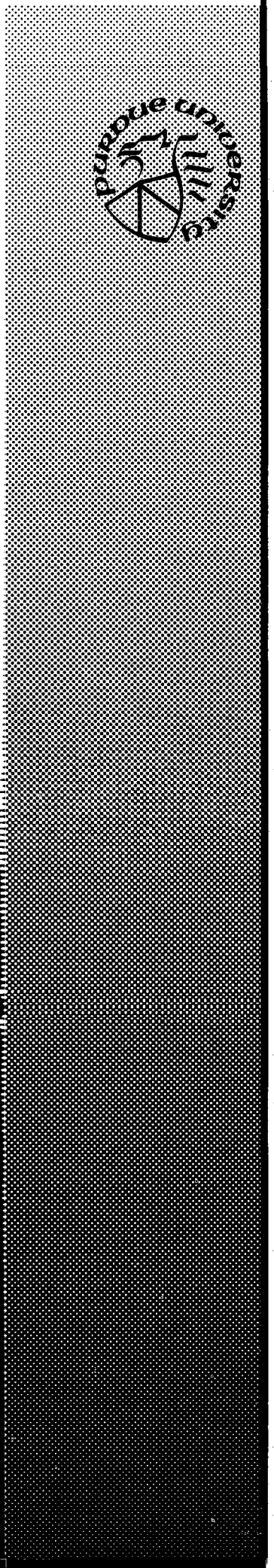
4-1-1987

Mathematics for Visual Guidance of Robots

YiuCheung Shiu
Purdue University

Shaheen Ahmad
Purdue University

Richard P. Paul
Purdue University



Mathematics for Visual Guidance of Robots

YiuCheung Shiu
Shaheen Ahmad
Richard P. Paul

TR-EE 87-14
April 1987

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

Mathematics for Visual Guidance of Robots

YiuCheung Shiu
Shaheen Ahmad
Richard P. Paul

School of Electrical Engineering
Purdue University
West Lafayette, IN 47907

ABSTRACT

Vision can be used to position a robot relative to a known object or a known environment in 3D. If the object has enough feature points, one view is sufficient for determining the relative position between the object and the camera, otherwise, multiple views are required. We discuss the mathematics of viewpoint determination, using a combination of calibration matrix decomposition and space resection. The combined method has low noise sensitivity and does not require knowledge of camera parameters. If the object does not have enough features, multiple views are required to determine its position and orientation; an example of this is be given. The formulation of homogeneous transform equations to drive the manipulator to the goal position is also be given.

1. Introduction

It is possible to find the position of an object using one view by identifying and finding the 2D locations of unobstructed visual features (such as points and lines). This set of image coordinates will be fed to a viewpoint determination algorithm to find the 3D position and orientation of the object. We will address only the viewpoint determination problem; the 2D locations of visual features are assumed to be provided by a computer vision system. In this report, we will only address point features.

Viewpoints can be determined by the combination of calibration matrix decomposition [Gana83] and space resection [Wolf83]; the results of calibration matrix decomposition are used as an initial estimate for the iterative space resection method. The first method alone is noise sensitive (a technical report will be published to compare noise sensitivity of various viewpoint determination methods). The second method requires an initial estimate of the viewpoint because it uses the Newton-Raphson algorithm [Dode78] to solve for a non-linear system of equations Another disadvantage of the second method is that the solutions may not be unique [Fisc81]. These problems will be solved when the two methods are used together.

In some cases, there may not be enough feature points for one-view methods to work; we will show an example that does not require as many features but requires multiple views. Finally, we will show how the relative position between an object and the camera can be incorporated into a transform equation necessary for the manipulation of the object.

2. Finding Object Position with a Single Image

We want to find the 3-D position and orientation of an object using model-based vision. It is possible to find the 3-D position using only one view because of the knowledge of object dimensions. We will limit ourselves to methods that use image coordinates of local features such as corners and small holes instead of methods that uses extended features such as lines and curves. Let $(u_i, v_i)^t$ be image coordinates of feature number i and $(x_i, y_i, z_i)^t$ be the coordinates of the feature with respect to the object frame *OBJ*, Ω be the set of 2D positions of image features and Γ be the set of corresponding 3D positions of the object features relative to the object. If there are n corresponding points,

$$\Omega = \left\{ (u_1, v_1) , (u_2, v_2) , \cdots (u_n, v_n) \right\} \tag{2.1}$$

$$\Gamma = \left\{ (x_1, y_1, z_1) , (x_2, y_2, z_2) , \cdots (x_n, y_n, z_n) \right\} \tag{2.2}$$

We require a function **F** which takes the arguments Ω, Γ, n , and produce the homogeneous transform ${}^{CAM}\mathbf{T}_{OBJ}$ which specifies the position of the *OBJ* frame with respect to the *CAM* frame.

$$\mathbf{F} (\Omega,\Gamma,n) . \tag{2.3}$$

We will discuss calibration matrix decomposition [Gana84] and space resection [Wolf83]. Calibration matrix decomposition is a non-iterative method that solves for both the transform ${}^{CAM}\mathbf{T}_{OBJ}$ and the camera parameters uniquely. This method requires at least 6 feature points to be seen on the image. Space resection is an iterative method used in photogrammetry to locate a camera-equipped airplane with respect to some known ground control points. It requires a minimum of three feature points, camera parameters and an initial estimate. The space resection solution may not be unique if less than 6 feature points are used [Fisc81].

We have found from simulation that calibration matrix decomposition is much more noise sensitive than space resection. The noise sensitivity and the non-uniqueness problems can be solved by using space resection to find the final result, while the calibration matrix decomposition is used to find the initial estimate and the camera parameters.

2.1. Camera Matrix Decomposition

In this section, we will find that the calibration matrix **C** can be solved when given a set of image coordinates and the corresponding set of 3-D feature coordinates with respect to the object frame *OBJ*. Then we can decompose **C** to find the position of the *OBJ* with respect to *CAM* and also the camera parameters.

Figure 2.1 shows the geometry of a camera viewing an object. ${}^{CAM}T_{OBJ}$ is the homogeneous transform that relates the frames *OBJ* and *CAM*. The camera calibration matrix **C** transforms 3-D coordinates with respect to the object frame into 2-D image coordinates. In homogeneous transform notation, this is written as

$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} \\ C_{21} & C_{22} & C_{23} & C_{24} \\ C_{31} & C_{32} & C_{33} & C_{34} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.4}$$

where. $(x,y,z)^t$ is the feature position measured with respect to *OBJ*, and $(u',v',w')^t$ is the feature position in homogeneous transform notation. Because of the nature of homogeneous transformation, Equation (2.4) still holds if **C** is scaled. In order to have a unique **C**, we will fix C_{34} to 1. The actual image position $(u,v)^t$ in pixel units can be calculated by

$$u = \frac{u'}{w'} \; ; \qquad v = \frac{v'}{w'}. \tag{2.5}$$

Given Ω (set of image coordinates) and Γ (set of 3-D coordinates with respect to *OBJ*), we want to find the elements of the calibration matrix **C**. From the position (u_i, v_i) of each image feature, we have from (2.5)

$$u'_i - u_i\, w'_i = 0; \qquad v'_i - v_i\, w'_i = 0. \tag{2.6}$$

But u'_i, v'_i, and w'_i can be expressed in terms of elements of **C** as:

$$u'_i = C_{11}\, x_i + C_{12}\, y_i + C_{13} \tag{2.7a}$$

$$v'_i = C_{21}\, x_i + C_{22}\, y_i + C_{23} \tag{2.7b}$$

$$w'_i = C_{31}\, x_i + C_{32}\, y_i + C_{33} \tag{2.7c}$$

Substitution (2.7) into (2.6) and using 6 points, we will arrive at the following matrix

and can solve for the elements of C.

$$\begin{vmatrix} x_1 & y_1 & z_1 & 1 & 0 & 0 & 0 & 0 & -u_1x_1-u_1y_1-u_1z_1 \\ x_2 & y_2 & z_2 & 1 & 0 & 0 & 0 & 0 & -u_2x_2-u_2y_2-u_2z_2 \\ x_3 & y_3 & z_3 & 1 & 0 & 0 & 0 & 0 & -u_3x_3-u_3y_3-u_3z_3 \\ x_4 & y_4 & z_4 & 1 & 0 & 0 & 0 & 0 & -u_4x_4-u_4y_4-u_4z_4 \\ x_5 & y_5 & z_5 & 1 & 0 & 0 & 0 & 0 & -u_5x_5-u_5y_5-u_5z_5 \\ x_6 & y_6 & z_6 & 1 & 0 & 0 & 0 & 0 & -u_6x_6-u_6y_6-u_6z_6 \\ 0 & 0 & 0 & 0 & x_1 & y_1 & z_1 & 1 & -v_1x_1-v_1y_1-v_1z_1 \\ 0 & 0 & 0 & 0 & x_2 & y_2 & z_2 & 1 & -v_2x_2-v_2y_2-v_2z_2 \\ 0 & 0 & 0 & 0 & x_3 & y_3 & z_3 & 1 & -v_3x_3-v_3y_3-v_3z_3 \\ 0 & 0 & 0 & 0 & x_4 & y_4 & z_4 & 1 & -v_4x_4-v_4y_4-v_4z_4 \\ 0 & 0 & 0 & 0 & x_5 & y_5 & z_5 & 1 & -v_5x_5-v_5y_5-v_5z_5 \end{vmatrix} \begin{vmatrix} C_{11} \\ C_{12} \\ C_{13} \\ C_{14} \\ C_{21} \\ C_{22} \\ C_{23} \\ C_{24} \\ C_{31} \\ C_{32} \\ C_{33} \end{vmatrix} = \begin{vmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{vmatrix} \quad (2.8)$$

