Invited Paper

Model-based Determination of Object Position and Orientation without Matching

KEN-ICHI KANATANI* and KAZUYUKI YAMADA*

This paper presents an algorithm for computing, from a single image, the 3D position and orientation of an object whose 3D shape is known. This algorithm does not require "matching" of the object with its image. The 3D position and orientation of the object is determined by computing the hypothetical "3D motion" of the object from a known position to the position of the observed image. The motion is iteratively estimated and updated from the values of "observables" globally defined over the model image and the observed image. Various technical issues are discussed, and examples based on real images are shown. The technique can be applied to various manufacturing processes by industrial robots where the positioning control of machine parts is not very precise.

1. Introduction

We present a solution to the problem of determining, by vision, the location of an object, say a machine part in a robotic manufacturing situation, whose 3D shape is known and stored in a database. It is important to solve this problem in many flexible manufacturing circumstances where low precision, general purpose fixtures and positioners are employed. In a typical application, the uncertainty in the location of the object is small relative to its dimensions, but sometimes errors that are more than an order of magnitude greater than the allowed tolerances may occur. In some circumstances, the machine parts are not placed precisely in the same location every time, but randomly placed within a certain area. In order to relax the need to constrain the environment, the robot must autonomously adapt to its environment.

The problem of object localization is usually solved by first *matching* a database description of the object with the observed image. Then, the robot computes the position and orientation of the object in analytical terms [14]. Alternatively, the robot iteratively generates images of the object model, each time updating its position and orientation so that the discrepancy, say the sum of the squares of the distances between specified feature points and their corresponding points on the image, always decreases. The motion from the initial position to the observed position is established if the generated model image sufficiently coincides with the observed image [12].

However, it is not easy to match the object model with its image. Many combinatorial possibilities must be exhausted, and some justification is necessary to ensure that the matching we choose is in fact correct. One approach is to find possible *partial matches* and compute the 3D motion solution for each partial match. Most of these candidates may be false matches, but they should contain multiple true matches. Hence, the correct match can be chosen by the *majority rule*. This strategy is known as the *generalized Hough transform* [14]. An alternative approach is to actually generate an object image on the assumption that the chosen matching is correct and see if the generated image precisely coincides with the observed image. In any case, whatever approach is used, the matching is confirmed only when the *motion* is correctly computed. It follows that a method to compute the motion *without matching* is very much desired.

In this paper, we present a method for computing motion which does not require matching. Instead of matching images, we measure *observables* of the generated model image and the observed image: The 3D motion is computed from the numerical values of the observables. This idea was first proposed by Amari [1] with regard to pattern recognition, and applications to motion detection of planar surfaces have extensively been studied by Kanatani [6–8], Amari and Maruyama [2], and others. In particular, Kanatani used, as observables, the statistics of surface textures [6], the characteristics of the contour shape [7], and the integrals along contour or over closed planar regions [8]. However, although the knowledge of *point-to-point correspondence* is not required, his method requires the knowledge of *face-to-face correspondence*, since observables is defined over some identified *planar surface*. Moreover, his method is based on the *optical flow equations*, which theoretically give *instantaneous velocities*

*Department of Computer Science, Gunma University

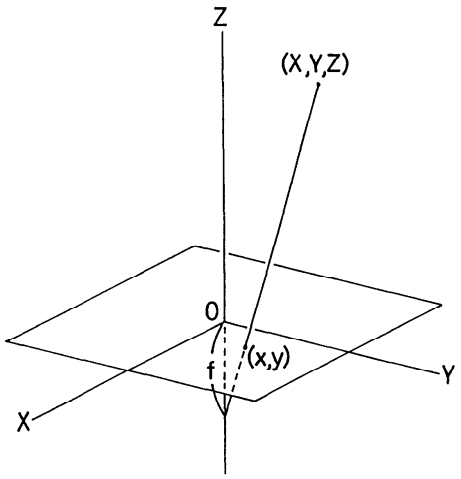

Fig. 1 Perspective projection of the scene onto the image plane $Z=0$ from the viewpoint $(0, 0, -f)$ on the negative side of the Z-axis.

on the image plane. Hence, the discrepancy between the generated model image and the observed image must be very small so that the displacements can be approximated by instantaneous velocities (assuming that the motion takes unit time).

Later, this method was modified to allow large discrepancies by introducing iterations [9], and a method which does not require identification of planar faces was suggested by Chou and Kanatani [4]. This paper extends their idea and gives a consistent algorithm of computing motion without requiring matching at all. In particular, the object need not have planar faces or straight edges; curved edges and curved surfaces can exist.

