35-43, 1982

3) P.H.Winston, "The Psychology of Computer Vision", McGraw-Hill, Inc., 1975

4) D.H.Ballard and C.M.Brown, "COMPUTER VISION", PRENTICE-HALL, INC., 1982

5) N.J.Nilsson, "Principles of Artificial Intelligence", Tioga Publishing Company, 1982

6) N.Ayache and O.D.Faugeras, "HYPER : A New Approach for thr Recognition and Positioning of Two-Dimensional Objects", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.PAMI-8, no.1, 1986