コンピュータビジョン 99－7
（1996. 3. 21）

Preprint from Technical Report of SIG-CV meeting of IPSJ, CV99
21th March, 1996 at Tokyo University of Marinterm

投票による3次元運動解析

井宮 淳, イリス フェルミン, 市川 熹

千葉大学 工学部 情報工学科

〒263 千葉市稲毛区弥生町 1-33

imiya@ics.tj.chiba-u.ac.jp, firis@@icsd4.tj.chiba-u.ac.jp

あらまし　従来, 物体表面の基準点の動きを追跡したり, 基準点の画像上での対応が決定されていることを仮定して物体の運動変数を推定してきた. しかし, 実際の運動解析では, 必ずしも物体の基準点が事前に決定されているとは限らない. そこで本論文では対象物体が多面体である場合に, 運動前と運動後との各々の物体において, 平面上にない4つの頂点の決める4面体の頂点を対応点の候補として運動変数を計算する手法を提案する. 提案する手法は, 窓掛け確率ハフ変換の考えを3次元の運動解析に応用したものである.

キーワード　剛体運動, 運動変数, 確率算法, 対応点, 多面体

3-D Motion Detection by Randomized Search

Atsushi Imiya, Iris Fermin, and Akira Ichikawa

Department of Information and Computer Sciences

Chiba University

1-33 Yayoi-cho, Inage-ku 263, Chiba, Japan

e-mail: imiya@ics.tj.chiba-u.ac.jp, firis@icsd4.tj.chiba-u.ac.jp

ichikawa@ics.tj.chiba-u.ac.jp

Abstract

Geometric hashing is a technique for the recognition of partially occluded objects from noisy data using a voting process. We use this voting method to detect the motion parameters in three-dimensional Euclidean space. We present two formal approaches for motion detection, one model-based and the other non-model-based. The first approach solves the motion equation directly. The second approach involves two steps as in classical geometric hashing. The first step is a preprocessing stage for the generation of the object model and the second step is the voting process to solve the motion equation.

Key words: motion parameters, probabilistic algorithm, geometric hashing, congruence checking

1. Introduction

Computational geometry focuses on the analysis of complexities of algorithms which solve geometric problems such as construction of convex hulls, generation of Voronoi diagrams and congruence checking. Congruence checking provides fundamental techniques for pattern recognition of planar and spatial point sets. In congruence checking, inputs are basically pairs of point sets in Euclidean space and outputs are binary Boolean values. On the other hand, in motion detection we estimate the motion parameters, rotation and translation, in three-dimensional Euclidean space using a series of images of the same object. However, during object tracking, some parts of each object may be occluded, due to for instance, the configuration of the cameras and objects in the space. This means that we obtain a partial point set from each image frame. Thus, the classical technique of congruence checking developed in computational geometry is not sufficient for motion estimation. Here we present an extension of classical congruence checking and two probabilistic algorithms for motion detection.

The paper is organized as follows. In section 2, we give a brief review of geometric hashing. In section 3, we describe the formal model developed for motion detection under a Euclidean transformation (rotation and translation). Section 4 describes the probabilistic algorithms, section 5 shows some results obtained using synthetic data, and finally conclusions are presented in section 6.

2. Geometric Hashing

Geometric hashing is a model-based approach for object recognition which is based on the idea of storing transformation-invariant information of an object in a hash table (Grimson and Huttenlocher[1], Wolfson[2]). Geometric hashing involves two stages. The first is a preprocessing stage in which a model representation is constructed by computing and storing transformation-invariant information of the object in a hash table. The second stage is recognition in which the same invariants are computed for the objects in a scene and used as indexing keys to retrieve information from the hash table. If there are a sufficient number of matching entries and invariants of an object the algorithm concludes that the object is similar to the model stored in the table.

Lamdan and Wolfson[3] described the recognition of 2-D and 3-D models under similarity transformation (invariant under rotation, translation and scale), by defining an orthogonal coordinate frame based on an ordered pair of points from the set of image points and representing all other points by their coordinates in this frame. For the determination of translation and rotation in 2-D a two-point basis is required. Assuming range data, the 3-D rigid motion can be determined by finding a correspondence between coordinate frames, using a three-point basis. Lamdan and Wolfson also defined the object recognition problem under affine transformation.

