(scaling and skewing), including the projective. The semantics of the matrix are summarized in Fig. A1.10.

Since the results of applying a transformation to a row vector is another row vector, transformations may be concatenated by repeated matrix multiplication. Such composition of transformations follows the rules of matrix algebra (it is associative but not commutative, for instance). The semantics of

$$\mathbf{x}' = \mathbf{x}ABC$$

is that $\mathbf{x}'$ is the vector resulting from applying transformation A to $\mathbf{x}$, then B to the transformed $\mathbf{x}$, then C to the twice-transformed $\mathbf{x}$. The single 4×4 matrix $D = ABC$ would do the same job. The inverses of geometric transformation matrices are just the matrices expressing the inverse transformations, and are easy to derive.

A1.8. CAMERA CALIBRATION AND INVERSE PERSPECTIVE

The aim of this section is to explore the correspondence between world and image points. A (half) line of sight in the world corresponds to each image point. Camera calibration permits prediction of where in the image a world point will appear. Inverse perspective transformation determines the line of sight corresponding to an image point. Given an inverse perspective transform and the knowledge that a visible point lies on a particular world plane (say the floor, or in a planar beam of light), then its precise three-dimensional coordinates may be found, since the line of sight generally intersects the world plane in just one point.

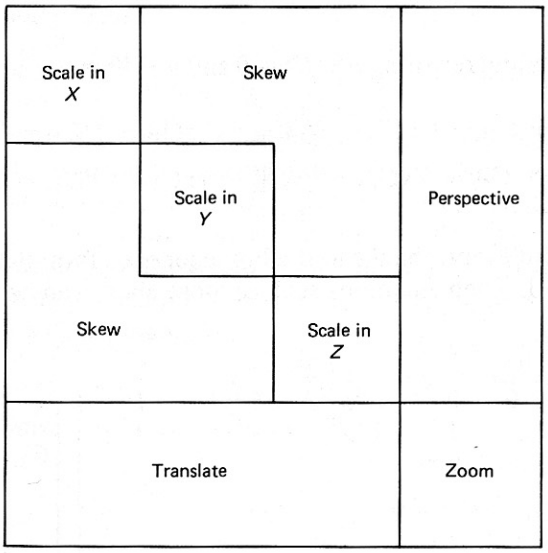

Fig. A1.10 The 4×4 homogeneous transformation matrix.

A1.8.1 Camera Calibration

This section is concerned with the "camera model"; the model takes the form of a 4×3 matrix mapping three-dimensional world points to two-dimensional image points. There are many ways to derive a camera model. The one given here is easy to state mathematically; in practice, a more general optimization technique such as hill climbing can be most effective in finding the camera parameters, since it can take advantage of any that are already known and can reflect dependencies between them.

Let the image plane coordinates be U and V; in homogeneous coordinates an image plane point is (u,v,t). Thus

$$U = \frac{u}{t}$$

$$V = \frac{v}{t}$$

Call the desired camera model matrix C, with elements C_{ij} and column four-vectors C_j. Then for any world point (x, y, z) a C is needed such that

$$(x, y, z, 1)C = (u, v, t)$$

So

$$\begin{aligned} u &= (x, y, z, 1)C_1 \\ v &= (x, y, z, 1)C_2 \\ t &= (x, y, z, 1)C_3 \end{aligned}$$

Expanding the inner products and rewriting $u - Ut = 0$ and $v - Vt = 0$,

$$\begin{aligned} xC_{11} + yC_{21} + zC_{31} + C_{41} - UxC_{13} - UyC_{23} - UzC_{33} - UC_{43} = 0 \\ xC_{12} + yC_{22} + zC_{32} + C_{42} - VxC_{13} - VyC_{23} - VzC_{33} - VC_{43} = 0 \end{aligned}$$

The overall scaling of C is irrelevant, thanks to the homogeneous formulation, so C_{43} may be arbitrarily set to 1. Then equations such as those above can be written in matrix form:

$$\begin{bmatrix} x^1 & y^1 & z^1 & 1 & 0 & 0 & 0 & 0 & -U^1x^1 & -U^1y^1 & -U^1z^1 \\ 0 & 0 & 0 & 0 & x^1 & y^1 & z^1 & 1 & -V^1x^1 & -V^1y^1 & -V^1z^1 \\ x^2 & y^2 & z^2 & 1 & \cdot & \cdot & \cdot & & & & \cdot \\ \cdot & & & & & & & & & & \cdot \\ \cdot & & & & & & & & & & \cdot \\ \cdot & & & & & & & & & & \cdot \\ 0 & 0 & 0 & 0 & x^n & y^n & z^n & 1 & -V^nx^n & -V^ny^n & -V^nz^n \end{bmatrix} \begin{bmatrix} C_{11} \\ C_{21} \\ \cdot \\ \cdot \\ \cdot \\ \\ C_{34} \end{bmatrix} = \begin{bmatrix} U^1 \\ V^1 \\ \cdot \\ \cdot \\ \cdot \\ U^n \\ V^n \end{bmatrix}$$

