Behavior Research Methods, Instruments, & Computers
1988, 20 (4), 372-389

A method for calibrating the spatial coordinates of a visual display to high accuracy

LAURENCE T. MALONEY and KYUNGHEE KOH
University of Michigan, Ann Arbor, Michigan

High-resolution CRT displays are subject to *geometric distortion*: lines that are straight in the internal coordinates of the graphics software and hardware are curved when projected onto a plane in the observer's line of sight. As the available resolution of CRT screens increases, it becomes more difficult to measure and correct for this distortion. We present a simple, highly accurate method for determining the mapping between internal coordinates and the viewing plane. It requires that an observer, using a calibration program, adjust triples of displayed points until they are collinear. A metal straightedge placed between the observer and the screen aids in this judgment. We describe the calibration of an IBM Enhanced Graphics Display in high-resolution mode (350×640 pixels), and illustrate how to use the estimated mapping to choose internal coordinates to draw undistorted figures that are accurate to within 0.5 pixel. The method can be used to assess or to correct the accuracy of visual displays. The method is relevant to experiments in spatial vision, spatial perception, perception of dot patterns, and any application in which geometrically accurate stimuli are required.

Consider the viewing geometry of Figure 1. The observer, whose head is immobilized by a chinrest or bite bar, views a CRT screen that is behind a large white surround. With proper choice of lighting, the observer sees only the information displayed and cannot see the (curved) surface of the screen or other details of the CRT. For the experimenter's purposes, the CRT information is effectively projected onto a plane perpendicular to the line of sight, the *viewing plane*. Suppose a grid pattern of lines that are horizontal and vertical in the internal coordinate system used to address locations on the CRT is then displayed.[1] The straight, orthogonal lines typically correspond to loci in the viewing plane of the observer that are neither straight nor orthogonal. The deviations from linearity and orthogonality represent *geometric distortion* (Rosenfeld & Kak, 1976, pp. 175-179).

Figure 2, for example, shows loci of points in internal coordinates that all map to the left edge or to the top edge of a rectangle in the viewing plane that occupies about 90% of the vertical and 90% of the horizontal extent of the addressable area of an IBM Enhanced Graphics Display (EGD) in high-resolution mode. The rectangle is shown in heavy black. The two plots are drawn next to the part of the rectangle to which they correspond. The points were aligned on the viewing plane with a straightedge adjusted to be horizontal or vertical. If no geometric distortion were present, the internal coordinates for the left and top edge would lie on straight vertical and horizontal lines, respectively. The largest error found in mapping the edges of the rectangle is 6 pixels in the horizontal direction; the largest in the vertical direction is 4 pixels. The degree of distortion is small but readily visible, the two edges are nonorthogonal in internal coordinates, and the top edge is visibly curved. It is these small but readily visible distortions across the entire screen that we sought to eliminate.

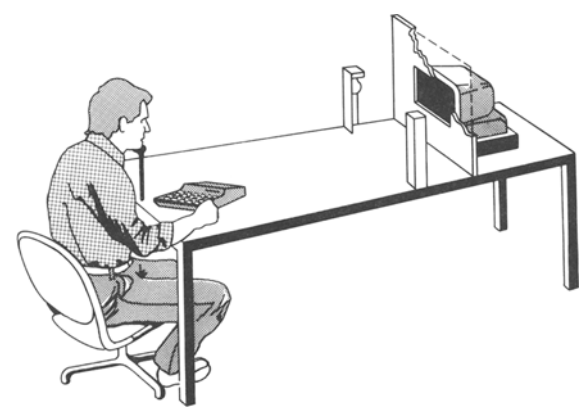

Figure 1. The observer viewing monocularly with a chinrest sees points on the CRT screen behind the brightly lit baffle but cannot see the curved surface of the CRT screen. The points are effectively projected onto a *viewing plane* perpendicular to the line of sight toward the CRT.

For many experimental purposes, such distortions may be tolerated. Alternatively, stimuli can be confined to a small region of the screen that is "flat." When the experimental stimuli require the full extent and resolution of the screen, however, the distortion encountered in map-

This research was supported in part through funds provided to the first author by the University of Michigan. We are especially grateful to Robert A. Mayans of AT&T Bell Laboratories for providing proofs of the two propositions in the text. Requests for reprints should be sent to Laurence T. Maloney, Department of Psychology, New York University, 6 Washington Place, New York, NY 10003.

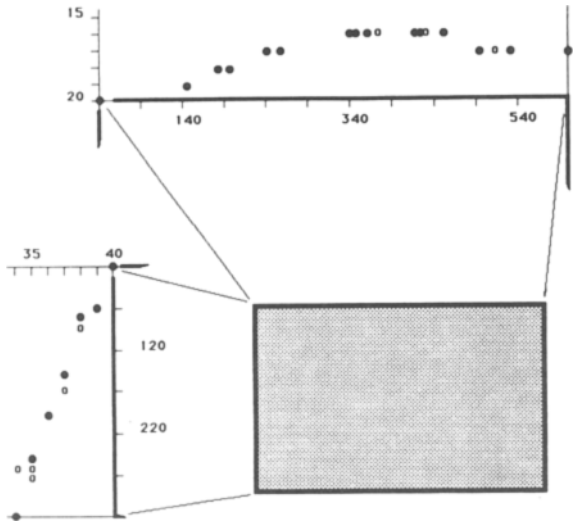

Figure 2. Plots of the internal coordinate pairs that map to the left and top edges of a rectangle in the viewing plane that occupies about 90% of the vertical and 90% of the horizontal extent of the addressable area of the IBM Enhanced Graphics Display (EGD) used. The vertical internal coordinate axis on the IBM EGD runs from top to bottom as shown. The upper left corner of the rectangle corresponds to the internal coordinates (20,40). The two plotting symbols identify the two observers.

ping internal coordinates to the viewing plane must be measured and countered.

If the locations at which points are actually displayed on the screen can be measured, the geometric distortion can be countered as follows: Establish an external coordinate system in the viewing plane of the observer. Accurately measure the position of many points with known internal coordinates (a test pattern) in external coordinates. *Geometric distortion functions* (usually polynomials) that map the internal coordinates to the external coordinates are fit. The geometric distortion functions are of the form

$$u = D^u(x,y)$$
$$v = D^v(x,y), \qquad (1)$$

and they indicate at what location the point (x,y) in internal coordinates will map in external coordinates (u,v). The two maps D^u and D^v can be thought of as a single vector-valued function:

$$(u,v) = \bar{D}(x,y). \qquad (2)$$

Rosenfeld and Kak (1976) described how to use accurate measurements of external coordinates to fit the function $\bar{D}$. Once an estimate of $\bar{D}$ is available, geometric correction is straightforward. All figures to be displayed are generated in terms of external u,v coordinates. For each point (u,v), an internal point (x,y) is found that maps as closely as possible to (u,v) by inverting $\bar{D}$:

$$(x,y) = \bar{D}^{-1}(u,v). \qquad (3)$$

$\bar{D}^{-1}$ is called the *geometric correction function*. The computed internal coordinates are displayed through the usual graphics software/hardware. The geometric correction function and the geometric distortion function cancel, and the point appears precisely where desired (up to the resolution of the internal coordinates).