Lamdan, Schwartz and Wolfson[4] considered the problem of object recognition under affine transformation. If there is an affine transformation between a pair of point sets, there is an invertible matrix $\boldsymbol{A}$ and a translation vector $\boldsymbol{b}$ for which the pair of sets are related by $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$, where $\boldsymbol{x}$ and $\boldsymbol{y}$ are corresponding points. For the 2-D case, $\boldsymbol{A}$ is a 2×2 matrix and $\boldsymbol{b} \in R^2$. The problem is to recognize the objects in the scene and to find the affine transformation for each recognized object. Although the complexity of the hashing method is exponential, it can be used to recognize many partially occluded objects simultaneously. Also, the model can be described using many kinds of features, including points, lines, and curves (Tsai[5]).

3. Motion Dectection

Two methods have been used for 3-D motion detection. The first involves iterative solution of nonlinear equations and the second solves the problem using linear equations. Here, we present a linear approach assuming that at least one quadruplet of point correspondences is known (Longuet-Higgins[6]).

3.1 Point Correspondences

Let $\{\boldsymbol{x}_i\}_{i=1}^k$ and $\{\boldsymbol{y}_i\}_{i=1}^k$ be a pair of point sets in n-dimensional Euclidean space R^n. The congruence checking is described as follows: for a rotation matrix $\boldsymbol{R}$, translation vector $\boldsymbol{a}$ and permutation σ, if

$$\boldsymbol{y}_i = \boldsymbol{R}\boldsymbol{x}_{\sigma(i)} + \boldsymbol{a}$$

then return true, otherwise return null.

In computer vision, we deal with objects in three-dimensional Euclidean space R^3. Let visible sets of points be $\boldsymbol{X}$ and $\boldsymbol{Y}$, where $\boldsymbol{X} \subseteq \{\boldsymbol{x}_i\}_{i=1}^k$ and $\boldsymbol{Y} \subseteq \{\boldsymbol{y}_i\}_{i=1}^k$. If $\{\boldsymbol{X}\} = \{\boldsymbol{x}_i\}_{i=1}^k$ and $\{\boldsymbol{Y}\} = \{\boldsymbol{y}_i\}_{i=1}^k$ there are no occluded parts. Otherwise, there are occluded parts in each frame. Usually, $|\boldsymbol{Y}| \neq |\boldsymbol{X}|$ holds, where $|.|$ is the number of points.

In the following, we assume $|\boldsymbol{Y}| = M$ and $|\boldsymbol{X}| = N$. Setting $k = \min(M,N)$, we define the permutation of kth order σ_k. For $\boldsymbol{x}_i \in \boldsymbol{X}$ and $\boldsymbol{y}_i \in \boldsymbol{Y}$ if we know that

$$\boldsymbol{y}_i = \boldsymbol{R}\boldsymbol{x}_{\sigma_k(i)} + \boldsymbol{a} \tag{1}$$

for at least four sets of i and $\sigma_k(i)$, we can solve equation (1) and determine the motion parameters. This problem is called the correspondence problem.

Equation (1) means that $\boldsymbol{Y}$ is obtained from $\boldsymbol{X}$ by a transformation

$$\boldsymbol{T} = \left(\begin{array}{c|c} \boldsymbol{R} & \boldsymbol{a} \\ \hline \boldsymbol{0}^{\mathrm{T}} & 1 \end{array}\right). \tag{2}$$

Thus, using the constraints $\boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{I}$ and $|\boldsymbol{R}| = 1$, one can solve equation (1)

$$\boldsymbol{y}_i = \boldsymbol{T}\boldsymbol{x}_i \tag{3}$$

given a quadruplet of corresponding points.

Let $\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k, \boldsymbol{x}_m$ and $\boldsymbol{y}_i, \boldsymbol{y}_j, \boldsymbol{y}_k, \boldsymbol{y}_m$ be corresponding points such that

$$\begin{pmatrix} \boldsymbol{y}_i & \boldsymbol{y}_j & \boldsymbol{y}_k & \boldsymbol{y}_m \\ 1 & 1 & 1 & 1 \end{pmatrix} = \left(\begin{array}{ccc|c} & \boldsymbol{R} & & \boldsymbol{a} \\ 0 & 0 & 0 & 1 \end{array}\right) \begin{pmatrix} \boldsymbol{x}_i & \boldsymbol{x}_j & \boldsymbol{x}_k & \boldsymbol{x}_m \\ 1 & 1 & 1 & 1 \end{pmatrix}. \tag{4}$$