Let us abbreviate the Equation (2.8) to

$$\mathbf{X}\,\mathbf{C}' = \mathbf{U}. \quad (2.9)$$

$\mathbf{C}'$ can be solve as

$$\mathbf{C}' = \mathbf{X}^{-1}\,\mathbf{U}. \quad (2.10)$$

If there are more than 6 points, we will have redundant equations. In this case, we can minimize the mean square error of the image coordinates by

$$\mathbf{C}' = (\mathbf{X}^{\mathrm{T}}\,\mathbf{X})^{-1}\,\mathbf{X}^{\mathrm{T}}\,\mathbf{U}. \quad (2.11)$$

Now we will express C in terms of the elements of the elements of ${}^{CAM}\mathbf{T}_{OBJ}$ and the camera parameters. To do this, we need to understand the geometry of the system. Shown in Figure 2.1 are the coordinates $(x_f,y_f,z_f)^t$ of a feature point with respect to *OBJ* , and the coordinates $(x'_f,y'_f,z'_f)^t$ of the same feature point with respect to the *CAM*. The ray that emanates from the focal point to the feature point intersects the image plane at $(x'_u, y'_u)^t$. The image coordinates read out from the vision system is not $(x'_u, y'_u)^t$ because they are expressed in pixels instead of millimeters and the origin may not be at the center of the image. Thus we can view $(u,v)^t$ as the results of scaling and translating $(x'_u,y'_u)^t$. Let S_u and S_v be the scaling factors in the u and v

directions, and u_0 and v_0 be the translations. We have

$$u = S_u x'_u + u_0 , \tag{2.12a}$$

$$v = S_v y'_u + v_0 . \tag{2.12b}$$

Using similar triangles,

$$x'_u = F \frac{x'_f}{z'_f}, \tag{2.13a}$$

$$y'_u = F \frac{y'_f}{z'_f}. \tag{2.13b}$$

where F is the focal length of the camera. Substituting (2.13) into (2.12) and rewritting in homogeneous notations,

$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \begin{bmatrix} S_u F & 0 & u_0 & 0 \\ 0 & S_v F & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x'_f \\ y'_f \\ z'_f \\ 1 \end{bmatrix} \tag{2.14}$$

$$\text{and} \qquad u = \frac{u'}{w'} , \qquad v = \frac{v'}{w'} .$$

We need to express the 3-D coordinates with respect to *OBJ* instead of *CAM*; this can be done with homogeneous transformations:

$$^{CAM}\begin{bmatrix} x'_f \\ y'_f \\ z'_f \\ 1 \end{bmatrix}^{CAM} = \left[\mathbf{T} \right]_{OBJ} {}^{OBJ}\begin{bmatrix} x_f \\ y_f \\ z_f \\ 1 \end{bmatrix} , \tag{2.15}$$

$$\text{where} \qquad {}^{CAM}\left[\ \mathbf{T}\ \right]_{OBJ} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Substituting (2.15) into (2.14) allows C to be expressed in terms of the components of ${}^{CAM}T_{OBJ}$ and the camera parameters (F, S_u, S_v,u_0, v_0). The C_{34} element is p_z so we need to divide all elements by p_z to have $C_{34} = 1$.

$$\begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} \\ C_{21} & C_{22} & C_{23} & C_{24} \\ C_{31} & C_{32} & C_{33} & C_{34} \end{bmatrix} \tag{2.16}$$

$$= \begin{bmatrix} \dfrac{n_x S_u F + n_z u_0}{p_z} & \dfrac{o_x S_u F + o_z u_0}{p_z} & \dfrac{a_x S_u F + a_z u_0}{p_z} & \dfrac{p_x S_u F + p_z u_0}{p_z} \\ \dfrac{n_y S_v F + n_z v_0}{p_z} & \dfrac{o_y S_v F + o_z v_0}{p_z} & \dfrac{a_y S_v F + a_z v_0}{p_z} & \dfrac{p_y S_v F + p_z v_0}{p_z} \\ \dfrac{n_z}{p_z} & \dfrac{o_z}{p_z} & \dfrac{a_z}{p_z} & 1 \end{bmatrix}$$

The elements C_{11}, C_{12}, C_{13} etc. are known because they can be solved by (2.8). We will equate both sides of (2.16) to solve for the components of ${}^{CAM}T_{OBJ}$ and the camera parameters. From (2.16), we can see that the parameters S_u, S_v, and F occurs in either S_uF or S_vF and therefore cannot be solved for individually. We will solve for k_u and k_v instead, where

$$k_u = S_u F \tag{2.17a}$$

$$k_v = S_v F \tag{2.17b}$$

From the last row of (2.16),

$$C_{31}^2 + C_{32}^2 + C_{33}^2 = \frac{n_z^2}{p_z^2} + \frac{o_z^2}{p_z^2} + \frac{a_z^2}{p_z^2} = \frac{1}{p_z^2}. \tag{2.18}$$

We will have two solutions for p_z from (2.18). However, if the z'-axis of the camera is pointing to the direction of sight and *OBJ*'s origin is within the object volume, p_z will be physically constrained to be positive. Thus,

$$p_z = \frac{1}{\sqrt{C_{31}^2 + C_{32}^2}} \tag{2.19}$$

Once p_z is found, n_z, o_z, and a_z can be found by equating the last row of equation (2.16).

$$n_z = p_z\, C_{31} \tag{2.20}$$

$$o_z = p_z\, C_{32} \tag{2.21}$$

$$a_z = p_z\, C_{33} \tag{2.23}$$

To solve for k_v, we use the first and second rows of C:

$$C_{11}\, C_{32} - C_{31}\, C_{12} = \frac{k_u}{p_z^2}(n_x\, o_z - o_x\, n_z) \tag{2.24a}$$

$$C_{11}\, C_{33} - C_{31}\, C_{13} = \frac{k_u}{p_z^2}(n_x\, a_z - n_z\, a_x) \tag{2.24b}$$

$$C_{12}\, C_{33} - C_{32}\, C_{13} = \frac{k_u}{p_z^2}(o_x\, a_z - o_z\, a_x) \tag{2.24c}$$

Taking sums of squares of (2.24a)-(2.24c), we obtain

$$(C_{11}C_{32}-C_{31}C_{12})^2+(C_{11}C_{33}-C_{31}C_{13})^2+(C_{12}C_{33}-C_{32}C_{13})^2 = \frac{k_u^2}{p_z^4}\ . \tag{2.25}$$

The sign of k_u is implementation dependent. In our system, we have assigned the coordinate frame to the camera such that u points to the same direction as x' and v points to the same direction as y'. In this case, both k_u and k_v are positive. Thus,

$$k_u = p_z^2\sqrt{(C_{11}C_{32}-C_{32}C_{12})^2+(C_{11}C_{33}-C_{31}C_{13})^2+(C_{12}C_{33}-C_{32}C_{13})^2} \tag{2.26}$$

To find u_0, we sum the squares of the elements of the first column.

$$C_{11}^2+C_{12}^2+C_{13}^2 = \frac{1}{p_z^2}[k_u^2(n_x^2+o_x^2+a_x^2) \tag{2.27}$$

$$+2k_u u_0(n_x n_z+o_x o_z+a_x a_z)+u_0^2(n_z^2+o_z^2+a_z^2)].$$

Simplifying (2.27) using properties of n, o, and **a**, and choosing the correct sign, we can solve for u_0. From Equation (2.12), and from the assumption that k_u and k_v are positive, we can interprete a positive u_0 to be the distance of the u—v origin left of the image center, and a positive v_0 to be the distance of the origin below the image center. In our vision system, the origin of the coordinate axes is at the lower left hand corner of the image plane so our u_0 and v_0 are positive.

$$u_0 = \sqrt{p_z^2(C_{11}^2+C_{12}^2+C_{13}^2)-k_u^2} \tag{2.28}$$

Finding k_v is similar to finding k_u except that we are working with the second and third rows of the C matrix. We will just state the results here.

$$k_v = p_z^2\sqrt{(C_{21}C_{32}-C_{31}C_{22})^2+(C_{21}C_{33}-C_{31}C_{23})^2+(C_{22}C_{33}-C_{32}C_{23})^2}. \tag{2.29}$$