The inverse of the function $\bar{D}$ at (u,v) can be computed by any of a number of numerical techniques. In particular, the problem of inverting $\bar{D}$ can be reduced to the equivalent problem of finding a root (x,y) of the function $\bar{D}(x,y) - (u,v)$, since, of course, this function is 0 only if $\bar{D}(x,y) = (u,v)$. The program INVERT2 (see Appendix A) uses a generalization of the Newton-Raphson root-finding method to compute $\bar{D}^{-1}$ when the geometric distortion function $\bar{D}$ is specified by two quadratic polynomials in (x,y). INVERT2 is easily generalized to compute $\bar{D}^{-1}$ for any differentiable $\bar{D}$ as described by Press, Flannery, Teulosky, and Vetterling (1986, pp. 269-272).

For example, suppose we wish to draw $n+1$ points that are equally spaced along the quarter circle of radius r centered on (0,0). In (u,v) coordinates, the coordinates of the points are computed as follows:

$$u_i = r\cos\frac{2\pi i}{n}, \; i = 0,n$$

$$v_i = r\sin\frac{2\pi i}{n}.$$

Next, the corresponding internal coordinates (x_i,y_i) corresponding to each (u_i,v_i) pair are computed using Equation 3 and a program similar to INVERT2. Then the internal coordinates (x_i,y_i) that lie on a geometrically corrected circle are displayed.

The problematic step in the use of this method is the estimation of $\bar{D}$, the geometric distortion function. To measure (u,v) directly for any (x,y) requires measurement of the absolute location of points displayed in the view plane. The higher the resolution available for a given screen size, the more accurate the measurement must be to get full benefit from the added resolution. The physical spacing between pixels in the horizontal directions on an IBM EGD, for example, is about 0.4 mm. Accurate calibration of the screen by the methods outlined by Rosenfeld and Kak (1976) requires that one measure absolute location in the external view plane to comparable resolutions. In attempting to calibrate an IBM EGD, we could not achieve the level of accuracy required with the equipment available to us. We attempted, for example, to photograph test patterns on the CRT screen and make calibration measurements on enlargements of the photographs. The photographs proved to be too distorted (by the optics of the camera used) to permit calibration by this method.

In this article, we describe a novel method that does not require measurements of position or separation and that permits accurate estimation of $\bar{D}$ and $\bar{D}^{-1}$ to the limits of accuracy of the graphic device considered (0.5 pixel). The method requires only a metal straightedge. With experience, initial calibration measurements of a CRT can

be completed in 2 h. An abbreviated procedure that takes under 30 min can then be used to monitor the calibration of the CRT to see whether it drifts over time. We present the results of our calibrations of an IBM EGD, one of the more common types of CRT available. The calibration method is based on mathematical results that are summarized in a later section. We also describe the method for testing the results of the calibration procedure to determine the accuracy and stability of the estimated geometric distortion function.

All of the estimation and correction algorithms described in this article are implemented in FORTRAN 77. Listings of the programs are available from the authors.[2]

The Calibration Method

To begin calibration, an observer must be positioned at a particular vantage point, preferably with a chinrest or headrest to stabilize viewing. The estimated distortion function applies only for that vantage point. If the surface of the screen is curved, lines that appear straight when viewed from straight ahead will generally be curved if viewed from an off-axis vantage point. We performed all calibrations with two observers, both viewing the screen monocularly. The method requires only that an observer adjust points displayed on the viewing plane to lie along a straightedge interposed between the observer and the CRT. No judgments of distance between points are needed. The calibration data are used to estimate the geometric distortion function $\bar{D}$.

The observer's subjective judgments of "straightness" are not required in these measurements. The only requirement is that the observer bring the points into alignment with an objective criterion of straightness, the straightedge. If the observer wears markedly thick glasses or has vision uncorrected for astigmatism, he/she may actually perceive the aligned points as noncollinear. The coincident straightedge will be seen as curved to the same degree. Whether the observer perceives the final alignment as collinear is irrelevant.

The geometric function $\bar{D}$ can be thought of as being composed of two components, the linear distortion and the nonlinear distortion. If $\bar{D}$ consisted only of linear distortion, then lines in internal coordinates would map to lines in external coordinates, but orthogonal lines in internal coordinates need not map to orthogonal lines in external coordinates. Figure 3a illustrates linear distortion. A linear distortion of a flat screen is equivalent to viewing the screen from some other viewpoint: it induces an affine transformation on the external coordinates. Figure 3b illustrates nonlinear distortion: lines in internal coordinates are curved in external coordinates. If the screen is flat, then the nonlinear component will be corrected by the calibration procedure for all vantage points, but of course the linear component will change with viewing perspective.

We describe two types of calibration measurement, which we term Procedure 1 and Procedure 2. Procedure 1 gathers the information needed to estimate the nonlinear component of $\bar{D}$, and Procedure 2 gathers the information needed to determine the linear component. It is easy to estimate the linear component: as few as two measurements using Procedure 2 are needed. The bulk of the measurements taken will therefore be of the first kind. Both measurements can be performed by means of a simple calibration program, called CALIB, whose capabilities we describe next.

CALIB displays three points (pixels) on the CRT screen. The observer is able to individually move the three points around the screen. We used the cursor pad of the IBM PC/AT to which our CRT was attached. A joystick, if available, can also be used. The observer must be able to select any one of the three points and move it to any position on the screen. At any point, the observer must be able to signal the computer to record the locations of the three points on an output file. These recorded positions are the calibration information.

For Procedure 2, only two points are actually needed, and one of the two will never be moved. Therefore, a modified version of CALIB was used for Procedure 2 (although we could have used CALIB).

Procedure 1. A straightedge is placed near the screen between the screen and the observer. We positioned the straightedge by simply clamping it to two 2×4 boards that were clamped in turn to each other. The resulting complex could easily be slid about to reposition the straightedge.

Once the straightedge is positioned, the observer uses the program CALIB to position the three points along the straightedge as accurately as possible. We found that the physical width of a single point displayed in isolation (the *point-spread* of the CRT) was several pixels wide, and that we could use the edge of the straightedge to bisect the point and reduce its apparent intensity. If one point of the three was much dimmer or brighter than the others, it was not an equivalent distance from the straightedge. We therefore used brightness cues to check that we had successfully aligned the three points. Accuracy using combined cues was superior to the resolution of the screen. We determined experimentally that observers could make the required judgments of collinearity with high reliability (< 0.6 sec of arc). (We were able to repeatedly reposition a point at exactly the same horizontal position along the straightedge for a given vertical position.)

The only requirements regarding the position and angle of the straightedge are that at least three significantly different angles be used and the positions should more or less cover the screen. The exact angles and positions of the straightedge are not important to this method; they may be set by eye without ruler or protractor. Each triple should span the width and/or height of the screen. The triples of collinear points should be uniformly distributed across the screen. The basis for these claims are outlined below when we present the mathematical results underlying this method.

We made 145 collinearity judgments in order to provide data to estimate the nonlinear component of the geo-

metric distortion $\bar{D}$. These judgments indicate nothing about the linear component, because a linear transformation preserves collinearity. Put another way, the Procedure 1 measurements tell about the distortion of the screen up to a linear transformation. If we correct for these components, we only guarantee that the unit square in corrected coordinates would map to a parallelogram, as in Figure 3a. What must be done next is to pin down the unit square of the corrected coordinates.