This leads to the solution

$$\left(\begin{array}{ccc|c} & \boldsymbol{R} & & \boldsymbol{a} \\ 0 & 0 & 0 & 1 \end{array}\right) = \begin{pmatrix} \boldsymbol{y}_i & \boldsymbol{y}_j & \boldsymbol{y}_k & \boldsymbol{y}_m \\ 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_i & \boldsymbol{x}_j & \boldsymbol{x}_k & \boldsymbol{x}_m \\ 1 & 1 & 1 & 1 \end{pmatrix}^{-1}. \tag{5}$$

3.2 Determination of the Motion Parameters

In the following we assume that the correspondences are not determined. Here, we apply the basic idea of the hashing method to motion detection. The idea behind the hash table is that each k-tuple of model points forms a basis for a coordinate system that is invariant under possible model transformations (Grimson and Huttenlocher[1]). Thus, all model points can be stored in a transformation-invariant form by rewriting them in terms of this reference frame. The rewritten model points are used as indices in the hash table, where corresponding basis k-tuples are stored. We derive a method for estimating the correspondence of points using the idea of geometric hashing. For a pair of quadruplets $\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k, \boldsymbol{x}_m$ and $\boldsymbol{y}_i, \boldsymbol{y}_j, \boldsymbol{y}_k, \boldsymbol{y}_m$, setting

$$\begin{aligned} \boldsymbol{x}_{\gamma m} &= \boldsymbol{x}_i - \boldsymbol{x}_m \\ \boldsymbol{y}_{\gamma m} &= \boldsymbol{y}_i - \boldsymbol{y}_m, \end{aligned} \tag{6}$$

for $\gamma = i, j, k$, we define a set of orthonormal bases

$$\boldsymbol{\xi}_{im} = \frac{\boldsymbol{x}_{im}}{|\boldsymbol{x}_{im}|} \tag{7}$$

$$\boldsymbol{\xi}_{jm} = \frac{\boldsymbol{x}_{jm} - (\boldsymbol{x}_{jm}^T\boldsymbol{\xi}_{im})\boldsymbol{\xi}_{im}}{|\boldsymbol{x}_{jm} - (\boldsymbol{x}_{jm}^T\boldsymbol{\xi}_{im})\boldsymbol{\xi}_{im}|} \tag{8}$$

$$\boldsymbol{\xi}_{km} = \frac{\boldsymbol{x}_{im} \times \boldsymbol{x}_{jm}}{|\boldsymbol{x}_{im} \times \boldsymbol{x}_{jm}|} \tag{9}$$

and

$$\boldsymbol{\eta}_{im} = \frac{\boldsymbol{y}_{im}}{|\boldsymbol{y}_{im}|} \tag{10}$$

$$\boldsymbol{\eta}_{jm} = \frac{\boldsymbol{y}_{jm} - (\boldsymbol{y}_{jm}^T\boldsymbol{\eta}_{im})\boldsymbol{\eta}_{im}}{|\boldsymbol{y}_{jm} - (\boldsymbol{y}_{jm}^T\boldsymbol{\eta}_{im})\boldsymbol{\eta}_{im}|} \tag{11}$$

$$\boldsymbol{\eta}_{km} = \frac{\boldsymbol{y}_{im} \times \boldsymbol{y}_{jm}}{|\boldsymbol{y}_{im} \times \boldsymbol{y}_{jm}|} \tag{12}$$

where $|.|$ is the length of a vector and $\boldsymbol{x}^T\boldsymbol{y}$ is the inner product of $\boldsymbol{x}$ and $\boldsymbol{y}$. If we set

$$\alpha_{im} = \boldsymbol{x}_{km}^T\boldsymbol{\xi}_{im}, \quad \alpha_{jm} = \boldsymbol{x}_{km}^T\boldsymbol{\xi}_{jm}, \quad \alpha_{km} = \boldsymbol{x}_{km}^T\boldsymbol{\xi}_{km} \tag{13}$$

and

$$\beta_{im} = \boldsymbol{y}_{km}^T\boldsymbol{\eta}_{im}, \quad \beta_{jm} = \boldsymbol{y}_{km}^T\boldsymbol{\eta}_{jm}, \quad \beta_{km} = \boldsymbol{y}_{km}^T\boldsymbol{\eta}_{im}, \tag{14}$$

we have

$$\begin{aligned} \boldsymbol{x}_{km} &= \alpha_{im}\boldsymbol{\xi}_{im} + \alpha_{jm}\boldsymbol{\xi}_{jm} + \alpha_{km}\boldsymbol{\xi}_{km} \\ \boldsymbol{y}_{km} &= \beta_{im}\boldsymbol{\eta}_{im} + \beta_{jm}\boldsymbol{\eta}_{jm} + \beta_{km}\boldsymbol{\eta}_{km}. \end{aligned} \tag{15}$$

For a pair of triplets of real values $(\alpha_{im}, \alpha_{jm}, \alpha_{km})$ and $(\beta_{im}, \beta_{jm}, \beta_{km})$ which are determined from the quadruplets, the following theorem holds.