To find v_0, it is similar to the method for finding u_0 except that we are working with the second row. We get

$$v_0 = \sqrt{p_z^2(C_{21}^2+C_{22}^2+C_{23}^2)-k_v^2} \tag{2.30}$$

Now that n_z, o_z, a_z, p_z, k_u, k_v, u_0, and v_0 are found, we can use equate the eight elements of both sides of (2.16) and solve for the eight remaining unknowns.

$$n_x = \frac{p_z}{k_u}(C_{11}-n_z u_0), \tag{2.31}$$

$$o_x = \frac{p_z}{k_u}(C_{12}-o_z u_0), \tag{2.32}$$

$$a_x = \frac{p_z}{k_u}(C_{13} - a_z u_0), \tag{2.33}$$

$$p_x = \frac{p_z}{k_u}(C_{14} - p_z u_0), \tag{2.34}$$

$$n_y = \frac{p_z}{k_v}(C_{21} - n_z v_0), \tag{2.35}$$

$$o_y = \frac{p_z}{k_v}(C_{22} - o_z v_0), \tag{2.36}$$

$$a_y = \frac{p_z}{k_v}(C_{23} - a_z v_0), \tag{2.37}$$

$$p_y = \frac{p_z}{k_v}(C_{24} - p_z v_0). \tag{2.38}$$

2.2. Space Resection

This technique involves the formulation of nonlinear equations involving the position and orientation parameters, and then solving for the solution iteratively by the Newton-Raphson algorithm.

Figure 2.2 shows the relationship between the frames *CAM* and *OBJ*. We want to find ${}^{CAM}\mathbf{T}_{OBJ}$; this is equivalent to finding a sequence of transformations which takes a frame from *CAM* to *OBJ*:

$$ {}^{CAM}\mathbf{T}_{OBJ} = \mathbf{Rot}(\mathbf{z},-\kappa)\mathbf{Rot}(\mathbf{y},-\phi)\mathbf{Rot}(\mathbf{x},-\omega)\mathbf{Trans}(-x_c,-y_c,-z_c). \tag{2.39} $$

CAM is rotated about its **z**-axis by $-\kappa$, followed by a rotation about the **y**-axis of the rotated frame by $-\phi$, a rotation about the **x**-axis of the rotated frame by $-\omega$, followed by a translation of $(-x_c,-y_c,-z_c)^t$ with respect to the resulting frame; the final frame will coincide with *OBJ*. Multiplying the matrices in (2.39) together we have the **n**, **o**, **a**, and **p** vectors of ${}^{CAM}\mathbf{T}_{OBJ}$ as follows:

$$ {}^{CAM}\left[\ \mathbf{T}\ \right]_{OBJ} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.40} $$

$$ = \begin{bmatrix} C_\phi C_\kappa & S_\omega S_\phi C_\kappa + C_\omega S_\kappa & -C_\omega S_\phi C_\kappa + S_\omega S_\kappa & -n_x x_f - o_x y_f - a_x z_f \\ -C_\phi S_\kappa & -S_\omega S_\phi S_\kappa + C_\omega C_\kappa & C_\omega S_\phi S_\kappa + S_\omega C_\kappa & -n_y x_f - o_y y_f - a_y z_f \\ S_\phi & -S_\omega C_\phi & C_\omega C_\phi & -n_z x_f - o_z y_f - a_z z_f \\ 0 & 0 & 0 & 1 \end{bmatrix}. $$

A set of collinearity equations are formed when we rotate *CAM* by the three rotations so that *CAM* and *OBJ* have the same orientation. Notice that we have not translated *CAM* at this point so it is still at a different position from *OBJ*, as shown in Figure 2.3. The *OBJ* frame has coordinate axes **x**–**y**–**z**; $(x_f,y_f,z_f)^t$ and $(x_c,y_c,z_c)^t$ are the coordinates of the feature point and the *CAM* origin with respect to *OBJ*. *CAM* has coordinate axes **x′**–**y′**–**z′** and $(x'_u,y'_u,z'_u)^t$ is the point where the line that joins the feature point and the focal point intersects the image plane, with respect to *CAM*. After the rotation, as shown in Figure 2.3, the resulting frame is **x″**–**y″**–**z″**, and the intersection point expressed in this coordinate axes is $(x''_u,y''_u,z''_u)^t$. From similar triangles in Figure 2.3, we have

$$\left(\frac{x''_u}{x_f-x_c}\right) = \left(\frac{y''_u}{y_f-y_c}\right) = \left(\frac{z''_u}{z_f-z_c}\right) \tag{2.41}$$

These equations can be rewritten as

$$x''_u = \left[\frac{x_f-x_c}{z_f-z_c} z''_u\right], \tag{2.42a}$$

$$y''_u = \left[\frac{y_f-y_c}{z_f-z_c} z''_u\right], \tag{2.42b}$$

$$z''_u = \left[\frac{z_f-z_c}{z_f-z_c} z''_u\right], \tag{2.42c}$$

Since we have rotated *CAM* to the same orientation as *OBJ*, we can use the rotation vectors $(\mathbf{n},\mathbf{o},\mathbf{a})^t$ of ${}^{CAM}\mathbf{T}_{OBJ}$ to relate $(x''_u,y''_u,z''_u)^t$ and $(x'_u,y'_u,z'_u)^t$. We have

$$\begin{bmatrix} x'_u \\ y'_u \\ z'_u \\ 1 \end{bmatrix} = \begin{bmatrix} n_x & o_x & a_x & 0 \\ n_y & o_y & a_y & 0 \\ n_z & o_z & a_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x''_u \\ y''_u \\ z''_u \\ 1 \end{bmatrix}. \tag{2.43}$$

Multiplying out, we have

$$x'_u = n_x x''_u + o_x y''_u + a_x z''_u \tag{2.44a}$$

$$y'_u = n_y x''_u + o_y y''_u + a_y z''_u \tag{2.44b}$$

$$z'_u = n_z x''_u + o_z y''_u + a_z z''_u \tag{2.44c}$$

Substituting (2.43a, 2.43b, 2.43c, 2.44c) into (2.44a, 2.44b), and substituting z'_u by the focal length F, we have

$$x'_u = F\left[\frac{n_x(x_f-x_c)+o_x(y_f-y_c)+a_x(z_f-z_c)}{n_z(x_f-x_c)+o_z(y_f-y_c)+a_z(z_f-z_c)}\right] \tag{2.45a}$$

$$y'_u = F\left[\frac{n_y(x_f-x_c)+o_y(y_f-y_c)+a_y(z_f-z_c)}{n_z(x_f-x_c)+o_z(y_f-y_c)+a_z(z_f-z_c)}\right] \tag{2.45b}$$

In these two non-linear equations, $(x_f,y_f,z_f)^t$ is known, and $(x'_u,y'_u)^t$ can be calculated from $(u,v)^t$ with (2.12) assuming the camera parameters are known. The unknowns we want to solve are the rotations ω, ϕ, and κ (concealed in **n**, **o**, **a**), and the origin of *CAM* $(x_c,y_c,z_c)^t$. We will arrange these two equations in suitable form before applying the Newton-Raphson algorithm. Equations (2.45a,b) are rewritten as

$$G = qx'_u - rf = 0 \tag{2.46a}$$

$$H = qy'_u - sf = 0 \tag{2.46b}$$

where

$$q = n_z(x_f-x_c)+o_z(y_f-y_c)+a_z(z_f-z_c)$$

$$r = n_x(x_f-x_c)+o_x(y_f-y_c)+a_x(z_f-z_c)$$

$$s = n_y(x_f-x_c)+o_y(y_f-y_c)+a_y(z_f-z_c)$$

Equations (2.46a,b) are linearized using Taylor's theorem,

$$(G)_0+\left(\frac{\partial G}{\partial x'_u}\right)_0 dx_a+\left(\frac{\partial G}{\partial \omega}\right)_0 d\omega+\left(\frac{\partial G}{\partial \phi}\right)_0 d\phi+\left(\frac{\partial G}{\partial \kappa}\right)_0 d\kappa \tag{2.47a}$$
$$+\left(\frac{\partial G}{\partial x_c}\right)_0 dx_c+\left(\frac{\partial G}{\partial y_c}\right)_0 dy_c+\left(\frac{\partial G}{\partial z_c}\right)_0 dz_c = 0$$

$$(H)_0+\left(\frac{\partial G}{\partial y'_u}\right)_0 dx'_u+\left(\frac{\partial H}{\partial \omega}\right)_0 d\omega+\left(\frac{\partial H}{\partial \phi}\right)_0 d\phi+\left(\frac{\partial H}{\partial \kappa}\right)_0 d\kappa \tag{2.47b}$$
$$+\left(\frac{\partial H}{\partial x_c}\right)_0 dx_c+\left(\frac{\partial H}{\partial y_c}\right)_0 dy_c+\left(\frac{\partial H}{\partial z_c}\right)_0 dz_c = 0$$

