CGTM # 28, October 1967
R. Conny W. Fridh

An Attempt to Put Parts of Roberts' "Machine Perception of Three-Dimensional Solids" into the Framework of Picture Calculus.

The goal of this research is to make the program described in [2] more general. Specifically one would need only:

1. A PDL-grammar for 2-D line drawings.
2. A PDL-grammar for 3-D models.
3. Properties of the primitives in the 3-D grammar.

to generate a system. The system will accept digitized pictures and a desired change of transformation and produce a line drawing.

Input for the system

The input is parsed for correct pictures according to the 2-D grammar. One great problem is to put the search strategies into the syntax (See comment in Appendix A). After a successful test the found lines are matched against the lines of the 3-D grammar which has the same label as the primitive in the 2-D grammar, e.g. a line that is called a^i by the 2-D grammar should be matched against the p^i of the 3-D grammar.

Matching

The goal of the matching is to determine which model the picture represents, if any. The picture represents a model if a correct transformation can be found that takes a part of the model into the picture. A correct transformation could consist of separate scale-factors for the x-, y-, and z- directions, rotation, translation, and a perspective transformation or it could be more general, e.g. allowing skew deformations. When a picture of a model is formed the picture and the remaining picture are searched for other correct pictures. The non-perspective transformation could then be altered to produce a picture of the same models from another view-point. This is done in a final drawing step that has to be able to calculate hidden lines.

Input grammars

The input for the picture recognizer could be a grammar in PDL, [1], [3], and [4], in which the primitives are any straight lines at all. So far the program is restricted to straight lines and convex bodies. The matching data should then be a PDL- description of each model in 3-D, with specific properties prescribed

for the primitives.

If skew deformations are not allowed, then the 3-D grammar must have more than one description of e.g. a wedge, in order to obtain a transformation for all correct pictures. If the transformation obtained when matching against the first description has too great an error then the next should be taken, etc.

Skew deformations

The reason for the above procedure is that by allowing skew deformations, any triangular prism can be derived from a wedge that has one 90° angle. Whereas if the model of the wedge is placed with one of the legs of the right angle in the z-direction, this angle will always remain 90° during skew-free transformations. See fig. 1.

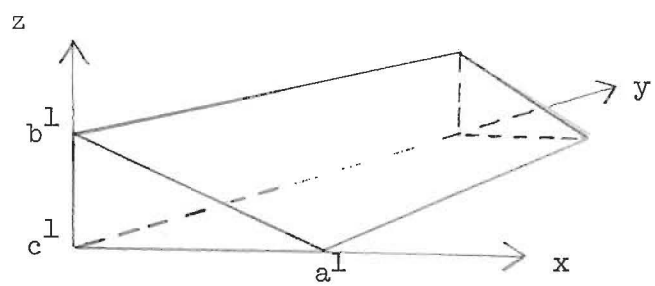


Fig. 1

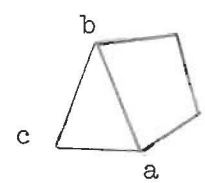


Fig. 2

Assume that the picture in Fig. 2 has been found to be topologically equal to a picture of the wedge in Fig. 1. Assume further that in fact the point c is a picture of the point c'. If the point a(or b) is matched against the point c' of the model, one cannot find a skew-free transformation that produces the analyzed picture.

Examples

A grammar designed to recognize cubes and wedges is shown in appendix A. Q means that the picture is topologically equivalent to a picture of a cube only. W stands for wedge and QW means that the picture could represent either a cube or a wedge and that the cube test is to be made first.

Suppose the third test for W is successful, that is the picture is found to be a (TRILEG $+\sim a^2$) or $(a^1 + (a^6 \times a^{10}) + \sim a^8 + \sim a^{16} + \sim a^2)$. If skew is allowed the five found lines from the picture should be matched against x^1, y^6, z^{10}, $\sim w^8$, and $\sim x^2$ (taken in the same order) from the first description (of the wedge) only. If skew is not allowed then both descriptions must be considered and the second

should in this case be against: x^1, z^0, w^{10}, $\sim y^8$, and $\sim x^2$ of the second description.