Theorem 1 *If*

$$\boldsymbol{y}_i = \boldsymbol{R}\boldsymbol{x}_i + \boldsymbol{a},$$

then $\alpha_{\gamma m} = \beta_{\gamma m}\ \forall\ \gamma = i, j, k.$

Proof. Since

$$\boldsymbol{y}_i = \boldsymbol{R}\boldsymbol{x}_i + \boldsymbol{a}$$

and

$$\boldsymbol{y}_{\gamma m} = \boldsymbol{R}\boldsymbol{x}_{\gamma m},$$

we have

$$\boldsymbol{\eta}_{\gamma m} = \boldsymbol{R}\boldsymbol{\xi}_{\gamma m}. \tag{16}$$

This leads to

$$\begin{aligned} \beta_{\gamma m} &= \boldsymbol{y}_{\gamma m}^T\boldsymbol{\eta}_{\gamma m} \\ &= \boldsymbol{x}_{\gamma m}^T\boldsymbol{R}^T\boldsymbol{R}\boldsymbol{\xi}_{\gamma m} \\ &= \boldsymbol{x}_{\gamma m}^T\boldsymbol{\xi}_{\gamma m} \\ &= \alpha_{\gamma m}, \quad \textit{where } \gamma = i, j, k. \end{aligned} \tag{17}$$

(Q.E.D.)

4. Algorithms for Motion Detection

From theorem 1 we can conclude that if a part of $\boldsymbol{Y}$ is the result of the application of a Euclidean motion to a part of $\boldsymbol{X}$, then $\alpha_{\gamma m} = \beta_{\gamma m}$ and $\boldsymbol{X}$ is congruent to $\boldsymbol{Y}$. However, if $\alpha_{\gamma m} = \beta_{\gamma m}$, we cannot conclude that $\boldsymbol{y}_i = \boldsymbol{R}\boldsymbol{x}_i + \boldsymbol{a}$, since the rigid motion is a sufficient condition for $\alpha_{\gamma m} = \beta_{\gamma m}$ of which correspondences comprise a many-to-one mapping. However, using the idea of voting, we can define an inverse transformation from $\alpha_{\gamma m} = \beta_{\gamma m}$ to $\boldsymbol{y}_i = \boldsymbol{R}\boldsymbol{x}_i + \boldsymbol{a}$.

In this section we describe two different algorithms to solve the problem of determining motion parameters in three-dimensional space using the idea of the voting process of the geometric hashing method.

4.1 One-Step Algorithm

In this algorithm we have no *a priori* assumptions about the object model, and the motion parameters are determined directly using two corresponding sets of image points. Thus, we call this algorithm a non-model-based algorithm. Figure 1 shows the concept involved in this algorithm. We describe this algorithm as follows.

Algorithm 1

1. For all quadruplets $\{\boldsymbol{x}_1,\boldsymbol{x}_2,\boldsymbol{x}_3,\boldsymbol{x}_4\} \in \boldsymbol{X}$ and $\{\boldsymbol{y}_1,\boldsymbol{y}_2,\boldsymbol{y}_3,\boldsymbol{y}_4\} \in \boldsymbol{Y}$, where $\boldsymbol{X}$ and $\boldsymbol{Y}$ are the sets of the object vertices:

2. Compute $\{\ \boldsymbol{x}_{\gamma m},\boldsymbol{\xi}_{\gamma m},\alpha_{\gamma m}\ \}$ and $\{\boldsymbol{y}_{\gamma m},\boldsymbol{\eta}_{\gamma m},\beta_{\gamma m}\ \}$ where $\gamma = 1,2,3$ and $m = 4$.

3. If
$$\sum_{\gamma=1}^{3}(\alpha_{\gamma m} - \beta_{\gamma m})^2 < \epsilon^2 \qquad (18)$$
then set
$$\boldsymbol{R} = [\boldsymbol{\eta}_{14}\boldsymbol{\eta}_{24}\boldsymbol{\eta}_{34}]\,[\boldsymbol{\xi}_{14}\boldsymbol{\xi}_{24}\boldsymbol{\xi}_{34}]^T. \qquad (19)$$
If
$$\sum_{i<j} \|\boldsymbol{y}_{ij} - \boldsymbol{R}\boldsymbol{x}_{ij}\|^2 < \delta^2 \qquad (20)$$
then set
$$\boldsymbol{a} = \frac{1}{4}\sum_{i=1}^{4}(\boldsymbol{y}_i - \boldsymbol{R}\boldsymbol{x}_i). \qquad (21)$$

4. Increment $\boldsymbol{R}$ and $\boldsymbol{a}$ by one.

5. If a threshold in the parameter space ($\boldsymbol{R}$, $\boldsymbol{a}$) is reached, then stop, otherwise go to step 1.