2. Image Flow Resulting from Object Motion

Take an XYZ-coordinate system in the scene. The XY-plane is identified with the image plane, and point $(0, 0, -f)$ is identified with the center of perspective projection, which we call the *viewpoint* (Fig. 1). A point (X, Y, Z) is projected onto the intersection (x, y) of the image plane with the ray starting from the viewpoint and passing through the point (X, Y, Z). From Fig. 1, we see that

$$x=\frac{fX}{f+Z}, \quad y=\frac{fY}{f+Z}. \tag{2.1}$$

If the 3D shape of an object is known, it can be mapped, by computation, in an arbitrary position in the scene. Suppose the object has some distinctive points (e.g., corner vertices). We call such points *feature points*. As is well known, a 3D rigid motion is specified by a translation (A, B, C) at a *reference point* and a rotation $\boldsymbol{R}=(r_{ij})$ (orthogonal matrix) around that point. Let (X, Y, Z) be a feature point, and (X_0, Y_0, Z_0) the reference point. If the object undergoes a 3D motion specified by translation (A, B, C) and rotation $\boldsymbol{R}$, the point (X, Y, Z) moves into position (X', Y', Z') given by

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix} + \begin{bmatrix} A \\ B \\ C \end{bmatrix} + \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} X-X_0 \\ Y-Y_0 \\ Z-Z_0 \end{bmatrix}. \tag{2.2}$$

Now, let us define the following quantity:

$$z=\frac{fZ}{f+Z}. \tag{2.3}$$

We call z the *reduced depth* (as opposed to the *depth* Z). Let us put

$$x_0\equiv\frac{fX_0}{f+Z_0}, \quad y_0\equiv\frac{fY_0}{f+Z_0}, \quad z_0\equiv\frac{fZ_0}{f+Z_0}. \tag{2.4}$$

Then, it is easy to show that the image coordinates (x, y) and the reduced depth z change their values, after the motion of eqn (2.2), in the form

$$\begin{aligned} x' &= f\frac{x+(1-z/f)A+(r_{11}-1)\hat{x}(x,z)+r_{12}\hat{y}(y,z)+r_{13}\hat{z}(z)}{f+(1-z/f)C+r_{31}\hat{x}(x,z)+r_{32}\hat{y}(y,z)+(r_{33}-1)\hat{z}(z)}, \\ y' &= f\frac{y+(1-z/f)B+r_{21}\hat{x}(x,z)+(r_{22}-1)\hat{y}(y,z)+r_{23}\hat{z}(z)}{f+(1-z/f)C+r_{31}\hat{x}(x,z)+r_{32}\hat{y}(y,z)+(r_{33}-1)\hat{z}(z)}, \\ z' &= f\frac{z+(1-z/f)C+r_{31}\hat{x}(x,z)+r_{32}\hat{y}(y,z)+(r_{33}-1)\hat{z}(z)}{f+(1-z/f)C+r_{31}\hat{x}(x,z)+r_{32}\hat{y}(y,z)+(r_{33}-1)\hat{z}(z)}, \end{aligned} \tag{2.5}$$

where we defined

$$\hat{x}(x,z)\equiv x-\frac{f-z}{f-z_0}x_0, \quad \hat{y}(y,z)\equiv y-\frac{f-z}{f-z_0}y_0, \quad \hat{z}(z)\equiv z-\frac{f-z}{f-z_0}z_0. \tag{2.6}$$

A 3D rotation is specified by its axis (n_1, n_2, n_3), which is taken to be a unit vector, and the angle of rotation Ω around it screw-wise. As is well known, the corresponding rotation matrix is given by

$$R=\begin{bmatrix} \cos\Omega+(1-\cos\Omega)n_1^2 & (1-\cos\Omega)n_2n_1+\sin\Omega n_3 & (1-\cos\Omega)n_3n_1-\sin\Omega n_2 \\ (1-\cos\Omega)n_1n_2-\sin\Omega n_3 & \cos\Omega+(1-\cos\Omega)n_2^2 & (1-\cos\Omega)n_3n_2+\sin\Omega n_1 \\ (1-\cos\Omega)n_1n_3+\sin\Omega n_2 & (1-\cos\Omega)n_2n_3-\sin\Omega n_1 & \cos\Omega+(1-\cos\Omega)n_3^2 \end{bmatrix}. \tag{2.7}$$

If a rotation is infinitesimal, i.e., if the angle of rotation Ω is infinitesimally small, the rotation matrix takes the form $\boldsymbol{R}=\boldsymbol{I}+\Delta\boldsymbol{R}$ where

$$\Delta R=\begin{bmatrix} 0 & \Omega_3 & -\Omega_2 \\ -\Omega_3 & 0 & \Omega_1 \\ \Omega_2 & -\Omega_1 & 0 \end{bmatrix}+O(\Omega^2). \tag{2.8}$$

Here, we put $\Omega_i=\Omega n_i$, $i=1, 2, 3$, and $O(\Omega^2)$ denotes terms of orders equal to or higher than the power 2 of Ω.