First, the positions of two points are defined. These are not measurements, but rather arbitrary choices. Basically they pin down, say, the upper left and lower right corners of the unit square. We assert that the internal point (x_0,y_0) will map to the external point (u_0,v_0) and that the distinct internal point (x_1,y_1) will map to the external point (u_1,v_1). The two x-coordinates must be different, the two u-coordinates must be different, the two y-coordinates must be different, and the two v-coordinates must be different. Any two points subject to these restrictions can be selected, but it is best to pick points near the edges of the screen that are widely separated in both the x- and y-dimensions. Figure 4 shows our choice of two points for the IBM EGD. We set up our external (u,v) coordinates so that the specific point (x_0,y_0) is mapped to (0,0) and a second point (x_1,y_1) is mapped to (1,1). To complete the calibration, only two more points need to be specified, as shown in Figure 4. The horizontal and vertical directions are defined by specifying a point directly below (u_0,v_0) and one on a level with (u_1,v_1). More formally, we specify an internal point (x_2,y_2) that maps to the external point (u_0,v_2) and an internal point (x_3,y_3) that maps to the external point (u_3,v_1). Procedure 2 is used to obtain two such points.

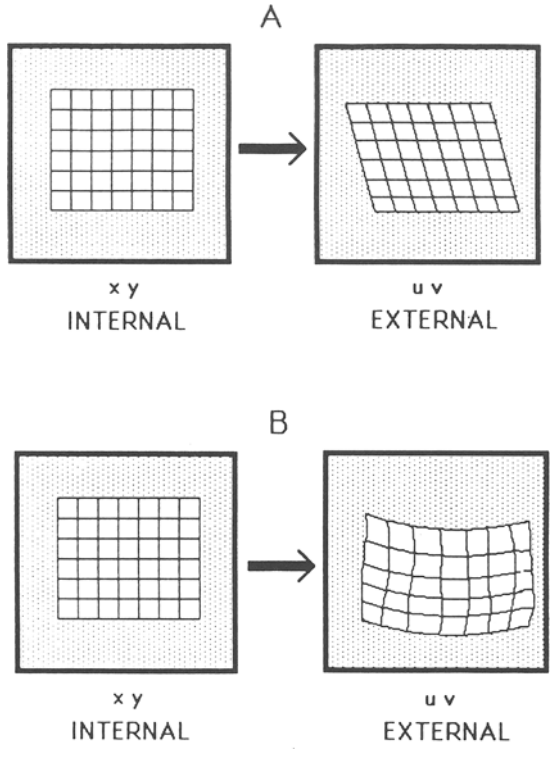

Figure 3. Linear and nonlinear geometric distortion. The left-hand figures are in internal coordinates, the right-hand figures in external coordinates. The top pair of plots exhibits a purely linear distortion in passing from internal to external coordinates. The bottom pair exhibits a nonlinear distortion.

Procedure 2. The program CALIB must first be set up to display one point at (u_0,v_0) by displaying the point (x_0,y_0) in internal coordinates. With the aid of an assistant who moves a vertical straightedge, the observer positions the straightedge in front of this point. (We used a large carpenter's square with one side set on a reference surface [a table] to define vertical.) The observer then adjusts a second point using CALIB to be directly below the first using the straightedge aligned with the first point. This gives (x_2,y_2). Then (x_3,y_3) is estimated analogously using a horizontal reference. These measurements may be repeated to guard against error, but only two measurements are actually needed. Figure 2 shows the internal coordinates of the many measurements made by two observers in our calibrations of the IBM EGD. We ultimately used only two of these. Note that we do not know (or need to know) v_2 or u_3. We know only information about collinearity and orthogonality plus the coordinates we have defined. No length measurements or measurements of absolute position have been taken. The two kinds of information described are all the information needed to estimate $\bar{D}$, as described in the next section.

Mathematical Basis of the Method

The mathematical justification for this method is summarized in the following propositions.[3]

Proposition 1. Let D: $\mathbf{R}^2 \rightarrow \mathbf{R}^2$ be continuous and one-to-one. Suppose D maps any set of collinear points to collinear points. Then D is a linear (affine) mapping.

Proposition 2. Let D: $\mathbf{R}^2 \rightarrow \mathbf{R}^2$ be continuous and one-to-one. Let θ_i $(i = 1, 2, 3)$ be distinct angles in $(0,\pi)$. Suppose D maps lines of angle θ_1 to lines of angle ϕ_1, lines of angle θ_2 to lines of angle θ_2, and lines of angle θ_3 to lines of angles ϕ_3, where the ϕ_i is also a distinct angle in $[0,\pi)$. Then D is a linear (affine) mapping.

The first proposition tells us that all possible collinearity measurements taken together determine the nonlinear part of D. The second proposition is the basis for the claim above that we must take data at at least three different angles.

These results do not guarantee that any finite set of measurements will result in a reliable estimate of the distortion function of a specific CRT. We have to verify that any estimated distortion function is accurate and stable, as described below. These results simply motivate the design of a measurement technique based on judgments of collinearity.

Estimation of the Geometric Distortion Function

Rosenfeld and Kak (1976) outlined how to approximate $\bar{D}$ by polynomials in two variables by fitting measurements

of the locations of known points displayed on the screen. We do not have such measurements, but rather information concerning collinearity and orthogonality. Polynomial approximations to D^u and D^v are fitted, subject to the constraints that they preserve measured collinearity and orthogonality (recall that D^u and D^v are the two scalar functions that together make up the vector function $\bar{D}$). In this section, we describe the estimation procedure. To make the presentation clearer, assume that the two reference points (x_0,y_0) and (x_1,y_1) are mapped to (0,0) and (1,1), as suggested in Figure 4. We used both quadratic and cubic polynomials in x and y to approximate D^u and D^v. The cubic approximating functions were denoted as follows:

$$\begin{aligned} D^u(x,y) &= a_0+a_1x+a_2y+a_3x^2+a_4xy+a_5y^2+a_6x^3 \\ &\quad +a_7x^2y+a_8y^2x+a_9y^3 \\ D^v(x,y) &= b_0+b_1x+b_2y+b_3x^2+b_4xy+b_5y^2 \\ &\quad +b_6x^3+b_7x^2y+b_8y^2x+b_9y^3. \end{aligned} \tag{4}$$

The quadratic equations are simply the cubic equations with the last four coefficients set to zero. The cubic equations have a total of 20 unknown coefficients $(a_0,\ldots,a_9)$ and $(b_0,\ldots,b_9)$; the quadratic have a total of 12. Two of the coefficients can be removed in either set of equations by a simple trick. The choice of the origin in the internal coordinate system is built into the graphics hardware/software interface. We can, however, establish by convention that the fixed reference point (x_0,y_0) is the internal origin (0,0). We need only remember to add the offsets x_0 and y_0 to any internal coordinates before display. Then we choose (u_0,v_0) to be (0,0), which can always be done as we may locate the origin in the external coordinate system wherever we wish. As a consequence, the polynomial $\bar{D}$ maps (0,0) internal to (0,0) external and it is easy to verify that a_0 and b_0 are always 0 and need not be estimated. Only the 10 (quadratic) or 18 (cubic) remaining coefficients need to be estimated.

