コンピュータビジョン 26－1
（1983. 9. 26）

プロダクション・システムによる線画の解釈

Rule-Based Interpretation of Line Drawing

松原 仁　井上 博允（東京大学工学部）

Hitoshi Matsubara　Hirochika Inoue

(Faculty of Engineering, University of Tokyo)

Abstract

This paper describes a method of rule-based interpretation of line drawing that uses a hierachical production system with certainty factor. All the knowledges for region segmentation, labeling, symbolic model description, and so on, are represented by production rules. Rules for control strategies are also imbedded into the meta-level production system. By introducing certainty factor, local interpretations are evaluated and an adequate module of production rules is applied. This mechanism enables us to combine data-driven processing with model-driven processing effectively in line drawing interpretation.

1. はじめに

コンピュータによる画像理解のシステムでは，低レベルの画像処理から高レベルのモデル記述までの幾つものレベルに渡る複雑な知識を扱わなければならない。視覚情報は極めてあいまいで量も多いので，画像理解システムをより知的なものにするには，画像理解のための新しい処理手法を開発してそれを知識として取り入れることももちろん重要であるが，それ以上に，幾つものレベルに渡る複雑な知識をどのように表現し制御するかということが鍵になると考えられる。

多面体の線画解釈の分野においてはロバーツの有名な研究以来たくさんの処理手法が開発されてきたが，知識の表現と制御という観点から見た研究は少なかった。最近，複雑な対象に関する知識表現のあり方を研究する学問として知識工学が脚光を浴びているが，知識工学の方法論を線画解釈に適用することは有望な手段だと思われる。

以上のような考察から，我々はプロダクション・システム（以下PSと略す）による線画解釈のシステムを作成した。システムに対する入力は不完全なものを含めた多面体線画のリストである。知識表現の枠組としてはフレームとPSが有力な候補であるが，フレームでなくPSを選んだ理由の説明は別の機会に譲り[1,2]，ここでは省略する。線画解釈システムは二階層のPSから成り立っており，PSには確信度が取り入れられている。

本報告ではこの線画解釈システムの概容を述べる。さらに，線画解釈システムを作成する際に留意すべきだと思われる問題点を三つ取り上げて，このシステムでそれらの問題点がどのように扱われているかにも言及する。

2. 線画解釈における問題点

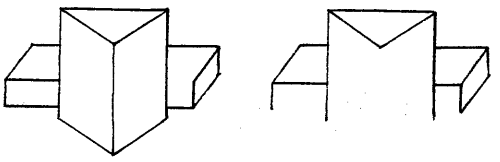
ここでは線画解釈のシステムを作成

する際に留意すべきだと思われる三つの重要な問題点を取り上げる。これら以外にもヒューリスティックの取り扱い[2]などの問題点が存在する。

2．1 data-drivenとmodel-drivenの制御

画像（線画）からモデルに向って処理をする過程をdata－driven（bottom－up），逆にモデルから画像に向って処理をする過程をmodel-driven（top-down）という。どちらも一方だけでは十分な解釈は不可能である。data-drivenだけでは探索空間が広いために計算量が多くなって効率が悪く，また意味づけ無しに処理をするために結果の信頼性が低い。とはいえ，一般にはmodel-drivenだけで処理が済むほど対象は制限されていない。したがってdata－drivenとmodel-drivenの両方を用いて，中間のレベルでデータ（画像）とモデルを照合することになる。この際常に一定のレベルで照合するのは適当ではない。照合のレベルは状況に応じて変えられるようになっていることが望ましい。対象が非常に制限されている場合には，画像にごく近いレベルでモデルをmodel-drivenで変換したものと照合するのが適当である。逆にデータの信頼性が高く対象が広い場合には，なるべくモデルに近いレベルで照合するのが適当である。また，どのレベルで照合するかは画像理解の目的としてどのような情報を抽出したいかにも依存するはずである。

線画解釈システムは，画像の状態や対象に関する知識や理解の目的に応じて，自由にdata－drivenとmodel-drivenの制御ができるようにすることが重要である。



