コンピュータビジョン 53 − 4
(1 9 8 8. 3. 1 5)

一般化Hough変換を用いた並列多面体認識

田中弘美　　　辻　三郎
大阪大学基礎工学部制御工学科

あらまし　複数の三次元物体が存在するシーンにおいて、既知の複数多面体を同時に認識する２種類の並列アルゴリスムを述べる。三次元シーンと各既知物体のモデルの幾何学的構造や拘束条件は、平面と稜と角と頂点による多面表現を用いて、それぞれ観測者中心座標、物体中心座標で記述される。シーン中の各物体は、この異なる二座標系間の変換を決定することによって認識される。第一の認識法は、変換パラメータを回転(姿勢)と平行移動(位置)を示す各3自由度のパラメータグループに分割し、稜と頂点の対応付けから二処理で順次決定する。第二の認識法は、角を対応づけることにより、回転と平行移動パラメータを、直接決定する。アルゴリズムは、パラメータテーブルと呼ばれる効果的なデータ構造を入出力に用い、一般化Hough変換に基づいた標準様式で記述される。

Parallel Recognition of polyhedra Using Generalized Hough Transform

Hiromi T.Tanaka and Saburo Tsuji
Department of Control Engineering, Osaka University
Toyonaka, Osaka, 560, Japan

We propose two parallel recognition methods of multiple objects in a three-dimensional scene. Both methods adopt parallel algorithms based on the generalized Hough transform. Each object in the scene is recognized by determining the rigid transformation which specify the rotational and translational changes between the parts of the scene and an object model. The principal feature of the first method is that it decouples the interdependence of two subgroups of transformation parameters for rotation and translation. Two subgroups of parameters are obtained sequentially by matching edges then by matching vertices. In the second method, both rotation and translation parameters of multiple objects are determined in a single step by matching corners (pairs of 3-d unit vectors). Both methods are evaluated by analyzing scenes containing polyhedra and the performance of both methods is discussed.

1. Introduction

Much work in computer vision and robotics has concentrated to solve problems on scenes of highly constrained environments. Such environments can be made efficient on conventional, sequential von Neumann computers, but there is growing realization that, to handle general scenes of any complexity, a parallel machine architecture must be used. This has sparked an interest in parallel machines and algorithms for vision.

The shape recognition problem is logically broken up into two parts : (1) the creation of a 3-d geometric structure from sensed data ; and (2) the mapping of that data onto a library of known objects. We assume that first problem can be solved [1,3] : shapes are initially described in terms of various levels of geometrical primitives, such as faces, edges, corners and vertices. The problem of associating the image-based structures with model data, has been handled by others as a search problem in feature space [4,5]. Research [7] has shown the feasibility of computing the correspondence between the part of the scene and a known object in parallel for the 2-d case and also explored the feasibility of 3-d, particularly for rotation. The parallel method is based on capability of computing an explicit transformation mapping from the image data to the model data, by using generalized Hough transform techniques [6].

In earlier work [1,2], we have developed parallel algorithms that build a description of a polyhedral scene from image data, and have shown how to use it to find instances of known polyhedral prototypes. In this paper, we describe and evaluate two polyhedra recognition algorithms which are insensitive to occlusion and works with only a partial description of an object. The performance of both methods is demonstrated for the scenes with five objects.

2. Overview

The problem of recognizing 3-d objects in parallel can be thought of as finding seven free parameters, three for rotation, three for translation of transformation parameters and one for model index ($1 \leqq$ model index $\leqq k$) which is associated with each object model in the model library. To find transformation parameters of k object models, M1,M2,...,Mk, in parallel, the algorithm tries to build all possible transformation of k objects if there exists a feature match between a part of a scene and any k object models. For a match between a particular scene feature and a model feature extracted from the i-th object model Mi,$1 \leqq i \leqq k$, the algorithm computes the possible transformation parameters of i-th object model and collect them into the i-th accumulator to which all possible transformation values of i-th object model are collected. If that scene feature matches also to a model feature of j-th model Mj ($i \neq j$), computed values are considered as a transformation of j-th object model and collected into j-th accumulator.