4.2 Two-Step Algorithm

The second algorithm consists of two stages. The first is a preprocessing stage in which the object models are constructed using the image points of the first frame and stored them in a hash table. The second is a recognition stage where, for the points of the second image frame, the algorithm determines the model parameters in each iteration and verifies whether these parameters match one of the models stored in the hash table. If there is a match between an entry of the hash table and the parameters of the second image frame then the algorithm computes the motion parameters. Hence this algorithm is a model-based algorithm (see figure 2). We show the details as follows.

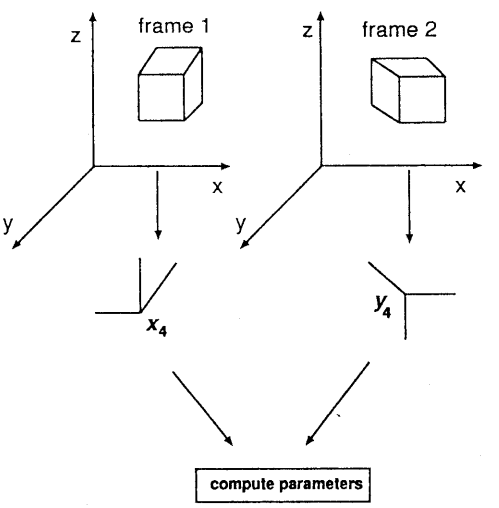


Figure 1: *Algorithm 1*

Algorithm 2

1.1 For all quadruplets $\{\ \boldsymbol{x}_1,\boldsymbol{x}_2,\boldsymbol{x}_3,\boldsymbol{x}_4\ \} \in \boldsymbol{X}$, where $\boldsymbol{X}$ is the set of vertices:

1.2 Compute $\{\ \boldsymbol{x}_{\gamma m},\boldsymbol{\xi}_{\gamma m},\alpha_{\gamma m}\ \}$ where $\gamma = 1,2,3$, $m = 4$ and make a table.

2.1 For a quadruplet $\{\ \boldsymbol{y}_1,\boldsymbol{y}_2,\boldsymbol{y}_3,\boldsymbol{y}_4\ \} \in \boldsymbol{Y}$, where $\boldsymbol{Y}$ is the set of vertices:

2.2 Compute $\{\ \boldsymbol{y}_{\gamma m},\boldsymbol{\eta}_{\gamma m},\beta_{\gamma m}\ \}$ where $\gamma = 1,2,3$ and $m = 4$.

2.3 If
$$\sum_{\gamma=1}^{3}(\alpha_{\gamma m} - \beta_{\gamma m})^2 < \epsilon^2$$
then
$$\boldsymbol{R} = [\boldsymbol{\eta}_{14}\boldsymbol{\eta}_{24}\boldsymbol{\eta}_{34}]\,[\boldsymbol{\xi}_{14}\boldsymbol{\xi}_{24}\boldsymbol{\xi}_{34}]^T.$$
If
$$\sum_{i<j} \|\boldsymbol{y}_{ij} - \boldsymbol{R}\boldsymbol{x}_{ij}\|^2 < \delta^2$$
then
$$\boldsymbol{a} = \frac{1}{4}\sum_{i=1}^{4}(\boldsymbol{y}_i - \boldsymbol{R}\boldsymbol{x}_i).$$

2.4 Increment $\boldsymbol{R}$ and $\boldsymbol{a}$ by one.

3 If a threshold is reached in the parameter space ($\boldsymbol{R}$, $\boldsymbol{a}$) then stop, otherwise go to step 2.1.

We can say that this algorithm knows *a priori* the object representation, whereas in contrast, the first algorithm computes the motion parameters directly, using two images simultaneously.

