10—2

# Successive Pose Clustering for Stereoscopic Object Recognition

Takeshi SHAKUNAGA Tohru OHNO
Department of Information Technology, Okayama University
Okayama-shi, Japan 700-8530
E-mail: shaku@it.okayama-u.ac.jp

**Abstract**

The concept of generalized trihedral vertex (GTV) is proposed for 3d object representation. The concept covers not only polygonal objects but also curved objects including rounded edges and rounded corners. An effective algorithm is shown for matching two GTVs, one of which is constructed from stereoscopic images and the other of which is precompiled in the GTV database. Finally, we construct an efficient algorithm for successive pose clustering based on the GTV matching. The pose space (6 DOF) is decomposed into 3d rotation space and 3d translation space. Successive algorithms are constructed to perform pose clustering in the two spaces without using any Hough-like voting or any peak detection.

## 1 Introduction

Object recognition is one of the most important themes in computer vision. Considerable work has been accomplished with a single image [1]-[7]. In the controlled environments, some of them can work well by utilizing the information on the environments. The single view approach, however, suffers from difficulties of initial correspondences between image and model primitives in the uncontrolled environments. To avoid this difficulty, this paper proposes a new approach to the 3-d object recognition using stereoscopic data.

## 2 Object Model Representation

### 2.1 Selection of Primitives

We have to determine what primitives to use for 3d object recognition. The primitives should cover a wide varieties of 3d model representation. On the other hand, the primitives should be extracted from both a stereoscopic image and a 3d model data. It is not global features but local features that satisfy the above two requirements.

Among local feature primitives, trihedral vertices are a very common concept in computer vision. A trihedral vertex is composed of three planes which strictly meet one another at the vertex. Although the trihedral vertex seems very week to cover a wide variety of 3d objects, we can extend the concept to cover them.

### 2.2 Generalized Trihedral Vertex(GTV)

To cover a more general object class, a new primitive should be introduced instead of the trihedral vertex. We call the new primitive a generalized trihedral vertex or a GTV in short.

A GTV is a generalized concept of trihedral vertex in the following meaning. First, planes are generalized to surfaces of which contour can be observed in the image. Second, the vertex is generalized to vertex-like local feature. A generalized trihedral vertex is composed of three surfaces which roughly meet at a point called a virtual vertex as shown in Fig. 1. Furthermore, three observed line segments, which are located near to one another, are regarded as a generalized trihedral vertex.

A virtual vertex can be calculated from three space lines by the least-squared error method. The virtual vertex is not stable when the three lines are in general position. But it is stably calculated when the three lines are near and they are not parallel. When the three lines strictly meet at the crossing point, the virtual vertex coincides with a real one. Otherwise, it is a virtual concept which is not observed in the image.

We can enumerate all the GTVs from stereoscopic data as well as from a given 3d object model. Locality constraint is applied to a three-line set in order to suppress the computational cost and erroneous combinations which are generated by occlusion. In our current implementation, three line segments are selected when the minimum distance between the terminal points of two lines are smaller than a threshold. This constraint can work very effectively to prevent combinatorial explosion of the number of GTVs.

## 3 GTV-based Object Recognition

### 3.1 System Overview

We can make up an object recognition system based on GTV matching. Rough scheme is as follows:

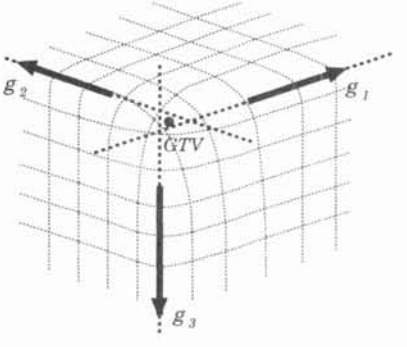



Figure 1: Generalized trihedral vertex.

a) GTV database is precompiled from given 3d object models.

b) A set of GTV is extracted from unknown stereoscopic data.

c) Each GTV is matched with GTV database, and a matched pair generates a possible object model and its pose.

d) Successive pose clustering is accomplished to make a set of feasible poses of a particular object.

e) The object identification and the pose estimation are accomplished using backprojection.

### 3.2 GTV Database

The GTV database is compiled from a given set of 3d object models. All the GTVs are extracted from each object model, and they are labeled with the object name and its object-centered coordinates. Each GTV has the following attributes for efficient matching. We use a N-vector notation $N(\mathbf{x}) = \mathbf{x}/||\mathbf{x}||$ for simple description.