If the motion is smooth, the instantaneous translation velocity (a, b, c) and the rotation velocity $(\omega_1, \omega_2, \omega_3)$ are obtained by taking the limit $\Delta t\to 0$ of $(A/\Delta t, B/\Delta t, C/\Delta t)$ and $(\Omega_1/\Delta t, \Omega_2/\Delta t, \Omega_3/\Delta t)$, respectively, where Δt is the time of the duration of the motion. This instantaneous rotation is also interpreted as a rotation around axis $(\omega_1, \omega_2, \omega_3)$ by *angular velocity* $\sqrt{\omega_1^2+\omega_2^2+\omega_3^2}$ (rad/sec) screw-wise around it.

Taking the limit $\Delta t\to 0$ of $((x'-x)/\Delta t, (y'-y)/\Delta t)$, we obtain, from eqns (2.5), the velocity $(\dot{x}, \dot{y})$ of point (x, y) on the image plane in the form

$$\begin{aligned} \dot{x}&=u_1(x, y, z)a+u_2(x, y, z)b+u_3(x, y, z)c \\ &\quad +u_4(x, y, z)\omega_1+u_5(x, y, z)\omega_2+u_6(x, y, z)\omega_3, \\ \dot{y}&=v_1(x, y, z)a+v_2(x, y, z)b+v_3(x, y, z)c \\ &\quad +v_4(x, y, z)\omega_1+v_5(x, y, z)\omega_2+v_6(x, y, z)\omega_3, \end{aligned} \tag{2.9}$$

where we put

$$\begin{aligned} &u_1(x, y, z)\equiv 1-\frac{z}{f}, \quad u_2(x, y, z)\equiv 0, \\ &u_3(x, y, z)\equiv -\frac{x}{f}\left(1-\frac{z}{f}\right), \\ &u_4(x, y, z)\equiv -\frac{x}{f}\hat{y}(y, z), \\ &u_5(x, y, z)\equiv \hat{z}(z)+\frac{x}{f}\hat{x}(x, z), \ u_6(x, y, z)\equiv -\hat{y}(y, z), \\ &v_1(x, y, z)\equiv 0, \quad v_2(x, y, z)\equiv 1-\frac{z}{f}, \\ &v_3(x, y, z)\equiv -\frac{y}{f}\left(1-\frac{z}{f}\right), \\ &v_4(x, y, z)\equiv -\hat{z}(z)-\frac{y}{f}\hat{y}(y, z), \\ &v_5(x, y, z)\equiv \frac{y}{f}\hat{x}(x, z), \ v_6(x, y, z)\equiv \hat{x}(x, z). \end{aligned} \tag{2.10}$$

3. Computing Motion from Observables

If the object model is placed in the scene by computation, its image can be generated by the projection equations (2.1). Let (x_i, y_i), $i=1, \ldots, n$, be the image coordinates of n feature points of the model image. We define the following *observables* of the feature points:

$$J=\sum_{i=1}^{n} w(x_i, y_i). \tag{3.1}$$

Here, $w(x, y)$ is an arbitrarily given weight function.

If a velocity field $(\dot{x}, \dot{y})$ exists on the image plane, observable J changes its value in the form

$$\frac{dJ}{dt}=\sum_{i=1}^{n}\left[\frac{\partial w}{\partial x}(x_i, y_i)\dot{x}_i+\frac{\partial w}{\partial y}(x_i, y_i)\dot{y}_i\right], \tag{3.2}$$

where $\dot{x}_i\equiv\dot{x}(x_i, y_i, z_i)$ and $\dot{y}_i\equiv\dot{y}(x_i, y_i, z_i)$ are given by eqns (2.9). Substitution of eqns (2.9) yields

$$\frac{dJ}{dt}=C_1a+C_2b+C_3c+C_4\omega_1+C_5\omega_2+C_6\omega_3, \tag{3.3}$$

where

$$\begin{aligned} C_k=\sum_{i=1}^{n}\Bigg[&\frac{\partial w}{\partial x}(x_i, y_i)u_k(x_i, y_i, z_i) \\ &+\frac{\partial w}{\partial y}(x_i, y_i)v_k(x_i, y_i, z_i)\Bigg] \end{aligned} \tag{3.4}$$

for $k=1, \ldots, 6$. The image coordinates (x_i, y_i), $i=1, \ldots, n$, of the feature points are known quantities. Their reduced depths z_i, $i=1, \ldots, n$, are also known for the model image. Hence, all C_k, $k=1, \ldots, 6$, are known quantities.

Now, consider the object image taken by the camera. Suppose it is very close to the generated image. Then, we can regard it as resulting from an infinitesimal "motion" of the object model. Hence, the time derivative dJ/dt can be approximated by the finite difference of the values of observable J for the two images. (We take the time laps between the two images to be unit time.) Thus, we can compute all quantities in eqn (3.3) except $a, b, c, \omega_1, \omega_2, \omega_3$. This means that eqn (3.3) imposes a *linear constraint* on the six velocity components $a, b, c, \omega_1, \omega_2, \omega_3$. Hence, if six or more independent weight functions $w(x, y)$ are used, a set of simultaneous linear equations are obtained, and the six unknowns $a, b, c, \omega_1, \omega_2, \omega_3$ are determined. The determination does not require any knowledge of which feature point corresponds to which feature point between the two images.