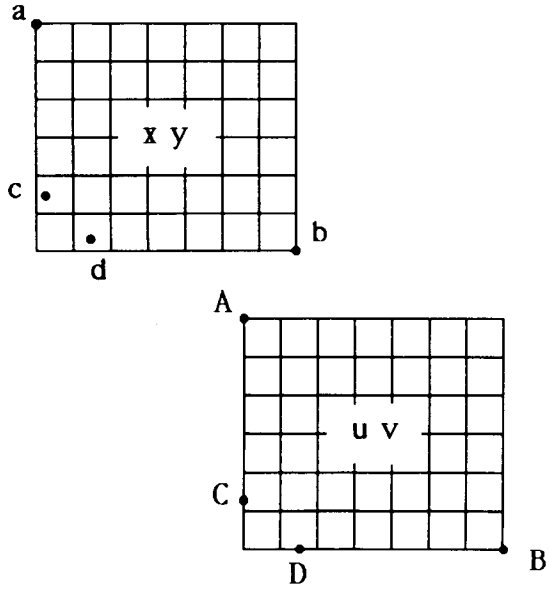

Figure 4. The points defined and measured to pin down the linear component of the distortion function. Points A and B in internal coordinates are declared to be the points (0,0) and (1,1) in external coordinates. Point C is aligned to be directly below point A (using Procedure 2), and point D is aligned to be on the same level as B. These points define a coordinate system.

The estimates of the unknown coefficients are chosen to minimize a *penalty function* that punishes failures to map points measured to be collinear to collinear points, fixed points to corresponding fixed points, and points measured to be directly above or to the side of each other to points that are so. The penalty function P will be a function of the choice of polynomials D^u and D^v and the calibration measurements. It is denoted as $P(\bar{D})$. We seek to find the polynomials $\bar{D}$ with coefficients that minimize $P(\bar{D})$.

Appendix B contains one of the estimation programs, EST2, as an example. EST2 is written in FORTRAN 77. It implements the estimation procedure in the quadratic case and uses the downhill simplex method described in Press et al. (1986, pp. 289-293) to minimize the penalty function. The primary virtue of this method is that it is extremely easy to implement. We also estimated polynomial approximations to D^u and D^v using the program STEPIT (Chandler, 1965).

We chose to use two different methods so that we might compare their performances. STEPIT's performance in simulations was comparable to that of the downhill simplex method except that the latter tended to become trapped in local minima more often than did STEPIT. As with any function minimization, we recommend computing the solution to the minimization several times with different initial values. Many common math packages for mainframe computers have function minimization programs that may be suited to this problem. EST2 is easily adapted to use other minimization programs, including STEPIT.

The penalty function P has two components:

$$P(\bar{D}) = C(\bar{D})+\beta O(\bar{D}). \tag{5}$$

C penalizes noncollinearity, and O represents the penalty for failing to fit the fixed points. β is a weight that determines the relative importance of the two kinds of constraints. Since the value of C is unaffected by a linear transformation and the value of O can be set to zero by a linear transformation, the value of β cannot affect the global minimum. β affects only the search process, speeding it or slowing it. We chose to make β equal to 10,000.

Next, the two constraints in the penalty equation are defined. The term O is easily specified. We want (x_0,y_0) to map to (0,0), (x_1,y_1) to map to (1,1), (x_2,y_2) to map to (0,?), and (x_3,y_3) to map to (?,1), where "?" represents "don't care." The corresponding penalty is

$$\begin{aligned} O(\bar{D}) &= D^u(x_0,y_0)^2 + D^v(x_0,y_0)^2 \\ &\quad +[D^u(x_1,y_1)-1]^2 + [D^v(x_1,y_1)-1]^2 \\ &\quad +D^u(x_2,y_2)^2 + (D^v(x_3,y_3)-1)^2. \end{aligned} \tag{6}$$

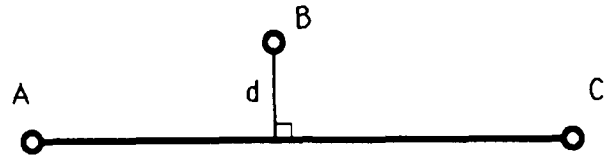

Figure 5. Three noncollinear points A, B, and C. The noncollinearity penalty is d^2.

The term C, which penalizes noncollinearity, is then defined. Figure 5 illustrates our choice of noncollinearity penalty. For any set of three points, the deviation from collinearity is the line segment of length d. The square of d is the desired measure of noncollinearity for this triple.[4] Suppose that the three points are $A = (x_a,y_a)$, $B = (x_b,y_b)$, and $C = (x_c,y_c)$ as shown in Figure 5, with B between the other two. Let $X_b = x_b - x_a, Y_b = y_b - y_a$, $X_c = x_c - x_a$, and $Y_c = y_c - y_a$. Let $N_b = \sqrt{X_bX_b + Y_bY_b}$, the length of the line denoted AB, and $N_c = \sqrt{X_cX_c + Y_cY_c}$, the length of the line denoted AC. Then

$$\cos\theta = \frac{X_bX_c + Y_bY_c}{N_bN_c}$$

and

$$d = N_b \sin\theta.$$

Suppose there are M triples indexed by $j = 1,\ldots,M$, each of which has penalty d_j. Then we define

$$C(\bar{D}) = \sum_{j=1}^{M} d_j^2.$$

The procedure for estimating the coefficients a_i,b_i involves writing a computer program that minimizes P for a given set of calibration data by choice of coefficients. As noted above, any of a number of standard function minimization programs can be used to find the coefficients that minimize P for the given calibration data. Appendix B contains an example of such a program.

We also performed simulations of the fitting process, not reported here, to determine how many calibration measurements were needed to reliably fit a quadratic or cubic geometric distortion function, given the level of distortion we encountered for the IBM EGD. One hundred calibration measurements were sufficient to fit quadratic geometric distortion functions similar to the one we estimated. About 140 suffice for the cubic. These values vary with the degree and kind of distortion present.

Results

The IBM EGD was attached to an IBM PC/AT. The observer's station from which calibration measurements were taken was a distance of 2.07 m from the middle of the screen. An illuminated white surround, as shown in Figure 1, rendered the pronouncedly curved glass surface of the EGD not visible. The observer saw only bright white points against a black background that had a white surround. The CRT was run in high-resolution mode, 350×640 pixel resolution. At the calibration distance, a horizontal step was on average 38 sec of arc and a vertical step was 50 sec of arc. The straightedge was placed at the distance of the white surround, 1.83 m from the observer. Both observers had corrected-to-normal vision.

The vision of each observer was tested and corrected to 20/15 or better by an optometrist in the week preceding the first calibration measurements. Both observers were slightly myopic and wore glasses throughout calibration. As noted above, distortions introduced by glasses or visual defects do not affect the accurary of the method. Both observers together made 145 calibration measurements using Procedure 1 and 64 calibration measurements using Procedure 2. Only two measurements using Procedure 2 were actually needed.

The remaining measurments taken using Procedure 2 allowed us to assess the reliability of each observer and the agreement between them. As the data in Figure 2 indicate, both observers were accurate to the pixel. The two observers agreed in their estimates to within 1 pixel. No better agreement is to be expected.