1) Location of the virtual vertex, $\tilde{\mathbf{x}}$

2) Locations of terminal points of the three line segments which form the GTV

3) Three unit vectors which show the directions of the lines, $\tilde{\mathbf{g}}_i (i = 1, 2, 3)$

4) Scalar triple product of the three unit vectors, $|(\tilde{\mathbf{g}}_1 \times \tilde{\mathbf{g}}_2) \cdot \tilde{\mathbf{g}}_3|$

5) Three angles generated by two lines of the three (in descendent order), $\tilde{\phi}_i (i = 1, 2, 3)$

6) Pointers from $\tilde{\phi}_i (i = 1, 2, 3)$ to two lines

7) Orthogonal bases, $\tilde{\mathbf{h}}_i (i = 2, 3)$, which are defined as follows:

$$\begin{array}{ll} \tilde{\mathbf{h}}_1 & = N(\sum_i \tilde{\mathbf{g}}_i) \quad \text{when } |(\tilde{\mathbf{g}}_1 \times \tilde{\mathbf{g}}_2) \cdot \tilde{\mathbf{g}}_3| \leq 0.3 \\ & = N(\tilde{\mathbf{g}}_1 \times \tilde{\mathbf{g}}_2) \quad \textit{otherwise} \end{array}$$

$$\tilde{\mathbf{h}}_2 = N(\tilde{\mathbf{g}}_1 \times \tilde{\mathbf{h}}_1) \qquad \tilde{\mathbf{h}}_3 = \tilde{\mathbf{h}}_1 \times \tilde{\mathbf{h}}_2$$

It is noted that 1) through 3) are given in the object-centered coordinates. Random sampling can be performed when the total numbers of GTVs are too large. These GTVs are classified in the GTV index table, which is coordinated by $\phi_1$ and $\phi_2$ for quick matching.

### 3.3 Extraction of GTVs

Given stereoscopic image data, GTVs can be extracted in the similar way. They are associated with the same attributes as shown in the previous subsection. It should be noted that 1) through 3) are given in the viewer-centered coordinates.

## 4 Pose Candidate Generation from GTVs

### 4.1 Matching GTVs

Each GTV extracted from stereoscopic data is matched to GTVs in the precompiled GTV database. This matching is efficiently performed on the GTV index table. If some candidates are found in the neighborhood of the index, $(\phi_1, \phi_2)$, the corresponding records in the GTV tables are checked. When $\phi_3$ and the scalar triple product of the GTV record in the database are similar to ones of the GTV, the GTV record is selected as a candidate.

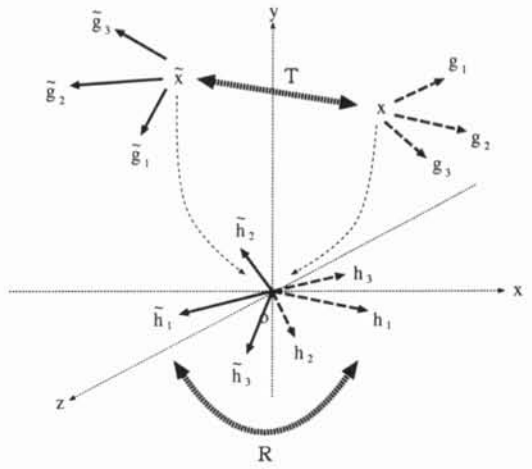



Figure 2: Candidate pose generation from matched GTVs.

### 4.2 Pose Candidate Generation

Rough scheme of the pose generation is shown in Fig. 2. A translation matrix $T$ can be calculated from correspondence of the virtual vertices:

$$T = \mathbf{x} - \tilde{\mathbf{x}}$$

where $\mathbf{x}$ and $\tilde{\mathbf{x}}$ denote positions of the virtual vertices in the given stereoscopic data and in the database, respectively.

For a matched pair of GTVs, a rotation matrix $R$ can be estimated from correspondence of three unit vectors. Let $\mathbf{g}_i (i = 1, 2, 3)$ denote three unit vectors in the given data, and $\tilde{\mathbf{g}}_i (i = 1, 2, 3)$ the corresponding three unit vectors in the database.