Next, suppose the observed image is not close to the

model image. This means that the "motion" is not infinitesimal. Then, we can apply iterations, iteratively moving the model image according to the estimated motion so that the model image approaches closer and closer to the observed image. This is numerically done as follows. First, we apply the method for infinitesimal motion and estimate the velocities a, b, c, ω_1, ω_2, ω_3. From these, the translation and the rotation are estimated to be $A=a\Delta t$, $B=b\Delta t$, $C=c\Delta t$, $\Omega=\sqrt{\omega_1^2+\omega_2^2+\omega_3^2}\Delta t$, $n_i=\omega_i\Delta t/\Omega$, $i=1, 2, 3$. The corresponding matrix is constructed by eqn (2.7). Then, a new model image is generated, and the image coordinates (x_i, y_i) and the reduced depths z_i for $i=0, 1, \ldots, n$ are updated by eqns (2.5).

If the newly generated model image sufficiently coincides with the observed image, our estimation is correct. (The coincidence of the two images is measured by the coincidence of the values of their observables.) Otherwise, the same process is repeated: From the values of the observables of the newly generated model image and the observed image, we can again estimate the translation (A', B', C') and the rotation $\boldsymbol{R}'$. An improved estimate is obtained by the combined motion specified by translation $(A+A', B+B', C+C')$ and rotation $\boldsymbol{R}'\boldsymbol{R}$. This process is repeated until no further improvement takes place.

4. Observables Based on Edge Images

Suppose there are m edges L_i, $i=1, \ldots, m$, in the model image. They need not be line segments; they can be arbitrarily shaped curves. Since the 3D positions of the corresponding edges of the object model are known, the image coordinates (x, y) and the reduced depth z are known for every point on edges L_i, $i=1, \ldots, m$. Consider an observable defined by the sum of curvilinear integrals along the edges

$$J=\sum_{i=1}^{m}\int_{L_i} w(x, y)\, ds, \tag{4.1}$$

where $w(x, y)$ is an arbitrarily given weight function.

Suppose a velocity field $(\dot{x}, \dot{y})$ exists on the image plane. If each edge is parameterized by the arc length s, it is easy to show that observable J changes its value in the form

$$\frac{dJ}{dt}=\sum_{i=1}^{m}\int_{L_i}\left[\frac{\partial w}{\partial x}\dot{x}(s)+\frac{\partial w}{\partial y}\dot{y}(s)+w(x, y)\times\left(\frac{d\dot{x}}{ds}l_1(s)+\frac{d\dot{y}}{ds}l_2(s)\right)\right]ds, \tag{4.2}$$

where $(l_1(s), l_2(s))$ is the unit tangent to the edge at $(x(s), y(s))$ in the direction of increasing arc length s. Substitution of eqn (2.9) yields

$$\frac{dJ}{dt}=C_1a+C_2b+C_3c+C_4\omega_1+C_5\omega_2+C_6\omega_3, \tag{4.3}$$

where

$$C_k=\sum_{i=1}^{m}\int_{L_i}\left[\frac{\partial w}{\partial x}(x, y)u_k(x, y, z)+\frac{\partial w}{\partial y}(x, y)v_k(x, y, z)+w(x, y)\left(\frac{du_k}{ds}(s)l_1(s)+\frac{dv_k}{ds}(s)l_2(s)\right)\right]ds, \tag{4.4}$$

for $k=1, \ldots, 6$.

Since the 3D shape, position and orientation of the object model are known, all C_k, $k=1, \ldots, 6$, can be computed immediately. Thus, if observables of the form of eqn (4.1) are computed for six or more independent weight functions $w(x, y)$ over *all* the edges, the 3D motion of the object is determined by exactly the same procedure as stated in the preceding section without doing any matching at all.

To be specific, consider an edge L, Take on it a sequence of dividing points (of not necessarily equal intervals) (x_K, y_K), $K=0, 1, \ldots, N$, such that (x_0, y_0) and (x_N, y_N) are the end points of L. Put $w_K\equiv w(x_K, y_K)$, $K=0, 1, \ldots, N$. Then, we can use approximation

$$\int_L w(x, y)\, ds\approx\frac{1}{2}\sum_{K=0}^{N-1}(w_K+w_{K+1})\Delta s_K \tag{4.5}$$

for the computation of the observable of eqn (4.1), where $\Delta s_K\equiv\sqrt{(x_{K+1}-x_K)^2+(y_{K+1}-y_K)^2}$, $K=0, 1, \ldots, N-1$. Note that the dividing points are taken along each edge in both the model image and the observed image, but *the dividing points need not correspond between the two images.* They can be chosen *arbitrarily* and *independently*, since eqn (4.5) is merely a discrete approximation of integration.