Eleven such equations allow a solution for C. Two equations result for every association of an (x, y, z) point with a (U, V) point. Such an association must be established using visible objects of known location (often placed for the purpose). If more than 5½ such observations are used, a least-squared-error solution to the overdetermined system may be obtained by using a pseudo-inverse to solve the resulting matrix equation (Section A1.9).

A1.8.2 Inverse Perspective

Finding the world line corresponding to an image point relies on the fact that the perspective transformation matrix also affects the z component of a world point. This information is lost when the z component is projected away orthographically, but it encodes the relation between the focal point and the z position of the point. Varying this third component references points whose world positions vary in z but which project onto the same position in the image. The line can be parameterized by a variable p that formally occupies the position of that z coordinate in three-space that has no physical meaning in imaging.

Write the inverse perspective transform P^{-1} as

$$(x', y', p, 1)P^{-1} = (x', y', p, 1 + \frac{p}{f})$$

Rewriting this in the usual way gives these relations between the (x, y, z) points on the line.

$$(x, y, z, 1) = \left[\frac{fx'}{f+p}, \frac{fy'}{f+p}, \frac{fp'}{f+p}, 1\right]$$

Eliminating the parameter p between the expressions for z and x and those for z and y leaves

$$x = \frac{x'}{y'}\, y = \frac{-x'}{f}\, (z - f)$$

Thus x, y, and z are linearly related; as expected, all points on the inverse perspective transform of an image point lie in a line, and unsurprisingly both the viewpoint $(0, 0, f)$ and the image point $(x', y', 0)$ lie on it.

A camera matrix C determines the three-dimensional line that is the inverse perspective transform of any image point. Scale C so that $C_{43} = 1$, and let world points be written $\mathbf{x} = (x, y, z, 1)$ and image points $\mathbf{u} = (u, v, t)$. The actual image points are then

$$U = \frac{u}{t}, \quad V + \frac{v}{t}, \qquad \text{so } u = Ut, \quad v + Vt$$

Since

$$\mathbf{u} = \mathbf{x}C,$$

$$u = Ut = \mathbf{x}C_1$$

$$v = Vt = \mathbf{x}C_2$$

$$t = \mathbf{x}C_3$$

Substituting the expression for t into that for u and v gives

$$UxC_3 = xC_1$$
$$VxC_3 = xC_2$$

which may be written

$$\mathbf{x}(C_1 - UC_3) = 0$$
$$\mathbf{x}(C_2 - VC_3) = 0$$

These two equations are in the form of plane equations. For any U, V in the image and camera model C, there are determined two planes whose intersection gives the desired line. Writing the plane equations as

$$a_1x + b_1y + c_1z + d_1 = 0$$
$$a_2x + b_2y + c_2z + d_2 = 0$$

then

$$a_1 = C_{11} - C_{13}U \qquad a_2 = C_{12} - C_{13}V$$

and so on. The direction (λ, μ, ν) of the intersection of two planes is given by the cross product of their normal vectors, which may now be written as

$$(\lambda, \mu, \nu) = (a_1, b_1, c_1) \times (a_2, b_2, c_2)$$
$$= (b_1c_2 - b_2c_1,\ c_1a_2 - c_2a_1,\ a_1b_2 - a_2b_1)$$

Then if $\nu \neq 0$, for any particular z_0,

$$x_0 = \frac{b_1(c_2z_0 + d_2) - b_2(c_1z_0 - d_1)}{a_1b_2 - b_1a_2}$$

$$y_0 = \frac{a_2(c_1z_0 + d_1) - a_1(c_2z_0 - d_2)}{a_1b_2 - b_1a_2}$$

and the line may be written

$$\frac{x - x_0}{\lambda} = \frac{y - y_0}{\mu} = \frac{z - z_0}{\nu}$$

A1.9. LEAST-SQUARED-ERROR FITTING

The problem of fitting a simple functional model to a set of data points is a common one, and is the concern of this section. The subproblem of fitting a straight line to a set of (x, y) points ("linear regression") is the first topic. In computer vision, this line-fitting problem is encountered relatively often. Model-fitting methods try to find the "best" fit; that is, they minimize some error. Methods which yield closed-form, analytical solutions for such best fits are at issue here.