After processing all possible matches between scene features and all model features (of M1,M2,...,Mk), if a prominent peak appeared in the i-th accumulator, it means that there exists an instance of i-th object model in the scene and the paremeter values associated with the peak (in the i-th accumulator), determine the transformation parameters of i-th object models, $i=1,2,\ldots,k$. These k transformation parameters,(each of which is determined from the peak in i-th accumulator,$i=1,2,\ldots,k$,) provide the necessary information to recognize all k objects in parallel, i.e., the orientations and locations of instances of all k objects in the scene.

Two methods for recognizing three-dimensional shapes are developed. In the first method, the view transformation parameters are decoupled into two components, one for rotation and the other for translation. The recognition process works in two stages. First the rotational parameters of multiple objects are estimated in parallel by matching edges, then the translational parameters are determined by matching appropriately oriented vertices which are obtained by applying estimated rotation parameters to original vertices. In the second method, a corner is used as a matching feature. A corner is characterized as a feature which can provides a necessary information to determine both rotation and translation. Therefore, the view transformation of k objects are obtained in one step. Their overviews are shown in Figure 1,2.

3. Polyhedral Representation

3.1 Coordinate Systems

Three 3-d coordinate systems should be concerned in representing 3-d shapes: (1) a primitive-centered system (x_p,y_p,z_p) ; (2) an object-centered system (x_b,y_b,z_b) ; (3) a viewer-centerd system (x_v,y_v,z_v). The scene is described in the viewer-centerd system. Objects are described in the object-centered system respectively. The object-centered system could be arbitrarily chosen, e.g., as one of the primitive-centered systems associated with each geometrical primitive.

3.2 Geometrical Primitives

The polyhedral representation consists of a hierarchy of four levels of geometrical primitives. Each primitive is extracted from a pair of (adjacent or connected) lower-level primitives and contain more information about local surface geometry.

(1) a plane : a plane $\mathbf{P}=(\mathbf{n},\mathbf{d})$ is described by a surface normal and the distance ;

(2) an edge : Adjacent planes $\mathbf{P}_i=(\mathbf{n}_i,d_i)$ and $\mathbf{P}_j=(\mathbf{n}_j,d_j)$ determine an edge $\mathbf{E}$. Each edge is described as follows: an angle θe between planes, a direction $\mathbf{e}_\theta=<\mathbf{n}_i\times\mathbf{n}_j>$, an orientation $\mathbf{n}_e=<\mathbf{n}_i+\mathbf{n}_j>$, a length l, a direction $\mathbf{e}_d$ of a perpendicular from the origin to $\mathbf{E}$, and its length d. The operator $<>$ represents normalization of its vector argument. The axes of the primitive-centered system (x_p,y_p,z_p) are chosen such that $x_p=\pm\mathbf{e}_\theta$ $y_p=\mathbf{n}_e$,and $z_p=x_p\times y_p$. ;

(3) a corner : A pair of connected edges $(\mathbf{E}_l, \mathbf{E}_r)$ of different directions intersect at a point and determine a corner $\mathbf{C}$. Each corner is described as follows: the position $\mathbf{pc}=(x_c,y_c,z_c)$, two unit vectors $\mathbf{el}$ and $\mathbf{er}$ representing the direction of $\mathbf{E}_l,\mathbf{E}_r$,an angle θc between $\mathbf{el}$ and $\mathbf{er}$, a surface normal of a corner plane $\mathbf{n}_p=<\mathbf{el}\times\mathbf{er}>$, a corner direction $\mathbf{n}_c=-<\mathbf{el}+\mathbf{er}>$. The axes of the primitive-centered system (x_p,y_p,z_p) are chosen such that $x_p=\mathbf{n}_c$, $y_p=\mathbf{n}_p$,and $z_p=\mathbf{n}_c\times\mathbf{n}_p$.