The flowchart in appendix 3 shows a way to link the topology test and the matching. The matrices A_1 and B_1 are of the size (4 x m) and (3 x m), where m is the greatest number of expected points in one picture (7 in our example).
The procedure to find the transformation is given in [2], and is also given with another and in a concatenated form in appendix C.

Construction step

The construction and drawing step is already very general since the information that is needed is partly contained in the 3-D grammar, partly a result of the topology test and the matching step. Given the connection matrices of the 3-D grammar there should be a fairly straightforward way to number the points and the lines, find and number the polygons and find the normal vectors of the polygons. This forms the main part of the input for the final construction and drawing step. The transformation is the only independent input for this step.

Following a way slightly different from Roberts, I have written a program that uses the coordinates of points in three dimensions, the components of the normal vectors, the topological data that could be derived from the 3-D grammar, and a desired transformation. The program decides which lines and parts of lines are hidden and the output is the end-point of the unhidden lines. For details see appendix D. Since the output was on a printer I was not concerned about any picture frame. The main difference between Roberts' description of his method and mine lies in the treatment of lines and points lying behind another model. Roberts uses two parameters, one to move a point along a line and one to move it forward in the positive x-direction through the hiding body, and the so moved point is checked against the volume matrix of that body. I think a simpler, time-saving way would be to use only one parameter (along the line) and to check the point against the volume matrix of the hidden volume.

My first step is to delete lines hidden by their own body. The second step concerns the points and it takes care of lines that are completely hidden behind a body and lines that are touching lines of the hiding body. The third step takes care of lines that are partly hidden and not previously deleted. For comparison see [2].

The program is written in ALGOL for the IBM 360 model 75. The program has not yet had a completely successful run. And because of lack of time and trouble with the compiler, I don't think I will be able to complete the work during my stay at SLAC.

References:

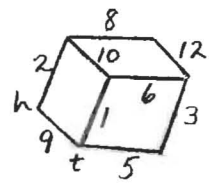
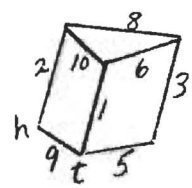
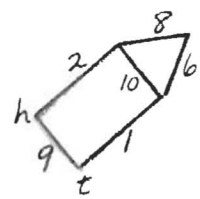
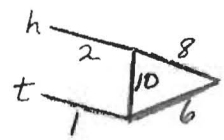
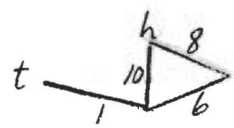
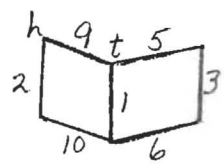
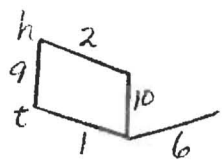
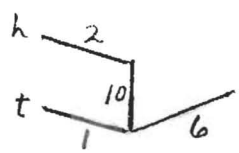
[1] W.F. Miller and Alan C. Shaw: A Picture Calculus. GSG No. 40, SLAC, Computation Group, Stanford, California

[2] L.G. Roberts: Machine Perception of Three-Dimensional Solids. Optical and Electro-Optical Information Processing (1965), Ch. 9.

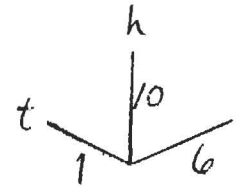
[3] Alan C. Shaw: A Proposed Language for the Formal Description of Pictures. GSG Memo 28 (Feb. 1967), SLAC, Computation Group, Stanford, California

[4] Alan C. Shaw: A Picture Calculus-- Further Definitions and Some Basic Theorems. GSG. Memo 46 (June, 1967) SLAC, Computation Group, Stanford, California