The relevant "error" to minimize is determined partly by assumptions of dependence between variables. If x is independent, the line may be represented as $y = mx + b$ and the error defined as the vertical displacement of a point from the line. Symmetrically, if x is dependent, horizontal error should be minimized. If neither variable is dependent, a reasonable error to minimize is the perpendicular distance from points to the line. In this case the line equation $ax + by + 1 = 0$ can be used with the method shown here, or the eigenvector approach of Section A1.9.2 may be used.

A1.9.1 Pseudo-Inverse Method

In fitting an $n \times 1$ observations matrix y by some linear model of p parameters, the prediction is that the linear model will approximate the actual data. Then

$$Y = XB + E$$

where X is an $n \times p$ formal independent variable matrix, B is a $p \times 1$ parameter matrix whose values are to be determined, and E represents the difference between the prediction and the actuality: it is an $n \times 1$ error matrix.

For example, to fit a straight line $y = mx + b$ to some data (x_i, y_i) points, form Y as a column matrix of the y_i.

$$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ & \cdot \\ & \cdot \\ & \cdot \end{bmatrix}$$

$$B = \begin{bmatrix} b \\ m \end{bmatrix}$$

Now the task is to find the parameter B (above, the b and m that determine the straight line) that minimizes the error. The error is the sum of squared difference from the prediction, or the sum of the elements of E squared, or E^TE (if we do not mind conflating the one-element matrix with a scalar). The mathematically attractive properties of the squared-error definition are almost universally taken to compensate for whatever disadvantages it has over what is really meant by error (the absolute value is much harder to calculate with, for example).

To minimize the error, simply differentiate it with respect to the elements of B and set the derivative to 0. The second derivative is positive: this is indeed a minimum. These elementwise derivatives are written tersely in matrix form. First rewrite the error terms:

$$\begin{aligned} E^TE &= (Y - XB)^T(Y - XB) \\ &= Y^TY - B^TX^TY - Y^TXB + B^TX^TXB \\ &= Y^TY - 2B^TX^TY + B^TX^TXB \end{aligned}$$

(here, the combined terms were 1 × 1 matrices.) Now differentiate: setting the derivative to 0 yields

$$0 = X^T X B - X^T Y$$

and thus

$$B = (X^T X)^{-1} X^T Y = X^{\dagger} Y$$

where $X^{\dagger}$ is called the pseudo-inverse of X.

The pseudo-inverse method generalizes to fitting any parametrized model to data (Section A1.9.3). The model should be chosen with some care. For example, Fig. A1.11 shows a disturbing case in which the model above (minimize vertical errors) is used to fit a relatively vertical swarm of points. The "best fit" line in this case is not the intuitive one.

A1.9.2 Principal Axis Method

The principal axes and moments of a swarm of points determine the direction and amount of its dispersion in space. These concepts are familiar in physics as the principal axes and moments of inertia. If a swarm of (possibly weighted) points is translated so that its center of mass (average location) is at the origin, a symmetric matrix M may be easily calculated whose eigenvectors determine the best-fit line or plane in a least-squared-perpendicular-error sense, and whose eigenvalues tell how good the resulting fit is.

Given a set $\{\mathbf{x}^i\}$ row of vectors with weights w^i, define their "scatter matrix" to be the symmetric matrix M, where $\mathbf{x}^i = (x_1^i, x_2^i, x_3^i)$:

$$M = \sum_i \mathbf{x}^{i^T} \mathbf{x}^i$$

$$M_{kp} = \sum_i x_k^i x_p^i \qquad 1 \leqslant k,\ p \leqslant 3$$

Define the dispersion of the $\mathbf{x}^i$ in a direction $\mathbf{v}$ (i.e., "dispersion around the plane whose normal is $\mathbf{v}$") to be the sum of weighted squared lengths of the $\mathbf{x}^i$ in the direction $\mathbf{v}$. This squared error E^2 is

$$E^2 = \sum_i w^i (\mathbf{x}^i \cdot \mathbf{v})^2 = \mathbf{v} \left(\sum_i w^i \mathbf{x}^{iT} \mathbf{x}^i\right) \mathbf{v}^T = \mathbf{v} M \mathbf{v}^T$$

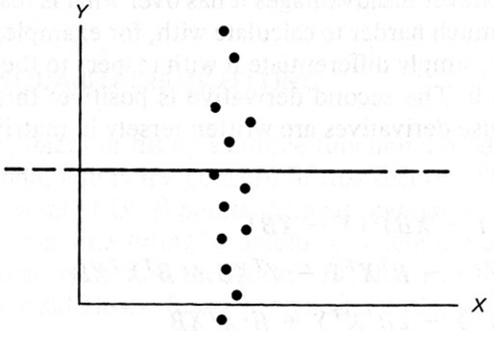

Fig. A1.11 A set of points and the "best fit" line minimizing error in Y.