(4) a vertex : Two or more edges intersect (or corners meet) at a vertex$\mathbf{V}$. Each vertex is described as follows: a position $\mathbf{p}_v=(x_v,y_v,z_v)$ of a vertex $\mathbf{V}$, a total number n of edges meeting at $\mathbf{V}$, a total length L of n edges meeting at $\mathbf{V}$, a set of direction vectors $\mathbf{A}=\{\mathbf{e}_\theta\}$ of n edges ($|\mathbf{A}|=n$). The edge and corner representation is shown in Figure 3, 4.

4. Algorithms

The algorithms that follow take the same form of the table-based Generalized Hough transform [2]. The algorithms use an efficient data structure termed a parameter table both for input and output (as an accumulator).

Parameter Table : A parameter table is a table of ordered parameter lists called Plist and each table is specified by following : (1) a variable $\mathbf{v}$ of k features and its parameter vector $\mathbf{v}=(p1,p2,\ldots,pk)$; (2) the table index function $f(\mathbf{v})$; (3) the quantization of each parameter p_i in terms of a range $(minp_i,maxp_i)$ and its accuracy p_i, $i=1,2,\ldots,k$.

To store a general variable $\mathbf{v}$ (i.e.,to store various values of $\mathbf{v}$), a i-th parameter p_i is chosen for a table index. Then the function $f(\mathbf{v})$ converts p_i to an index Np_i that ranges between minp$_i$ and maxp$_i$. Each Plist (N_{pi}) storesdifferent values of **v** in the decreasing order of "votes". The following example shows the construction of a 2-d edge table. The table specification for edges is shown in Figure 5 (a) and the resultant edge table in (b). In this example the angle θc is chosen for a table index.

Generic Form of Algorithms

Algorithms can be made generic and described in the following way. Given a pair of input T1 and T2 represented in the parameter tables, algorithms check if a pair of values (**u,v**) each from T1 and T2 satisfies conditions (Conds). If it does then compute the value of an output parameter w from the ternary constraints (Rules) which relate **u**, **v** and **w**. Then the algorithm adds **w** in to the output parameter table T3 by incrementing "votes" n in the appropriate place of Plist (N_{pi}), the entry indexed by the index function f (**w**) $= N_{pi}$. .

Next **w** in T3 is thresholded. This means that each Plist (N_{pi}) is thresholded against ThresholdVal [N_{pi}], i.e., if below ThresholdVal [Npi], deleted from T3 by Delete (**w**,T3).

```
IncrementTable (u,v,w,T1,T2,T3,n,Conds,Rules)
  Foreach u in T1 do
    Foreach v in T2 do
        if Condition (u;v,T1,T2,Conds) then
      {w=ApplyRules (u,v,T1,T2,Rules) ;
        Increment (w,T3,n) }
```

```
ThresholdTable (w,T3,ThresholdVal)
  Foreach w in T3 do
  {if w < ThresholdVal then
    Delete (w,T3)}
```

View Transformation

The recognition method is based of the fact that each object can be recognized if the view transformation from its object model is known. Since the scene is described in the viewer-centered system and object models in the object-centered system, the task is reduced to determine the parameter which specify the orientation and the translation between two different systems.

The orientation of an object model is described as a rotation $\mathbf{R}=(\mathbf{w},\theta)$ of angle θ around an axis $\mathbf{w}=(lw,mw,nw)$. In the algorithm, the x_p and y_p axes of the primitive centerd system are compared as a pair for detecting orientation change. The rotation axis **w** is computed from the interesting geometry that vectors which specify the displacement of xp and yp are both perpendicular to w, since they are both rotated about the same axis w.

$$\mathbf{w} = (\mathbf{D}x_p \times \mathbf{D}y_p) \;/\; \| \mathbf{D}x_p \times \mathbf{D}y_p \| \quad \text{where } \mathbf{D}x_p = x_{ps}\text{-}x_{pm} \text{ and } \mathbf{D}x_p = y_{ps}\text{-}y_{pm}. \qquad \text{(Eq. 1)}$$

Then rotation angle θc is obtained from the following equations.

$$\cos\theta = (\mathbf{w}\times y_{ps}) \cdot (\mathbf{w}\times y_{pm}),\ \ \text{sine}\,\theta\ \mathbf{w} = (\mathbf{w}\times\ y_{ps}) \times (\mathbf{w}\times y_{pm}) \qquad \text{(Eq. 2)}$$