We fit quadratic and cubic polynomials to the calibration data (145 collinear triples) to estimate the geometric distortion function. Two methods were used to assess the quality of fit of the geometric distortion.

1. *Refitting.* The collinearity data on which the estimates were based were refit. If the geometric distortion function accurately captured the distortion present, then all triples of collinear points used in the estimation procedure should be rendered collinear. We consequently wrote evaluation programs EVAL2 (quadratic) and EVAL3 (cubic), which took each triple of measured collinear points and transformed them by means of the estimated geometric distortion function. We then computed (in pixel units) how far the transformed middle point was from perfect collinearity. For each calibration measurement, there can be as many as two errors, one in the horizontal coordinate, the other in the vertical. The number and magnitude of errors is the measure of goodness of fit. Because of the discrete nature of the internal (pixel) coordinates, a small deviation in estimation or correction can create a 1-pixel error in the x- or y-coordinate due to "rounding." The best indication of a good fit then is an absence of errors greater than 1 pixel and no obvious spatial pattern in the occurrence of 1-pixel errors across the extent of the CRT.

The quadratic polynomial rendered 123 of the 145 collinear triples exactly collinear. There were no errors greater than 1 pixel. We could discern no pattern in the location on the CRT of the 22 remaining 1-pixel errors. The cubic polynomial improved upon this fit by reducing the number of 1-pixel errors to 19. The lack of pattern in the quadratic errors, together with simulation results indicating that the 22 1-pixel errors are consistent with the number of errors due to rounding, suggest that the quadratic fit is adequate.

2. *Remeasurement.* The estimated geometric distortion functions are of value to the extent that they can be used to correct novel data, data not used in the process of esti-

mation. We took two additional sets of collinearity measurements: 28 after 18 days and 26 after 56 days. We took these additional measurements over a period of several weeks in order to assess the stability over time of the geometric distortion. We applied EVAL2/EVAL3 to these measurements. The number and magnitude of errors were again the measure of goodness of fit.

The fit was superb. A geometric correction function based on the quadratic perfectly corrected 54 out of 54 of the new x-coordinates and 53 out of 54 of the new y-coordinates. The remaining y-coordinate was in error by 1 pixel. For the cubic, there was also only a single 1-pixel error across both sets. Both estimated polynomials were able to correct independent data with high precision. On the basis of these results we would have no reason to prefer the cubic to the quadratic.

Summary

The mapping between the internal Cartesian coordinates specified by graphics hardware and software and position on a CRT screen typically exhibits some degree of geometric distortion. Other methods for estimating this distortion rely on precise measurements of position of points on the display. As the available resolution of CRT screens increases, it becomes increasingly more difficult and expensive to measure the distortion in this mapping by these methods.

We present a simple method for determining the mapping between internal coordinates and the viewing plane that is highly accurate and requires only that an observer, using a calibration program, adjust triples of displayed points until they are collinear. Human observers are extremely accurate at this alignment task. No measurements of absolute position are needed.

We describe the calibration of an IBM EGD in high-resolution mode (350×640 pixels). We illustrate how to use the estimated mapping to choose internal coordinates to draw undistorted figures that are accurate to within 0.5 pixel. We test the accuracy of the calibration empirically and outline the mathematical results that underlie it. This method should readily extend to other CRT displays and flat-panel displays (see Tannas, 1985) that experience geometric distortion.

REFERENCES

CHANDLER, J. P. (1965). STEPIT. *Quantum Chemistry Program Exchange*. Bloomington: Indiana University, Department of Chemistry.

ROSENFELD, A., & KAK, A. C. (1976). *Digital picture processing*. New York: Academic Press.

PRESS, W. H., FLANNERY, B. P., TEULOSKY, S. A., & VETTERLING, W. T. (1986). *Numerical recipes: The art of scientific computing*. Cambridge, England: Cambridge University Press.

TANNAS, L. E., JR. (Ed.). (1985). *Flat-panel displays and CRTs*. New York: Van Nostrand Reinhold.

NOTES

1. The internal coordinate system is the coordinate system used by the graphics software on the computer, the x,y-coordinates referenced in graphics commands. We consistently use x,y in referring to internal coordinates, and u,v in referring to external coordinates (described below). The IBM EGD software and documentation available to us refers to the horizontal axis as the y-axis and the vertical axis as the x-axis. The vertical (x) axis increases in going from top to bottom. We follow these conventions in description.

2. Requests for listings of the programs should be sent to Laurence T. Maloney, Department of Psychology, New York University, 6 Washington Place, New York, NY 10003.

3. We have not located published proofs of these propositions. Robert A. Mayans of AT&T Bell Laboratories has proven both of these propositions.

4 .We can get three different measures of noncollinearity, since any of the three points can be considered as the deviant projecting onto the line determined by the other two points. In practice, if triples of collinear points are spaced appreciable distances across the screen, one of these noncollinearity measures will be small and the others large, and it suffices to take the minimum as the penalty. We took a different approach. In taking calibration measurements, we made sure that the second of the three points recorded was always the point that had the smallest deviation from the line determined by the other two. This convention simplified the computation of this measure.

APPENDIX A
INVERT2

```
c ------------------------------------------------------------------------
c ------------------------------------------------------------------------
c
c       invert2 quadratic version.      Copyright (1988) Laurence T. Maloney
c
c       Find x,y such that f(x,y) = (u,v) where f is the quadratic
c       specified  by subroutines fx, fy and commons xcoeff, ycoeff.
c       The method used is a two-dimensional extension of Newton-Raphson
c       root-finding applied to the function f(x,y)-(u,v). For details,
c       see:
c
c       Press, W. H., Flannery, B. P., Teulosky, S. A., & Vetterling, W. T.,
c       NUMERICAL RECIPES; THE ART OF SCIENTIFIC COMPUTING.
c       (Cambridge, England: Cambridge University Press, 1986).
c
c       Tolerance values for proximity to root (btol) and magnitude of
c       iterated change (xtol) must be supplied.
c       A tolerance for a singular Jacobian should also be provided.
c
c ------------------------------------------------------------------------
```

APPENDIX A (Continued)

```
c       This program written for the Fortran 77 compiler provided with
c       UNIX* 4.2 BSD on a SUN 3/160 workstation.
c
c       Portability: I/O units must be assigned appropriately.

c       *UNIX is a trademark of AT&T Bell Labs.

c -------------------------------------------------------------------------

c       All floating point computation is in double precision.

c -------------------------------------------------------------------------


        subroutine invert(u,v,x,y,xtol,btol,dettol)
        double precision u,v,x,y,xtol,btol,dettol

        integer stderr, stdin, stdout
        parameter(stderr=0,stdin=5,stdout=6)

        double precision fx,fy,dfxdx,dfxdy,dfydx,dfydy
        double precision bx,by,dxx,dxy,dyx,dyy,detJ

        double precision dx,dy

c -------------

        x=u
        y=v
1       continue

        bx = fx(x,y)-u
        by = fy(x,y)-v

        if (abs(bx)+abs(by).le.btol) return
c       Compute Jacobian at (x,y)
        dxx = dfxdx(x,y)
        dxy = dfxdy(x,y)
        dyx = dfydx(x,y)
        dyy = dfydy(x,y)

c       solve JdX = b

        detJ = dxx*dyy-dxy*dyx
        if (abs(detJ).le.dettol) then
                write(stderr,80) x,y
80              format(' invert: singular Jacobian at ',2f12.4)
                stop
        endif

        dx = (-dyy*bx+dxy*by)/detJ
        dy = (dyx*bx-dxx*by)/detJ

        x = x + dx
        y = y + dy

        if(abs(dx)+abs(dy).lt.xtol) return

        goto 1
        end

c -------------------------------------------------------------------------
c
c       rdcoef          Read and echo coefficients
c
c -------------------------------------------------------------------------

        subroutine rdcoef

        integer stderr, stdin, stdout
        parameter(stderr=0,stdin=5,stdout=6)
```