Grammar in PDL for 2-D Pictures of Cubes and Wedges

Syntax	Type of Picture
$Q \rightarrow FOLD + (\overset{\lceil 2}{a} + a^{8} + \sim a^{12} + \sim \overset{\lceil 6}{a})$	
$W \rightarrow FOLD + (\overset{\lceil 2}{a} + \sim a^{8} + \sim \overset{\lceil 6}{a})$ |	
$(TRILEG + \sim a^{2}) * a^{9}$ |	
$TRILEG + \sim a^{2}$ |	
TRILEG	
$QW \rightarrow FOLD$ |	
QUADLEG |	
$STAR + \sim a^{2}$ |	

STAR



FOLD $\rightarrow$ QUADLEG * ($\overline{a^{1}}$ + $\overline{a^{6}}$ + $\sim a^{3}$ + $\sim a^{5}$ + $\overline{a^{9}}$)

QUADLEG $\rightarrow$ (STAR + $\sim a^{2}$) * a^{9}

TRILEG $\rightarrow$ STAR + $\sim a^{8}$ + $\overline{\sim a^{6}}$

STAR $\rightarrow$ a^{1} + (a^{6} $\times$ a^{10})

<u>3-D grammar in PDL for models (cube and wedge).</u>

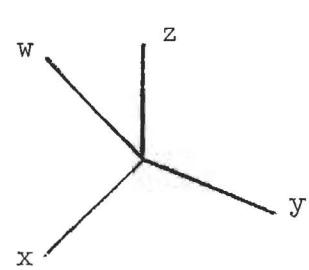
CUBE x^{1} * (z^{9} + x^{2} + $\sim z^{10}$) * (y^{5} + x^{3} + $\sim y^{6}$) *
($\overline{y^{5}}$ + z^{11} + x^{4} + $\sim z^{12}$ + $\overline{\sim y^{6}}$) * ($\overline{z^{9}}$ + y^{7} + $\overline{x^{4}}$ + $\sim y^{8}$ + $\overline{\sim z^{10}}$)

WEDGE x^{1} * (z^{9} + x^{2} + $\sim z^{10}$) * (y^{5} + x^{3} + $\sim y^{6}$) *
($\overline{y^{5}}$ + w^{7} + $\overline{x^{2}}$ + $\sim w^{8}$ + $\overline{\sim y^{6}}$) |

x^{3} * (z^{5} + x^{1} + $\sim z^{6}$) * (y^{7} + x^{2} + $\sim y^{8}$) *
($\overline{y^{7}}$ + w^{9} + $\overline{x^{1}}$ + $\sim w^{10}$ + $\overline{\sim y^{8}}$)

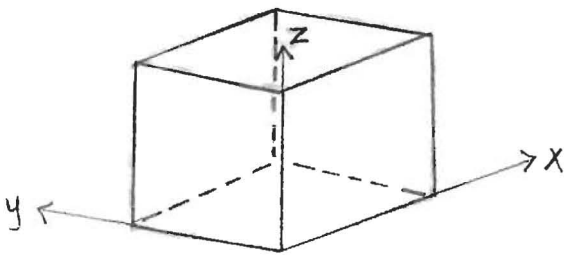
The primitives of the 3-D grammar have the following cartesian components:

x	(1, 0, 0)
y	(0, 1, 0)
z	(0, 0, 1)
w	(0, -1, 1)

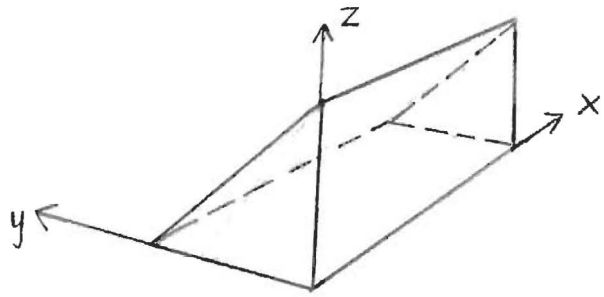


Pictures of the Models

cube:



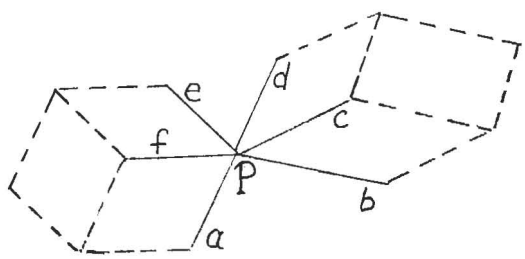
wedge:



Note: The parser keeps track of which test has been successful, i.e., if the picture is a (STAR) then the fourth test of QW is successful and if the picture is a (STAR + $\sim a^2$) then the third test of QW is successful.