（a）　（b）

図1　不完全線画

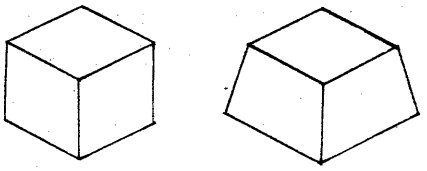
2．2 不完全線画の解釈

画像から完全な線画を抽出することは非常に難しい。よって線画解釈システムは不完全な線画を修復する努力をしなければならない。data－drivenの処理だけでは足りずにmodel-drivenの処理をすることもある。状況に応じ，いろいろなレベルの知識を用いて線画を修復するような仕組になっていることが望ましい。

また，不完全線画と一口にいってもその「不完全さ」には段階があることにも注意しなければならない。図1の(a)と(b)はうまく修復できれば同じ完全線画になるが，その不完全さには大きな差がある。両者を同等に扱うのは適当でない。不完全線画を修復する方法も一通りとは限らない。不完全線画と修復した線画との定量的な「距離」を指標として定義できれば便利である。この指標は修復した線画の信頼性に深く関わっており，それ以降の解釈の進め方に影響を与えることになる。

2．3 定量的な解釈[3]

物体の位相に関してのみ解釈することを定性的な解釈といい，辺の長さとか傾きなどをも含めて解釈することを定量的な解釈という。従来の研究では定量的な解釈はほとんどなされていなかった。図2（次ページ）の(a)と(b)は定性的には同じものと解釈されるが，定量的には異なったものと解釈される



(a)　　(b)

図2　立方体と四角錐台

のである。意味のある解釈をするためにはこの程度が区別できなくては困るので，定量的な解釈は必要である。

定量的な解釈は定性的な解釈よりも誤差の影響を受けやすい。すなわち定量的な情報には誤差のはいる余地が大きい。したがって，定量的な解釈をするためには誤差の扱いが重要になる。それも閾値を定めてその値までを許容するというような単純な方法でなく，誤差の大きさを示す指標を用意して，それ以降の解釈に利用できるようになっていることが望ましい。

3．プロダクション・システム

PSはSimonとNewellが最初人間の思考の枠組として提案したものであるが，現在フレームと並んで最も標準的な知識工学の道具である[4,5]。我々は線画解釈における知識表現の枠組としてPSを選び，そのインタープリターをLISPで記述した。PSでは知識を「if～，then～」という形のルールで表わす。図3はこのシステムにおけるルールの一例である。

ルールには以下の三つの手続きが記述可能である。?で始まるアトムはルールの局所変数を意味する。

1）　LISP関数

$で始まる手続きはLISP関数である。

```
($unchecked-frame? cube ?object)
```

2）　代入文

(:= オ1引数　オ2引数)　の形

```
(rule object-rule5
 (if
  (?object is hexahedron)
  ($unchecked-frame? cube ?object)
  (:= ?face1 ($get-first-face ?object))
  (:= ?face2 ($get-second-face ?object))
  (:= ?face3 ($get-third-face ?object))
  (?face1 is rhomb)
  (?face2 is rhomb)
  (?face3 is rhomb)
 )
 (then
  (?object is cube)
  ($check-frame cube ?object)
 )
 0.8)
```

図3　ルールの例

をしている。オ2引数の値をオ1引数に代入する。

```
(:= ?face1 ($get-first-face ?object))
```

3）　パターン

1，2以外のパターンは事実を表わす。PSの作業用記憶（データベース）が対象とするデータはこのパターンである。

```
(?object is hexahedron)
```

元来のPSにおいては命題は成り立つか成り立たないかのどちらかであるが，線画解釈はあいまいな知識を扱うのでそのように単純では意味をなさない。そこでMYCIN[6]の確信度の概念を導入する。確信度は命題の主観的なもっともらしさを示したもので，0から1までの実数である。事実とルールにはその確信度が必ずついている。図3の0.8という数字がこのルールの確信度である。これは，