To find the direction of minimum dispersion (the normal to the best-fit line or plane), note that the minimum of $\mathbf{v}M\mathbf{v}^T$ over all unit vectors $\mathbf{v}$ is the minimum eigenvalue λ_1 of M. If $\mathbf{v}_1$ is the corresponding eigenvector, the minimum dispersion is attained at $\mathbf{v} = \mathbf{v}_1$. The best fit line or plane of the points goes through the center of mass, which is at the origin; inverting the translation that brought the centroid to the origin yields the best fit line or plane for the original point swarm.

The eigenvectors correspond to dispersions in orthogonal directions, and the eigenvalues tell how much dispersion there is. Thus with a three-dimensional point swarm, two large eigenvalues and one small one indicate a planar swarm whose normal is the smallest eigenvector. Two small eigenvalues and one large one indicate a line in the direction of the normal to the "worst fit plane", or eigenvector of largest eigenvalue. (It can be proved that in fact this is the best-fit line in a least squared perpendicular error sense). Three equal eigenvalues indicate a "spherical" swarm.

A1.9.3 Fitting Curves by the Pseudo-Inverse Method

Given a function $f(\mathbf{x})$ whose value is known on n points $\mathbf{x}_1, \ldots, \mathbf{x}_n$, it may be useful is to fit it with a function $g(\mathbf{x})$ of m parameters $(b_1, \ldots, b_m)$. If the squared error at a point x_i is defined as

$$(e_i)^2 = [f(\mathbf{x}_i) - g(\mathbf{x}_i)]^2$$

a sequence of steps similar to that of Section A1.9.1 leads to setting a derivative to zero and obtaining

$$0 = G^T G\mathbf{b} - G^T\mathbf{f}$$

where $\mathbf{b}$ is the vector of parameters, $\mathbf{f}$ the vector of n values of $f(\mathbf{x})$, and

$$G = \frac{\partial \mathbf{g}}{\partial \mathbf{b}} = \begin{bmatrix} \dfrac{\partial g(x_1)}{\partial b_1} & \dfrac{\partial g(x_2)}{\partial b_2} & \cdots \\ \vdots & & \\ \cdot & \cdots & \dfrac{\partial g(x_n)}{\partial b_m} \end{bmatrix}$$

As before, this yields

$$\mathbf{b} = (G^T G)^{-1} G^T\mathbf{f}$$

Explicit least-squares solutions for curves can have nonintuitive behavior. In particular, say that a general circle is represented

$$\mathcal{G}(x, y) = x^2 + y^2 + 2Dx + 2Ey + F$$

this yields values of D, E, and F which minimize

$$e^2 = \sum_{i=1}^{n} \mathcal{G}(x_i, y_i)^2$$

for n input points. The error term being minimized does not turn out to accord with our intuitive one. It gives the intuitive distance of a point to the curve, but weighted by a factor roughly proportional to the radius of the curve (probably not desirable). The best fit criterion thus favors curves with high average curvature, resulting in smaller circles than expected. In fitting ellipses, this error criterion favors more eccentric ones.

The most successful conic fitters abandon the luxury of a closed-form solution and go to iterative minimization techniques, in which the error measure is adjusted to compensate for the unwanted weighting, as follows.

$$e^2 = \sum_{i=1}^{n} \left(\frac{f(x_i, y_i)}{|\nabla f(x_i, y_i)|} \right)^2$$

A1.10 CONICS

The conic sections are useful because they provide closed two-dimensional curves, they occur in many images, and they are well-behaved and familiar polynomials of low degree. This section gives their equations in standard form, illustrates how the general conic equation may be put into standard form, and presents some sample specific results for characterizing ellipses.

All the standard form conics may be subjected to rotation, translation, and scaling to move them around on the plane. These operations on points affect the conic equation in a predictable way.