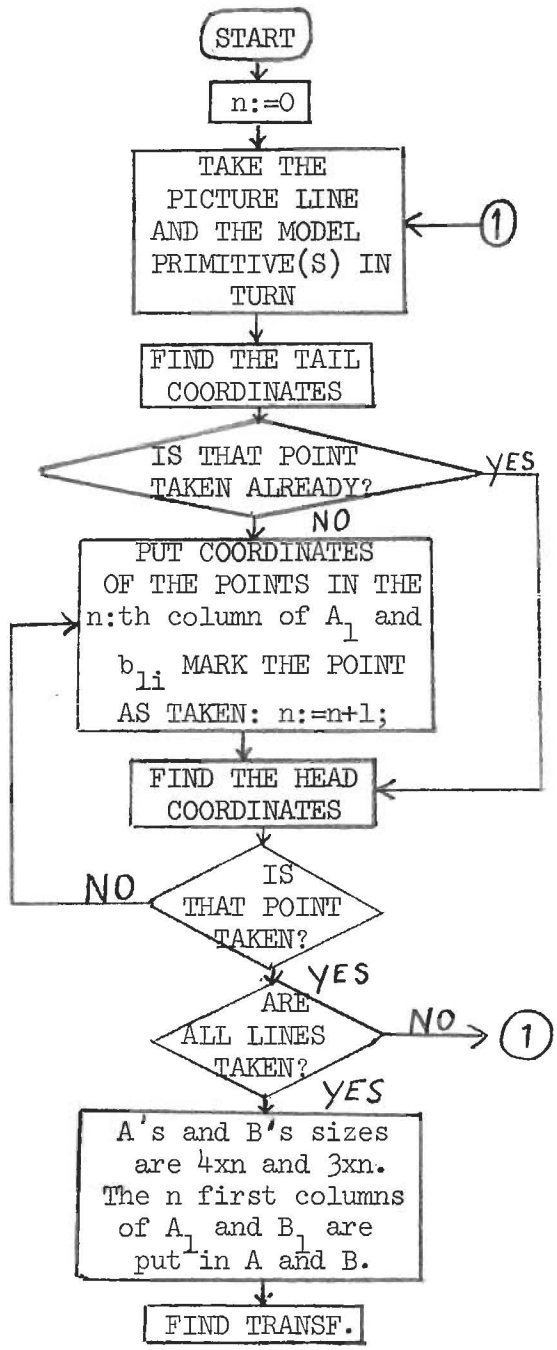
Comment: One difficult problem is to put back-tracking and "try-another-way" routines into the PDL-grammar.

Example



If this spider-like figure is part of an input picture the parser will perhaps try to find a STAR centered at the point P. If the STAR is composed of the lines a, b, and c, the attempt to find a FOLD will fail and so the point P will be discarded as a candidate for a centerpoint of STAR. The parser should instead try with the lines d, c, and b.

Flow chart over the link between a successful topology test and the start of the matching.



Appendix C.

Procedure for Finding a Transformation in Compact Form

Matrix	Size	Meaning
H	3×4	the wanted transformation
A	$4\times n$	has the model points as columns
B	$3\times n$	has the picture points ---"---
D	$n\times n$	diagonal
I	$n\times n$	identity matrix
E	$3\times n$	error matrix
G	$n\times n$	
Q	$n\times n$	
S	$n\times n$	

Problem: Find H and D such that HA = BD;

Define: $Q = B^T B;$

$G = A^T (AA^T)^{-1} A - I;$

$s_{ij} = q_{ij} \; g_{ji};$

s_{ij} are the elements of S.