On the other hand, eqn (4.4) need be computed only for the model image. Consider an edge L. Take on it a sequence of points (of not necessary equal intervals) (X_K, Y_K, Z_K), $K=0, 1, \ldots, N$, such that (X_0, Y_0, Z_0) and (X_N, Y_N, Z_N) are the end points. Let (x_K, y_K) and z_K be the image coordinates and the reduced depths, respectively, of these points computed by the projection equations (2.1) and eqn (2.3). Then, each term in eqn (4.4) is approximated by finite summation. For example, if we put $u_{k,K}\equiv u_k(x_K, y_K, z_K)$, $w_K\equiv w(x_K, y_K)$ and $\partial w/\partial x|_K\equiv\partial w/\partial x(x_K, y_K)$ for $K=0, 1, \ldots, N$, we can use approximations

$$\int_L\frac{\partial w}{\partial x}(x, y)u_k(x, y, z)ds\approx\frac{1}{2}\sum_{K=0}^{N-1}\left(\frac{\partial w}{\partial x}\Big|_K u_{k,K}+\frac{\partial w}{\partial x}\Big|_{K+1}u_{k,K+1}\right)\Delta s_K, \tag{4.6}$$

$$\int_L w(x, y)\frac{du_k}{ds}(s)l_1(s)\, ds\approx\frac{1}{2}\sum_{K=0}^{N-1}(w_{K+1}+w_K)(u_{k,K+1}-u_{k,K})\frac{\Delta x_K}{\Delta s_K}, \tag{4.7}$$

where $\Delta x_K\equiv x_{K+1}-x_K$ and $\Delta s_K\equiv\sqrt{(x_{K+1}-x_K)^2+(y_{K+1}-y_K)^2}$ for $K=0, 1, \ldots, N-1$. Other terms are similarly computed. If a motion is estimated, the image coordinates (x_K, y_K) and the reduced depths z_K for $K=0, 1, \ldots, N$ are updated by eqn (2.5).

5. Discussions and Examples

Our algorithm is summarized as follows. First, an image of the object model placed in a known position is generated by using its 3D shape data stored in a database. If this generated image coincides with the observed image, the object is correctly placed. Otherwise, the real position and orientation of the object is determined by computing the *motion* that brings the object from that position to the position where it appears in the image.

The observables to be computed can be defined with respect to either feature points or object edges. (If the object is a polyhedron, the corner vertices can be used as typical feature points.) However, although we need not know which feature point or edge corresponds to which feature point or edge, the model image and the observed image must have the same set of feature points or edges. In view of this, the use of the object edges for defining observables has the advantage that the computation is robust even if the detected edges are not complete: The curvilinear integration is not so much affected as long as the total length of the missing or false edges is very small.

A critical issue about our method is the possibility of *occlusion*. If the manufacturing environment is properly controlled, the discrepancy of the object position from the supposed position is expected to be very small. However, if the discrepancy of the object orientation is large, some part of the model image is occluded in the observed image, while some part hidden in the model image appears in the observed image. As a result, the set of feature points or edges do not correspond as a whole between the model image and the observed image.

One solution is to prepare, in the database, "typical views" of the object, and search for an appropriate one by means of some *criterion of similarity*. This strategy was used in another context by Ikeuchi [5] (also see [3, 10, 11] for object image representation). If no criterion is available as to which to choose, we can try, as a last resort, all the candidates one by one to see if the iteration converges.

In our iteration procedure, the degree of convergence is measured by the differences of the values of "observables" between the two images. In fact, our motion estimation is performed by "matching" observables instead of the images themselves. However, convergence of observables does not necessarily mean convergence of the images. It is desirable to introduce a *measure of convergence* which is capable of assuring convergence of the images, yet such that its computation does not require the matching. A simple one that satisfies this requirement is the "area of the intersection of the *convex hulls*". Given a set of feature points (or edges), we first compute their convex hull. An efficient algorithm is available, requiring only $O(n \log n)$ steps of computation, where n is the number of feature points [13]. After computing the convex hulls for both the model image and the observed image (separately and independently), we measure the area of their intersection. Again, there exists an efficient algorithm for computing the intersection of two convex polygons, requiring only $O(n+n')$ steps of computation, where n and n' are the numbers of vertices of the respective polygons [13].

Since in general the iterations converge more rapidly as the generated model image is closer to the observed image, the following preprocessing is effective: Before starting the iterations, the generated object model is translated in the scene in such a way that the *centroid* of its projection image coincides with that of the observed image and also the approximate *size* of the model image becomes equal to that of the observed image (cf. Appendix).

Fig. 2 shows an object model. Fig. 3 is a real image of this object placed in an unknown position. In our experiment, we skipped the image processing stage and picked out corner vertices and edges by hand. Fig. 4 shows the iterations to make the model image coincide with the observed image. The first step is the preliminary translation. Fig. 5 is another real image of the same object placed in a different position. Fig. 6 shows the iterations. Figs. 7 and 8 show other examples for the same images of Figs. 3 and 5 but from different starting positions. In all these examples, we used ten observables and solved the ten resulting equations by the least square method (cf. Appendix).