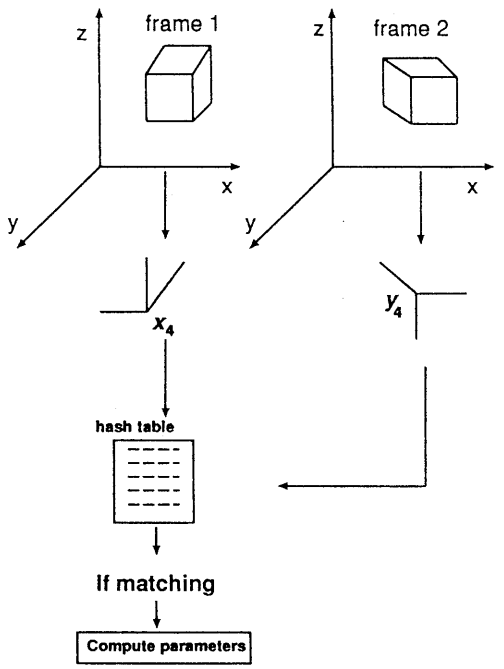


Figure 2: *Algorithm 2*

5. Implementation and Results

In this section we describe some assumptions concerning the objects and points and summarize the results of the implementation of algorithm 1.

5.1 Implementation

Assuming that our object is a trihedral polyhedron and knowing the locations of the vertices in each image frame, the algorithm in figure computes motion parameters in a three-dimensional space. This is the implementation of the first algorithm mentioned above. Here we randomly select points in each frame and compute the parameters and vote them until a threshold is reached.The parameter space is a twelve-dimensional space which corresponds to the elements of the rotation matrix and the translation vector.

The rotation matrix used to generate the transformation is expressed as

$$\boldsymbol{R} = \begin{pmatrix} \sin\psi\sin\theta & \sin\psi\cos\theta & -\cos\psi \\ \cos\psi\sin\theta & \cos\psi\cos\theta & \sin\psi \\ \cos\theta & -\sin\theta & 0 \end{pmatrix}, \quad (22)$$

and the translation vector is

$$\boldsymbol{a} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}. \quad (23)$$

The rotation angles are ψ, θ, and ϕ about z, x and y axes, respectively. We assume that $0 \leq \psi < 2\,\pi$ and $0 \leq \theta < \pi$.

5.2 Results

We present the results of computation of the motion parameters for three different synthetic objects using algorithm 1. For each object we show tables for translation estimation, and rotation estimation for x and z axes. We assume that the rotation about the y axis is $\frac{\pi}{2}$. Furthermore, we show the computation thresholds and coordinates of the polyhedron vertices. The consumption time is 1.8 seconds in the worst case, when the number of iteration cycles is 8500.

6. Conclusions

We have presented two probabilistic algorithms for motion estimation using the ideas of congruence checking and geometric hashing. The first algorithm has the advantage of requiring less memory than the second and the complexity of both algorithms increases with the number of polyhedron vertices. We obtained accurate estimates of motion parameters. However in testing we noticed that neither the estimation of rotation angle using $\sin^{-1}$ nor the estimation of the translation along the z axis is stable. This implies that the algorithm must execute more iteration cycles, and sometimes be rerun.

Acknowledgement

The authors thank Professor. T. Wada of Okayama University for helpful discussions on the difference between the two algorithms.

References

[1] Grimson, W. E. and Huttenlocher, D., On the sensitivity of geometric hashing, in *IEEE Proc. 3rd Int. Conf. on Computer Vision*, pp. 334-338, 1990.

[2] Wolfson, H., Model-based object recognition by geometric hashing, *Lecture Notes in Computer Science 427, Computer Vision-ECCV 90*, pp.526-536, Springer-Verlag, 1990.

[3] Lamdan, Y. and Wolfson, H., Geometric hashing: a general and efficient model-based recognition scheme, in *IEEE Proc. Second International Conference of Computer Vision (Tampa, FL)*, 1988.

[4] Lamdan, Y., Schwartz, J. and Wolfson, H., Affine invariant model-based object recognition, *IEEE Transactions on Robotics and Automation*, **6**, 5, pp. 578-589, 1990.

[5] Tsai, F., Geometric hashing with line features, *Pattern Recognition* **27**, 3, pp. 377-389, 1994.

[6] Longuet-Higgins, H.C., A computer algorithm for reconstructing a scene from two projections, *Nature*, **293**, pp. 133-135, 1981.