Circle: r = radius $\quad x^2 + y^2 = r^2$

Ellipse: a, b = major, minor axes $\quad \dfrac{x^2}{a^2} + \dfrac{y^2}{b^2} = 1$

Parabola: $(p, 0)$ = focus, p = directrix $\quad y^2 = 4px$

Hyperbola: vertices $(\pm a, 0)$, asymptotes $y = \pm \left(\dfrac{b}{a}\right) x \quad \dfrac{x^2}{a^2} - \dfrac{y^2}{b^2} = 1$

The general conic equation is

$$Ax^2 + 2Bxy + Cy^2 + 2Dx + 2Ey + F = 0$$

This equation may be written formally as

$$(x \quad y \quad 1) \begin{bmatrix} A & B & D \\ B & C & E \\ D & E & F \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \mathbf{x}M\mathbf{x}^T = 0$$

Putting the general conic equation into one of the standard forms is a common analytic geometry exercise. The symmetric 3×3 matrix M may be diagonalized, thus eliminating the coefficients B, D, and E from the equation and reducing it to be close to standard form. The diagonalization amounts to a rigid motion that puts the conic in a symmetric position at the origin. The transformation is in fact the 3×3 matrix E whose rows are eigenvectors of M. Recall that if $\mathbf{v}$ is an eigenvector of M,

$$\mathbf{v}M = \lambda \mathbf{v}$$

Then if D is a diagonal matrix of the three eigenvalues, $\lambda_1, \lambda_2, \lambda_3$,

$$EM = DE$$

but then

$$EME^{-1} = DEE^{-1} = D$$

and M has been transformed by a similarity transformation into a diagonal matrix such that

$$\mathbf{x}D\mathbf{x}^T = 0$$

This general idea is of course related to the principal axis calculation given in Section A1.9.2, and extends to three-dimensional quadric surfaces such as the ellipsoid, cone, hyperbolic paraboloid, and so forth. The general result given above has particular consequences illustrated by the following facts about the ellipse. Given a general conic equation representing an ellipse, its center (x_c, y_c) is given by

$$x_c = \frac{BE - 2CD}{B^2 - 4AC}$$

$$y_c = \frac{2EA - BD}{B^2 - 4AC}$$

The orientation is

$$\theta = ½ \tan^{-1}\left(\frac{B}{A-C}\right)$$

The major and minor axes are

$$\frac{-2G}{(A + C) \pm [B^2 + (A - C)^2]^{½}}$$

where

$$G = F - (Ax_c^2 + B_{x_c y_c} + Cy_c^2)$$

A1.11 INTERPOLATION

Interpolation fits data by giving values between known data points. Usually, the interpolating function passes through each given data point. Many interpolation methods are known; one of the simplest is Lagrangean interpolation.

A1.11.1 One-Dimensional

Given $n + 1$ points (x_j, y_j), $x_0 < x_1 < \cdots < x_n$, the idea is to produce an nth-degree polynomial involving $n + 1$ so-called Lagrangean coefficients. It is

$$f(x) = \sum_{j=0}^{n} L_j(x) y_j$$

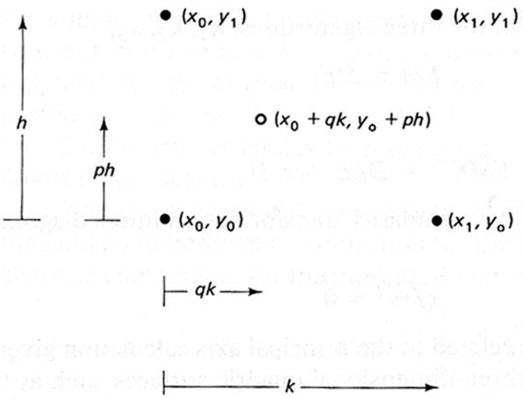

Fig. A1.12 Four point lagrangean interpolation on rectangular grid.

where $L_j(x)$ is the jth coefficient;

$$L_j(x) = \frac{(x-x_0)\ (x-x_1)\ \cdots\ (x-x_{j-1})\ (x-x_{j+1})\ \cdots\ (x-x_n)}{(x_j-x_0)\ (x_j-x_i)\ \cdots\ (x_j-x_{j-1})\ (x_j-x_{j+1})\ \cdots\ (x_j-x_n)}$$

Other interpolative schemes include divided differences, Hermite interpolation for use when function derivatives are also known, and splines. The use of a polynomial interpolation rule can always produce surprising results if the function being interpolated does not behave locally like a polynomial.