The origin **p** of the primitive-centered system is used to detect the translational change. The "appropriately oriented origin" $\mathbf{p}_{mrotated}$ of the model-primitive-centerd system is calculated by applying the rotation to the origin $\mathbf{p}_m$ of the model-primitive-centerd system. The displacement between the origins $\mathbf{p}_s,\mathbf{p}_{mrotated}$ of the scene-primitive-system and the rotated model-primitive-system corresponds to the translation between the viewer-centered system and the object-centerd system. Then the origin of the object-centered system for an object model mi must be at,

$$\mathbf{t} = \mathbf{p}_s - \mathbf{p}_{mrotated} \qquad \text{(Eq. 3)}$$

4.1 Parallel Recognition of Polyhedra by Matching Edges and Vertices

4.1.1 Finding Orientation

The edges in the scene and the edges of all k objects are stored in Scene-Edge-Table and Integrated-Model-Edge-Table. We use an edge angle and a length to match a scene edge

Es to a model edge **Em**. Thus, RotConds= $\{(\theta_s = \theta_m), (ls \leqq lm)\}$. In order to facilitate examining the first matching condition, both edge tables are indexed with the edge angle θ_s, θ_m respectively. For each matching of a edge pair, the algorithm computes the estimate of two rotations $\mathbf{r}_1 = (\mathbf{w}_1, \theta_1)$ and $\mathbf{r}_2 = (\mathbf{w}_2, \theta_2)$, where $\mathbf{w}_1 = \mathbf{w}_2$ and $\theta_1 - \theta_2 = \pm\pi$, from the orientation change in **x** and **y** axes of the primitive-centered frames of $\mathbf{E}_s$ and $\mathbf{E}_m$ (by RotRules= {(Eq. 1),(Eq.2)}). Two three-dimensional cell representing two rotation $\mathbf{r}_1$, $\mathbf{r}_2$ and a model index $m_i = i$ with a weight field=ls is added into the RotationTable (RotTable).

<u>Algorithm : Finding Rotation of Multiple Objects</u>

IncrementTable (**Es**,**Em**,**r**,,Scenel-Edge-Table,Integrated-Model-Edge-Table,RotTranTable, ls,RotConds,RotRules)

ThresholdTable (**r**,RotTable,RotThreshold).

4.1.2 Finding Translation

The vertices in the scene and the "rotated" vertices of all k objects are stored in Scenel-Vertex-Table and Integrated-Model-Vertex-Table. We use directions, total number,and total length of intersecting edges (at a vertex), to match a scene vertex **Vs** to a "rotated"model vertex $\mathbf{V}_{mrotated}$. Thus, TranConds= $\{(\mathbf{A}_s \subseteq \mathbf{A}_{mrotated}), (n_s \leqq n_{mrotated}), (L_s \leqq L_{mrotated})\}$. For each matching of a vertex pair, the algorithm computes the translation $\mathbf{t} = (\Delta x, \Delta y, \Delta z,)$ from the location change in the origins of the primitive-centered frames of $\mathbf{V}_s$ and $\mathbf{V}_{mrotated}$ (by TranRules= {(Eq. 3)}). A cell representing translation parameters and a model index $m_i = i$ with a weight=L_s is added into the TranslationTable (TranTable).

<u>Algorithm : Finding Translation of Multiple Objects</u>

IncrementTable (**Vs**,$\mathbf{V}_{mrotated}$,**t**,Scenel-Vertex-Table,Integrated-Model-Vertex-Table,TranTable, L_s,TranConds,TranRules)

ThresholdTable (**t**,TranTable,TranThreshold).