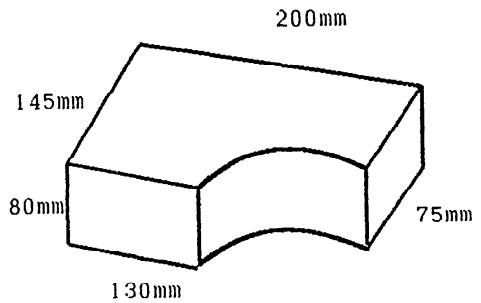

Fig. 2 The 3D shape of the object used in our experiment. The unit is millimeter.

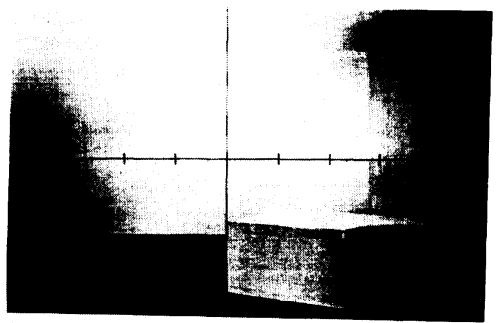

Fig. 3 A real image of thd object of Fig. 2 placed in an unknown position. The focal length is $f=29.25$ (mm).

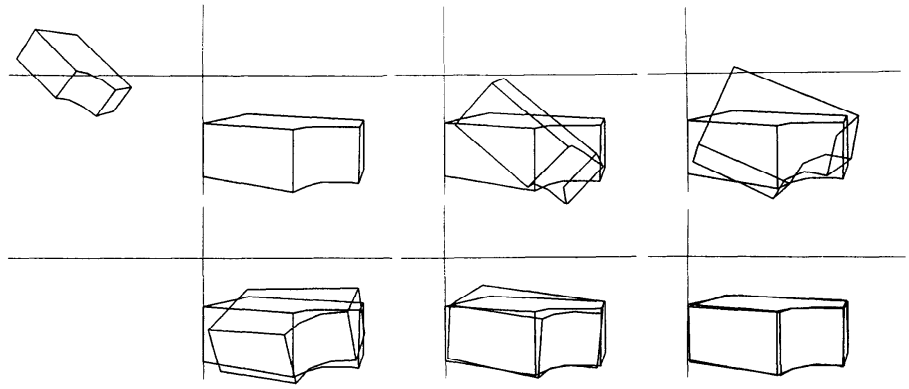

Fig. 4 The iterations for moving the object model of Fig. 2 so that its projection image coincides with the object image of Fig. 3. Observables based on corner vertices are used. The initial arrangement, the preliminary step, and the first through the fourth iteration steps are shown.

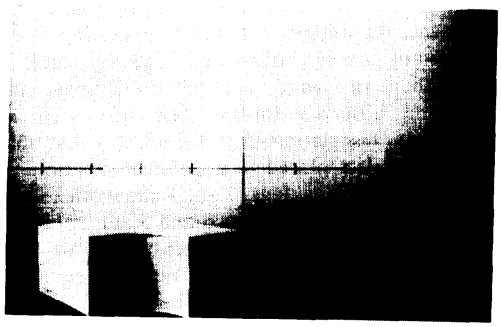

Fig. 5 Another real image of the object of Fig. 2 placed in an unknown position. The focal length is $f=29.25$ (mm).

Our experiment shows that the convergence is smoother for observables based on edges than for observables based on corner vertices. The iterations of Figs. 4 and 6 are tracing motions which have six full degrees of freedom—three translation components and three rotation components. In the iterations of Figs. 7 and 8, the starting positions are chosen by assuming that the object is placed on a "horizontal table" whose height is approximately known. This means that the "motion" has approximately three degrees of freedom—translations along the table surface and rotations around an axis perpendicular to it. Although this knowledge is never used in the iterations (hence all six nonzero motion components are actually computed), the convergence is very smooth. Since displacements of objects occurs on a horizontal table in many industrial manufacturing circumstances, the use of this knowledge can facilitate the convergence.