A1.11.2 Two-Dimensional

The four-point Lagrangean method is for the situation shown in Fig. A1.12. Let $f_{ij} = f(x_i, y_j)$. Then

$$f(x_0 + qk, y_0 + ph) = (1-p)\ (1-q)\ f_{00} + q\,(1-p)\ f_{10} + p\,(1-q)\ f_{01} + pqf_{11}$$

A1.12 THE FAST FOURIER TRANSFORM

The following routine computes the discrete Fourier transform of a one-dimensional complex array XIn of length N = $2^{\log N}$ and produces the one-dimensional complex array XOut. It uses an array W of the N complex Nth roots of unity, computed as shown, and an array Bits containing a bit-reversal table of length N. N, LogN, W, and Bits are all global to the subroutine as written. If the logical variable Forward is TRUE, the FFT is performed; if Forward is FALSE, the inverse FFT is performed.

```
SUBROUTINE FFT(XIn, KOut, Forward)
GLOBAL W, Bits, N, LogN
LOGICAL Forward
COMPLEX XIn, Xout, W, A, B
INTEGER Bits
ARRAY(0:N) W, Bits, XIn, XOut
```

```
DO (I = 0, N − 1) XOut(I) = XIn(Bits(I))
JOff = N/2
JPnt = N/2
JBk = 2
IOFF = 1
DO  (I = 1, LogN)
.     DO  (IStart = 0, N − 1, JBk)
.     .     JWPnt = 0
.     .     DO  (K = IStart, IStart + IOff − 1)
.     .     .     WHEN (Forward)
.     .     .     .     A = XOut(K + IOff) * W(JWPnt) + XOut(K)
.     .     .     .     B = XOut(K + IOff) * W(JWPnt + JOff) + XOut(K)
.     .     .     ...   FIN
.     .     .     ELSE
.     .     .     .     A = XOut (K + IOff) * CONJG(W(JWPnt)) + XOut(K)
.     .     .     .     B = XOut(K + IOff) * CONJG(W(JWPnt + JOff)) + XOu
.     .     .     ...   FIN
.     .     .     XOut(K) = A
.     .     .     XOut(K + IOff) = B
.     .     .     JWPnt = JWPnt + JPnt
.     .     ...   FIN
.     ...   FIN
.     JPnt = JPnt/2
.     IOff = JBk
.     JBk = JBk * 2
...   FIN
UNLESS (Forward)
.     DO  (I = 0, N − 1) XOut(I) = XOut(I)/N
...   FIN
END
```

```
TO INIT-W
.     Pi = 3.14159265
.     DO  (K = 0, N − 1)
.     .     Theta = 2 * Pi/N
.     .     W(K) = CMPLX(COS(Theta * K), SIN(Theta * K))
.     ...   FIN
...   FIN
```

```
TO BIT-REV
.     Bits(0) = 0
.     M = 1
.     DO  (I = 0, LogN − 1)
.     .     DO  (J = 0, M − 1)
.     .     .     Bits(J) = Bits(J) * 2
.   .   .   Bits(J + M) = Bits(J) + 1
.   .   ... FIN
.   .   M = M * 2
.   ... FIN
... FIN
```

A1.13 THE ICOSAHEDRON

Geodesic dome constructions provide a useful way to partition the sphere (hence the three-dimensional directions) into relatively uniform patches. The resulting polyhedra look like those of Fig. A1.13.

The icosahedron has 12 vertices, 20 faces, and 30 edges. Let its center be at the origin of Cartesian coordinates and let each vertex be a unit distance from the center. Define

$$t, \text{ the golden ratio} = \frac{1+\sqrt{5}}{2}$$

$$a = \frac{\sqrt{t}}{5^{1/4}}$$

$$b = \frac{1}{(\sqrt{t}\ 5^{1/4})}$$

$$c = a + 2b = \frac{1}{b}$$

$$d = a + b = \frac{t^{3/2}}{5^{1/4}}$$

$$A = \text{angle subtended by edge at origin} = \arccos(\frac{\sqrt{5}}{5})$$

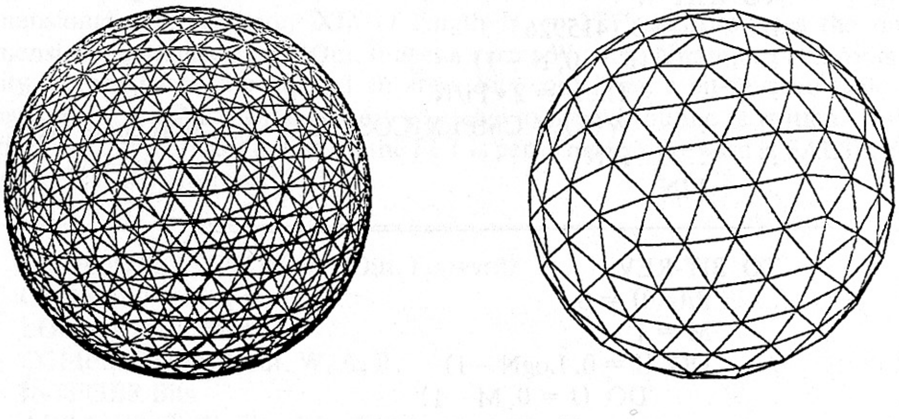
Fig. A1.13 Multifaceted polyhedra from the icosahedron.