4.1.3 Experiments

The first method has been evaluated by analyzing the scene containing polyhedra, shown in Figure 6. Model polyhedra were generated by specifying 3-d positions of the vertices. Each polyhedron has about 20 to 30 vertices. These values were then rotated and translated by the amount specified as the desired values of the transformation. Then noise (approximately 10% of shortest edge length or 1% of range of the entire scene), is added to 3-D position of each vertex. Noised vertices were then quantized and used as scene vertices. The planes, (inner) edges, and corners are extracted from these input. The algorithm is written in C-Language and implemented on a Sun-3 workstation.The processing time for the first scene was about 6 seconds for rotation and 4 seconds for translation computation.The result, after thresholded by 60% of Maxima of RotTable, TranTable (used as accumulators) is shown in Figure 7.

4.2 Parallel Recognition of Polyhedra by Matching Corners

4.2.1 Finding Rotation and Translation

The corners in the scene and the corners of all k objects are stored in Scenel-Corner-Table and Integrated-Model-Corner-Table. We use a corner angle to match a scene corner **Cs** to a model corner **Cm**. Thus, RotConds= $\{(\theta_{cs} = \theta_{ms})\}$. Both corner tables are indexed with the corner angle θ_{cs}, θ_{ms} respectively. For each matching of a corner pair, the algorithm computes the estimate of rotation $\mathbf{r} = (\mathbf{w}, \theta)$ from the orientation change in **x** and **y** axes of the primitive-centered frames of $\mathbf{C}_s$ and $\mathbf{C}_m$ (by RotRules= {(Eq.1),(Eq. 2)}). Then "rotated" primitive-centerd frame of $\mathbf{C}_{mrotated}$ is computed by applying the estimate of rotation **r** to the original frame of $\mathbf{C}_m$. For each rotated corner $\mathbf{C}_{mrotated}$, the algorithm checks TranConds= { $\mathbf{C}_{mrotated}$ is "visible"}. If so, we proceed to compute the translation **t**= (Δ x,Δ y,Δ z,) from the location change in the origins of the primitive-centered frames of $\mathbf{C}_s$ and $\mathbf{C}_{mrotated}$ (by TranRules = {(Eq.3)}). A seven-dimensional cell representing three rotational paramete, three translational parameters and a model index m_i=i with a weight field=1.0 is added into the TransformationTable (RotTranTable).

<u>Algorithm : Finding Transformation of Multiple Objects</u>

IncrementTable (**Cs**,**Cm**,**T**= (**r**,**t**),Scenel-Corner-Table,Integrated-Model-Corner-Table,RotTranTable, 1.0,RotConds,TranConds,RotTranRules)

ThresholdTable (**T**,RotTranTable,RotTranThreshold).

4.2.2 Experiment

The second method has been demonstrated on the scene (Figure 8). The processing time for the scene was about 90 seconds. All objects except WRENCH are detected correctly. The result (after thresholded by 60% of the maximum value) is shown in Figure 9.

5. Conclusion

The performance of two methods for parallel shape recognition has been evaluated analyzing scenes of polyhedra. The frame work of the two methods is rather simple and described as <valid hypothesis accumulation>, which does not require verification of values of agreeing hypothesis. The key points to make valid hypothesis are : (1) to <u>select a good feature</u> as <key> matching feature which reflect the local surface geometry of an object ; and (2) to <u>select good conditions</u> for establishing and supporting the matching, and to exclude false matches. A 3-d angle is a invariant to transformation and is not affected by occlusion.

In the first method, the second computation for translation is <filtered> by values of the first subgroup for rotation candidates. Only correct rotation supports the correct translation. In the second method, although we succeeded in estimating six free parameters of five objects in a single step, accuracy is rather poor especially for WRENCH object. From experimental result, we conclude that the strategy of decoupling high-dimensional parameters into a set of subgroups, and estimating each subgroups sequentially gives us <u>more accurate, stable, and reliable result</u> than the direct estimation of a whole set of parameters.