「ある物体が六面体で，三つの面が見えていて，その三つの面すべてがひし形ならば，その物体は立方体である」

というヒューリスティックのもっともらしさが0.8であることを示す。前提の確信度とルールの確信度（減衰係数という）から結論の確信度が計算される。

PSの制御の方向には前向きと後ろ向きがある。前提から結論へ，ifからthenの方向へ推論を進めるのが前向きである。ある結論を仮定してその仮定が成り立つかどうかthenからifの方向へ推論を進めるのが後ろ向きである。線画解釈には両方向のインタープリターを併用する。さらに，探索にあたってバックトラックするかしないかの区別もあるが，これについても両方のインタープリターを用意する。

PSの制御ではルール競合の解消が重要な問題である。インタープリターは逐次的にルールの前提条件が成り立っているかどうかを調べるが，同時に複数のルールの前提条件が成り立っている場合にその中のひとつを選択することを，ルール競合の解消という。いろいろな解消法があるが，ここでは以下の二つの方法のどちらかを用いる。

1）最初に見つけた，前提条件の確信度がある閾値を越えているルールを選ぶ。
2）すべてのルールを調べ，そのうちで前提条件の確信度が最大のルールを選ぶ。

4. メタレベルのプロダクション・システム

PSには基本的に階層がないので，そのままでは線画解釈に要求される複雑な制御が不可能である。

線画解釈のための知識は幾つかのモジュール（レベル）に分けることができるが，それらの知識をすべて平等なルールとして扱うのでは，インタープリターは常にルール全体を探索することになって非常に効率が悪い。

3章でPSのインタープリターとして，前向きと後ろ向き，バックトラックするかしないか，ルール競合の解消法の区別があることを述べたが，それらの意味のあるすべての組合せが使えるようになっている。しかし階層のないPSでは，最初に一度あるインタープリターを指定すると，最後までそのインタープリターだけで処理される。途中までは前向きに，それ以降は後ろ向きに推論するというような柔軟な制御はできない。

これらの問題点を解決するために知識に階層を導入する[7]。上位レベルは探索空間と探索方法を制御する知識で，下位レベルは対象（この場合は線画解釈）に関する知識である。フレームで対象に関する知識を階層化することとは意味が異なる。

この階層を図式化すると図4になる。黒板モデルの考え方はHearsay-IIという音声認識をPSで行なうシステムで用いられたものである[8]。黒板はPSの作業用記憶と考えてよい。大黒板とそれに付随するルール集合とインタープリターは，システム全体の制御を行なう。これがメタレベルのPSである。小黒板とそれに付随するルール集合は，それぞれひとつの分野を受け持つ専門家（モジュール）である。

大黒板のルール集合には，

「大黒板がある状況になっていれば，
　あるインタープリターを選び，
　ある小黒板に制御を移しなさい。」

という形のルールが書いてある。これ

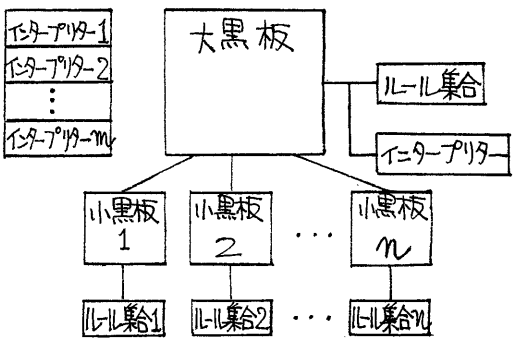


図4　PSの階層

```
(rule meta-rule8
 (if
  (:= ?rule-flag ($get-rule-flag))
  ($forward-chain-flag? ?rule-flag)
  (:= ?interpreter
      ($get-interpreter ?rule-flag))
  ($check-preconditions
      ?rule-flag ?interpreter)
 )
 (then
  ($load-rule ?rule-flag)
  ($load-interpreter ?interpreter)
  ($forward-chain)
  ($check-postconditions
      ?rule-flag ?interpreter)
 )
 1.0)
```