(1) Calculate orthonormal bases, $\mathbf{h}_i (i = 1, 2, 3)$:

$$\begin{array}{ll} \mathbf{h}_1 & = N(\sum_i \mathbf{g}_i) \quad \text{when } |(\tilde{\mathbf{g}}_1 \times \tilde{\mathbf{g}}_2) \cdot \tilde{\mathbf{g}}_3| \leq 0.3 \\ & = N(\mathbf{g}_1 \times \mathbf{g}_2) \quad \textit{otherwise} \end{array}$$

$$\mathbf{h}_2 = N(\mathbf{g}_1 \times \mathbf{h}_1) \qquad \mathbf{h}_3 = \mathbf{h}_1 \times \mathbf{h}_2$$

(2) Calculate the rotation matrix, $R$

$$R = (\ \mathbf{h}_1 \quad \mathbf{h}_2 \quad \mathbf{h}_3\ ) \begin{pmatrix} \tilde{\mathbf{h}}_1^T \\ \tilde{\mathbf{h}}_2^T \\ \tilde{\mathbf{h}}_3^T \end{pmatrix}$$

Thus a pose candidate can be estimated from a GTV pair. Note that $\tilde{\mathbf{h}}_i$ is compiled in the GTV database for diminishing the cost in the pose generation.

# 5 Successive Pose Clustering

## 5.1 Problems in Conventional Clustering

It is known that pose clustering techniques, such as the Hough transform and geometric hashing ([1]-[4]), are effective for robust object recognition[5][6]. Although these voting methods are easy to implement, there are two problems with these techniques when the pose space is big. In our problem, the pose space is six dimensional space. We can decompose the pose space into two subspaces of rotation and translation. However, the voting spaces are still three dimensional. If we use these three dimensional spaces for the voting, we have to quantize the space into an appropriate number of cells.

Ikeuchi et al.[7] and Shakunaga et al.[8] proposed successive clustering schemes for appearance based object recognition, for SAR images and for camera images, respectively. Although these schemes have not covered 3d rotation yet, an efficient algorithm can be constructed for the stereoscopic object recognition if the successive clustering is applied to the 3d rotation space. We propose the successive algorithm in this section.

The rough description of the pose clustering is as follows: When a new candidate pose is obtained from a feature correspondence described in the previous section, this pose candidate is examined whether it is within a certain distance, a clustering threshold distance, to one of the existing clusters. If it is within the distance from multiple clusters, the largest cluster will be preferred. Then, the average pose and the size of the cluster will be updated to include the candidate pose. Although a new candidate will be absorbed in the largest cluster even if another cluster is nearer to it, this is not a significant problem, if the order of pose generation is sufficiently randomized.

## 5.2 Equations for Successive Clustering

It is very easy and straightforward to implement the successive clustering in the translation space[7]. When a translation $T$ is generated in the neighborhood of a cluster $T_i$, and the size of the cluster is $n_i$, then $T_i$ should be updated to

$$T_i' = (n_i T_i + T)/(n_i + 1). \qquad (1)$$

On the other hand, a complete clustering algorithm in the rotation space is not easily made. However, an incomplete but effective algorithm can be constructed using Chasles' theorem[9].

When a rotation $R$ is generated in the neighborhood of a rotation cluster $R_i$, and the size of the cluster is $n_i$, then the two rotations can be transformed from one to the other by a rotation around an axis. Let $\mathbf{e} = (\ e_1 \quad e_2 \quad e_3\ )^T$ denote a unit vector along the axis, and $\theta$ denote the amount of rotation. Then a $\theta$-rotation around the vector $\mathbf{e}$ can be shown in:

$$Q(\mathbf{e},\theta) = M \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{pmatrix} M^T \qquad (2)$$

where

$$M = \begin{pmatrix} e_1 & a_1 & b_1 \\ e_2 & a_2 & b_2 \\ e_3 & a_3 & b_3 \end{pmatrix}$$

when $(\ e_1 \quad e_2 \quad e_3\ )^T$, $(\ a_1 \quad a_2 \quad a_3\ )^T$ and $(\ b_1 \quad b_2 \quad b_3\ )^T$ are the orthogonal bases.
By the way, $Q(\mathbf{e},\theta)$ is calculated from $R$ and $R_i$:

$$Q(\mathbf{e},\theta) = RR_i^T = \begin{pmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{pmatrix} \qquad (3)$$

From Eqs. (2) and (3), $c_{ij}$ is expressed by $\mathbf{e}$ and $\theta$ as follows.

$$c_{ii} = e_i^2 + (1 - e_i^2)\cos\theta$$

$$c_{ij} = e_i e_j(1-\cos\theta) + e_{6-i-j}\sin\theta \quad \text{when} \quad i-j = 3k+1$$

$$c_{ij} = e_i e_j(1-\cos\theta) - e_{6-i-j}\sin\theta \quad \text{when} \quad i-j = 3k-1$$