$(G)_0$, $(H)_0$, $(\partial G/\partial \omega)_0$, etc., are the functions G, H and their partial derivatives evaluated at the initial approximations. The $d\omega$, $d\phi$, $d\kappa$, dx_c, dy_c and dz_c are corrections to the initial approximations. From Equations (2.46a,b),

$$\frac{\partial G}{\partial x'_u} = \frac{\partial H}{\partial y'_u} = q \tag{2.48}$$

Performing the partial differentiations and simplifying, Equations (2.47a,b) can be written as follows:

$$dx'_u = b_{11}d\omega + b_{12}d\phi + b_{13}d\kappa - b_{14}dx_c - b_{15}dy_c - b_{16}dz_c + J \tag{2.49a}$$

$$dy'_u = b_{21}d\omega + b_{22}d\phi + b_{23}d\kappa - b_{24}dx_c - b_{25}dy_c - b_{26}dz_c + K \tag{2.49b}$$

where

$$\Delta x = x_f - x_c$$

$$\Delta y = y_f - y_c$$

$$\Delta z = z_f - z_c$$

$$b_{11} = \frac{x'_u}{q}(o_z\Delta z - a_z\Delta y) + \frac{f}{q}(a_x\Delta y - o_x\Delta z)$$

$$b_{12} = \frac{x'_u}{q}(\Delta x\cos\phi + \Delta y\sin\omega\sin\phi - \Delta z\sin\phi\cos\omega)$$

$$+ \frac{f}{q}(\Delta x\sin\phi\cos\kappa - \Delta y\sin\omega\cos\phi\cos\kappa)$$

$$- \Delta z\cos\omega\cos\phi\cos\kappa$$

$$b_{13} = -\frac{f}{q}(n_y\Delta x + o_y\Delta y + a_y\Delta z)$$

$$b_{14} = \frac{x'_u}{q}n_z - \frac{f}{q}n_x$$

$$b_{15} = \frac{x'_u}{q}o_z - \frac{f}{q}o_x$$

$$b_{16} = \frac{x'_u}{q}a_z - \frac{f}{q}a_x$$

$$J = \frac{qx'_u - rf}{q}$$

$$b_{21}=\frac{y'_u}{q}(o_z\Delta z-a_z\Delta y)+\frac{f}{q}(a_y\Delta y-o_y\Delta z)$$

$$b_{22}=\frac{y'_u}{q}(\Delta x\cos\phi+\Delta y\sin\omega\sin\phi-\Delta z\sin\phi\cos\omega)$$

$$+\frac{f}{q}(-\Delta x\sin\phi\sin\kappa+\Delta y\sin\omega\cos\phi\sin\kappa)$$

$$+\Delta z\cos\omega\cos\phi\sin\kappa$$

$$b_{23}=-\frac{f}{q}(n_x\Delta x+o_x\Delta y+a_x\Delta z)$$

$$b_{24}=\frac{y'_u}{q}n_z-\frac{f}{q}n_y$$

$$b_{25}=\frac{y'_u}{q}o_z-\frac{f}{q}o_y$$

$$b_{26}=\frac{y'_u}{q}a_z-\frac{f}{q}a_y$$

$$K=\frac{qy-sf}{q}$$

Equations (2.49a,b) can be used to solve for $d\omega$, $d\phi$, $d\kappa$, dx_c, dy_c and dz_c, the corrections for the orientation and position of the camera in order to improve the accuracy after each iteration. These changes are then added on to the previous estimate of ω, ϕ, κ, x_c, y_c, and z_c to get more accurate values. The iterations are terminated when the differential changes are smaller than the desired tolerance.

Three points are required to set up six equations in order to solve for the six unknowns. If there are more than three unknowns, mean square method can be used to minimize the sum of squares of the errors on the image positions $(x'_{u_i},y'_{u_i})^t$. For three different image features, Equations (2.49a,b) can be written in a matrix form.

$$\begin{bmatrix} J_1 \\ K_1 \\ J_2 \\ K_2 \\ J_3 \\ K_3 \end{bmatrix} = \begin{bmatrix} -b_{11_1} & -b_{12_1} & -b_{13_1} & b_{14_1} & b_{15_1} & b_{16_1} \\ -b_{21_1} & -b_{22_1} & -b_{23_1} & b_{24_1} & b_{25_1} & b_{26_1} \\ -b_{11_2} & -b_{12_2} & -b_{13_2} & b_{14_2} & b_{15_2} & b_{16_2} \\ -b_{21_2} & -b_{22_2} & -b_{23_2} & b_{24_2} & b_{25_2} & b_{26_2} \\ -b_{11_3} & -b_{12_3} & -b_{13_3} & b_{14_3} & b_{15_3} & b_{16_3} \\ -b_{21_3} & -b_{22_3} & -b_{23_3} & b_{24_3} & b_{25_3} & b_{26_3} \end{bmatrix} \begin{bmatrix} d\omega \\ d\phi \\ d\kappa \\ dx_c \\ dy_c \\ dz_c \end{bmatrix} + \begin{bmatrix} dx'_{u_1} \\ dy'_{u_1} \\ dx'_{u_2} \\ dy'_{u_2} \\ dx'_{u_3} \\ dy'_{u_3} \end{bmatrix} \tag{2.50}$$

This equation is rewritten as

$$\mathbf{L} = \mathbf{A}\mathbf{X}_c + \mathbf{V} \tag{2.51}$$

where the error $\mathbf{V}$ is commonly known as residue. To minimize the residue with more than 3 points, $\mathbf{X}_c$ can be solved as follows.

$$\mathbf{X}_c = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{L} \tag{2.52}$$

2.3. Combining Space Resection with Camera Matrix Decomposition

Since space resection has inherently better behaviour in a noisy environment, it is used to calculate the position of an object. However, this method does require an initial estimate of the position and requires the camera parameters; this can be achieved using calibration matrix decomposition. Camera parameters only have to be found once as long as the same camera and the same lenses are used; a calibration object with abundant features should be used to find the parameters as accurately as possible. When camera calibration is used to find the initial estimate of the position of an object, the camera parameters are re-calculated as a by-product; these values can be ignored because they are not accurate as the one calculated initially.

In order to combine the two techniques, we need to be able to convert their different notations to one another. Calibration matrix used homogeneous transforms while space resection expresses positions and orientations in terms of $(x_c,y_c,z_c,\omega,\phi,\kappa)^t$.

The results of space resection will be in terms of $(x_c,y_c,z_c,\omega,\phi,\kappa)^t$, we can convert them into a homogeneous transform using (2.40) .

The initial estimate found by calibration matrix decomposition is a homogeneous transform. To convert it to $(x_c,y_c,z_c,\omega,\phi,\kappa)^t$, we will use a method used by Paul, Renald, and Stevenson [Paul83]. We will first solve for the rotation part of the matrix. Let

$$\mathbf{M} = \begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{bmatrix} \tag{2.53}$$

Let us define **U** and **V** matrices as in [Paul83].

$$\mathbf{V}_0 = \mathbf{R}_1\mathbf{R}_2\mathbf{R}_3=\mathbf{M}=\mathbf{U}_1 \tag{2.54}$$

$$\mathbf{V}_1 = \mathbf{R}_2\mathbf{R}_3=\mathbf{R}_1^{-1}\mathbf{M}=\mathbf{U}_2 \tag{2.55}$$

$$\mathbf{V}_2 = \mathbf{R}_3 = \mathbf{R}_2^{-1}\mathbf{V}_1 = \mathbf{U}_3. \tag{2.56}$$

where $\mathbf{R}_1$, $\mathbf{R}_2$, and$\mathbf{R}_3$ are the rotational matrices about the z, y, and x axes by $-\kappa$, $-\phi$, and $-\omega$, respectively. Let $\mathbf{V}_{i_j}$ denote the jth column of $\mathbf{V}_i$ and let $\mathbf{M}_j=[s_i,r_i,t_i]^T$ stands for the jth column of **M**, i.e.,

$$\mathbf{M}_1=[n_x\ n_y\ n_z]^T,\mathbf{M}_2=[o_x\ o_y\ o_z]^T,\mathbf{M}_3=[a_x\ a_y\ a_z]^T.$$

From (2.55),

$$\mathbf{V}_{1_j}=\mathbf{R}_1^{-1}\mathbf{M}_j=\begin{bmatrix} C_\kappa & -S_\kappa & 0 \\ S_\kappa & C_\kappa & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} s_j \\ r_j \\ t_j \end{bmatrix} \tag{2.57}$$

$$=\begin{bmatrix} C_\kappa s_j - S_\kappa r_j \\ S_\kappa s_j + C_\kappa r_j \\ t_j \end{bmatrix}=\begin{bmatrix} v_{11j} \\ v_{12j} \\ t_j \end{bmatrix}$$

where

$$v_{11j}=C_\kappa s_j - S_\kappa r_j \tag{2.58}$$

$$v_{12j}=S_\kappa s_j + C_\kappa r_j \tag{2.59}$$

From (2.56), to express $\mathbf{V}_{2_j}$ in terms of $\mathbf{M}_j$,

$$\mathbf{V}_{2_j}=\mathbf{R}_2^{-1}\mathbf{M}_j=\begin{bmatrix} C_\phi v_{11}+S_\phi t_j \\ v_{12} \\ -S_\phi v_{11}+C_\phi t_j \end{bmatrix} \tag{2.60}$$