図5　メタルールの例

がメタルールである（図5）。制御は最初メタレベルにあり，大黒板の前向きのインタープリターがメタルールを調べていく。あるルールの前提条件が成り立っていれば，そのルールの指示するとおりに必要な情報を大黒板からある小黒板に書き移し，インタープリターを選んで制御をその小黒板に移す。小黒板での処理が終れば，処理結果が大黒板に書き移されて，制御はメタレベルに戻る。大黒板に停止のフラッグが立つか，すべてのメタルールの前提条件が成り立たなくなるまで，このことが繰り返される。

メタレベルのPSを導入することによって，探索空間を意味のあるモジュールだけに限定することと探索方法を自由に制御することが可能になる。制御の流れは前もって人間が決めておくこともできるが，大黒板の状況に任せてしまうこともできる。なお，この知識表現の枠組は線画解釈とは独立である。

5. 線画解釈の実例

階層化したPS上に，大黒板と小黒板のルールとして線画解釈のシステム

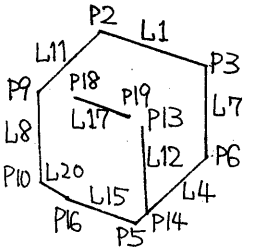


図6　入力した線画

を作成した。小黒板（モジュール）の数は十数個で，総ルール数は数百個である。

ここでは「立方体」の線画を例にとってシステムの動作を追うことにする。制御の流れは一定ではないので，あくまでもひとつの例に過ぎない。人間は大黒板のフラッグや閾値を操作して制御の流れを指定することができる。ルールを説明に持ち出すと複雑になるので，努めて言葉で述べることにする。また簡単のために，ルールの減衰係数以外の確信度は1.0とする。

前処理の済んだ，始点と終点の座標値から成る線分のリストが大黒板に入力されると，まず線分と頂点に名前がつけられる（図6）。次に線分を連結するモジュールに渡って図7のように連結される。このモジュールにおける処理の履歴は図8（次ページ）のとおりである。事実はパターン，確信度，パターンを支持するルール名の集合から成り，履歴は事実を判明した順番に並べたものである。

図7の線画をラベルづけ[9]のモジ

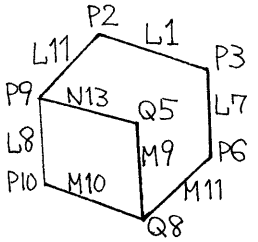


図7　連結した線画

```
((L20 and L15 have been merged into N4)
 1.0 (line-rule2))
((P13 and P19 have been replaced by Q5)
 1.0 (line-rule3))
((P14 and P5 have been replaced by Q8)
 1.0 (line-rule3))
((N12 has been joined between P18 and P9)
 1.0 (line-rule4))
((N12 and M7 have been merged into N13)
 1.0 (line-rule2))
```

図8　連結の履歴

ュールで完全かどうか調べると，不完全という結果が返ってくる。そこで不完全線画修復モジュールに渡すと図9のように修復される（この処理については6.2節で後述）。幸い図9の線画は完全なので面と物体に名前がつけられて最終的な線画と見なされる。このシステムでは線画を解釈した結果をフレーム記述言語のひとつであるFRL[5]でも表わすので，この時点でその初期化を行なう。

面が何角形かを判断するモジュールで図10の履歴が得られる。このときの面F1のフレームが図11である。最後に物体解釈モジュールに渡って，図12の履歴が最終的な解釈結果として得られる。物体O1のフレームが図13であるから，図6の入力線画は立方体と解釈されたことになる。減衰係数以外の確信度をすべて1.0と置いたために，立方体の確信度が0.64と大きい値になっているが，実際の値ははるかに小さい。

複数の物体から成る線画を解釈する

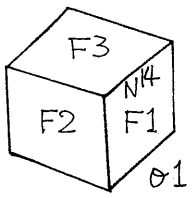


図9　修復した線画

```
((F1 is quadrilateral) 1.0 (face-rule2))
((F2 is quadrilateral) 1.0 (face-rule2))
((F3 is quadrilateral) 1.0 (face-rule2))
```