Therefore, $\theta$ and $\mathbf{e}$ can be calculated by

$$\theta = \cos^{-1}\frac{c_{11} + c_{22} + c_{33} - 1}{2} \qquad (4)$$

$$\mathbf{e} = \left(\frac{c_{32} - c_{23}}{2\sin\theta} \quad \frac{c_{13} - c_{31}}{2\sin\theta} \quad \frac{c_{21} - c_{12}}{2\sin\theta}\right)^T \qquad (5)$$

Using $Q(\mathbf{e},\theta)$, $R_i$ can be updated in the similar manner as the translation:

$$R_i' = Q(\mathbf{e}, \theta/(n_i + 1))R_i. \qquad (6)$$

Geometric meaning of this operation is illustrated in Fig. 3. It should be noted that the clustering result is dependent on the order of candidate poses because the pose space is not linear. The stability is, however, experimentally confirmed in 6.1, when all the candidate poses are near enough in the pose space.

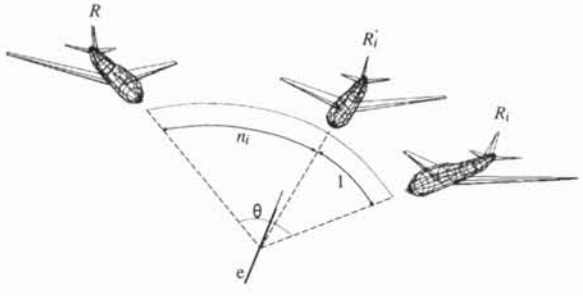
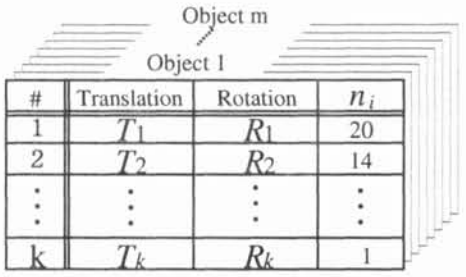



Figure 3: Geometric meaning of successive clustering with $R$.




| # | Translation | Rotation | $n_i$ |
|---|---|---|---|
| 1 | $T_1$ | $R_1$ | 20 |
| 2 | $T_2$ | $R_2$ | 14 |
| ⋮ | ⋮ | ⋮ | ⋮ |
| k | $T_k$ | $R_k$ | 1 |

Figure 4: Pose cluster list.

## 5.3 Efficient Implementation

Using equations described in the previous subsection, an efficient algorithm can be constructed as follows:

### 5.3.1 Pose Cluster List

The pose cluster list is defined as a list of pose cluster. Each record in the list consists of a record number $i$, a translation $T_i$, a rotation $R_i$ and the number $n_i$ of member poses in the cluster as shown in Fig. 4. No record is initially included in the list, and the list is kept sorted in the descending order of $n_i$ as shown in 5.3.4.

The maximum limit $k_{max}$ can be used for shortening the calculation cost where $k_{max}$ is controlled so as to be not too large and not too small.

### 5.3.2 Finding the Nearest Pose

When a candidate pose is generated, each record in the pose cluster list is sequentially checked whether it is similar to the candidate pose. To cut the calculation cost, $||T - T_i||$ is first checked. If it is bigger than a threshold, $\epsilon_T$, the record is skipped. Only if it is smaller than the threshold, $\theta$ is calculated by using Eqs. (4) and (3) from $R$ and $R_i$. If $|\theta|$ is bigger than a threshold, $\epsilon_\theta$, the pose cluster is selected. Otherwise the record is skipped.

When no similar pose can be found in the list, the candidate pose is registered as a new pose cluster unless the number of record, $k$, is over the limit, $k_{max}$.

### 5.3.3 Updating $T_i$, $R_i$ and $n_i$

When the similar pose is selected as described in the previous subsection, $T_i$, $R_i$ and $n_i$ are updated in this order. $T_i$ is simply updated by Eq.(1). To update $R_i$, $\mathbf{e}$ is first calculated by Eq.(5), then $Q(\mathbf{e}, \theta/(n_i+1))$ is calculated, and $R_i$ is updated by Eq. (6). The cluster size $n_i$ is incremented at last.