Table 1: *Translation estimation*

Polyhedron 1	translation		
	x	y	z
Real	3	5	7
Estimation 1	2.95	5.30	5.30
Estimation 2	2.85	5.44	4.76
Estimation 3	2.89	5.34	5.24

Table 2: *Rotation Angle X axis*

Polyhedron 1\Theta 17.0°	$\cos^{-1}$	$\sin^{-1}$
Estimation 1	17.00°	17.00°
Estimation 2	16.63°	16.63°
Estimation 3	16.97°	16.97°

Table 3: *Rotation Angle Z axis*

Polyhedron 1\ Psi 27.0°	$\cos^{-1}$	$\sin^{-1}$
Estimation 1	27.00°	27.48°
Estimation 2	27.00°	29.17°
Estimation 3	27.00°	23.81°

Table 4: *Thresholds for polyhedron 1*

Polyhedron 1	Iterations	threshold
Estimation 1	3297	10
Estimation 2	8329	10
Estimation 3	3588	10

Table 5: *Vertices of polyhedron 1*

frame 1			frame 2		
x	y	z	x	y	z
0	0	0	3.0000	5.0000	7.0000
4	0	0	3.5309	6.0420	10.8252
4	0	5	-0.9240	8.3119	10.8252
2	0	7	-2.9715	8.6989	8.9126
0	0	5	-1.4550	7.2699	7.0000
4	6	0	6.1358	11.1544	9.0709
4	6	5	1.6808	13.4244	9.0709
2	6	7	-0.3666	13.8113	7.1583
0	6	5	1.1498	12.3823	5.2457
0	6	0	5.6049	10.1124	5.2457

Table 6: *Translation estimation*

Polyhedron 2	translation		
	x	y	z
Real	3	5	7
Estimation 1	2.66	6.06	5.29
Estimation 2	3.42	5.37	5.72
Estimation 3	3.66	5.28	4.79

Table 7: *Rotation Angle X axis*

Polyhedron 2\Theta 17.0°	$\cos^{-1}$	$\sin^{-1}$
Estimation 1	16.90°	16.90°
Estimation 2	17.00°	17.00°
Estimation 3	17.00°	17.00°

Table 8: *Rotation Angle Z axis*

Polyhedron 2\ Psi 27.0°	$\cos^{-1}$	$\sin^{-1}$
Estimation 1	27.00°	31.48°
Estimation 2	27.00°	26.79°
Estimation 3	26.69°	24.20°

Table 9: *Thresholds polyhedron 2*

Polyhedron 2	Iterations	threshold
Estimation 1	5465	10
Estimation 2	5819	10
Estimation 3	4720	10

Table 10: *Vertices of polyhedron 2*

frame 1			frame 2		
x	y	z	x	y	z
0	0	0	3.0000	5.0000	7.0000
4	0	0	3.5309	6.0420	10.8252
4	0	4	-0.0330	7.8579	10.8252
2	0	2	1.4834	6.4289	8.9126
0	0	4	-0.5640	6.8159	7.0000
4	6	0	6.1358	11.1544	9.0709
4	6	4	2.5718	12.9704	9.0709
2	6	2	4.0883	11.5414	7.1583
0	6	4	2.0408	11.9284	5.2457
0	6	0	5.6049	10.1124	5.2457

Table 11: *Translation estimation*

Polyhedron 3	translation		
	x	y	z
Real	3	5	7
Estimation 1	3.12	5.18	4.89
Estimation 2	3.12	5.21	5.65
Estimation 3	3.56	5.25	5.79

Table 12: *Rotation Angle X axis*

Polyhedron 3\Theta 30.0°	$\cos^{-1}$	$\sin^{-1}$
Estimation 1	30.00°	30.00°
Estimation 2	30.00°	30.00°
Estimation 3	30.00°	30.00°

Table 13: *Rotation Angle Z axis*

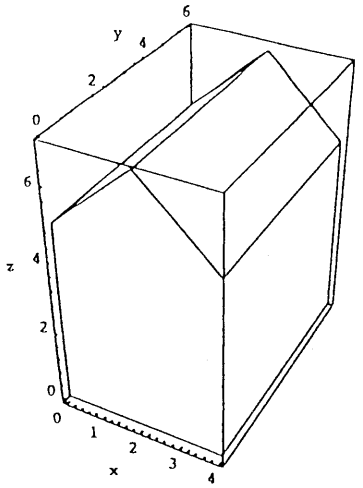
Polyhedron 3\ Psi 45.0°	$\cos^{-1}$	$\sin^{-1}$
Estimation 1	45.00°	47.33°
Estimation 2	45.00°	47.33°
Estimation 3	45.00°	45.80°

Table 14: *Thresholds for polyhedron 3*

Polyhedron 3	Iterations	threshold
Estimation 1	736	10
Estimation 2	866	10
Estimation 3	1561	10