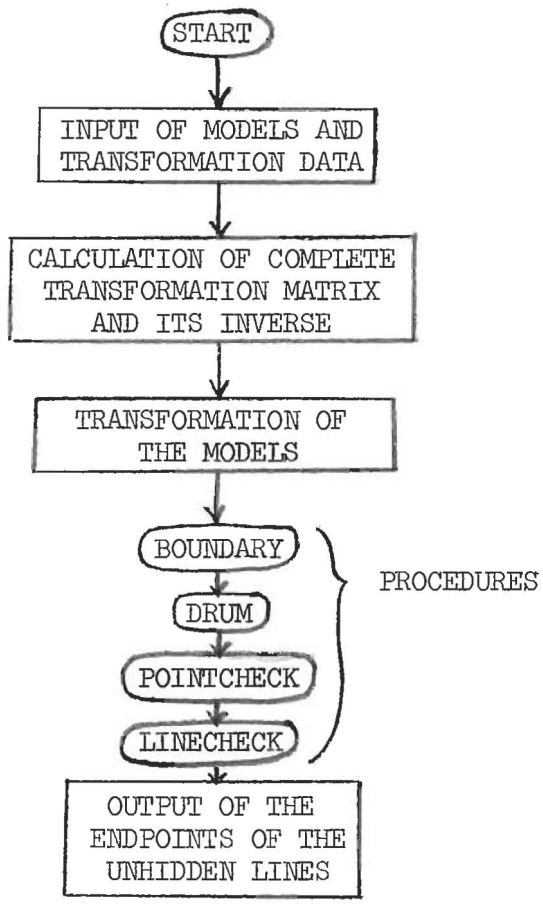
Solve: $S\bar{d} = 0$; where $\bar{d}$ is the diagonal vector of D. Define $d_i \equiv 1$ if d_i is not defined by $S\bar{d} = 0$.

Then: $H = BDA^T \; (AA^T)^{-1};$

and: $E = BDG;$

The above procedure is the same as the one in [2, APPENDIX B], but with another matrix notation. In Roberts' notation, HA = BD would be written AH = DB and the matrix H, which has three rows and four columns or is a (3×4) in my notation would have four rows and three columns and still be called a (3×4) in Roberts' notation. A transformation of a vector is written [row vector] · [matrix] by Roberts and [matrix] · [column vector] by me.

Flowchart Over a Program that Deletes Hidden Lines



The procedures:

BOUNDARY: Finds out which lines are hidden by their own model and which lines are lying on the boundary of this picture of the model.

DRUM: Calculates the volume matrix of the volume that is hidden by the model. This volume has the shape of a tunnel, closed at one end by the visible polygons of the model and open at the other end. The direction of the tunnel is the negative x-direction, which is away from the observer into the picture. Anything that is inside the drum is then hidden by the model.

POINTCHECK: The coordinates of all points are checked against the volume matrix of the drum of other models. If the point is hidden, i.e. the dot product between the point coordinates and the normalvectors (the rows of the volume matrix of the drum) are all ≤ 0, the points that are connected to it by one line are tested.

If one of these points also is inside the drum, that line is deleted. If both endpoints of a line give zero when multiplied by the same normal vector, the procedure PARALLELCHECK will check if one of the polygons adjacent to the line is parallel to and facing the same direction as that particular "drumwall" belonging to that normal vector. If that is true, the procedure COINCLINCHECK will calculate if the line should be completely or partly deleted or not at all deleted. The proper parts of the line are then deleted.

LINECHECK: This procedure will find the intersections between the remaining lines and the "drumwalls". The procedure XCHECK tells if the intersection is taking place on the actual wall of the drum and not on an extension of it. When a correct intersection is found a new point and a new line are inserted and the old line is deleted. If the middle part of the line is hidden, two points and two lines are inserted.

OUTPUT: When all points and all lines of a model have been checked against all other models, the output is a list of the endpoints of the lines that have not been deleted.