APPENDIX A (Continued)

```
      double precision xx,xy,xxx,xxy,xyy
      common /xcoeff/ xx,xy,xxx,xxy,xyy

      double precision yx,yy,yxx,yxy,yyy
      common /ycoeff/ yx,yy,yxx,yxy,yyy

c ---------------

      read(stdin,*) xx,xy,xxx,xxy,xyy
      read(stdin,*) yx,yy,yxx,yxy,yyy

      write(stdout,'(/)')
      write(stdout,80) xx,xy,xxx,xxy,xyy
      write(stdout,80) yx,yy,yxx,yxy,yyy
      write(stdout,'(/)')

80    format(5f15.8)
      end

c ------------------------------------------------------------------------
c
c     fx              fitting function (x coord)
c
c ------------------------------------------------------------------------

      double precision function fx(xv,yv)

      double precision xv,yv

      double precision xx,xy,xxx,xxy,xyy
      common /xcoeff/ xx,xy,xxx,xxy,xyy

      fx = xx*xv + xy*yv + xxy*xv*yv + xyy*yv*yv + xxx*xv*xv

      return
      end

c ------------------------------------------------------------------------
c
c     fy              fitting function (y coord)
c
c ------------------------------------------------------------------------

      double precision function fy(xv,yv)

      double precision xv,yv

      double precision yx,yy,yxx,yxy,yyy
      common /ycoeff/ yx,yy,yxx,yxy,yyy

      fy = yx*xv + yy*yv + yxy*xv*yv + yyy*yv*yv + yxx*xv*xv

      return
      end

c ------------------------------------------------------------------------
c
c     dfxdx           fitting function (x coord)
c
c ------------------------------------------------------------------------

      double precision function dfxdx(xv,yv)

      double precision xv,yv

      double precision xx,xy,xxx,xxy,xyy
      common /xcoeff/ xx,xy,xxx,xxy,xyy

      dfxdx = xx + xxy*yv + 2d0*xxx*xv

      return
      end
```

APPENDIX A (Continued)

```
c ------------------------------------------------------------------------
c
c       dfxdy           fitting function (x coord)
c
c ------------------------------------------------------------------------

        double precision function dfxdy(xv,yv)

        double precision xv,yv

        double precision xx,xy,xxx,xxy,xyy
        common /xcoeff/ xx,xy,xxx,xxy,xyy

        dfxdy = xy + xxy*xv + 2d0*xyy*yv

        return
        end

c ------------------------------------------------------------------------
c
c       dfydx           fitting function (y coord)
c
c ------------------------------------------------------------------------

        double precision function dfydx(xv,yv)

        double precision xv,yv

        double precision yx,yy,yxx,yxy,yyy
        common /ycoeff/ yx,yy,yxx,yxy,yyy

        dfydx = yx + yxy*yv + 2d0*yxx*xv

        return
        end

c ------------------------------------------------------------------------
c
c       dfydy           fitting function (y coord)
c
c ------------------------------------------------------------------------

        double precision function dfydy(xv,yv)

        double precision xv,yv

        double precision yx,yy,yxx,yxy,yyy
        common /ycoeff/ yx,yy,yxx,yxy,yyy

        dfydy = yy + yxy*xv + 2d0*yyy*yv

        return
        end
```

APPENDIX B
EST2

```
c ------------------------------------------------------------------------
c
c       est2    quadratic version.      Copyright (1988) Laurence T. Maloney
c
c       Find the quadratic geometric distortion function (Du,Dv) that
c       provides best fit to a given set of collinearity data.
c
c       Input data format specified in routine rddata below.
c       Uses the downhill simplex minimization algorithm described in:
c
c       Press, W. H., Flannery, B. P., Teulosky, S. A., & Vetterling, W. T.,
c       NUMERICAL RECIPES; THE ART OF SCIENTIFIC COMPUTING.
c       (Cambridge, England: Cambridge University Press, 1986).
c
```

APPENDIX B (Continued)

```
c ------------------------------------------------------------------------

c       This program written for the Fortran 77 compiler provided with
c       UNIX* 4.2 BSD on a SUN 3/160 workstation.
c
c       Portability: Calls to 'flush()' are specific to UNIX and may
c       simply be omitted.  I/O units must be assigned appropriately.

c       *UNIX is a trademark of AT&T Bell Labs.

c ------------------------------------------------------------------------

c       All floating point computation is in double precision.

c       I/O units [UNIX]:
c               stdin   input all data, initial values
c               stdout  output describing fit
c               stderr  errors and debugging output

c       stderr may be set to stdin on non-UNIX systems.

        integer stderr, stdin, stdout
        parameter(stderr=0,stdin=5,stdout=6)

c -------------------

c       A. ORIGIN, SCALE of the graphics device coord system.
c       These four constants control scaling and translation between
c       usual graphics coordinate system measured in pixel units and
c       the coordinate system we use internally, the unit square.
c       (x0,y0) is the pixel origin which maps to (0,0) internally.
c       delx, dely are the sizes of the sides of the unit square in
c       pixels.

        double precision x0, y0, delx, dely
        common /origin/ x0, y0, delx, dely

c       We keep two copies of all data read in, one copy in pixel
c       units ('raw') as integer variables, the other in unit
c       square units as double precision variables.

c -------------------
c       B. FIXED POINT DATA. These data constrain the linear component
c       of the transformation. See 'rddata' for format.

        integer NFIX
        parameter(NFIX=3)
        double precision  fixx(NFIX), fixy(NFIX)
        double precision  fixu(NFIX), fixv(NFIX)
        common /fix/ fixx,fixy,fixu,fixv

        integer  rfixx(NFIX), rfixy(NFIX)
        common /rawfix/ rfixx,rfixy

c -------------------

c       C. COLLINEARITY DATA. Triples of (x,y) coords intended to
c       be collinear after transform applied. These data constrain
c       the nonlinear component of the transform. See 'rddata' for
c       format.

c       The projection of the point v2 onto the line determined by
c       v1, v3 must lie between v1 and v3 (i.e. v2 is the middle
c       point of the nearly collinear triple v1, v2, v3. Routine
c       'colpen' below tests to be sure this condition holds.

        integer NCOL
        parameter(NCOL=1000)

        integer n
```