図10　位相判定の履歴

```
(F1
 (apo (value (O1)))
 (ako (value (quadrilateral)))
 (vertex (value (P6 P3 Q5 Q8)))
 (edge (value (M11 L7 N14 M9))))
```

図11　F1のフレーム

```
((F1 is parallelogram) 1.0 (f2-rule3))
((F2 is parallelogram) 1.0 (f2-rule3))
((F3 is parallelogram) 1.0 (f2-rule3))
((F1 is rhomb) 1.0 (f2-rule4))
((F2 is rhomb) 1.0 (f2-rule4))
((F3 is rhomb) 1.0 (f2-rule4))
((O1 is hexahedron) 0.8 (o-rule7))
((O1 is cube) 0.64 (o-rule5))
((O1 is rectangular parallelepiped)
 0.64 (o-rule6))
```

図12　物体解釈の履歴

```
(O1
 (ako (value (picture)
             (hexahedron) (cube)
             (rectangular-parallelepiped)))
 (co (value (F1) (F2) (F3))))
```

図13　O1のフレーム

能力もあるが，解釈の過程は非常に複雑になり，それに従って信頼性も低くなる。

6．問題点の扱い

ここでは，2章で取り上げた問題点がこのシステムでどのように扱われているかを説明する。

6．1　data-drivenとmodel-drivenの制御

前向きのインタープリターを用いることによってdata-drivenの処理を，後ろ向きのものを用いることによって

```
((O1 is hexahedron) 0.8 (o-rule7))
((F1 is parallelogram) 1.0 (f2-rule3))
((F1 is rhomb) 1.0 (f2-rule4))
((F2 is parallelogram) 1.0 (f2-rule3))
((F2 is rhomb) 1.0 (f2-rule4))
((F3 is parallelogram) 1.0 (f2-rule3))
((F3 is rhomb) 1.0 (f2-rule4))
((O1 is cube) 0.64 (o-rule5))
```

図14　後ろ向き推論の履歴

model-drivenの処理をすることができる。5章の例は実は最初から最後まで前向きのインタープリターでdata-drivenの処理をしたものであった。ここでは同じ線画を途中からmodel-drivenで処理した例を取り上げる。図10，図11の段階までは5章と同様にdata-drivenで処理し，それ以降をmodel-drivenで処理することにする。

物体O1が立方体である可能性が高いとして，物体解釈モジュールに対し，物体O1は立方体という仮定を立てて後ろ向きのインタープリターを走らせる。model-drivenで制御が進んで，仮定は正しいという結果とともに図14の履歴が得られる。図12と図14を比べると，「物体O1は立方体である」という解釈結果は同じであるが，事実が判明した順番は異なる。結果が得られるまでの時間もmodel-drivenの処理の方が短い。

この場合は仮定が正しかったが，正しくない場合には新たな仮定を立てるか，断念してdata-drivenの処理をすることになる。

data-drivenの処理とmodel-drivenの処理の両方が同一の知識（ルール）に基づいて行なわれることが，両方向PSの利点である。

6．2　不完全線画の解釈

このシステムには現在不完全線画を修復するためのモジュールが二つある。状況に応じてこの二つを使い分けることができ，確信度によって不完全さを数量化することができる。枠組としての準備は整っているが，残念ながら不完全線画の修復をルール化することは非常に難しく，たちのよいものしか修復できないのが現状である。

```
(rule incomplete-rule4
 (if
  (:= ?hexagon ($get-hexagon))
  ($closed-line-drawings?)
  (:= ?line
      ($get-hexagon-division-line ?hexagon))
  ($division-check ?line ?hexagon)
 )
 (then
  ($divide-hexagon ?line ?hexagon)
  (?hexagon has been divided by ?line)
 )
 1.0)
```