Now we can equate $\mathbf{V}_2$ to $\mathbf{R}_3$,

$$\begin{bmatrix}\mathbf{V}_{2_1}\mathbf{V}_{2_2}\mathbf{V}_{2_3}\end{bmatrix}\begin{bmatrix} C_\phi v_{111}+S_\phi n_z & C_\phi v_{112}+S_\phi o_z & C_\phi v_{113}+S_\phi a_z \\ v_{121} & v_{122} & v_{123} \\ -S_\phi v_{111}+C_\phi n_z & -S_\phi v_{112}+C_\phi o_z & -S_\phi v_{113}+C_\phi a_z \end{bmatrix}=\begin{bmatrix} 1 & 0 & 0 \\ 0 & C_\omega & S_\omega \\ 0 & -S_\omega & C_\omega \end{bmatrix} \tag{2.61}$$

$$C_\omega=v_{122}=S_\kappa o_x+C_\kappa o_y \tag{2.62}$$

$$S_\omega=S_\phi v_{112}-C_\phi o_z=S_\phi(C_\kappa o_x-S_\kappa o_y)-C_\phi o_z \tag{2.63}$$

A unique value of ω can be found by the atan2 function using sine and cosine as arguments.

$$\omega=\tan^{-1}\frac{S_\omega}{C_\omega} \tag{2.64}$$

To solve for ϕ, we equate $\mathbf{V}_1$ to $\mathbf{U}_2$.

$$\begin{bmatrix} C_\kappa x-S_\kappa y \\ S_\kappa x-C_\kappa y \\ z \end{bmatrix}=\mathbf{R}_2\mathbf{R}_3 \tag{2.65}$$

$$= \begin{bmatrix} C_\phi & S_\phi S_\omega & -S_\phi C_\omega \\ 0 & C_\omega & S_\omega \\ S_\phi & -C_\phi S_\omega & C_\phi C_\omega \end{bmatrix}$$

$$C_\phi = C_\kappa n_x - S_\kappa n_y \tag{2.67}$$

$$S_\phi = n_z \tag{2.68}$$

$$\phi = \tan^{-1} \frac{S_\phi}{C_\phi} \tag{2.69}$$

Finally, to solve for κ, we equate V_0 to U_1,

$$\begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{bmatrix} = \begin{bmatrix} C_\kappa C_\phi & ? & ? \\ -S_\kappa C_\phi & ? & ? \\ S_\phi & ? & ? \end{bmatrix} \tag{2.70}$$

where the terms with question marks are not used.

$$C_\kappa = \frac{n_x}{C_\phi} \tag{2.71}$$

$$S_\kappa = -\frac{n_y}{C_\phi} \tag{2.72}$$

It can be concluded that

$$\kappa = \tan^{-1} \frac{-n_y}{n_x} \qquad \text{if } C_\phi > 0 \tag{2.73}$$

$$\kappa = \tan^{-1} \frac{n_y}{-n_x} \qquad \text{if } C_\phi < 0$$

Note that κ is undefined when $C_\phi = 0$. There are multiple solutions to κ, ϕ and ω. We only have to find one set of these solutions since we are only translating one set of notation to another and are not concerned about the mechanical configurations of an arm.

To find x_c, y_c and z_c is straightforward once one realizes that they are the origin of the camera coordinate axes with respect to the object. The transform of the camera with respect to the object is just ${}^{CAM}T_{OBJ}^{-1}$.

$$\begin{bmatrix} n_x & n_y & n_z & -\mathbf{n}.\mathbf{p} \\ o_x & o_y & o_z & -\mathbf{o}.\mathbf{p} \\ a_x & a_y & a_z & -\mathbf{a}.\mathbf{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.74}$$

Therefore,

$$x_c = -\mathbf{n}.\mathbf{p} \tag{2.75}$$

$$y_c = -\mathbf{o}.\mathbf{p} \tag{2.76}$$

$$z_c = -\mathbf{a}.\mathbf{p} \tag{2.77}$$

3. Finding Object Position with Two Views

We have discussed finding an object position with a single view in the last section. Ganapathy's method requires six points while space resection method requires three points. Fischler and Bolles [Fisc 81] showed that space resection with less than 6 points may have multiple solutions. To disambiguate the solutions, we can either check the consistencies of the solutions with the image, or we can use more than one view.

Multiple views are also useful when we deal with an object with very few local features, or when the viewpoint is degenerate (when the features and the focal point lie on the same circle), or when some of the features are occluded. An example of this kind of application is locating a wire or a thin peg; a wire has only two features.

we will discuss a method which uses two views to find the positions of individual feature points. Figure 3.1 shows a thin rod and its image on two cameras whose positions and orientations are known. CAM and CAM' are the coordinate frames of the two cameras. We can find the 3D coordinates of **P1** and **P2** by finding intersection of lines l_1 and l'_1 and intersection of lines l_2 and l'_2 respectively. For demonstration purposes, we will find **P1** with respect to *CAM* whose coordinate frame is x—y—z. The parametric equation of l_1 is

$$(x,y,z)^t = (0,0,-f)^t + t(x_1,y_1,f)^t, \tag{3.1}$$

where $(x_1,y_1)^1$ are the image coordinates of the point **P1** and f is the focal length. The equation of l'_1 with respect to *CAM'* is

$$(x',y',z')^t = (0,0,-f)^t + t'(x'_1,y'_1,f)^t. \tag{3.2}$$

We can specify this equation with respect to *CAM* by specifying the point $(0,0,-f)^t$ and the vector $(x'_1,y'_1,f)^t$ with respect to CAM. If

$$\mathbf{T} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

is the transform matrix from *CAM* to *CAM'*, then the point ${}^{CAM'}(0,0,-f)^t$ is transformed too

$${}^{CAM}(-a_xf+p_x,-a_yf+p_y,-a_zf+p_z)^t,$$

and the vector ${}^{CAM'}(x'_1,y'_1,f)^t$ is transformed to

$$n_yx'_1+o_yy'_1+a_yf,n_zx'_1+o_zy'_1+a_zf)^t$$

Thus the equation of l'_1 is

$$(x,y,z)^t = (-a_x f+p_x,-a_y f+p_y,-a_z f+p_z)^t+ \tag{3.4}$$

$$t'(n_x x'_1+n_y y'_1+n_z f,n_y x'_1+o_y y'_1+a_y f,n_z x'_1+o_z y'_1+a_z f)^t$$

The two lines generally may not intersect because of errors; we will find the mid-point of the shortest line segment joining l_1 and l'_1. Let us rewrite the two lines as

$$l_1:(x,y,z)^t = (x_0,y_0,z_0)^t+t(a,b,c)^t, \tag{3.5}$$

$$l'_1:(x,y,z)^t = (x'_0,y'_0,z'_0)^t+t'(a',b',c')^t, \tag{3.6}$$

Let D be the distance form a point on l_1 to l'_1.

$$D^2 = (x_0+at-x'_0-a't')^2+(y_0+bt-y'_0-a't')^2+(z_0+ct-z'_0-c't')^2 \tag{3.7}$$

Taking partial derivatives with respect to t and t', we will have two linear equations from which we can solve for t and t':

$$a'(x_0-x'_0)+b'(y_0-y'_0)+c'(z_0-z'_0)+t(aa'+bb'+cc')-t'(a'^2+b'^2+c'^2)=0. \tag{3.8}$$

$$a(x_0-x'_0)+b(y_0-y'_0)+c(z_0-z'_0)-t'(aa'+bb'+cc')+t(a^2+b^2+c^2)=0. \tag{3.9}$$

From the values of t and t', we can find points on the two lines that are closest to one another. We will take the mid-point as the estimate of the intersection of these two lines. Once **P1** and **P2** are found by this method, we can find the p and **a** vector of the homogeneous frame of the peg. There are infinite number of solutions for the **n** and **o** vector since the peg is symmetric about the z-axis. If $\mathbf{P1} = (p1_x,p1_y,p1_z)^t$ and $\mathbf{P2} = (p2_x,p2_y,p2_z)^t$,

$$(p_x,p_y,p_z)^t = (p2_x,p2_y,p2_z)^t, \tag{3.10}$$

$$(a_x,a_y,a_z)^t = \frac{(p1_x-p2_x,p1_y-p2_y,p1_z-p2_z)^t}{\sqrt{(p1_x-p2_x)^2+(p1_y-p2_y)^2(p1_z-p2_z)^2}} \tag{3.11}$$

4. Formulating The Manipulator Sequences in terms of Homogeneous Transform Equations

We will show two examples of formulating transform equations. In the first example, we assume the camera is mounted on the robot wrist and the task is to pick up an object in a two-move sequence. The second example shows an assembly task where the camera is located arbitrarily. We will also show the use of a time-variant $\mathbf{R}(\gamma)$ transform which is servoed visually.