APPENDIX B (Continued)

```
      double precision v1x(NCOL),v1y(NCOL)
      double precision v2x(NCOL),v2y(NCOL)
      double precision v3x(NCOL),v3y(NCOL)
      common /dat/ n, v1x, v1y, v2x, v2y, v3x, v3y

      integer rv1x(NCOL),rv1y(NCOL)
      integer rv2x(NCOL),rv2y(NCOL)
      integer rv3x(NCOL),rv3y(NCOL)
      common /rawdat/ rv1x, rv1y, rv2x, rv2y, rv3x, rv3y

c -------------------

c     D. COEFFICIENTS of the quadratic fitting functions Du and Dv.
c     Note there are no constant terms. We are assuming that the
c     pixel origin maps to (0,0) in unit square coords.

c     Du(x,y) = c(1)*x + c(2)*y + c(3)*x**2 + c(4)*xy + c(5)*y**2
c     Dv(x,y) = c(6)*x + c(7)*y + c(8)*x**2 + c(9)*xy + c(10)*y**2


      integer NCOEFFS
      parameter(NCOEFFS=10)
      double precision c(NCOEFFS)

c -------------------
c     Variables needed for the downhill simplex minimization algorithm.
c     See Press et al.
      integer NMAX, NMAX1
      parameter(NMAX=10,NMAX1=NMAX+1)

      double precision LAMBDA
      parameter(LAMBDA=0.05d0)

      double precision q(NMAX1), p(NMAX1,NMAX), tmp(NMAX)

c -------------------

      integer i, j, ndim, iter
      double precision ftol


c     The penalty function to be minimized.

      double precision loss
      external loss

      integer count
      common /count/ count

c -------------------

c     Read initial values of coefficients.
      read(stdin,*) (c(i),i=1,NCOEFFS)
      write(stdout,80) (c(i),i=1,NCOEFFS)
80    format(9f8.4)

c     Read, echo, and transform data to unit square coords.
      call rddata

c     Initialize the simplex
      ndim = NMAX
      do 170 i=0,ndim
      do 150 j=1,ndim
      if (i.eq.j) then
              p(i+1,j)=c(j)+LAMBDA
              tmp(j)=c(j)+LAMBDA
      else
              p(i+1,j)=c(j)
              tmp(j)=c(j)
      endif
150   continue
      q(i+1)=loss(tmp)
```

APPENDIX B (Continued)

```
170     continue

c       Find functions Du, Dv which minimize penalty function for the
c       given data.
        count = 0
        ftol =1d-10
        call amoeba(p,q,NMAX1,NMAX,ndim,ftol,loss,iter)

c       Write out the final coefficients [c(j) = p(1,j)]
        write(stdout,'(/)')
        write(stdout,82) (p(1,j),j=1,5)
        write(stdout,82) (p(1,j),j=6,10)
        write(stdout,'(/)')
82      format(5f15.8)

        end


c -------------------------------------------------------------------------
c
c       rddata          Read and echo data
c
c       Format:         First the origin and scale of the x,y coords are
c                       read. All reads use FORTRAN77 list format.
c
c               (A)     x0      y0      xdel    ydel
c
c                       Then NFIX x,y pairs termed 'fixed points' are read.
c                       Each is coded as a quadruple of integers as
c                       each pair x,y is accompanied by a u,v pair that
c                       it is constrained to map to.
c                       Typical quadruples would be
c
c               (B)     fixx(1) fixy(1)   1  1
c                       fixx(2) fixy(2)  -1  0
c                       fixx(3) fixy(3)   0 -1
c
c                       A value of -1 means no constraint.
c                       In the example, the third point must map to an x
c                       value of 0 but may be anywhere on that edge.
c
c                       These mapping represent constraints that the final
c                       fitting maps should satisfy.
c
c               (C)     The fixed points are followed by triples of x,y
c                       points (6 integers) which are taken to be collinear in
c                       external u,v space. The fitting program will minimize
c                       their deviations from collinearity.
c
c -------------------------------------------------------------------------

        subroutine rddata

        integer stderr, stdin, stdout
        parameter(stderr=0,stdin=5,stdout=6)

c -------------------

c       A. ORIGIN, SCALE of graphics device coord system.

        double precision x0, y0, delx, dely
        common /origin/ x0, y0, delx, dely

c -------------------
c       B. FIXED POINT DATA

        integer NFIX
        parameter(NFIX=3)
        double precision  fixx(NFIX), fixy(NFIX)
        double precision  fixu(NFIX), fixv(NFIX)
        common /fix/ fixx,fixy,fixu,fixv
```

APPENDIX B (Continued)

```
      integer  rfixx(NFIX), rfixy(NFIX)
      common /rawfix/ rfixx,rfixy

c -------------------

c     C. COLLINEARITY DATA

      integer NCOL
      parameter(NCOL=1000)

      double precision v1x(NCOL),v1y(NCOL)
      double precision v2x(NCOL),v2y(NCOL)
      double precision v3x(NCOL),v3y(NCOL)

      integer n

      common /dat/ n, v1x, v1y, v2x, v2y, v3x, v3y

      integer rv1x(NCOL),rv1y(NCOL)
      integer rv2x(NCOL),rv2y(NCOL)
      integer rv3x(NCOL),rv3y(NCOL)

      common /rawdat/ rv1x, rv1y, rv2x, rv2y, rv3x, rv3y

c -------------------

      integer i

c -------------------

c     The transformed values are double precision and fall
c     in the unit square. All computations are done in terms
c     of the transformed values.

c     Read origin, scale of graphics device. E.g. Delx is the
c     length of the line from (0,0) to (1,1) in pixels.
c     (x0,y0) is the origin in pixel coordinates.
      read(stdin,*) x0, y0, delx, dely
      write(stdout,'(/4f20.5//)') x0, y0, delx, dely

c     The origin maps to the origin automatically by choice of
c     the polynomial fitting functions. Three more points
c     are needed to pin the coordinate system. These points
c     define horizontal and vertical.

c     Read 'fixed points'.
      do 100 i=1,NFIX
      read(stdin,*) rfixx(i), rfixy(i), fixu(i), fixv(i)
      fixx(i) = rfixx(i)
      fixy(i) = rfixy(i)
100   continue

c     read data to be fit (3 points at a time)
      n=0
2     continue
      read(stdin,*,end=3) rv1x(n+1),rv1y(n+1)
      read(stdin,*) rv2x(n+1),rv2y(n+1)
      read(stdin,*) rv3x(n+1),rv3y(n+1)
      n=n+1
      v1x(n) = rv1x(n)
      v1y(n) = rv1y(n)
      v2x(n) = rv2x(n)
      v2y(n) = rv2y(n)
      v3x(n) = rv3x(n)
      v3y(n) = rv3y(n)
      if (n.ge.NCOL) pause 'rddata: too many data points.'
      goto 2

3     continue

c     Change coords so that output rectangle in the unit square
c     for convenience.
```