図15　不完全線画修復のルール

一方のモジュールは直接には上位レベルのモデルの知識を使わずに修復するもので，5章の例はこのモジュールを用いている。もう一方のモジュールでは直接モデルと照合する。物体の見え方に関する二次元のモデルを持っていて，不完全線画とそのモデルを重ねて修復を行なう。簡単な物体のモデルしか持っていないことと，モデルが二次元であることが弱点である。

図15は5章の例で用いたルールのひとつである。線画の中に六角形があれば，その六角形が不完全になっている原因だとして，四角形二つに分割するために線分を補おうとする。division-checkという関数はもとの線画の不完全さを確信度として返す。一般に確信度は補う線分が長いほど小さくなり，もとの線画の中に補う線分と平行な線分があればあるほど大きくなる（5章の例ではほぼ平行な線分が二本ある）。

6．3　定量的な解釈

このシステムでは確信度によって定

```
(rule f2-rule3
 (if
  (?face is quadrilateral)
  ($unchecked-frame? parallelogram ?face)
  (:= ?line1 ($get-first-line ?face))
  (:= ?line2 ($get-second-line ?face))
  (:= ?line3 ($get-third-line ?face))
  (:= ?line4 ($get-fourth-line ?face))
  ($parallel? ?line1 ?line3)
  ($parallel? ?line2 ?line4)
 )
 (then
  (?face is parallelogram)
  ($check-frame parallelogram ?face)
 )
 1.0)
```

図16　平行四辺形のルール

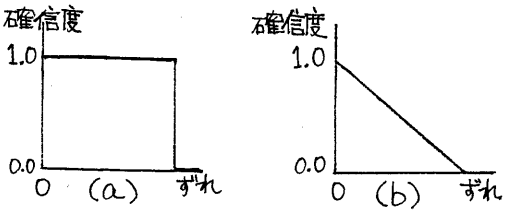


図17　確信度の関数

量的情報の誤差を柔軟に扱うことができる。図16は，

「相対する辺が両方とも平行な四角形は平行四辺形である」

というルールである。四角形の「平行四辺形らしさ」の確信度は，相対する辺の「平行らしさ」の確信度の積として得られる。この際の「平行らしさ」を求める関数は図17(a)のような単純な閾値関数ではなく，(b)のようになっているので，確信度の値に平行らしさが反映されているのである。(b)のような一次関数がいいか高次関数がいいかは個々の場合による。

7. おわりに

階層と確信度を取り入れたPSによる線画解釈システムの概容を述べた。階層と確信度によって柔軟な制御が可能になっているので，線画解釈の幾つかの問題点が統一的に扱えることを示した。

画像理解のシステムに完成ということはなく，今後もどしどしルールを追加，改良していく必要がある。特に，モデルを二次元から三次元にすることと，前処理の部分をルール化してシステムに取り込むことは，不完全線画の解釈を柔にして解釈の信頼性を高めるという意味で，非常に有意義だと思われる。

参考文献

[1] 松原　仁
PSによる線画の解釈，
東大工学部修士論文，1983.

[2] 松原　仁，井上　博允
知識工学の手法を用いた
線画の解釈，
情処全大27，1983.(予定)

[3] Kanade,T.
Recovery of the three-dimensional shape of an object from a single view,A.I.17,409-460,1981.

[4] 安西　祐一郎　他
LISPで学ぶ認知心理学2
問題解決，東大出版会，1982.

[5] Winston,P. & Horn,B.
LISP,
Addison-Wesley, 1981.

[6] Shortliffe,E.
Computer-based medical consultations :MYCIN,
American Elsevier, 1976.

[7] Bundy,A. & Welham,B.
Using meta-level inference for selective application of multiple rewrite rule sets in algebraic manipulation,
A.I.,16,189-211, 1981.

[8] McCracken,D.
A production system version of the Hearsay-II speech understanding system, UMI Research Press, 1981.

[9] Waltz,D.
Understanding line drawing of scenes with shadows,
The psychology of computer vision,
Winston,P.(ed.), Mcgraw-Hill, 1975.