In the first example, the camera is fixed to the robot wrist, and the object to be grasped is assumed to be within the view of the camera The relative position of the object with respect to the camera can be calculated using techniques described in the last section, let us denote this transformed as ${}^{CAM}\mathbf{T}_{OBJ}$. We also let **T6**(k) and **T6**(k+1) be the current position and the next position of the manipulator end, **HAND** be the gripper position with respect to manipulator wrist flange, ${}^{\mathbf{T6}}\mathbf{T}_{CAM}$ be the camera position

relative to manipulator wrist flange, and **APPR** be the predefined "approach" position of the gripper with respect to the object. From Figure 4.1, we have

T6(k+1) can be solved as

$$\mathbf{T6}(k+1) = \mathbf{T6}(k)\ {}^{\mathbf{T6}}\mathbf{T}_{CAM}$$

After the move to the "approach" position, the robot will move on to the "grasp" position. The **T6** will be calculated using an equation similar to the previous one, except that **APPR** is replaced by **GRASP**.

Figure 4.2 shows two robots assembling two parts and the camera is at an arbitrary position viewing the part-mating operation; visual Servoing is used to overcome the mismatch of the two robot spaces (see next section). The camera position is not assumed to be known since the method described will be independent of it. Let *OBJ*1 and *OBJ*2 be the top object and the bottom object respectively. The relative position of *OBJ*1 with respect to *OBJ*2 (${}^{OBJ2}\mathbf{T}_{OBJ1}$) can be found by vision.

Assume Robot2 holding *OBJ*2 remains stationary and we want to move the Robot1 which manipulates *OBJ*1. We want to move Robot1 such that *OBJ*1 is at the desired "approach" position with respect to *OBJ*2, and let this transform be ${}^{OBJ2}\mathbf{TAPPR}_{OBJ1}$. Initially, due to coordinate mismatch, *OBJ*1 is at ${}^{OBJ2}\mathbf{T}(k)_{OBJ1}$, we want to move Robot1 such that *OBJ*1 is at the correct "approach" position. Let **T6**(k) and **T6**(k+1) be the position of the current position and next

position of Robot1 respectively, and **GRASP1** and **HAND1** be defined similar to the previous example. From Figure 4.2, we can see that

$$\mathbf{T6}(k+1) = \mathbf{T6}(k)\ \mathbf{HAND1}\ \mathbf{GRASP1}^{-1}\ \mathbf{GRASP1}\ \mathbf{HAND1}^{-1}$$

Once *OBJ*1 is at the "approach" position, we now can proceed to the "mate" position. We can just use an equation similar to the last one except substituting **TAPPR** by **TMATE**. However, if the resolution of the vision system is better than the robot accuracy, we should visually servo on all the intermediate positions from the "approach" position to "mate" position in order to acquire better accuracy. This can be accomplished by defining a variable transform $\mathbf{D}(\gamma)$ such that it changes from **TAPPR** to **TMATE** when γ changes from 0 to 1. For example, if we want Robot1 to move down from **TAPPR** to **TMATE** at a constant speed, and if

$$^{OBJ}[\mathbf{TAPPR}]_{OBJ2} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{OBJ2}\mathbf{TMATE}_{OBJ1} =$$

we can defined $\mathbf{D}(\gamma)$ as

$$\mathbf{D}(\gamma) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & (1-\gamma)d1+\gamma d2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where γ changes from 0 to 1. For general transitions from one transform to another, we can use the more general "drive" function described in Paul's book [Paul83].

5. Conclusions

We have presented methods of calculating object positions from camera images. For objects with enough visible features, we used a combination of calibration matrix decomposition and space resection, and have achieved low noise sensitivity without requiring an initial estimate of the object position. For objects with very few features, we have presented a method which uses multiple views to find the 3D positions of feature points. We have also discussed the usage of the transforms to drive the manipulator during the visual servoeing.

Appendix: Simulations of Noise Sensitivities

We simulated the noise sensitivities of camera calibration matrix decomposition and space resection by disturbing the 2D position of an feature point at pixel increments, assuming we have a 256 by 256 image. Figure A1 shows the viewpoint of the object and the feature to be disturbed. Figure A2 shows the comparative noise sensitivities.

List of References

[Dode78]
I.A. Dodes, *Numerical Analysis for Computer Science*, North-Holland, New York, 1978.

[Fisc81]
M.A. Fischler and R.C.Bolles, "*Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography*", Communications of the ACM, June 1981, vol. 24, no. 6, pp. 381-395.

[Gana84]
S. Ganapathy, "Decomposition of Transformation Matrices for Robot Vision", *International Conference on Robotics*, Atlanta, Georgia, March 1984, pp. 130-139.

[Lowe84]
Perceptual Organization and Visual Recognition, Ph.D dissertation, Stanford University, September 1984.

[Paul81]
Richard P. Paul, *Robot Manipulators: Mathematics, Programming and Control*, MIT Press, 1981.

[Paul83]
Richard P. Paul, Marc Renaud, and Charles N. Stevenson, "A systematic Approach for Obtaining the Kinematics of Recursive Manipulators based on Homogeneous Transformations", *International Symposium on Robotics Research*, Brettonwoods, September 1983.

[Wolf83]
P.R. Wolf, *Elements of Photogrammetry*, McGraw-Hill, New York, 1983.