Then

angle between radius and an edge $= b = \arccos(b)$
edge length $= 2b$
distance from origin to center of edge $= a$
distance from origin to center of face $= \dfrac{ta}{\sqrt{3}}$

The 12 vertices may be placed at

$$\begin{array}{ccc} (\;0, & \pm a, & \pm b) \\ (\pm b, & 0, & \pm a) \\ (\pm a, & \pm b, & 0) \end{array}$$

Then midpoints of the 20 faces are given by

$$\begin{array}{l} \tfrac{1}{3}(\pm \mathrm{d},\ \pm \mathrm{d},\ \pm \mathrm{d}) \\ \tfrac{1}{3}(\;0,\ \pm \mathrm{a},\ \pm \mathrm{c}) \\ \tfrac{1}{3}(\pm \mathrm{c},\ 0,\ \pm \mathrm{a}) \\ \tfrac{1}{3}(\pm \mathrm{a},\ \pm \mathrm{c},\ 0) \end{array}$$

To subdivide icosahedral faces further, several methods suggest themselves, the simplest being to divide each edge into n equal lengths and then construct n^2 congruent equilateral triangles on each face, pushing them out to the radius of the sphere for their final position. (There are better methods than this if more uniform face sizes are desired.)

A1.14 ROOT FINDING

Since polynomials of fifth and higher degree are not soluble in closed form, numerical (approximate) solutions are useful for them as well as for nonpolynomial functions. The Newton–Raphson method produces successive approximations to a real root of a differentiable function of one variable.

$$x^{i+1} = x^i - \frac{f(x^i)}{f'(x^i)}$$

Here x^i is the ith approximation to the root, and $f(x^i)$ and $f'(x^i)$ are the function and its derivative evaluated at x^i. The new approximation to the root is x^{i+1}. The successive generation of approximations can stop when they converge to a single value. The convergence to a root is governed by the choice of initial approximation to the root and by the behavior of the function in the vicinity of the root. For instance, several roots close together can cause problems.

The one-dimensional form of this method extends in a natural way to solving systems of simultaneous nonlinear equations. Given n functions F_i, each of n parameters, the problem is to find the set of parameters that drives all the functions to zero. Write the parameter vector $\mathbf{x}$.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{bmatrix}$$

Form the function column vector **F** such that

$$\mathbf{F}(\mathbf{x}) = \begin{bmatrix} F_1(\mathbf{x}) \\ F_2(\mathbf{x}) \\ \cdot \\ \cdot \\ \cdot \\ F_n(\mathbf{x}) \end{bmatrix}$$

The Jacobean matrix J is defined as

$$J = \begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} & \cdots & \frac{\partial F_1}{\partial x_n} \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ \frac{\partial F_n}{\partial x_1} & & \cdots & \frac{\partial F_n}{\partial x_n} \end{bmatrix}$$

Then the extension of the Newton–Raphson formula is

$$\mathbf{x}^{i+1} = \mathbf{x}^i - J^{-1}(\mathbf{x}^i)F(\mathbf{x}^i)$$

which requires one matrix inversion per iteration.

EXERCISES

A1.1 $\mathbf{x}$ and $\mathbf{y}$ are two two-dimensional vectors placed tail to tail. Prove that the area of the triangle they define is $|\mathbf{x} \times \mathbf{y}|/2$.

A1.2 Show that points $\mathbf{q}$ in a plane defined by the three points $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ are given by

$$q \cdot \left[(\mathbf{y}-\mathbf{x}) \times (\mathbf{z}-\mathbf{x})\right] = \mathbf{x} \cdot (\mathbf{y} \times \mathbf{z})$$

A1.3 Verify that the vector triple product may be written as claimed in its definition.

A1.4 Given an arctangent routine, write an arcsine routine.

A1.5 Show that the closed form for the inverse of a 2 × 2 A matrix is

$$\frac{1}{\det A} \begin{bmatrix} a_{22} & -a_{21} \\ -a_{12} & a_{11} \end{bmatrix}$$

A1.6 Prove by trigonometry that the matrix transformations for rotation are correct.