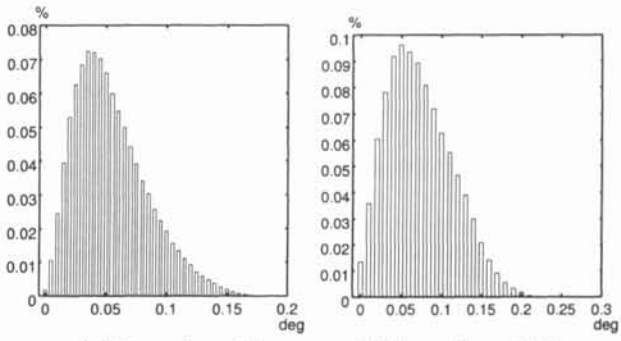
### 5.3.4 Maintenance of the sorted list

When $n_i$ is incremented, it should be compared with $n_j (j = i-1, i-2, ..., 1)$ in this order. If the $j$-th record is the first record that satisfies $n_j \geq n_i$, and if $j \neq i-1$, then the $i$-th record and the $(j-1)$th record are exchanged. Otherwise, no exchange is occurred.

### 5.3.5 Termination of the clustering

The clustering process terminates either when a size of the largest cluster is large enough or when the total number of generated pose candidates reaches a threshold. In the both cases, the largest cluster is selected as a candidate pose and confirmed by the backprojection.

We can alternatively select several large clusters as hypotheses and test them using a correspondence process. This may be very effective for object detection in very cluttered scenes.




(a)$\mathbf{R}_{10}$, $\theta_0 = 5deg.$ (b)$\mathbf{R}_{10}$, $\theta_0 = 10deg.$

Figure 5: Stability of the successive clustering in rotation space.

# 6 Experimental Results

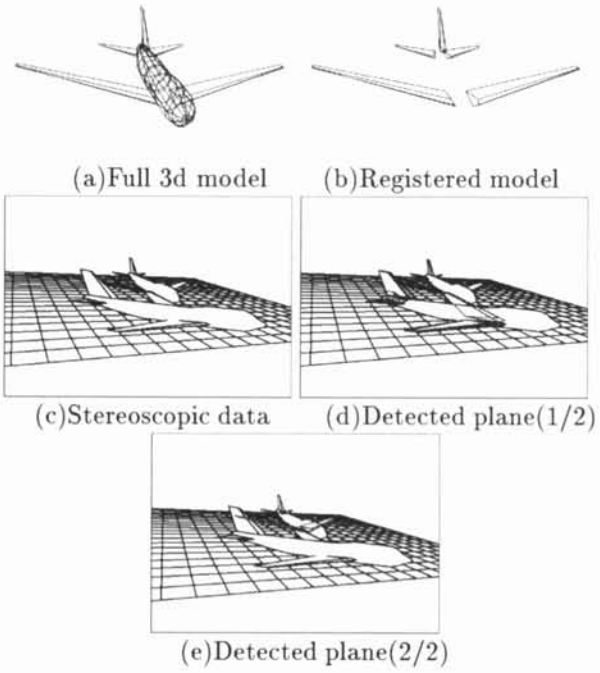
## 6.1 Stability of Successive Clustering

The clustering result of translation is independent of the order of translations unless the member translations change because some different clusters exist in the neighborhood of the cluster. On the other hand, the clustering result of rotation depends on the order of candidate poses. We confirmed the stability of the algorithm by the simulation.

In the experiment, a set of candidate poses, $\mathbf{R}_n = \{R\}$ are generated at random in $\theta_0$ from a given pose, $R_0$, where $n$ shows the number of poses. Then test sequences are made by randomly permuting the candidate poses and the successive pose clustering is tried for each test sequence. Let $R(j)$ denote the final pose for the $j$-th test sequence. Then the distribution of differences in $|\theta|$ is checked over all the pairs of $R(j)$ and $R(k)$ for $\mathbf{R}_5$, $\mathbf{R}_{10}$ and $\mathbf{R}_{50}$. The typical results are shown in Fig. 5, for $\theta_0 = 5$ and 10 deg. When $\theta_0$ is not smaller than 10 deg., the result pose difference never exceeds $\theta_0/20$ in the simulations. When the number of merged poses is 5, the result pose difference never exceeds $\theta_0/100$ for any $\theta_0$. These results show that the successive pose clustering is stable enough when $\theta_0$ is no more than 10 deg.

## 6.2 Experiment on Simulation Data

(1) Air plane detection



(a)Full 3d model (b)Registered model

(c)Stereoscopic data (d)Detected plane(1/2)

(e)Detected plane(2/2)

Figure 6: Experiment for airplane model.