List of Figures

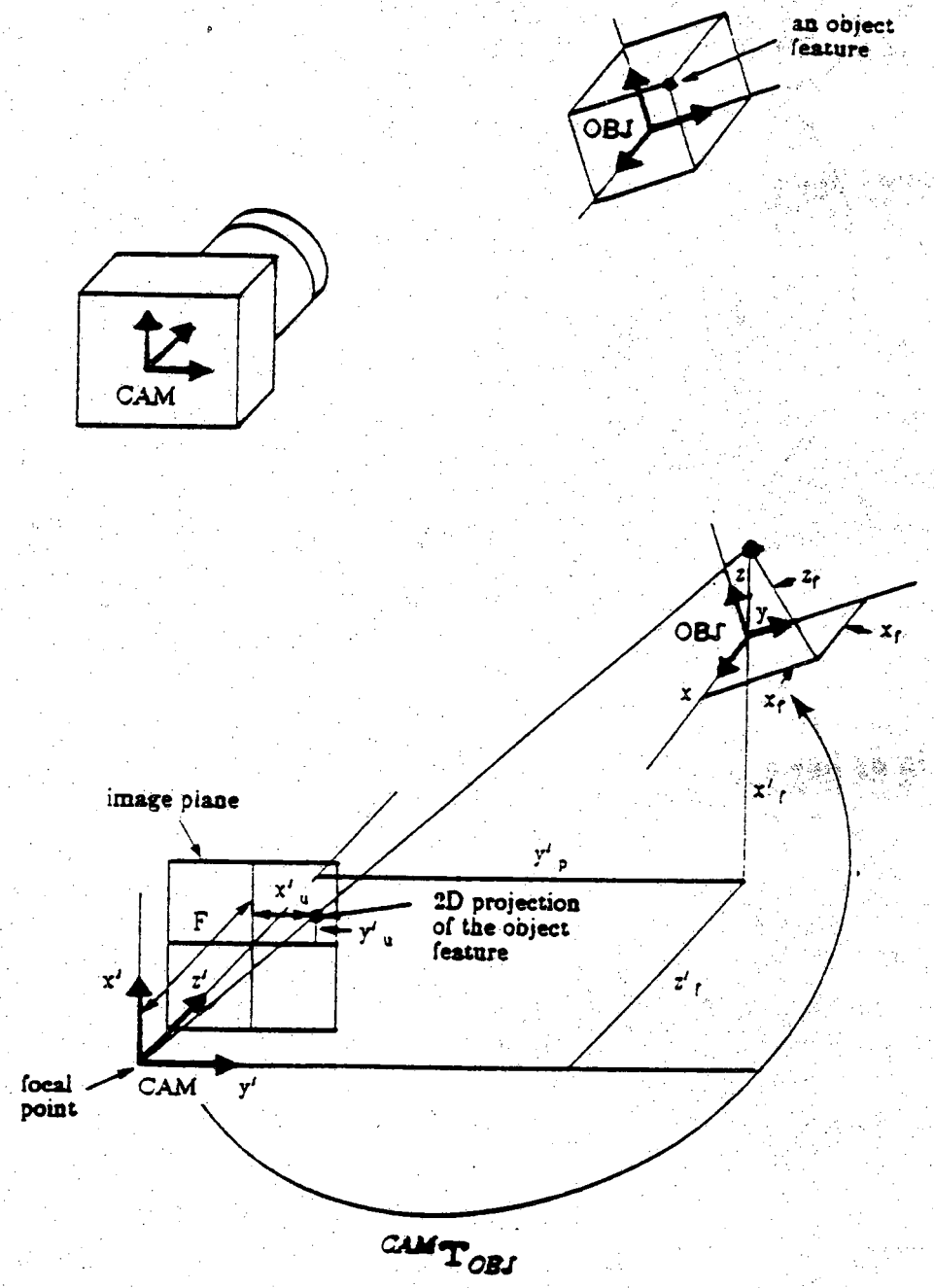


Figure 2.1. Geometry of the relationship between a camera and an object

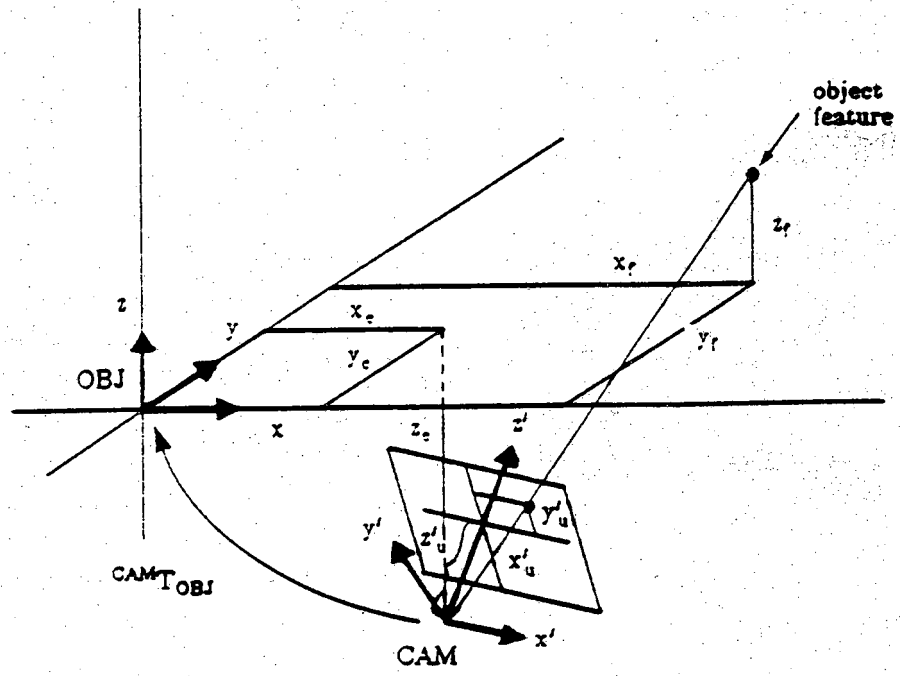


Figure 2.2. Viewpoint determination by space resection

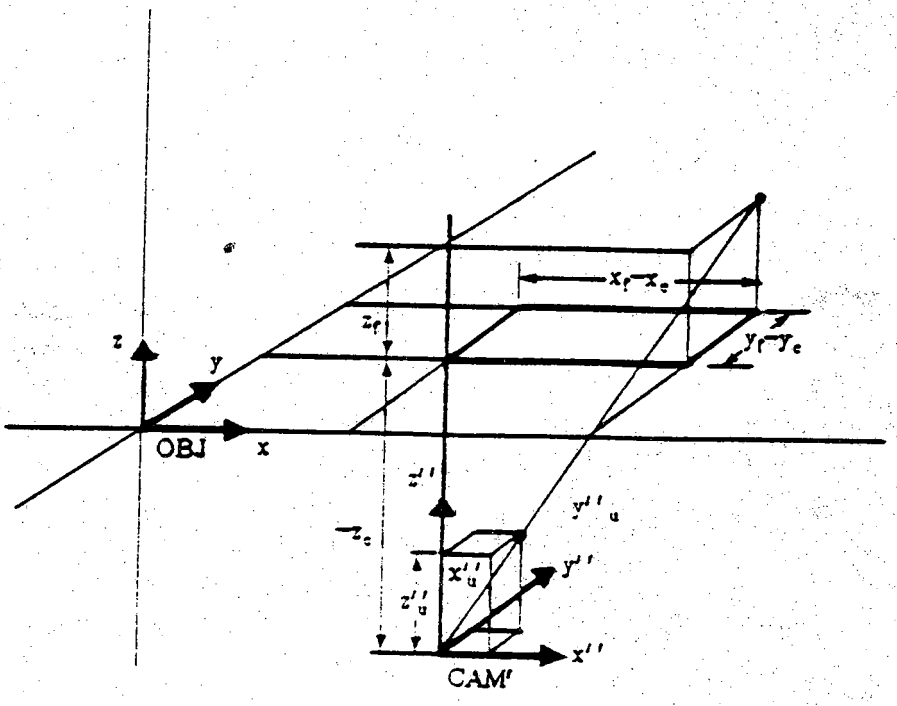


Figure 2.3 Aligning the camera frame with the object frame in order to write the collinearity equations