6. Concluding Remarks

We have presented an algorithm to compute, from a single image, the 3D position and orientation of an object whose 3D shape is known. Our algorithm does not require matching of points of the object with points of the observed image. The 3D position and orientation of the object is determined by computing the hypothetical 3D *motion* of the object from a known position to the position of the observed image: The motion of the object model is iteratively estimated and updated from the

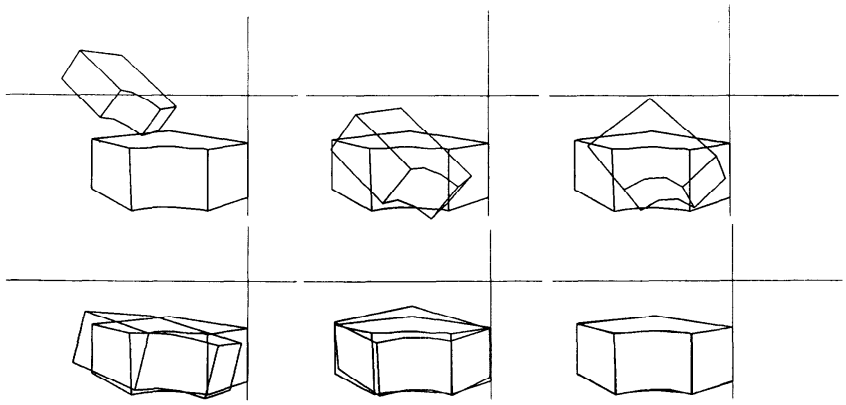

Fig. 6 The iterations for moving the object model of Fig. 2 so that its projection image coincides with the object image of Fig. 5. Observables based on edges are used. The initial arrangement, the preliminary step, and the first through the fourth iteration steps are shown.

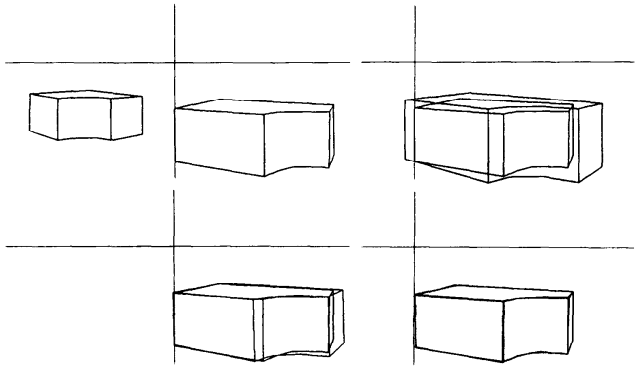

Fig. 7 The iterations for moving the object model of Fig. 2 so that its projection image coincides with the object image of Fig. 3. Observables based on corner vertices are used. The initial arrangement, the preliminary step, the first and the second iteration steps are shown.

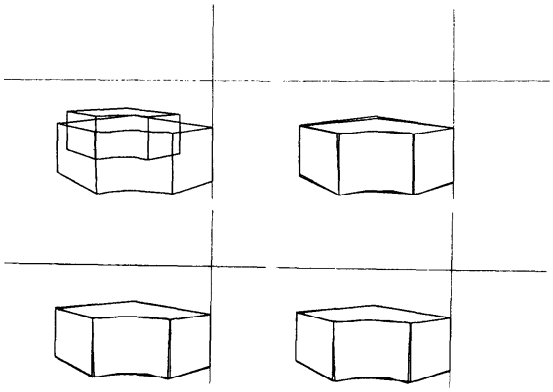

Fig. 8 The iterations for moving the object model of Fig. 2 so that its projection image coincides with the object image of Fig. 5. Observables based on edges are used. The initial arrangement, the preliminary step, the first and the second iteration steps are shown.

values of *observables* globally defined over the model image and the observed image. The matching of the object with the observed image is established *as a result* of this motion detection. We have also discussed various technical issues and shown some examples based on real images. Our technique can be applied to various manufacturing processes by industrial robots where the positioning control of machine parts is not very precise.

Acknowledgments

This research originates from the study of the first author (Kanatani) while he was visiting the University of Maryland in 1985–1986. He thanks Azriel Rosen-feld and Larry Davis for helpful discussions and Tsai-Chia Chou for numerical experiments. The authors also thank Kokichi Sugihara and Shun-ichi Amari of the University of Tokyo for helpful communications, and Seiji Uchiyama of Daikin Industries, Ltd. for his preliminary experiments. Part of this work was supported by Casio Science Promotion Foundation, Yazaki Memorial Foundation for Science and Technology, Inamori Foundation, and the Ministry of Education, Science, and Culture under Grant in Aid for Scientific Research (No. 63550268).