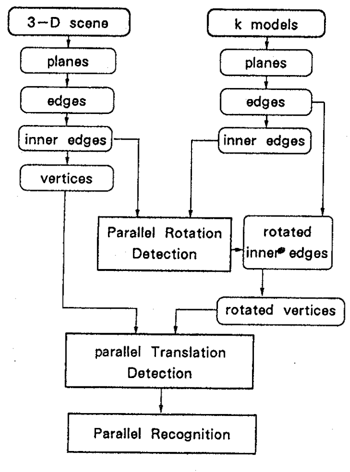


Figure 1 Over View of Two-Stage Approach

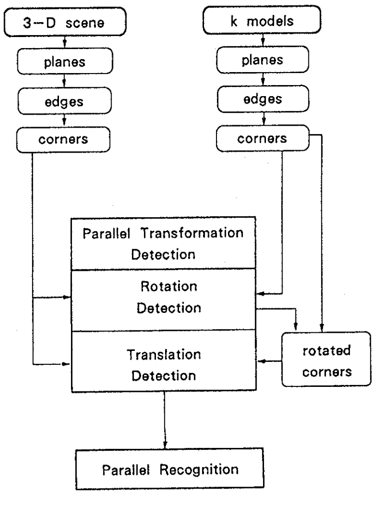


Figure 2 Over View of Direct Approach

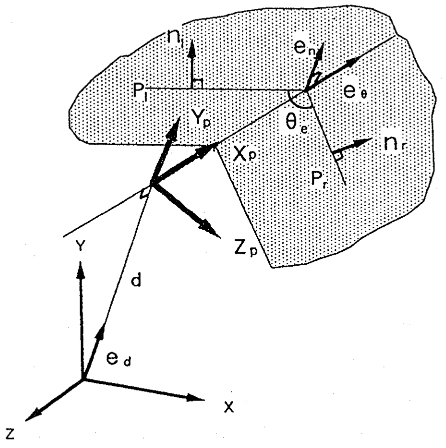


Figure 3.3 Edge Representation

Adjacent planes $P_i = (n_i, d_i, m_i)$ and $P_j = (n_j, d_j, m_i)$ determine an edge E_i. A rotation part E_g of the primitive-centered frame F_g is defined as $F_g = (E_g = (x_g, y_g, z_g), de_d) = ((e_{\theta e}, n_e, e_{\theta e} \times n_e), de_d)$.

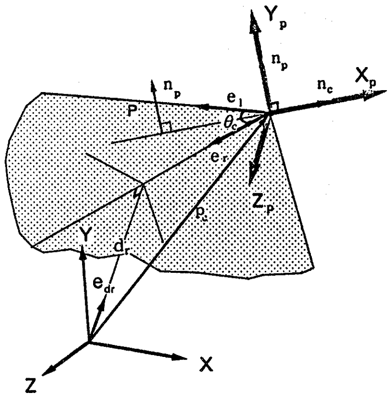


Figure4 Corner Representation

Coplanar edges E_l and E_r intersect at p_c and determine a corner C_i. The primitive-centered frame $F_p = (E_p = (n_c, n_p, n_c \times n_p), p_c)$, is defined such that $x_p = n_c$, $y_p = n_p$, and $z_p = (n_c \times n_p)$.

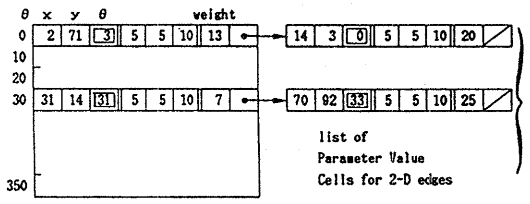
(1) a variable e for edge: $e = (x, y, \theta)$

(2) an index function: $f(e) = \theta$

(3) a quantization table Q_e

	parameter	min(pl)	max(pl)	Δpi
P_1	x	0	255	5
P_2	y	0	255	5
P_3	θ	0	360	10.