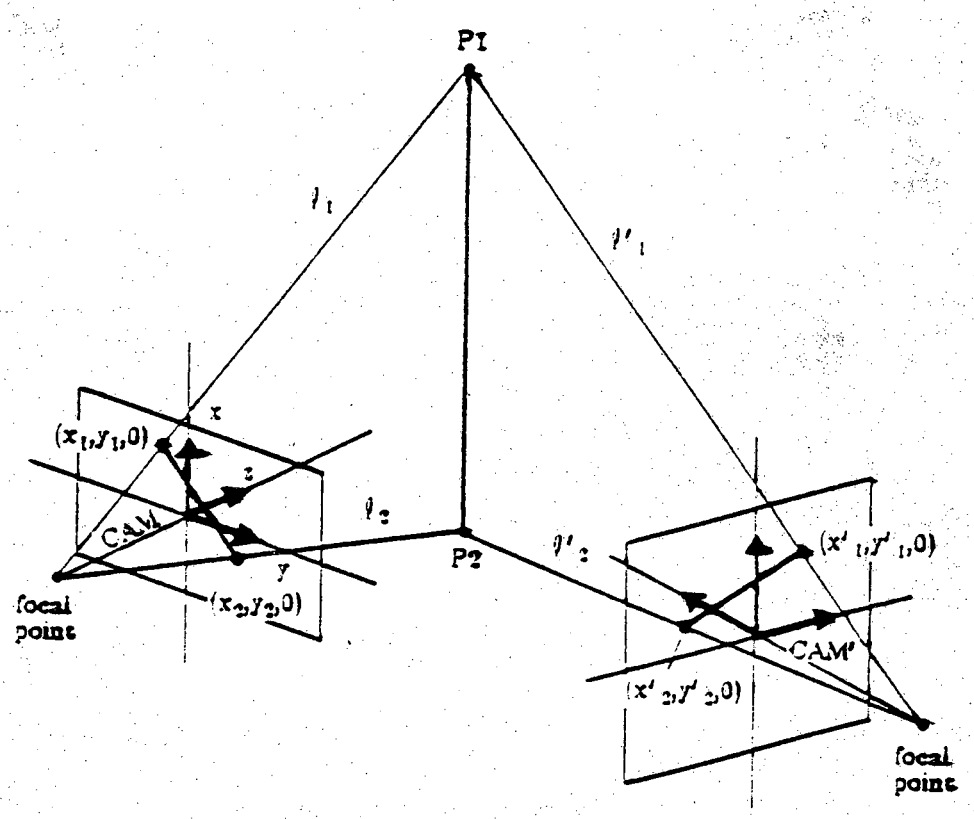


Figure 3.1. Finding object position with two views

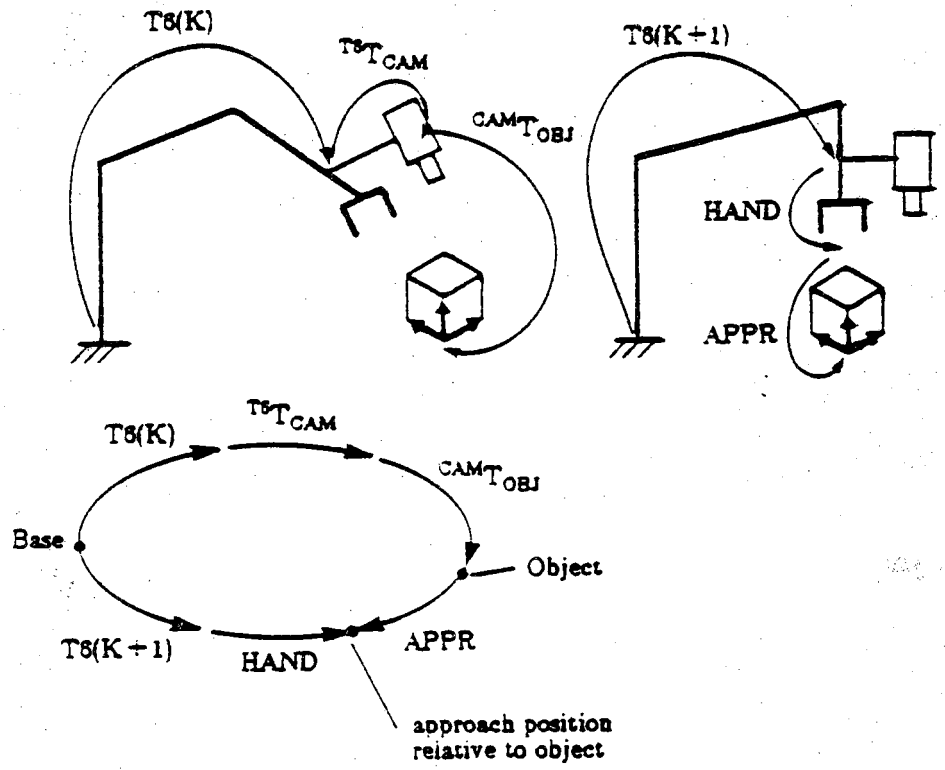


Figure 4.1. Homogeneous transform equation for approaching an object

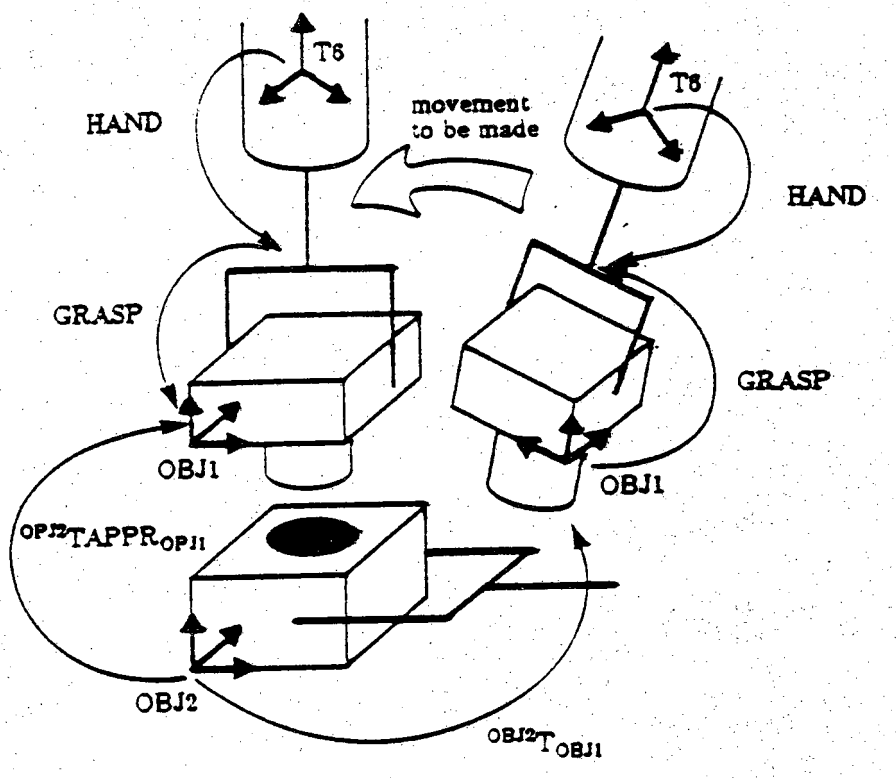


Figure 4.2. Transform equations for assembly

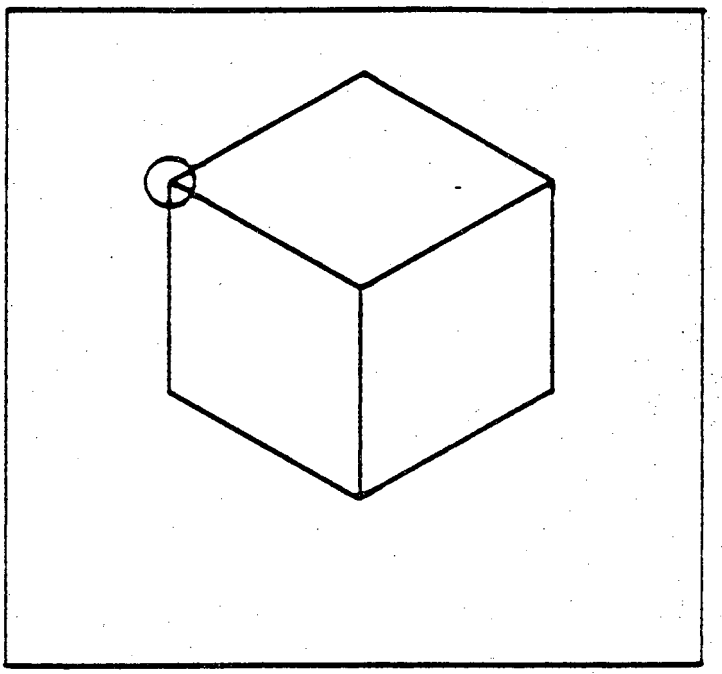
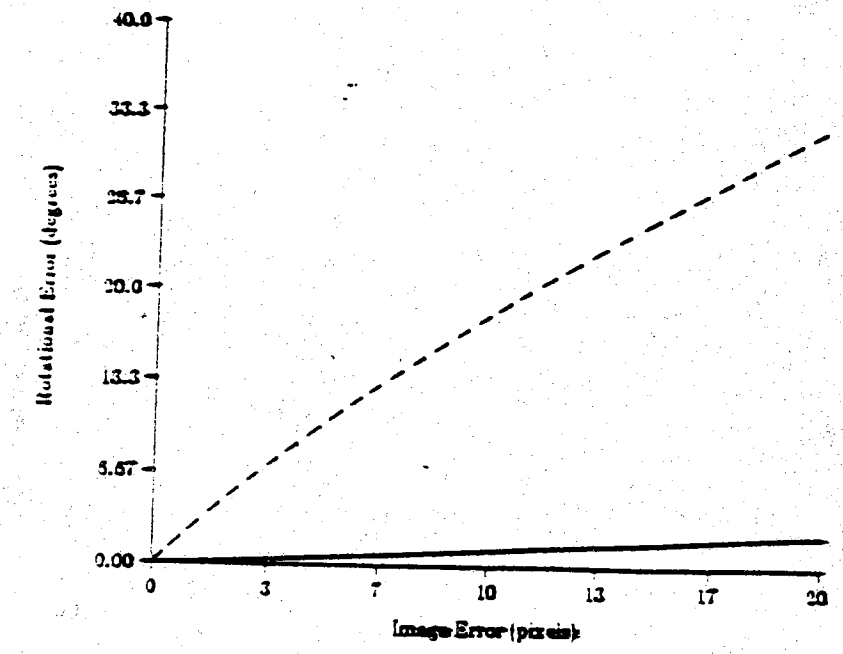


Figure A1 Simulated image of a cube. The feature to be perturbed is circled. The simulated image has a resolution of 256×256 pixels and the camera is 500 mm from the closest corner of the cube.



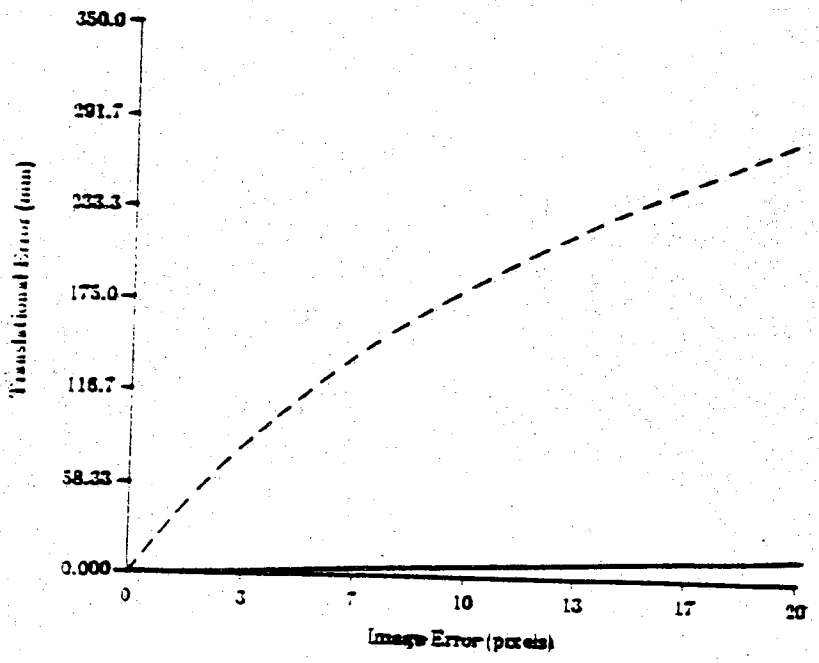


Figure A2. Rotational and translational errors resulting from perturbation of a feature position