Table 15: *Vertices of polyhedron 3*

frame 1			frame 2		
x	y	z	x	y	z
0	0	0	3.0000	5.0000	7.0000
3	0	0	4.0606	6.0606	9.5980
3	0	3	1.9393	8.1819	9.5980
0	0	3	0.8786	7.1213	7.0000
3	2	0	5.2854	7.2854	8.5980
3	2	3	3.1640	9.4067	8.5980
0	2	3	2.1034	8.3460	6.0000
0	2	0	4.2247	6.2247	6.0000



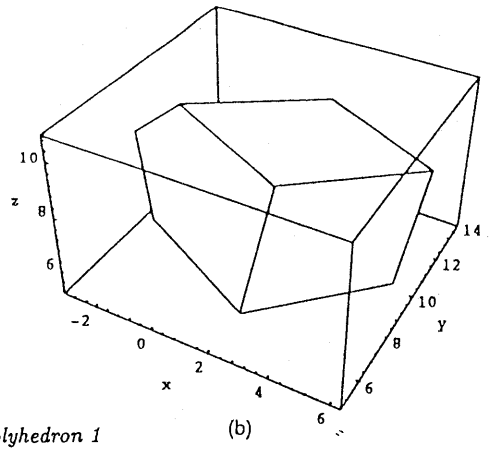
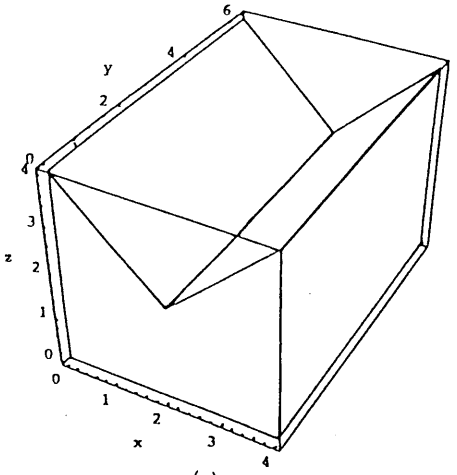


Figure 3: *Polyhedron 1*



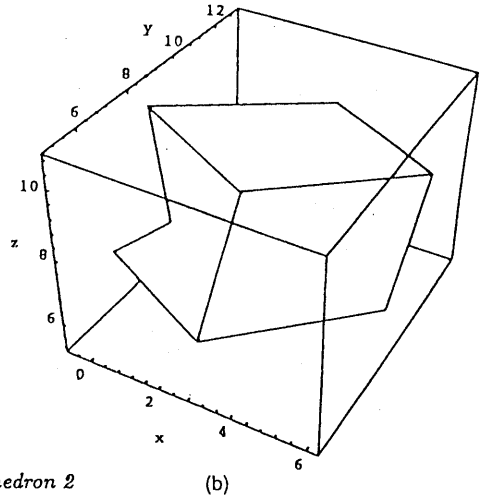
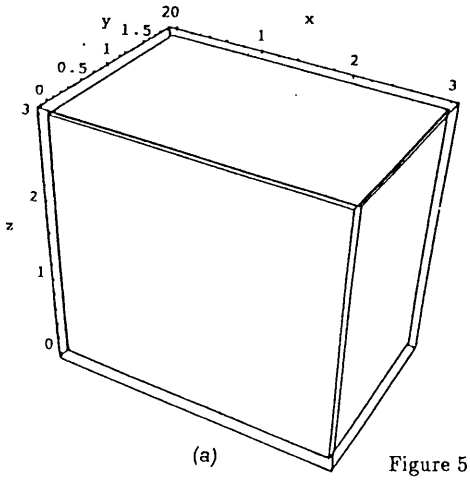


Figure 4: *Polyhedron 2*



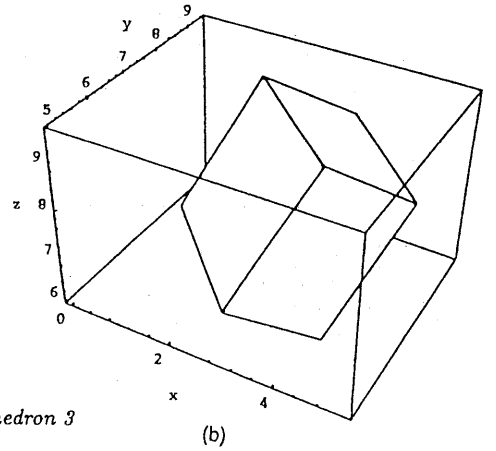


Figure 5: *Polyhedron 3*