A1.7 What geometric transformation is accomplished when a_{44} of a geometric transformation matrix A varies from unity?

A1.8 Establish conversions between the given line representations.

A1.9 Write a geometric transform to mirror points about a given plane.

A1.10 What is the line-equation representation of a line $L1$ through a point **x** and perpendicular to a line $L2$ (similarly represented)? Parallel to $L2$?

A1.11 Derive the ellipse results given in Section A1.10.

A1.12 Explicitly derive the values of D, E, and F minimizing the error term

$$\sum_{i=1}^{n} [f(x_i, y_i)]^2$$

in the general equation for a circle

$$x^2 + y^2 + 2Dx + 2Ey + F = 0$$

A1.13 Show that if points and lines are transformed as shown in Section A1.7.6, the transformed points indeed lie on the transformed lines.

A1.14 Explicitly derive the least-squared-error solution for lines represented as $ax + by + 1 = 0$.

A1.15 If three planes intersect in a point, is the inverse of

$$\begin{bmatrix} \mathbf{p1} & \mathbf{p2} & \mathbf{p3} & 0 \\ & & & 0 \\ & & & 0 \\ & & & 1 \end{bmatrix}$$

guaranteed to exist?

A1.16 What is the angle between two three-space lines?

A1.17 In two dimensions, show that two lines **u** and **v** intersect at a point **x** given by $\mathbf{x} = \mathbf{u} \times \mathbf{v}$.

A1.18 How can you tell if two line segments (defined by their end points) intersect in the plane?

A1.19 Find a 4 × 4 matrix that transforms an arbitrary direction (or point) to lie on the Z axis.

A1.20 Derive a parametric representation for planes based on three points lying in the plane.

A1.21 Devise a scheme for interpolation on a triangular grid.

A1.22 What does the homogeneous point $(x, y, z, 0)$ represent?

REFERENCES AND FURTHER READING

Computer Graphics

1. NEWMAN, W. M., and R. F. SPROULL. *Principles of Interactive Computer Graphics*, 2nd Ed. New York: McGraw-Hill, 1979.
2. CHASEN, S. H. *Geometric Principles and Procedures for Computer Graphic Applications.* Englewood Cliffs, NJ: Prentice-Hall, 1978.

3. FAUX, I. D., and M. J. PRATT. *Computational Geometry for Design and Manufacture*, Chichester, UK: Ellis Horwood Ltd, 1979.

4. ROGERS, D. F., and J. A. ADAMS. *Mathematical Elements for Computer Graphics*. New York: McGraw-Hill 1976.

Computer Vision

5. HORN, B. K. P. "VISMEM: Vision Flash 34." AI Lab, MIT, December 1972.

6. SOBEL, I. "Camera models and machine perception." AIM-21, Stanford AI Lab, May 1970.

7. DUDA, R. O. and P. E. HART. *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.

8. ROSENFELD, A., and A. C. KAK. *Digital Picture Processing*. New York: Academic Press, 1976.

9. PAVLIDIS, T. *Structural Pattern Recognition*. New York: Springer-Verlag, 1977.

Geometry, Calculus, Numerical Analysis

10. WEXLER, C. *Analytic Geometry: A Vector Approach*. Reading, MA: Addison Wesley, 1962.

11. APOSTOL, T. M. *Calculus.*, Vol. 2, Waltham, MA: Blaisdell, 1962.

12. CONTE, S. D. and C. DEBOOR. *Elementary Numerical Analysis: An Algorithmic Approach*. New York: McGraw-Hill, 1972.

13. RALSTON, A. *A First Course in Numerical Analysis*. New York: McGraw-Hill, 1965.

14. ABRAMOWITZ, M, and I. A. STEGUN. *Handbook of Mathematical Functions*. New York: Dover, 1964.

15. HODGEMAN, C. D. (Ed.). *CRC Standard Mathematical Tables*. West Palm Beach, FL: CRC Press.

Geodesic Tesselations

16. BROWN, C. "Fast display of well-tesselated surfaces." *Computers and Graphics 4*, 1979, 77–85.

17. CLINTON, J. D. "Advanced structural geometry studies, part I: polyhedral subdivision concepts for structural applications," NASA CR-1734/35, September 1971.

Fast Transformations

18. PRATT, W. K. *Digital Image Processing*. New York: Wiley-Interscience, 1978.

19. ANDREWS, H. C., and J. KANE. "Kronecker matrices, computer implementation, and generalized spectra." *J. ACM*, April 1970.