(a) A table specification for 2-D edge $e = (x, y, \theta)$



(b) The 2-D edge table

Figure5 A parameter table for 2-D edges
a) The table specification b) the 2-D edge table

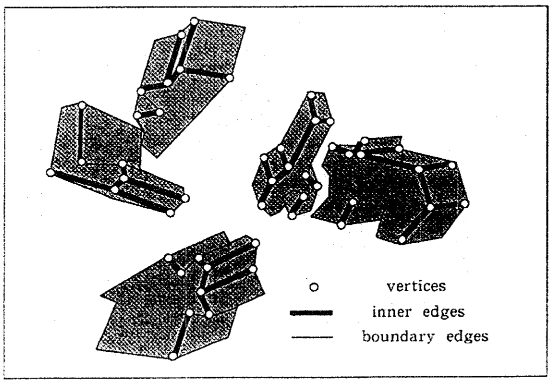


Figure 6 Scene of five different polyhedra

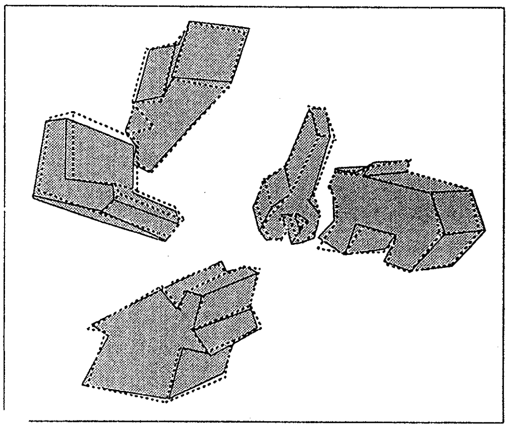


Figure 7 Experimental Result of Scene-1:
Projection of 5 polyhedra using estimated transformation

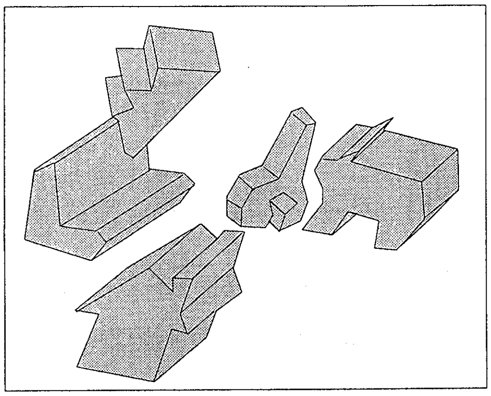


Figure 8 Scene of five different polyhedra

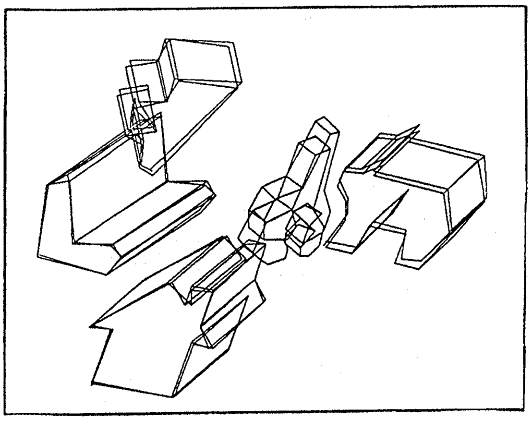


Figure 9 Experimental Result of Scene
Projection of five polyhedra using estimated Parameters

References

[1] Ballard D.H. and Tanaka T.H. " Transformational Form Perception : Constraints, Algorithms, Implementation," Proc. IJCAI−85 ,pp964−968,1985

[2] Tanaka,H.D, Ballrad,D.H.,Tsuji,S.,and Curtis,M., ; " Parallel Polyhedral Shape Recognition", Proc. CVPR,pp491-496, 1985

[3] Inokuchi,S,Sato,K, and Matsuda,D.,"Range Imaging system for 3−D object recognition," Proc. ICPR−84,pp806−808,1984

[4] Oshima M,Shirai Y., "Object Recognition Using Three-Dimensional Information", In Proc. IJCAI-81, 1981, pp601-606

[5] Silberberg,T.,Harwood, D. and Davis,L., ; " Object Recognition Using Oriented Model Points," Computer Vision, Graphics and Image Processing 35,1986

[6] Ballard D.H.,"Generalizing the Hough Transform to Detect -Arbitrary Shapes," Pattern Recognition 13, 2, 111-122, 1981

[7] Ballard D.H. and Sabbah D."Viewer Independent Shape Recognition" IEEE Trans. On Pattern Analysis and Machine Intelligence, Vol PAMI-5, No6, Nov. 1983