References

1. AMARI, S. Invariant structures of signal and feature spaces in pattern recognition problems, *RAAG Memoirs*, **4** (1968), 553–566.
2. AMARI, S. and MARUYAMA, M. A theory on the determination of 3D motion and 3D structure from features, *Spatial Vision*, **2** (1987), 151–168.
3. BESL, P. J. and JAIN, R. C. Three-dimensional object recognition, *Comput. Surveys*, **17** (1985), 75–145.
4. CHOU, T.-C. and KANATANI, K. Recovering 3D rigid motions without correspondence, *Proc. IEEE 1st Int. Conf. Computer Vision*, London, June (1987), 534–538.
5. IKEUCHI, K. Generating an interpretation tree from a CAD model for 3D-object recognition in bin-picking tasks, *Int. J. Comput. Vision*, **1** (1987), 145–165.
6. KANATANI, K. Detection of surface orientation and motion from texture by a stereological technique, *Artif. Intell.*, **23** (1984), 213–237.
7. KANATANI, K. Tracing planar surface motion from a projection without knowing the correspondence, *Comput. Vision Graphics Image Process.*, **29** (1985), 1–12.
8. KANATANI, K. Detecting the motion of a planar surface by line and surface integrals, *Comput. Vision Graphics Image Process.*, **29** (1985), 13–22.
9. KANATANI, K. and CHOU, T.-C. Tracing finite motions without correspondence, *Proc. IEEE Int. Workshop on Industrial Appl. Machine Vision and Machine Intell., Tokyo*, February (1987), 118–123.
10. KOENDERINK, J. J. and VAN DOORN, A. J. The singularities of the visual mapping, *Biol. Cybern.*, **24** (1976), 51–59.
11. KOENDERINK, J. J. and VAN DOORN, A. J. Internal representation of solid shape with respect to vision, *Biol. Cybern.*, **32** (1979), 211–216.
12. GUNNARSSON, K. T. and PRINZ, F. B. CAD model-based localization of parts in manufacturing, *Computer*, (August 1987), 66–74.
13. PREPARATA, F. P. and SHAMOS, M. I. *Computational Geometry*, Springer, New York, 1985.
14. SILBERBERG, T. M., HARWOOD, D. and DAVIS, L. Three dimensional object recognition using oriented model points, *Techniques for 3-D Machine Perception* (ed. A. Rosenfeld), North-Holland, Amsterdam, (1986), 271–320.

Appendix. Preliminary Translation and Weight Functions

Let $(\bar{x}, \bar{y})$ be the *centroid* of the corner vertices (i.e., their average) of the generated model image, and let

$(\bar{x}', \bar{y}')$ be that of the observed image. We define the *size* of the object image by the maximum of the distances of the corner vertices from their centroid. Let l and l' be the sizes of the generated model image and the observed image. Let $(\bar{X}, \bar{Y}, \bar{Z})$ be the average of the 3D coordinates of the corner vertices (i.e., their centroid) of the object model. We define the *apparent object size* L by

$$L=\left(1+\frac{\bar{Z}}{f}\right) l. \tag{A.1}$$

This is the size of the object regarded as a "flat plane" parallel to the image plane and passing through the centroid $(\bar{X}, \bar{Y}, \bar{Z})$ (Fig. A). From Fig. A, we see that the size of the model image becomes equal to that of the observed image if the centroid $(\bar{X}, \bar{Y}, \bar{Z})$ is displaced into the position $(\bar{X}', \bar{Y}', \bar{Z}')$ given by

$$\bar{Z}'=\left(\frac{L}{l'}-1\right) f, \quad \bar{X}'=\left(\frac{\bar{Z}'}{f}+1\right) \bar{x}',$$
$$\bar{Y}'=\left(\frac{\bar{Z}'}{f}+1\right) \bar{y}'. \tag{A.2}$$

Hence, as a preliminary step, the object model is translated in the scene by

$$A=\bar{X}'-\bar{X}, \quad B=\bar{Y}'-\bar{Y}, \quad C=\bar{Z}'-\bar{Z}. \tag{A.3}$$

Then, the iteration procedure is started as described in Sections 3 and 4. As the weight functions for the observables (for both corner vertices and edges), we used

$1/l, \quad (x-\bar{x})/l, \quad (y-\bar{y})/l,$

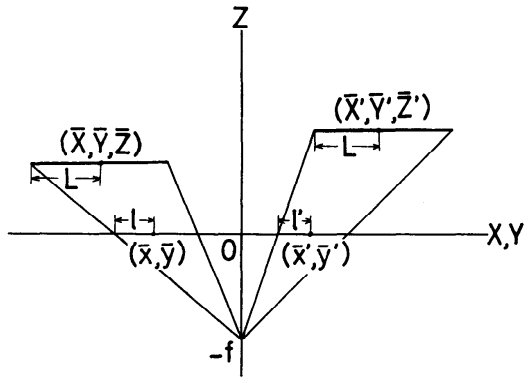

Fig. A The preliminary translation of the model object so that the centroid $(\bar{x}, \bar{y})$ of the generated model image coincides with the centroid $(\bar{x}', \bar{y}')$ of the observed image and the approximate sizes of the two images become the same.

$$(x-\bar{x})^2/l^2, \quad (x-\bar{x})(y-\bar{y})/l^2, \quad (y-\bar{y})^2/l^2,$$
$$(x-\bar{x})^3/l^3, \quad (x-\bar{x})^2(y-\bar{y})/l^3, \quad (x-\bar{x})(y-\bar{y})^2/l^3,$$
$$(y-\bar{y})^3/l^3. \tag{A.4}$$

Since these ten weight functions define ten observables for six unknowns, we used the least square method to solve the resulting ten simultaneous linear equations. These weight functions are revised at each iteration step, since the centroid $(\bar{x}, \bar{y})$ and the size l is altered each time a new model image is generated.

(Received June 1, 1988)