APPENDIX B (Continued)

```
        do 300 i=1,NFIX
        write(stdout,82) rfixx(i), rfixy(i), fixu(i), fixv(i)
        fixx(i) = (fixx(i)-x0)/delx
        fixy(i) = (fixy(i)-y0)/dely
300     continue

        write(stdout,'(/)')

c       Confirm remaining input and transform to the unit square
        do 400 i=1,n
        write(stdout,80) rv1x(i),rv1y(i),rv2x(i),rv2y(i),rv3x(i),rv3y(i)
        v1x(i) = (v1x(i)-x0)/delx
        v2x(i) = (v2x(i)-x0)/delx
        v3x(i) = (v3x(i)-x0)/delx
        v1y(i) = (v1y(i)-y0)/dely
        v2y(i) = (v2y(i)-y0)/dely
        v3y(i) = (v3y(i)-y0)/dely
400     continue

        write(stdout,'(/)')

        call flush(stdout)

        return
80      format(6i10)
82      format(2i10,2f10.3)
        end

c -------------------------------------------------------------------------
c
c       loss    Compute the penalty for the current function.
c
c -------------------------------------------------------------------------

        double precision function loss(c)

c -------------------

c       D. COEFFICIENTS of Du, Dv

        integer NCOEFFS
        parameter(NCOEFFS=10)

        double precision c(NCOEFFS)

c       Relative weight of fixed point penalty versus collinearity
c       penalty.
        double precision BETA
        parameter(BETA=10000d0)

c -------------------

c       B. FIXED POINT DATA.
c       Only the internal copies of the data needed.

        integer NFIX
        parameter(NFIX=3)
        double precision  fixx(NFIX), fixy(NFIX)
        double precision  fixu(NFIX), fixv(NFIX)
        common /fix/ fixx,fixy,fixu,fixv

c -------------------

c       C. COLLINEARITY DATA.

        integer NCOL
        parameter(NCOL=1000)
        double precision v1x(NCOL),v1y(NCOL)
        double precision v2x(NCOL),v2y(NCOL)
        double precision v3x(NCOL),v3y(NCOL)
        common /dat/ n, v1x, v1y, v2x, v2y, v3x, v3y
```

APPENDIX B (Continued)

```
c ------------------

c       Count calls to loss by minimization program. Used to
c       selectively print debugging output every 50 calls.
        integer count
        common /count/ count


c       Functions called (colpen is collinearity penalty).
        double precision Du, Dv, colpen

        double precision sum, fixed
        integer i, n, err1, err2, err3, sumer1, sumer2, sumer3
        double precision ax,ay,bx,by,cx,cy

c ------------------

c       Compute penalties for fixed points. Recall that a code of
c       -1 in fixv[xy] means no constraint and therefore no penalty.
        fixed=0d0
        do 100 i=1,NFIX
        if (fixu(i).gt.-0.5) then
                fixed = fixed+(fixu(i)-Du(fixx(i),fixy(i),c))**2
        endif
        if (fixv(i).gt.-0.5) then
                fixed = fixed+(fixv(i)-Dv(fixx(i),fixy(i),c))**2
        endif
100     continue

c       Now tranform each of the triples intended to be collinear
c       by the candidate function and compute the collinearity
c       penalty. For convenience, the routine 'colpen' returns
c       other pieces of information about the quality of the fit
c       in terms of pixel units. See colpen.
        sum = 0d0
        sumer1 = 0
        sumer2 = 0
        sumer3 = 0
        do 200 i=1,n
        ax = Du(v1x(i),v1y(i),c)
        ay = Dv(v1x(i),v1y(i),c)
        bx = Du(v2x(i),v2y(i),c)
        by = Dv(v2x(i),v2y(i),c)
        cx = Du(v3x(i),v3y(i),c)
        cy = Dv(v3x(i),v3y(i),c)
        sum=sum+colpen(ax,ay,bx,by,cx,cy,err1,err2,err3)
        sumer1 = sumer1 + err1
        sumer2 = sumer2 + err2
        sumer3 = sumer3 + err3
200     continue


        loss = sum + BETA*fixed
c       The following is debugging output which may be eliminated
c       or reduced in quantity without affecting the program.
        count = count + 1
        if ( 50*(count/50) .eq. count ) then
          write(0,70) count, sum, fixed, loss, (c(i),i=1,NCOEFFS)
70        format(/' ss ', i6, 3f20.15,/(5f15.7))
          write(0,72) sumer1
72        format(' Number collinearity errors exceeding one half pixel:  ',i8)
          write(0,74) sumer2
74        format(' Number collinearity errors exceeding one pixel:       ',i8)
          write(0,76) sumer3
76        format(' Number collinearity errors exceeding two pixels:      ',i8)
          call flush(0)
        endif

        return
        end
```

APPENDIX B (Continued)

```
c ------------------------------------------------------------------------
c
c       colpen  Return the squared norm of the distance between the point
c               bx,by and the line determined by ax,ay,cx,cy.
c
c ------------------------------------------------------------------------

        double precision function colpen(ax,ay,bx,by,cx,cy,err1,err2,err3)
        double precision ax,ay,bx,by,cx,cy
        integer err1, err2, err3

c -------------------

c       A. ORIGIN, SCALE of the graphics device coord system.

        double precision x0, y0, delx, dely
        common /origin/ x0, y0, delx, dely

        double precision dx,dy,ex,ey,errx,erry
        double precision norme2,alpha

c       change affine origin to vector a.
        dx = bx-ax
        dy = by-ay
        ex = cx-ax
        ey = cy-ay

c       alpha is proportion of vector e occupied by projection of
c       vector d. Let d.e denote the inner product of the vectors
c       d and e. then the length of the projection is d.e/|e|  and the
c        proportion is d.e/|e|^2.

        norme2 = ex*ex + ey*ey
        alpha = (dx*ex + dy*ey)/norme2

c       if alpha is not in (0,1) then then b is not between a
c       and c and we've blown it.

        if ((alpha.lt.0).or.(alpha.gt.1.0)) then
                write(0,76) alpha
76              format(' colpen: alpha out of range: ',f20.10)
        endif

c       Compute the error vector.

        errx = dx - alpha*ex
        erry = dy - alpha*ey

c       Count errors of various magnitudes (pixel units).

        err1 = 0
        if (abs(errx)*delx.gt.0.5) err1 = err1+1
        if (abs(erry)*dely.gt.0.5) err1 = err1+1

        err2 = 0
        if (abs(errx)*delx.gt.1.0) err2 = err2+1
        if (abs(erry)*dely.gt.1.0) err2 = err2+1

        err3 = 0
        if (abs(errx)*delx.gt.2.0) err3 = err3+1
        if (abs(erry)*dely.gt.2.0) err3 = err3+1

        colpen = errx*errx + erry*erry

        return
        end

c ------------------------------------------------------------------------
c
c       Du              Du function of quadratic (u coord)
c
c ------------------------------------------------------------------------
```

APPENDIX B (Continued)

```
      double precision function Du(x,y,c)

      integer NCOEFFS
      parameter(NCOEFFS=10)

      double precision c(NCOEFFS)
      double precision x,y

      Du = c(1)*x + c(2)*y + c(3)*x*x + c(4)*x*y + c(5)*y*y

      return
      end

c -------------------------------------------------------------------------
c
c       Dv              Dv function of quadratic (v coord)
c
c -------------------------------------------------------------------------

      double precision function Dv(x,y,c)

      integer NCOEFFS
      parameter(NCOEFFS=10)

      double precision c(NCOEFFS)
      double precision x,y

      Dv = c(6)*x + c(7)*y + c(8)*x*x + c(9)*x*y + c(10)*y*y

      return
      end
```

(Manuscript received February 18, 1988;
revision accepted for publication April 18, 1988.)