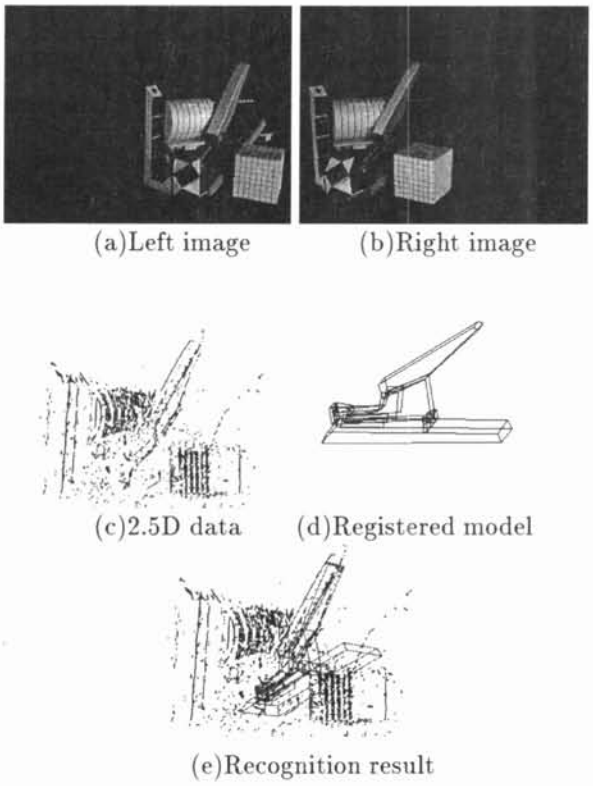
Figure 6 shows an experimental result for a plane model. In this experiment, a partial shape model as shown in Fig. 6(b) is registered instead of the full 3d model which is shown in Fig. 6 (a). The registered model consists of five wings, and doesn't include the body part. For the generated stereoscopic scene as shown in Fig. 6 (c), the object recognition system correctly extracted two planes as shown in (d) and (e).

### 6.3 Experiment on Real Image Data

Finally, the experiments were accomplished on real stereoscopic data as shown in Fig. 7. From a pair of images, (a) and (b), a stereoscopic line data shown in (c) is derived by a line-based stereo algorithm. Then the stapler model as shown in (d) is matched with the stereoscopic data. Figure 7(e) shows the correctly detected pose, which is superimposed to the 2.5D data.

## 7 Conclusions

A 3d object recognition algorithm is proposed for stereoscopic data. The successive pose clustering algorithm is confirmed to work with pose generation based on GTV matching in this paper. However, the algorithm can be applied to other types of 3d data such as 3d point set generated by the SVD algorithm.



(a)Left image (b)Right image

(c)2.5D data (d)Registered model

(e)Recognition result

Figure 7: Experiment on real data.

## References


[1] Ballard, D. H., "Generalizing the Hough Transform to detect arbitrary patterns," Pattern Recognition, vol. 13, no. 2, pp. 111-122, 1981.

[2] Lamdan, Y. and H. J. Wolfson, "Geometric Hashing: A general and efficient model-based recognition scheme," Proc. ICCV'88, pp.238-249, 1988.

[3] Stockman, G. S., "Object recognition and localization via pose clustering," Comp. Vision, Graphics, Image Proc., vol. 40, pp.361-387, 1987.

[4] Dhome, M., M. Richetin and G. Rives, "Model-based recognition and location of local patterns in polygonal contours via hypothesis accumulation," Pattern Recognition in Practice II, edited by E. S. Gelsema and L. N. Kanal, North-Holland, 1986.

[5] Grimson, W. E. L., "Object recognition by computer," The MIT Press, 1990.

[6] Wolfson, H. J. and Y. Lamdan, "Transformation invariant indexing", in Geometric Invariance in Computer Vision (J. Mundy and A.Zisserman edit, MIT Press), pp.335-353, 1992.

[7] Ikeuchi K., T. Shakunaga, M. Wheeler and T. Yamazaki, "Invariant Histograms and Deformable Template Matching for SAR Target Recognition," Proc. CVPR'96, pp.100-105, June 1996.

[8] Shakunaga T., K. Ikeuchi and T. Kanade, "Object Description and Recognition Using Invariant Map," Trans. IEICE, vol. J80-D-II, no. 6, pp.1466-1474, 1997(in Japanese).

[9] Chen, H. H., "A screw motion approach to uniqueness analysis of head-eye geometry," Proc. CVPR'91, pp.145-151, June 1991.